SP14 Green Chess AI

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

Course: CS 4850-02

Semester: Spring 2023

Professor Perry

Date: 2/12/23

Team:

Tyler Ebersold

Michael Dirksen

Logan Lane

Table of Contents

1.0 Introduction

1.1 Overview

Our project sets out to create a new variant of the popular board game chess. Our version of chess will feature a new unique ruleset and piece behavior. An AI opponent will also be incorporated into our project for the user to rival. The AI must be able to make educated moves by using logic to compare various moves and select the best one.

1.2 Project Goals

The goal of our project is to create a new and exciting way for users to play chess. This project also strives to make a challenging but fair AI opponent for the user to face.

1.3 Definitions and Acronyms

- AI - Artificial Intelligence
- UI - User Interface

1.4 Assumptions

The AI will assume the opponent will make good plays when considering its next move.

2.0 Design Constraints

2.1 Environment

We chose the environment Unity to host our game and design. Unity was picked due to its flexibility when incorporating visual elements, such as the board and the pieces.

2.2 User Characteristics

The main users of our application will be people who have played chess before and already have an understanding of the game. Our ruleset for our chess game will be more complicated and will have various twists when compared to the base game. The AI will also be of moderate difficulty where the user has a chance to win, but the AI should win the majority of games simply due to human error.

2.3 System

Our current design plan is to export the final version of our application into an exe file that will be downloadable from our website. Everything needed to play the game will be loaded into the executable. We'll be using Unity Collaborate to facilitate simultaneous workflow. The files will then be published to a Github repository that will be linked on our website.

3.0 Requirements

1.0 Game Ruleset

1.1 Backline pieces are placed in random positions. The chosen positions are the same for both players.

1.2 Random squares not containing a piece will be selected to contain mines, which destroy any pieces that move onto it. These mines are shown briefly at the beginning of the game, but never again. After a mine detonates, it is gone forever.

1.3 Pawns can move diagonally forward as well as one square forward. On a pawn's first move, they may still move forward two squares, but cannot move two squares diagonally.

1.4 All other regular chess rules are in effect

2.0 Building the game

2.1 The game will be built in Unity and intended for Windows systems.

2.2 The project will use two-dimensional graphics

2.3 Develop game board and piece movement

2.3 Have a way for the AI to translate its decisions to in-game actions

3.0 Building the AI

3.1 Find a way to recognize valid, legal moves only

3.2 Utilize min-max algorithm to evaluate move effectiveness

3.3 Employ alpha-beta pruning to improve performance and eliminate unneeded moves

3.4 Make sure the AI never spends too long on a single move

4.0 User experience

4.1 The user will play vs our AI

4.2 Before the game begins, the user is allowed to choose the number of mines from a range of zero to five

4.3 The UI will be lightweight and clear. Information about the game, such as the active player's turn, check status, and captured pieces will be clearly displayed to the player.

APPENDICES

Min-Max Algorithm: Recursive algorithm that evaluates a game backwards, assuming the opponent will play correctly, and finds the best way to reach that winning state.

Alpha Beta Pruning: Search algorithm that looks to prune a tree in order to make a traversal algorithm more efficient. When poorly performing moves are found, that entire branch of the tree will be trimmed off, meaning the min-max algorithm has much fewer moves to go through.