SP-14 Red Build Chess Game Using AI

Date: 4/5/23   Website: https://sites.google.com/view/sp14red/home

Roles	Name	Major responsibilities	Contact (Email and/or Phone)
Team leader	Aaron Dailey	Team lead, Documentarian, quality tester, and designer	aarondailey9@gmail.com 706-312-4017
Team members	Nicholas Kennel	UI designer and Implementer	nickovertime9@gmail.com 770-899-8206
	Haige Zhu	AI designer and implementer	hzhu5@students.kennesaw.edu 847-848-5735
Advisor / Instructor	Sharon Perry	Facilitate project progress; advise on project planning and management.	Sperry46 in D2L !!

Aaron Dailey   Nicholas Kennel   Haige Zhu

Contents

1.0 Introduction

The following project is a chess game that allows the user to play against an Ai on a Windows computer made from C#, Ai algorithms, and runs on the Unity engine.

1.1 Overview

The project is an AI chess game where the user will play against a CPU chess player.

1.2 Project Goals

The goal of the project is for the user to play a chess game with Ai.

2.0 Design Constraints

2.1 Environment

Chess is a class that controls the logical operation of the game and forms the core of the software. It has two independent auxiliary classes: ChessBoard and LegalMoveSet. ChessBoard mainly records the current state of the chessboard in various forms. LegalMoveSets is a chess rule class used to determine whether a move is legal and whether the game is over. It can also form the action set of each chess piece under the current state.
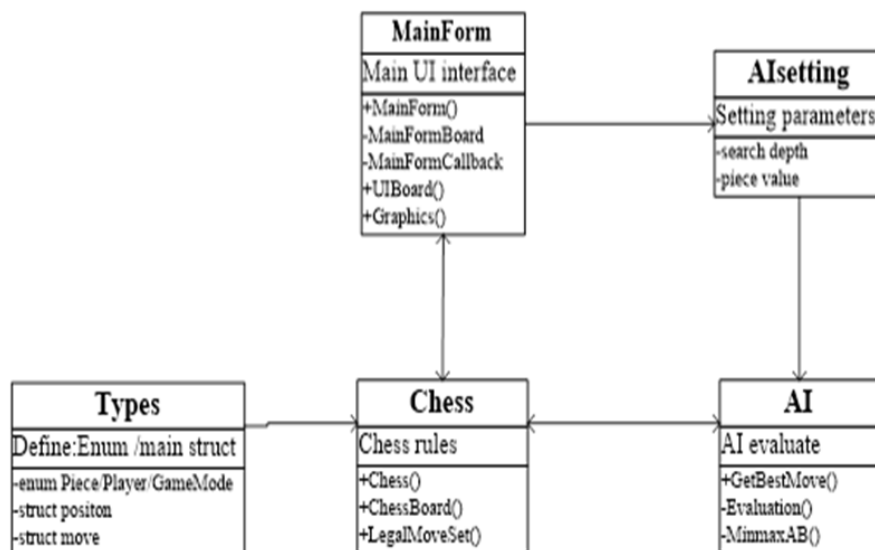
MainForm is the form that runs the game and serves as the UI. It has two shadow forms, MainFormBoard and MainFormCallbacks, to ensure smoother interface operation.

AI is an AI that evaluates values and uses a minimax search with alpha-beta pruning to make the best move decisions.

Types define various enumeration values and structures required by the system, such as chess piece enumeration, player enumeration, game mode enumeration, chess piece position structure, and chess piece movement model structure.

AIsetting allows users to set the game difficulty (search depth) and chess piece strength values through the FrmSetting form interface to affect the AI evaluation results.
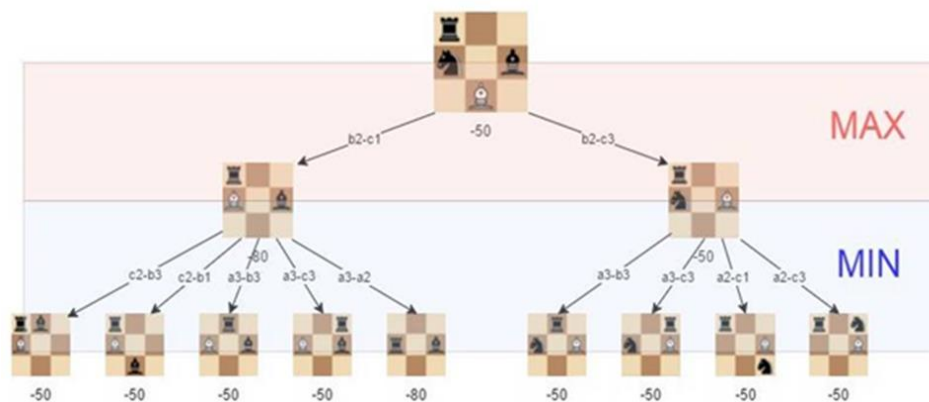
2.2 User Characteristics

2.3 System

```
        ┌─────────────┐
        │   AI turn   │
        └──────┬──────┘
               │
               ▼
   ┌───────────────────────┐
   │  Generate All Legal   │
   │  Moves for current    │
   │        Board          │
   └───────────┬───────────┘
               │
               ▼
   ┌───────────────────────┐
   │  Use search to simulate│
   │   some pieces moves    │
   └───────────┬───────────┘
               │
               ▼
   ┌───────────────────────┐
   │   Use the evaluation  │
   │  function to evaluate │
   │       the score       │
   └───────────┬───────────┘
               │
               ▼
   ┌───────────────────────┐
   │ Get best move and make│
   │          move         │
   └───────────┬───────────┘
               │
               ▼
   ┌───────────────────────┐
   │    Opponent's turn    │
   └───────────┬───────────┘
               │
               └─────────────────►
```

	White		Black
Pawn	10		-10
Knight	30		-30
Bishop	30		-30
Rook	50		-50
Queen	90		-90
King	900		-900

MAX

MIN

b2-c1   -50   b2-c3

c2-b3 c2-b1 a3-b3 a3-c3 a3-a2   a3-b3 a3-c3 a2-c1 a2-c3
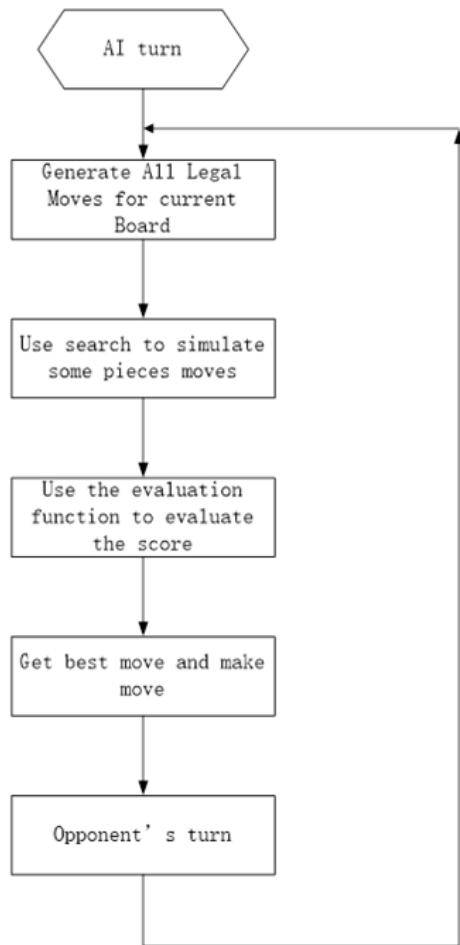
-50   -50   -50   -50   -80   -50   -50   -50   -50

3.0 Requirements

3.1 User Requirements

The game begins with a start screen where it would allow you to do 2 options, one start game which will bring up a canvas, or a different scene which in that case lets you pick which starting color you want. The starting color will be the player while the opposite color is the AI. From there on, it is going to go to the main chess game scene where the player can play the AI. The chess game scene is going to consist of some canvases for the UI, chess pieces, and the chess board. It is going to be a basic UI that lets you be

able to just click on a chess piece, then it is going to enable you to see where that piece can move as well as where it can move and be vulnerable to enemy pieces.

 After that you move a chess piece, it will automatically switch from the player's turn to the AI's turn, and then the AI's turn will then select a chess piece and then move their chess piece. It will switch back and forth until it gets to a point where one of the chess pieces will take another chess piece. When a piece takes another chess piece it would replace that piece or delete the enemy's chess piece. Then that will continue until you take the enemy (AI) king or until the enemy takes your king and at that point, it will change a win condition through either you or the AI to true. Once the king is taken, instead of going to the enemy's turn or your turn it will, then just go to the victory screen or defeat screen.

Finally, you could choose to replay the game again.

3.2 AI Requirements

We use the training data as chess games and rules. Our goal for the AI part is at least for the AI to be able to move the piece by the logic and the rules, while AI plays with the user. For example, if the player the user uses white and he makes a move with the pawn to E4, then the AI will use the minimax algorithm to evaluate the optimal solution based on the user's moves, and then the Alpha-beta algorithm will optimize the calculation time, trying to optimize the time to within a few seconds

Deep learning and data sets to train AI to achieve AI to play against people. It can set the difficulty. The AI part is to calculate the possibility of winning each step through the Alpha -Beta pruning algorithm and reduce the calculation time through algorithm optimization. The Minmax algorithm is used to evaluate the optimal solution based on the user's moves.

4.0 External Interface Requirements

4.1 User Interface Requirements

An interactive chess piece, available chess movement, pause screen, start screen, and a victory/lose menu.

5.0 Hardware Interface Requirements

A computer that has x64 architecture for its cpu, and gpu that has DX10, DX11, and DX12-capable GPUs.

5.1 Software Interface Requirements

Required software need is Windows 7 (SP1+) or Windows 10, 64-bit versions or Windows 11.

5.2 Tech platform

describe in detail the tools you are using – use TECHNICAL TERMS

5.3 Development of software project

Discuss your development effort – what went well, roadblocks, work arounds

6.1 Results

where are you with this software

6.2 Project Plan and Management

Meeting Schedule Date/Time

Mondays 4- 5 pm and Wednesdays 4-5 pm

Collaboration and Communication Plan

- Discord will be the main form of communication otherwise other forms of communication will be via email or phone. ▯ Regular meetings will be held on Mondays and Wednesdays 4-5 pm in discord if needed meetings can be arranged at other times. The requirements for meetings is a microphone and pc or phone to communicate in discord. Meeting and distributing notes will be handled by the Team leader to keep everyone on what is needed; otherwise, if the leader cannot attend, the notes will be delegated to one of the other team members.
- The tool we will use for communication will be Discord while file sharing of documents will be by OneDrive and sharing of code will be by GitHub. Discord is how we will communicate where all files are in OneDrive and GitHub as well as post shared links to the files. GitHub will be used as our website but also where we will share code and versions of our game. OneDrive will be the documentation for our deliverables will be stored.
- The regular status updates and progress will be reported bi-weekly in each meeting Mondays and Wednesdays. If needed other updates may come outside those frames are to keep the workflow ahead of schedule or to accomplish a certain task that was not available previously. All members of the team are responsible for reporting their updates to other members, so we all know what we need to work on for the project

Project Schedule and Task Planning (Ganatt chart)

Version Control Plan

The plan for the version control is to make the coding and software in Unity and then export it to our GitHub to maintain and document different versions of the game. The GitHub will have the most updated version of the game and will reflect what code has already been done.

6.3 Test Plan

The minimum amount of tests should be 2 for each level for each game type except player vs player will be tested 2 times since it's not reliant on artificial intelligence, and the tests must be conducted on a Windows computer. The total amount of tests should be around 32, and to see if it has any other problems on other computers. It will be tested on another computer for the same making it 64 tests. All ai will be timed to see if it's the game loading time is dependent on the CPU and GPU. If the CPU and GPU are, it shows that we need to change our current code to be more efficient. The overall test plan is

to make sure all game modes work and make sure the ai is efficient in playing the game of chess with the user.

6.4 Summary and Conclusion