Software Development

TSA State Competition

Shogi AI and Client

将棋

Seven Springs, PA

15 April 2015

Contents

1 Research **2**

2 Overview **5**

3 Documentation **6**

3.1 Project Requirements 6

3.2 Software Design 6

3.3 Testing 6

3.4 User Manual 6

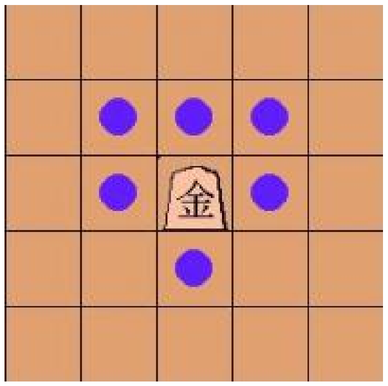
4 Evaluation **6**

5 Bibliography **6**

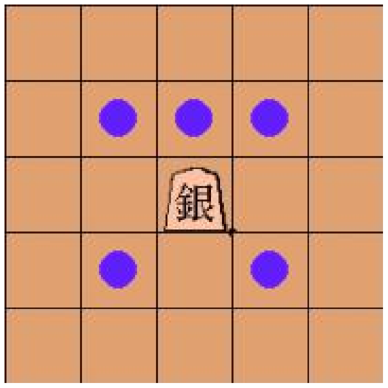
1 Research

Our research was focused on the rules and gameplay of shogi, the Japanese analogue of chess. The goal of the game, as with the familiar version of chess, is to place the opponent's king in checkmate. The game is played on a nine-by-nine board with eight kinds of pieces:

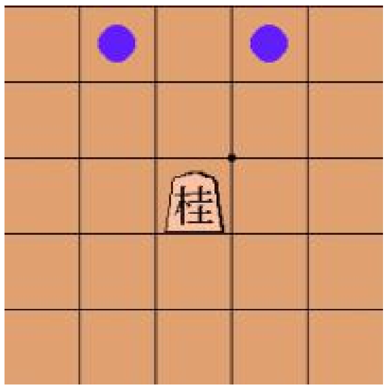
- Pawns move forward one square at a time
- A king can move one square in any direction, although it is subject to being placed in check and suffers the associated restrictions the same way as a king from chess
- Gold generals can move one square in any direction except diagonally backwards



- Silver generals can move straight or diagonally forward, as well as diagonally backwards



- A knight moves forward two spaces and one space to the side, jumping over any intervening pieces, mimicking the motion of its chess counterpart except for its inability to move sideways or backwards



- A lance moves any number of open spaces straight forward
- A rook moves any number of open spaces forward, backward, or to the sides
- A bishop moves any number of open spaces in any diagonal direction

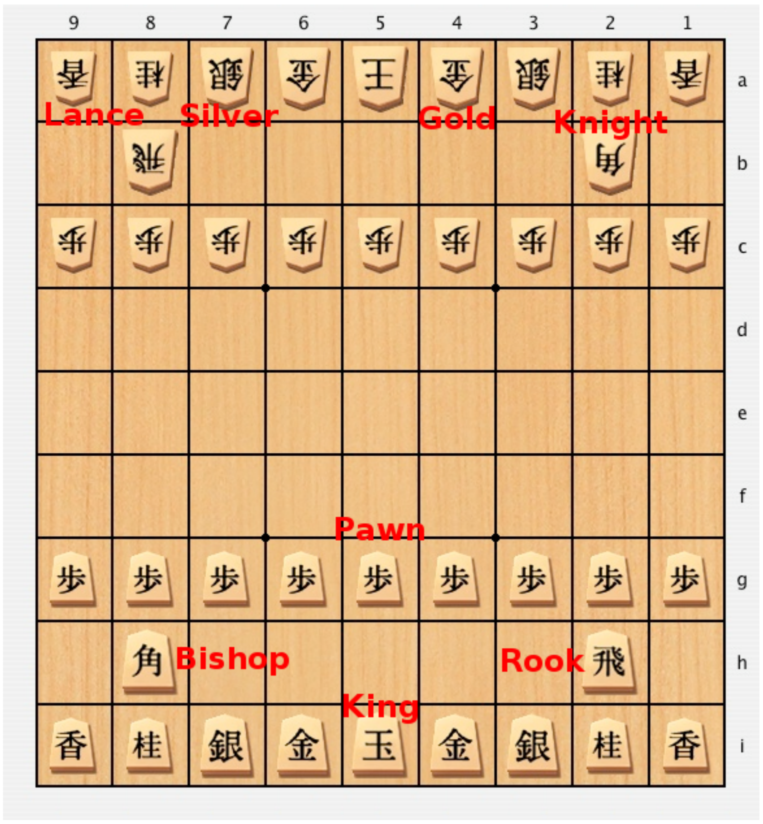
In addition to the original pieces are promoted pieces. Promotion is a feature of shogi similar to pawn promotion in standard chess, but each piece has a specified promotion. Two pieces (the king and the gold general) are not promoted at all if the conditions are met. Promotion may occur when a piece moves into any space among the far three rows of the board, although a piece does not have to be promoted if the player doesn‘t want to. Promoted pieces move as follows:

- A promoted pawn, silver general, knight, or lance moves as if it were a gold general.
- A promoted rook or bishop gains the ability to move one square in the directions it couldn’t prior to the promotion (so a promoted rook would be able to move one space diagonally, and a bishop could move one space vertically or horizontally)

Pieces are captured when one piece moves into a square occupied by an opposing piece, just like in chess. A piece’s promotion is lost upon being captured.

In addition to promotion, shogi adds another unique mechanic: “drops.” A drop is when the player places a piece previously captured from her opponent somewhere on the board. A drop takes a full turn, and the only restrictions on it are that a piece cannot be dropped somewhere it would be unable to move the next turn (so, a pawn cannot be dropped into the back row), and a player may not place a pawn in such a way that he will have two in the same column. A piece cannot be promoted by dropping it into the back three rows, and a piece loses its promotion when captured, so a piece will never be promoted on the turn it is dropped. After the drop the piece acts as it normally would, following all the rules as if it had moved to its current location normally.

The shogi board is laid out as follows, on a 9x9 board.



In this section, the mechanics of the AI itself were glossed over. However, these will be explored in detail in the Software Design section.

2 Overview

The main goal of the Shogi AI project is to provide an educational environment for newcomers to learn how to play the game. In summary, we hope to accomplish this by providing a few elements:

- Graphical User Interface for move input
- Basic level of Shogi AI to play against for beginners
- Game commentary provided by an AI

In future versions, we hope to include:

- Server integration so the user can play against opponents across the globe.
- Advanced AI that can challenge and beat the best players
- Shogi variants to mix up play

3 Documentation

3.1 Project Requirements

3.2 Software Design

Since this was a solo project, many of the pitfalls of group work were avoided, especially in fitting modules together and getting them to communicate properly. However, good modular design makes for code reuse and simplicity of bug fixing later on. It is a tenet of open source software development.

Image of hierarchy.

A large majority of project time was spent on the artificial intelligence. To simplify the process, a function library was created to determine the legality of moves, make moves, print the board to stdout, etc. Then the AI would call these functions, and use its data structures to build a decision tree. Each node on the tree is evaluated and "scored". From there, pruning is executed on the tree, to decrease the amount of memory and processing power consumed by the program. This is called alpha-beta pruning (more on that later). Finally, the "best" available move is executed.

To construct the tree, dynamic allocation was used along with C structs. During the runtime of the program, thousands of nodes are created. Each parent node starts with every possible legal move on the board at the current state below it. Then, alpha-beta pruning is used to cut down the amount of nodes that must be evaluated. Alpha-beta is very difficult to explain in prose, the wikipedia article has an excellent animation, as well as pseudocode.

Finally, interfacing the GUI with the AI program created a unique challenge. I wanted to keep the GUI and AI as two distinct elements, so other AIs could be developed to be compatible with our interface. However, as deadlines crept nearer, I realised that it would be very difficult to accomplish this under the time constraint.

3.3 Testing

3.4 User Manual

4 Evaluation

5 Bibliography