12/26/2021

Term Project Report

Kamran Musayev
CSE 462

1. GAME DESIGN

To make the game more understandable and versatile, it was built with simplicity in mind. It contains four classes, each with a distinct function. The classes are AI, Board, Game, and Window.

Game class includes all of the other classes' instances, initiates variables, and contains the game loop, which runs using the runnable interface. Furthermore, a win/loss condition method is implemented, which prohibits the game from continuing to play at the ending.

Window class offers a frame for the GUI as well as mouse movement capability, which is the primary mechanism for playing the game.

Board class is the most significant class, which contains all of the game's functions and mechanics such as illegal movements, possible moves, captures, and turn calculation methods.

AI class includes AI features and computing techniques.

2. GAME FLOW

The game consists of a 7x7 board with four triangle (AI) and four circle (Player) pieces. The player's movement requests are registered via the drag and drop mouse input technique. Each player can move twice every turn, but if one of the players only has one piece remaining, that player can only move once per round. The pieces may move both horizontally and vertically. Diagonal motions are not permitted. If a player possesses more than one piece, he or she should make two successive movements using different pieces. If a player just possesses one piece, he or she should only make one move. If one of your pieces/pieces is caught between a wall and an opponent piece or between two opponent pieces, that piece/pieces is captured. It is considered a draw when neither player has any pieces or has only one piece. When one player has certain pieces but the opposing player has not, the player wins, and vice versa. However, if both players have the same number of pieces after 50 moves, the game is a draw; if the player has more pieces, the player wins; otherwise, the player loses.

3. AI MECHANICS

The AI class utilizes the bestMove () method to obtain the potential moves for each piece in that turn, and then recursively performs the minimax algorithm for each potential move, which works as follows:

```
If depth = height or remaining moves = 0:
        Return evaluation ()
If turn is Triangle:
        Loop through board checking for triangle possible moves
        Check for captures
        Set last move
        Calculate whose next turn is
        Call minimax recursively with increased depth value in each recursion
        Set best minimum score value
        Set alpha beta values
        Return best minimum score value
Else if turn is Circle:
        Loop through board checking for possible circle moves
        Check for captures
        Set last move
        Calculate whose next turn is
        Call minimax recursively with increased depth value in each recursion
        Set best maximum score value
        Set alpha beta values
        Return best maximum score value
```

Minimax delivers the best possible score for each conceivable move, which is then fed into the algorithm through a loop. At the end of the loop, the best score is chosen, as well as the piece information and location coordinates for moving the piece.

Minimax Tree looks as follows:

White = AI

Black = Player