Christopher Murray, Kevin Vece, Juan Reyes
CSCE 315-501 Group F
Project 2: Design Document

Overview

This project combines artificial intelligence techniques with a client server model to produce a network based reversi game. Numerous problems emerge when we try to tackle this project. The artificial intelligence engine must use the previously established minimax tree method for decision making, which requires the development of an equation to describe favorability of the board state. The server must also accept multiple connections so many users can play concurrent reversi games. We envision many reversi enthusiasts using this program as their goto method for playing.

High Level Design Components

The high level design components for this project include the representation of the game board, the AI, a client-server model based on the command line, and a GUI client.

The game board will be in charge of representing, managing, flipping, and adding game pieces, as well as being able to display the current board state to the user when asked.

The AI component of the design will handle all possible moves that it is able to make based on the current game board configuration. There will be multiple difficulties that the AI can embody - Easy, Medium, and Hard. Each successively harder difficulty is expected to be able to defeat the easier difficulties the vast majority of the time, with an even higher chance for the Hard AI to defeat the Easy AI.

This implementation of Reversi can be played in several forms - AI vs Human, AI vs AI - and will be played by connected to a specified server through a client program. The client program will be what the Human player interacts through (as well as an AI in an AI vs AI game) to connect to the server and play against another player or the AI. The server will act as the AI player in a game where an AI is involved, and receive moves a player makes, and invoke the AI to make a move based on the difficulty choice.

The GUI will be the application the user uses to connect to the server and play the game. It will display the game board, as well as other information relevant to the game, as well as all of the options available to the user..

Low Level Design Components

The game board has several low level components - the current location of all pieces on the game board and their associated player, the player who is currently playing, and all available moves the current player can make. The representation of the pieces on the board will be controlled by a matrix with values representing each game board space, and whether it has no piece currently occupying it, a piece from player one, or a piece from player 2. The game board will also note which player is currently selecting a move and be able to determine, and display, all available moves that the current player is able to make. If the player selects to play a piece in an invalid space, the game board will be able to notify the player that he has selected an invalid move.

The game AI will be able to be invoked with varying difficulties - easy, medium, and hard - as well as utilize multiple different search algorithms to determine which space to play a piece in. The search algorithms as well as the AI difficulty levels will go hand in hand. The easy difficulty AI will pick randomly from all available moves to simulate a new player to the game. The medium difficulty AI will use a min-max tree search algorithm with a limited depth in order to select a move, to simulate a player thinking ahead several moves. The hard difficulty AI will also use a min-max tree search algorithm, but with a depth that is greater than the medium difficulty AI.

The client-server relationship will include the command line client that the user will initially use to connect to the game server. The server itself will be what the user connects connects to, and where the game board is stored, and where the AI will determine which moves to make. The client will initially be designed as a command line application that takes input in the form of commands (such as selecting AI difficulty, and issuing moves) which are then sent to the server, evaluated, and the resulting board state, or

other command output, is printed to the screen. After the command line implementation, a GUI will be created that will have all the same functionality as the command line version, but in a much more user-friendly and accessible representation. The server will be designed using TCP streaming sockets for connection over a network and will allow multiple client connections at once. The client will communicate with the server using the following defined grammar.

expr	::= *command* | *move* | *comment*
command	::= EXIT | DISPLAY | *difficulty* | UNDO | HUMAN-AI|AI-AI <server> <port> *difficulty difficulty*
difficulty	::= *EASY|MEDIUM|HARD*
move	::= *column row*
comment	::= ; *
column	::= a | b | c | d | e | f | g | h
row	::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8
digit	::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
server	::= IP address or hostname
port	::= digit { digit }

The client GUI application will be designed in OpenGL for C++, as that will provide an easy base to design and implement all features that need to be present in our GUI. The GUI will consist of a representation of the game board as well as all the available options that were implemented in the CLI version of the game. The available options that the user can select from are placing a piece, undo/redo their moves, quitting the game, and selecting the ai difficulty.

Benefits, Assumptions, Risks, and Issues

- Benefits
  - Many people can play reversi at once
  - Offers multiple difficulties for players of any skill
  - GUI offers easier interaction with the game
- Risks
  - Network must remain accessible
  - Player understands the rules of the game
  - Depending on depth of min-max algorithm, move picking may take more time that a player is willing to wait
- Assumptions
  - Minimum to no human error on GUI
  - Player is not intentionally attempting to break the program