Amazon Project Report

Group 11

Program Structure

Our program has a few classes that compensate each other: aiPlayer, ChessBoard, Moves, and NodeGraph. Each has their own unique purpose and usage. AiPlayer extends ChessBoard class and contains important functions such as: graph search method that calls relevant algorithms depending on how far the game goes, and minimax algorithm, which we will go over in this report. Moves class makes sure the chess pieces are moving based on the rules of the game. NodeGraph class extends ChessBoard class and generates nodes for the graph search method. Last, ChessBoard class creates the structure of the chess board, a randomMove() method in the ChessBoard class to allow black and white queens to move randomly by default, as well as some constructors.

Classes

Chessboard.java

The constructor of this class creates a chess board with 4 white queens and black queen. The position of white and black queen is set to the initial position. The constructor with the given parameter copies the same constructor with the given chessboard object.

Attributes:

- POS_BLACK: black queen
- POS_WHITE: white queen
- POS_ARROW: arrow
- N: total number of spaces of the chess board

- W_POS: position of white queen
- B_WHITE: position of black queen

Methods:

- getTile: returns the tile of a given row and column as an integer.
- 2 setTile methods: set the tile with a given position with given value, and the other setTile method set tile with different parameters, which are index and value.
- getBoard: returns the board array.
- checkFin: check if the game ends or not. It checks if there is a space to move or not. If no spaces are left, the game will finish, and the method will return true, otherwise return false.
- moveChess: moves the chess to the given position.
- randomMove: move the black or white chess randomly if the game hasn't finished. The random method is a helper method of randomMove. It generates random directions and distance and moves the chess to the generated position.
- getPositions: returns the position of given color chess.

NodeGraph.java

The NodeGraph class is a child class of Chessboard. Besides its parent class, it has 8 attributes. The oldQueen stores the old queen position in an arraylist, newQueen stores the new queen position in an arraylist, newArrow stores the new arrow position in an arraylist. Value, value1, value2, and depth are double and int type.

There are 4 NodeGraph constructors. The first one creates a NodeGraph with a given board. The second and third constructors create NodeGraph with a given value. The fourth

constructor creates NodeGraph with the given old position, new position, values, board, and depth.

Methods:

- addChild: adds a child to the nodeGraph.
- addChildren: adds a list of children to the nodeGraph.
- getValue: returns the value.

Moves.java

The moves class has 9 attributes. 8 of them are unchangeable integers, which store the 8 different directions: N, NE, E, SE, S, SW, W, NW. The other attribute is valid, which stores the directions in an arraylist.

The constructor of this class initializes the move object with an arraylist.

Methods:

- getMoves: returns an array of chessboards with the given position.
- moveN: returns how many spaces left of north direction, and moveNhelp is a helper method of moveN.
- moveNE: returns how many spaces left of north direction, and moveNEhelp is a helper method of moveNE.
- moveE: returns how many spaces left of north direction, and moveEhelp is a helper method of moveE.
- moveSE: returns how many spaces left of north direction, and moveSEhelp is a helper method of moveSE.

- moveS: returns how many spaces left of north direction, and moveShelp is a helper method of moveS.
- moveSW: returns how many spaces left of north direction, and moveSWhelp is a helper method of moveSW.
- moveW: returns how many spaces left of north direction, and moveWhelp is a helper method of moveW.
- moveNW: returns how many spaces left of north direction, and moveNWhelp is a helper method of moveNW.

aiPlayer.java

The constructor creates an aiPlayer with a given chessboard and initializes the nodeGraph attribute. The second constructor creates an aiPlayer with a given aiPlayer parent and chessboard.

The ai method calls relevant algorithms depending on how far the game goes by evaluating the counter.

Attributes:

- Parent (type: aiPlayer)
- Board (type: ChessBoard)
- Graph (type: NodeGraph)

Methods:

- createnNodes: creates new nodes to the graph.
- addToGraph: addes new children to the graph.
- updatePos: returns a new position based on the old position and distance.

- getOppColor: returns the opponent color of the given color chess.
- getOppPos: returns the opponent's position of given color chess.

COSC322Test.java

New attributes updated:

- aiPlayer
- ChessBoard
- moving
- myColor
- oppoColor
- count

Methods:

- play: takes several parameters and connects to the gui by calling the sendMoveMessage and updateGameState.
- handleGameMessage: called by the GameClient when it receives game related messages from the server. If the game starts, the method will keep track of the chess information of both sides by calling different relevant methods of other classes.

Implementation

Minimax algorithm

aiPlayer.java

line 96-131

The minMax method is the minimax algorithm to prioritize maximizing the moves by minimizing the opponent's available moves.

Formula

$$v_i = \max_{a_i} \min_{a-i} v_i(a_i, a - i)$$

i = index of the player of interest

$-i$ = denotes all other players except player i

a_i = action taken by player i

$a - i$ = actions taken by all other players

v_i = value function of player i

Challenges of the project

The challenges we faced is that we don't know how to connect to the gui after finishing implementation. We solved this problem later by discussing with other groups and reading the API.

Reference

Q. Jianning, Q. Hongkun, W. Yajie, L. Fei and Q. Shengran, "Application of UCT technologies for computer games of Amazon," *2016 Chinese Control and Decision Conference (CCDC)*, 2016, pp. 6896-6899, doi: 10.1109/CCDC.2016.7532241.