New Bulgarian University

NETB 375 Programming practice

AI Chess Game

by:
Atanas Chorbadjiiski, F70431

Project Description

Using the programming language C++ and programming framework Qt implement a computer program the famous chess game. The game must gave two modes of play: human against human (on the same computer), and human against CPU. The program must have the following features:

- fully functioning GUI that represents the chess board and all pieces;

- white piece can be moved if it is a white player's move, and the same for black pieces;

- each piece is allowed to perform only valid moves according to game rules;

- AI complexity is left to project developers, and it can be the simplest possible -- a random move.

Design and implementation of the AI.

The design of the AI is the most simple – it generates a random move out of the possible moves of the figures which are currently on the board.

My implementation consists of an ai class and 2 functions:

1. Controller function - **getAiTurn**

2. Class **ai**

3. ai function – **getAllRandomValidTurns**

1. getAiTurn

This function traverses the board and takes the coordinates of every figure on it in the vector **piecePositions**. After getting the coordinates of all the figures of the board, the function checks to see if the figure is the same color of the current player(which is the AI/Computer) then the coordinates of said figure will be pushed into a vector of coordinates. The vector **piecePositions** is then returned.

2. Class AI

The class's purpose is to generate a random move. The idea behind the movement of the pieces is to have a source coordinate of the figure and a target coordinate. What the AI class does is take a vector of the coordinates of all the pieces, which is provided by the controller function **getAiTurn**, on the board which fit the color of the AI(or Computer) and generate, through a random principle, the target coordinates which will be where a randomly chosen figure will be moved.

3. getAllRandomValidTurns

This function of the AI class generates the target coordinates by a random principle. The function takes the vector of coordinates from **getAiTurn** into a new vector called **allAiMoves**. A new two dimentional vector is created called **validAiMoves**, which will be needed to store all the possible moves of a piece from **allAiMoves**.

Two integers are created, named **sourceColumn** and **sourceRow**, which will be used to store the source coordinates of the figure which will be moved. Later these coordinates will be needed.

The vector **allAiMoves** is being traversed in order to find the coordinates which fit our requirements – have at least one possible move. First the row and the column on the board is generated from the coordinate functions getColumn() and getRow() by using the coordinates from the vector. After that the column and row which are the source coordinates of the figure are stored for later use, provided every other condition later on in the loop is met. In the case that the conditions are not met, then the values will be replaced with the next iteration of the loop. By using the coordinates, the piece at the location is checked to see what kind of piece it is. Depending on the type, different possible moves in the form of two dimentional vector are generated and stored into the two dimentional vector called **validAiMoves**. Here, if there are no possible moves for the figure, meaning the generated vector is empty, then the current itteration of the for cycle stops with break; and proceeds with the next one, in other words with the next coordinates from the vector allAiMoves.

Now that the vector **allAiMoves** has possible moves stored, a random coordinate can be chosen as a target for the movement of the piece. To choose a random coordinate from allAiMoves, first a random number is generated by using rand() with size of the vector being it's highest possible value.

With the random number generated with rand(), the vector of possible moves at the position of the random number is assigned to a new vector called **finalCoordinateList**. From there by using the same principle with rand() a coordinate is chosen as the target coordinate and stored in another vector called **sourceTarget**.

From earlier the **sourceRow** and **sourceColumn** integers which were used to store the initial source of the figure are assigned as a coordinate to the sourceTarget vector as a source coordinate of the figure.

The source coordinate is always stored on the first element, while the target coordinate is stored at the second element of the vector sourceTarget. The function returns sourceTarget.