 **New Bulgarian University**

Bachelor Faculty

Bachelor programme Network Technologies

NETB 375 Programming Practice

AI Chess Game

Prepared by:

Vladimir Petkov, F 74482

Sofia, 2017

Content:

Problem description ………………………………….……..………………….. 3

Project architecture overview ………… ..……………………………………… 4

The chess state …..…………………… ..……………………………………… 5

The chess logic ………..…….………………………………………………….. 6

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.

Project Architecture Overview

The chess game is organized as a three-layer application. The three main layers are the presentation logic (GUI), the ‘chess-logic’ which contains the business logic of the game and the ‘chess-state’ which stores the current state of the game.

The presentation logic (the GUI) knows and communicates only with the ‘controller’ which is the main part of the chess-logic. This way the GUI is responsible only for visualization and interaction with the user. It does not contain any chess specific logic, it does not perform any direct changes the chess state.

The ‘chess-logic’ contains all of the logic specific to a chess game. It is formed by the Controller, CheckChecker, SpecialMovesHandler and AI classes. They are responsible for enforcing the rules of the game and changing the state.

The ‘chess-state’ represents the state of a chess game. It is formed by the State, Player, Cell, GameType and all Piece classes. They model the chess game and store all the necessary information to replicate the state of a chess game.

The chess state

As stated earlier, the ‘chess-state’ represents the state of a chess game. It is formed by the State, Player, Cell, GameType and all Piece classes. In this document the Piece classes will not be explained as they are written by Vasil Yoshev.

The ‘**Cell**’ class represents a cell from a chess board. It has a default and a copy constructor.

It has the following member fields:

* Piece\* piece - A reference to a piece object. If there is no object on the cell the reference is a NULL reference.

And the following methods:

* void setPiece(Piece\* piece) – A setter for the piece reference.
* Piece\* getPiece() – A getter for the piece reference.

The ‘**GameType**’ is an enum that lists the possible game types:

* Not Selected - the initial value when no specific game type is selected.,
* Player vs Player - the game type when two players play against each other
* Player vs CPU - the game type when a player plays against the AI.

The ‘**Player**’ class represents a Player. It has a default and a copy constructor.

It has the following member fields:

* string name – the name of the player
* Color color – the color of the player
* bool inCheck - flag to indicate if the player is in check
* bool inCheckmate - flag to indicate if the player is in checkmate

And the following methods:

* void setName(string name) – setter for the name of the player
* string getName() – getter for the name
* void setColor(Color color) – setter for the color of the player
* Color getColor() – getter for the color
* void setInCheck(bool inCheck) – setter for the inCheck flag
* bool isInCheck() – getter for the inCheck flag
* void setInCheckmate(bool inCheckmate) - setter for the inCheckmate flag
* bool isInCheckmate() – getter for the inCheckmate flag

The ‘**State**’ class represents the actual state of the chess game. Just like the other classes it has a default and a copy constructor.

fIt contains the board which is two-dimensional vector of Cell objects. There are no getters and setters for this field, instead there are setPiece() and getPiece() methods which return or place the piece on the desired cell. The class stores the two players in a static array of size 2 as well as the index and a reference to the current player. There is a getter for the current player and a method that changes the current player to the other player. Also there are setters for the two players. The ‘State’ also holds the game type as an enum value. There is a getter and a setter for it. It also holds track if there is a pawn in promotion state (A pawn is in promotion when it reaches 8th rank. Then it should be changed with either a queen, rook, bishop or knight). This is the reason there is a flag inPawnPromotion and Coordinate field which stores the coordinates of the pawn that is in promotion. There are setters and getters for these two fields. There are also three private methods that help the initialization of the board – initPieces() which calls setPawnPieces() which sets Pawn pieces on entire row and setMajorPieces() which sets the major pieces on entire row. Thera is also shortcut for checking the check status of the current player – getCheckStatusCurrentPlayer(). The class also has a copy constructor which is used to copy the current state of the chess game.

The chess logic

As stated earlier the ‘chess-logic’ contains all of the logic specific to a chess game. It is formed by the Controller, CheckChecker, SpecialMovesHandler and AI classes. In this document the Controller class and the CheckChecker class will be explained, because SpecialMovesHandler is written by Vasil Yoshev and AI is written by Atanas Chorbadjiiski.

The ‘’