****

**Department of Computer Science**

**Final Project Report**

**Chess Game Design**

By:

Rodin Ashmoz

Supervisor:

Dr. Tamar Zemach

November 2020

Table of contents

1 [Abstract 3](#_Toc55585864)

2 [Introduction 4](#_Toc55585865)

3 [Game Description 4](#_Toc55585866)

3.1 [Game propeties 4](#_Toc55585867)

3.2 [Game loop 5](#_Toc55585868)

3.3 [Description of the flowchart 6](#_Toc55585869)

4 [Game modes 7](#_Toc55585870)

4.1 [AI player 7](#_Toc55585871)

4.1.1 [Algorithms 7](#_Toc55585872)

4.1.2 [Evaluation function 7](#_Toc55585873)

4.1.3 [Minimax search 8](#_Toc55585874)

4.1.4 [Alpha-Beta pruning 9](#_Toc55585875)

4.2 [Performances 9](#_Toc55585876)

4.3 [Online 9](#_Toc55585877)

4.4 [Multiplayer 10](#_Toc55585878)

5 [Game Graphics 10](#_Toc55585879)

6 [Game Development 11](#_Toc55585880)

7 [Removed Features in the game 11](#_Toc55585881)

8 [Final Remarks & Future work 11](#_Toc55585882)

8.1 [Future improvements 12](#_Toc55585883)

9 [References 12](#_Toc55585884)

# Abstract

This report documents the design and development of chess computer game application with graphic user interface, the algorithms and heuristics used to built an AI player, how the game runs behind the scenes and the graphics that operates the application . The program is written in Python language, without using any chess libraries, every function is written from scratch.

Further it discusses the different modes the game consist of - to play against an AI player which is based on Minimax algorithm with optimizations, an option to play over the network, locally or remote network, and a multiplayer game which two users can play each other on the same computer*.*

# Introduction

Chess is a [strategy](https://en.wikipedia.org/wiki/Abstract_strategy_game) [board game](https://en.wikipedia.org/wiki/Board_game), which is one of the oldest and most popular board games, played on a checkered board by two players white and black of 8x8.

Since the emergence of artificial intelligence, better and newer algorithms have been developed to improve and create different chess engines, the ability of a computer to play chess. This project is inspired by such developments.

The aim of this project is designing a chess game, which contains the following properties:

* Uses graphic user interface.
* A game that can be played against AI player which has a chess engine based on Minimax (Negamax form) algorithm. The difficulty based on depth of 1 – 3 in the Minimax algorithm tree, namely - easy medium and hard.
* LAN game (local network), by hosting a server and client connection.
* Multiplayer on the same computer.

# Game Description

## Game Properties

* Played on 8x8 board, represented as list board [8][8].
* 2 players participating, white and black.
* Each place in the board represented string of 2 chars:

"(w/b ( = color), w ( = kind))".

* Kinds: king = k, queen = q, rook = r, bishop = b, knight = n, pawn = p.
* Color: white = w, black = b.
* Empty places represented as " - -".
* Each piece has its own unique moves.
* No skipping turn - moving is compulsory.
* Each piece has its own legal moves, One valid move each turn.
* Can't jump over pieces, path is blocked unless it’s a knight piece when moving.
* The aim of the game is to capture opponent's king.
* The game can end with a Win/Lose or a Draw.

## Game Loop

Y

Reverse round

/Restart game

Reverse ?

/Restart

Exit game?

Y

Exit

Wait for action

Start game

Made Move?

Y

AI player

N

Is Valid Move?

Y

N

Change turn

Is game over?

Y

Display game over

AI player

Waiting for action from the user

## Description Flowchart

* *Start game* – begins with white playing the first turn. we can see the initial position of the game on the picture on the right side.
* *Wait of action* - the program waits for the user to choose between multiple options, interacting by the mouse/keyboard:
  + If pressed 'p' – undo the last move by reversing to the previous position round one before.
  + If pressed 'r' – the game initilisizes to the initial position of the game as the picture describes above.
  + If mouse clicked twice, checks if it’s a valid move or not and acts accordingly.
  + If clicked Esc, exits from the program.
* *Is Valid Move* – after 2 clicks, the first click in the board terms considered as "from which square" or selected square, the second click as "to which square" or where to move, and finds if it’s a valid move:
* *Is Game Over* – check if in this current position we got 1 of the 2 options:
* Checkmate – king's opponents got captured = winner.
* Stalemate - current player has no possible moves and its king not in check = draw.
* *Change turn* – change turn from white/black, AI/human player.
* *Display Game Over* – display on the screen which player won the game. or a draw.
* *AI player (optional)* - AI player calculated its best move based on Minimax algorithm.

# Game Modes

## AI Player

### Algorithms

* In order to make an AI to play chess on its own, it must be operated by a chess engine that runs by algorithms.
* The chosen algorithm is Minimax search with alpha-beta pruning and "reordering moves" for decrease search time.
* The every path of movements (possible moves) measured by Evaluation function.
* The AI chooses the best possible move namely the move with the highest value.

### Evaluation Function

* An heuristic evaluation function, used by game playing computer programs to estimate the value of certain action, in chess terms – a possible move(s). the better the fuction, the better choices the AI makes.

The formula used for the AI is Claude Shannon's formula [1]:

f(p) = 200(K-K') + 9(Q-Q') + 5(R-R') + 3(B-B' + N-N') + 1(P-P') - 0.5(D-D' + S-S' + I-I') + 0.1(M-M')

The values can be seen in the table below [2]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Symbol** | [Chess plt45.svg](https://en.wikipedia.org/wiki/File:Chess_plt45.svg) | [Chess nlt45.svg](https://en.wikipedia.org/wiki/File:Chess_nlt45.svg) | [Chess blt45.svg](https://en.wikipedia.org/wiki/File:Chess_blt45.svg) | [Chess rlt45.svg](https://en.wikipedia.org/wiki/File:Chess_rlt45.svg) | [Chess qlt45.svg](https://en.wikipedia.org/wiki/File:Chess_qlt45.svg) |
| **Piece** | [pawn](https://en.wikipedia.org/wiki/Pawn_(chess)) = p | [knight](https://en.wikipedia.org/wiki/Knight_(chess)) = n | [bishop](https://en.wikipedia.org/wiki/Bishop_(chess)) = b | [rook](https://en.wikipedia.org/wiki/Rook_(chess)) = r | [queen](https://en.wikipedia.org/wiki/Queen_(chess)) = q |
| **Value** | 1 | 3 | 3 | 5 | 9 |

**- Note the king equal 200 or in other words innumerable compared to the rest.**

D = Doubles - 2 pawns on the same column.

S = blocked pawns on the board (no possible move).

M = Mobility - the number of legal moves for current player (any moves by all pieces).

### Minimax Search

An algorithm or A tree search where the leaf nodes has values, formulated for n-player [zero-sum](https://en.wikipedia.org/wiki/Zero-sum) [game](https://en.wikipedia.org/wiki/Game_theory), where (assuming) the players always make the best choice relative to them in each split on the tree, in chess it can interpreted as follows:

* The splits are the possible moves.
* Each split represents player choices (each depth in the tree equals turns).
* Player 1 tries to maximize the value, by choosing the highest path (move) each split.
* Player 2 tries to minimize the value, by choosing the lowest path (move) each split.
* Depth of the tree equals the number of moves the AI can see ahead.

### Alpha-Beta Pruning

* This technique used to improve minimax algorithm to decrease the time search, which implementing brute force (searching all possible options).
* Skips certain places in search tree where it can inferred the best choice to make.
* The AI searches only where it necessary to get the best possible move.
* Uses also algorism called "reordering moves" where the list of possible moves sorted by piece values which make the pruning move efficient.

## Performances

The graph below shows the average time in seconds it take for each depth, or path of moves the AI can search ahead:

**Depth**

As it can be seen, the alpha beta pruning makes the search much more faster and efficient.

* At depth 4 alpha-beta pruning gets the Agv of 6-7 seconds, for Minimax alone it takes a lot of time to measure.

## 

## LAN

* This option makes possible for two players to connect with each other by a server and client and play the game.
* The host creates a server.
* The client connects to the server by the IP address of the host.

## 

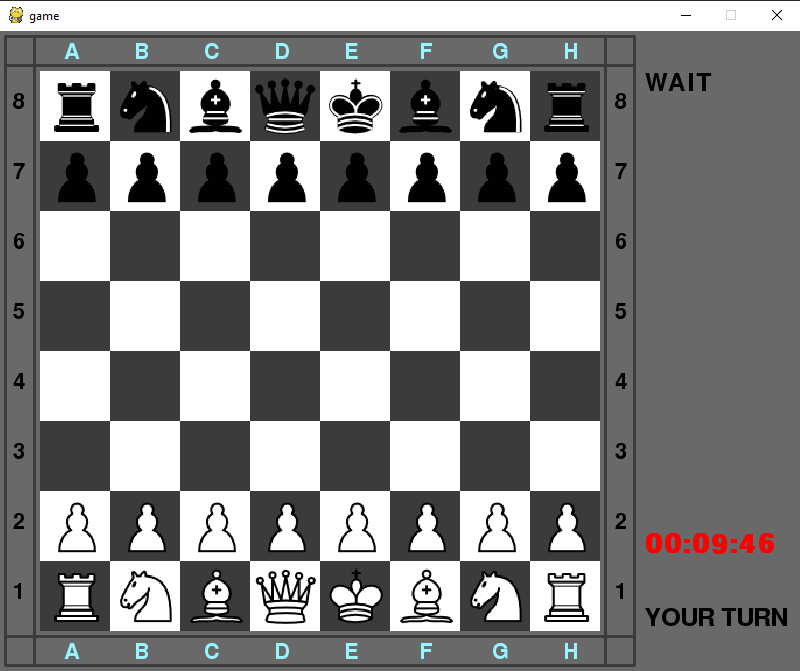
## Multiplayer

* Offline mode, which two players can play on the same computer.

# Game Graphics

* The graphic design built by the library "Pygame" [3] .
* Pygame is library of set python modules for creating a 2D games, which includes computer graphs and also sound libraries.
* The image pieces taken from Wikipedia, 6 for each color.
* Color board and the game environment graphics created by using Pygame library.
* The user has the option to change the graphic inside the game.
* Note each piece represented by two chars: first = color, second = kind





# Game Development

* The first version of the game was made without graphics, in order to fit it latter.
* It wasn’t efficient, with too many functions and unnecessary code, that could be written better.
* All the pieces had a own class
* Didn’t include move class, a very important element for games like chess.
* In the second version (the current program), all the mistake were rectified.
* Removed the classes for each piece, replaced only with string representation (much faster and easier to use).

# Removed Features

* There are some canceled features that were included in the first design, but dropped for the main reason that it impacts the run time or requires changing the structure of the current game in and out.
* Piece Square Tables - assign values to specific [pieces](https://www.chessprogramming.org/Pieces) on specific [squares](https://www.chessprogramming.org/Squares), a unique board for each piece.
* Replace the board game (and the game structure as consequence) to use bits, which are way faster than lists or arrays.
* Write in faster computer language for the game to run much faster, like C or C++.
* Hash tables with selected algorithms
* Opening book – saves in data the previous positions of the board and their best moves played before (history) as numeric value. Each time player makes a move, it won't need to search again by minimax search.
* And there are so many approaches and algorithms for chess engines that are not mentioned here [4].

# Final Remarks & Future work

* In this project, we have built a chess game that supports 3 modes, game vs AI player, playing online or multiplayer on the same machine.
* For the AI player, the deeper the search tree (searching for possible moves x moves ahead, the more time it takes to calculate.
* Different algorithms and heuristics have impact the quality of the AI and run time of the game.
* Minimax algorithm alone its not efficient for large search trees.
* Alpha-beta pruning improves drastically the performance as shown in figure
* This version of the game can be upgrade a lot as mentioned above.

## Future Improvements

There are innumerable ways to improve any (obviously including this current program) chess game, not to mention even the best and popular chess engines are improving over time with new algorithms and heuristics, considering ideas from the best chess engines out there, and learn from the best.

* As mentioned above, in all of these ways this program can improve a lot, starting a "new version" with faster program language, new algorithms to decrease the run time and add move algorithms, With these changes, the game can run with depth of 5-6.
* Another different and interesting approach is to build a game based on **Reinforcement learning,** which is the computer learns to play by itself after training it by trial and error.
* Adding new modes to the game, other versions of chess game out there.
* Adding chat box for 2 human players can text with each other while playing
* Convert the game into a website application, where any amount of people can enter and play.

# References

[1] [**Claude Shannon**](https://www.chessprogramming.org/Claude_Shannon), [*Programming a Computer for Playing Chess*](http://www.pi.infn.it/~carosi/chess/shannon.txt) http://www.pi.infn.it/~carosi/chess/shannon.txt

[2] Chess piece relative value <https://en.wikipedia.org/wiki/Chess_piece_relative_value>

[3] Pygame [https://pythonprogramming.net](https://pythonprogramming.net/)

[4] Rrepository of information about programming computers to play chess https://www.chessprogramming.org/