Final Project Report

CS101 Projects 2014

Team ID: 201

Chess

Team leader: Rahul Govind, 140010011
Sanket Doshi, 140010001
Tanmay Deshmukh, 140110011
Saurabh Gajula, 140010040

Table of Contents

1 Introduction

Chess is a two player strategy board game played on a 8x8 checkered board. Each player has 16 pieces: 8 pawns, 2 rooks, 2 bishops, 2 knights, a queen and a king. The objective of the game is to 'checkmate' the opponent by placing his king under the threat of an inescapable capture.

With the advent of computers and the development of the mathematics related to game theory, it has become possible to design programs which can make intelligent moves against other computers or other players. There are many chess engines now which can play as good as the top Chess Players in the world, if not better.

1.1 Definitions, Acronyms and Abbreviations

Core Engine -	Refers to the part of the program which validates the moves.
AI Engine -	Refers to the part of the program which evaluates which move should be made by the computer.
Front-end -	Refers to the part of the program, which displays the board and the chess pieces, and handles the interface.
Pawn -	A piece that can move only one step ahead at a time, or two from its initial position. It captures a piece diagonally.
Bishop -	A piece that can move and capture diagonally
Rook -	A piece that can move and capture, vertically and horizontally.
Knight -	A piece that moves 2 steps vertically and 1 step horizontally, or 1 step vertically and 2 steps horizontally. It captures in similar fashion, and can jump over other pieces.
Queen -	A piece that can move and capture in any direction.
King -	A piece that can move and capture only one step in any direction.
Check -	The situation when the king is under direct attack.
Checkmate -	The situation when the king is under direct attack, but cannot avoid capture.
Stalemate -	The situation when the king is not direct attack, but no other move can be made which does not put the king under direct attack.
Pawn promotion -	The situation where a player's pawn reaches the end of the opponent's side of the board, and can be promoted to any other piece.

1.2 References

E. N. Barron, An Introduction to Game Theory, Wiley - India, 2012
Jon Kleinberg, Algorithm Design, Pearson Education, 2006

2 Overall description

2.1 Product Perspective

The purpose of the project is to design a chess engine which can be used by amateur and professional chess players, and also for research purposes. Another purpose of the project is to design a portable front-end which can communicate with other commercial and amateur chess engines

2.2 Product Functions

The program implements a graphical user interface with which the user can control his moves, change game settings and play against the computer. The computer also validates each move. It also implements an AI engine which can evaluate different moves, and compute the best possible move that can be played against a human competitor.

2.3 User Characteristics

It would be recommended that the user knows the basic rules of chess. However, the front-end is designed to highlight the possible moves that can be made from a given position.

2.4 System requirements

OpenGL API Version 3.2 or higher
Hard Disk: 20 MB of free hard disk space

Other requirements for Developer:
MinGW Developer Environment with g++ version 4.7.1 or higher
CMake Version 2.8 or higher

3 Details

3.1 Objectives planned and completed

3.1.1 Initial objectives

Goal 1: To build an elegant front end.
Goal 2: Construct an AI Chess Engine for single player version of the game.
Goal 3: Improve the depth and efficiency of the Chess Engine and minimize the time consumed for move analysis.
Goal 4: To top the AI Engine performance.

3.1.2 Status of initial objectives

All initial objectives were completed. The implementation is discussed under the section "Implementation".

3.1.3 Extra features

1) Debugging option for advanced users and developer. All games are stored in debug.dat. The saved game can be reloaded by running Chess.exe with debug.dat as an argument.

2) Custom GL library to decrease the complexity of programming using OpenGL in C++. The custom GL library also makes it easier to make UI elements using OpenGL and can be used to teach OpenGL.

3.2 Code format and supportability

- The code is formatted in the All Man style.
- Variables have all been named in lower case letters.
- Classes, functions and structures have been named in Camelcase starting with upper case letters

3.3 Design constraints

The Engine is capable of handling all moves which are possible in a standard game of chess. Software and hardware constraints are listed in the System Requirements section.

3.4 User documentation and Help system

It is only recommended that the end user know the basic rules of chess. However, the interface is self-explanatory and no end-user help system is required.

3.5 Interfaces

- Mouse input. (To select and move pieces)

- Keyboard input (For shortcuts and to pause game)

3.5.1 User Interface

Pre-game interface:	Interface shown before the game has started. It gives the user the option to select the player mode.
Interface during game:	Interface shown while the game is going on. It cosists of the interface to control the pieces.
Pause game interface:	Interface shown when the user(s) pauses the game. It gives the user the option to restart the game, undo moves and exit the program.
Post-game interface:	Interface shown after the game ends. It gives the user the option to restart the game,and displays the result of the game.

3.5.2 Hardware Interface

The following devices are required for the interface -
- A mouse
- A keyboard
- A monitor

3.5.3 Software interface

Minimalistic GNU for Windows (MinGW) for developer

4 Implementation

4.1 Front end

Dependencies: The GUI and graphics were all rendered using OpenGL through the GLFW (OpenGL Framework) utility library. OpenGL extensions i.e. Modern OpenGL functions were loaded using GLEW (OpenGL Extension Wrangler) library. The GLM (OpenGL Mathematics) library was used to implement the various mathematical functions and data structures for rendering.

Final rendering: The chess board consists of 64 squares. Each square was rendered as two separate triangle. The chess pieces were "drawn" onto the board by using an image of the chess pieces as a texture for the triangles.

4.2 Engine

The Board class contains all the necessary functions and data to define and control a board. It contains functions to determine if a move is valid, whether the player is "checkmated", etc.

The Engine class is built upon the Board class to create an interface with the front-end. It was designed to handle one-way communication from the front end i.e. it does not need to know any information about the front-end and doesn't send any input to the front end.

4.3 Move evaluation and Artificial Intelligence

The EngineAI class is built upon the Board class and is used to determine the best possible move by the computer. The EngineAI class also creates a separate class to look for the best move so that the main loop i.e. the GUI can still take other input.

The basic algorithm used is the Minimax algorithm. It has been optimized using Alpha-Beta pruning and a customized move sorting algorithm.

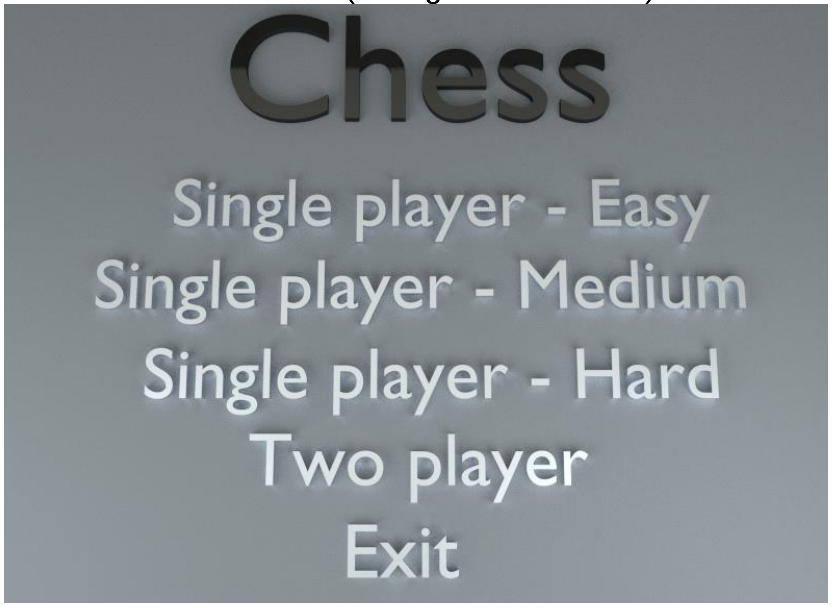
5 Challenges

- Adding human touch to the Chess Engine and implementing different strategies in the game according to the stages in the game.
- Adding special Chess moves i.e. En passant, Castling and Pawn Promotion.
- Switching preferences between positional and material advantages in the game depending on the number of moves played in the game. i.e: According to different junctures of the game.
- Creating a User Interface using GLFW and OpenGL since they do not provide any functionality to create user interface elements.

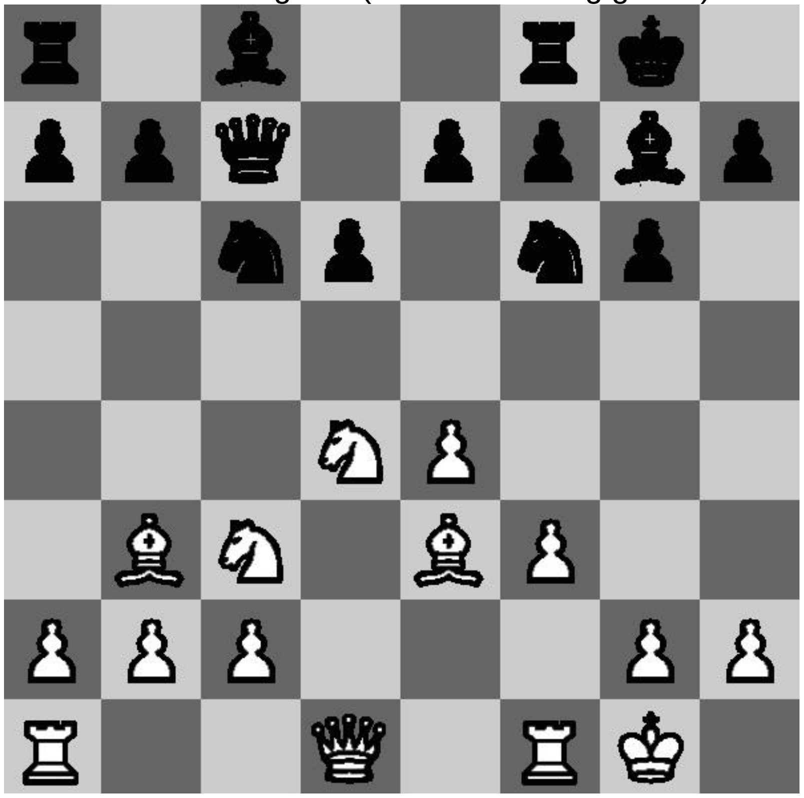
6 Final product overview

6.1 Images:

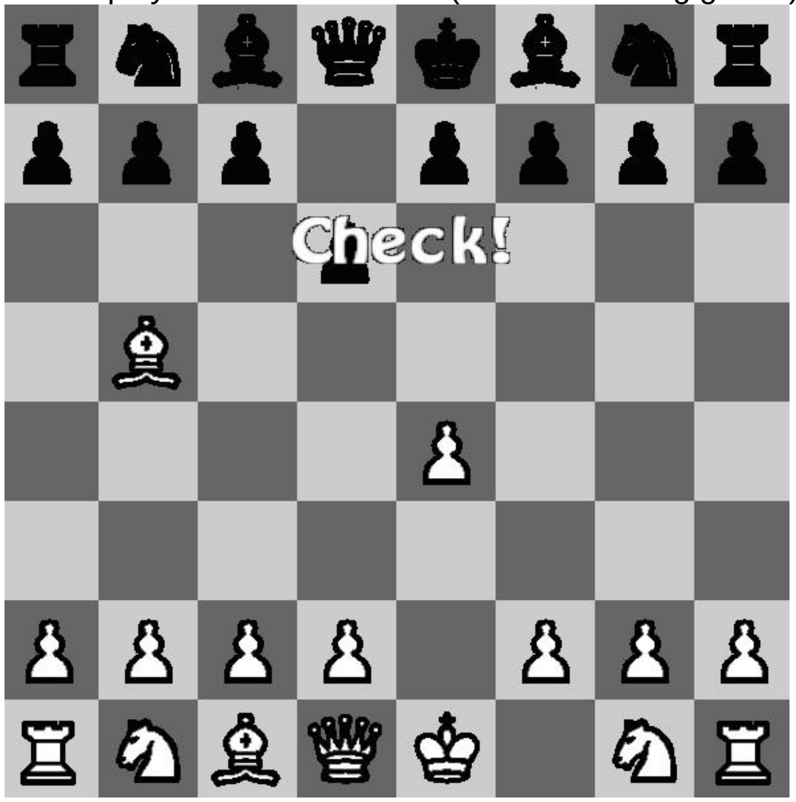
Main Menu (Pre-game interface)



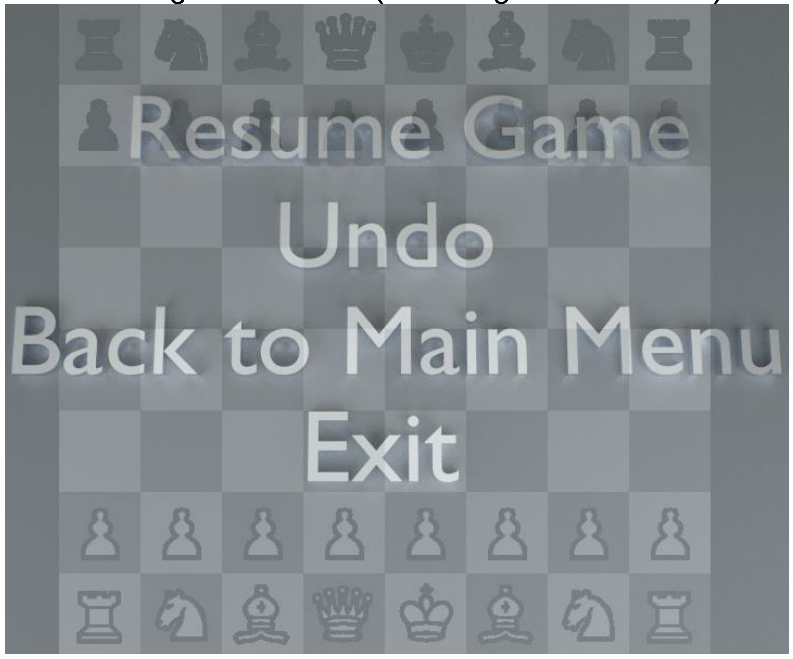
Game In Progress (Interface during game)



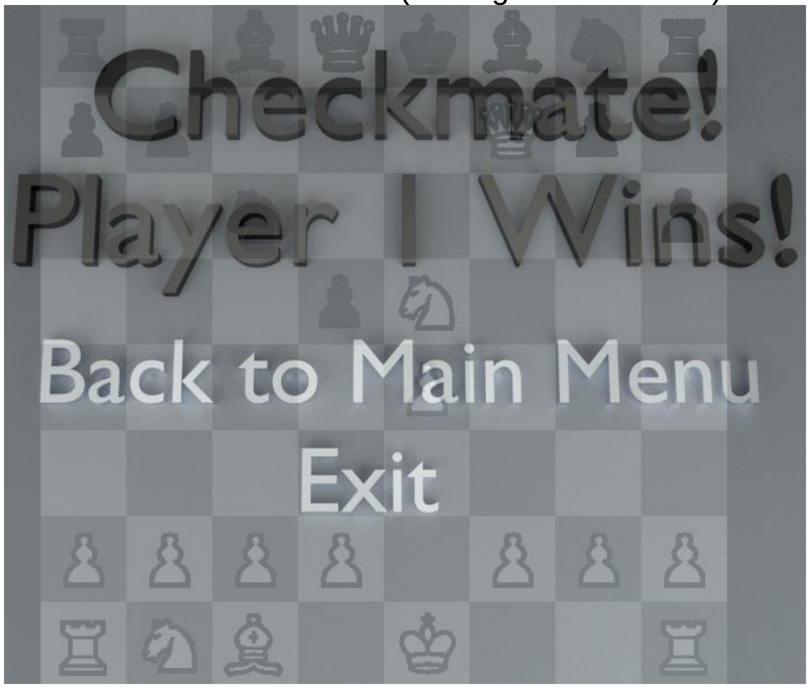
Text displayed in case of check (Interface during game)



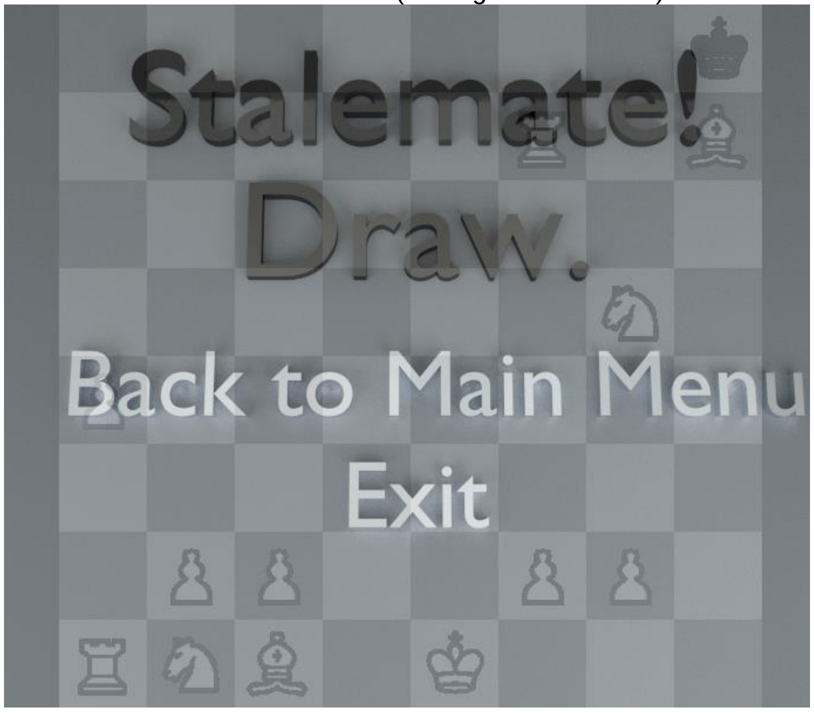
Pause game window (Pause game interface)



In case of checkmate (Post game interface)



In case of stalemate (Post game interface)



6.2 Video:

Product overview video link: https://www.youtube.com/watch?v=2MzxpV0E8iI
Compilation video link: https://www.youtube.com/watch?v=SWTjVSlrBQA

7 Developer guide

7.1 Project layout

- Function of each code file
  The functions performed by the code in each of the code files is given below.
  - main.cpp - Arguments passed to program checked here and the Chess window is loaded.
  - Chess2D.h - Change window settings and load it, load graphics, and process keyboard and mouse input.
  - board.h and Board.cpp - Basic validation of moves and save positions of all chess pieces
  - engine.h and Engine.cpp - Provide interface between the front end, the board and the AI
  - engine_ai.h and Engine_AI.cpp - Predict the best possible move for the computer i.e. implementation of Minimax and the necessary evaluation functions
  - image_loader.h and image_loader.cpp - Loads BMP image into memory
  - gl.h and gl.cpp - Custom GL Library to load shaders, text and menus.
  - shader_source.h - GLSL code required for OpenGL Shader
  - CMakeLists.txt - Instructions for making MAKEFILE
  - compile.bat - Automate the process to build the final executable

- Other folders
  - images - Contains all the necessary images for the graphics
  - dlls - Contains the necessary dlls to run the executable
  - include - Contains the header files of the dependencies
  - lib - Contains the precompiled binaries required for linking

7.2 Building and compiling

- To compile directly
  - Run install.bat
  - In case, MinGW folder is not present in the PATH, the batch file will prompt you to enter the MinGW folder For e.g. "C:\Program files (x86)\MinGW\bin" (Quotation marks required)
  - In case, CMake folder is not present in the PATH, the batch file will prompt you to enter the CMake folder For e.g. "C:\Program files (x86)\CMake\bin" (Quotation marks required)

- To make MAKEFILE and compile separately
  - For making MAKEFILE
    - Make a new folder
    - Run cmake -G "MinGW Makefiles" from this folder (Make sure that MinGW and CMake directories are present in the PATH)
  - For compiling
    - Run mingw32-make from the new folder

8 Quality control

Debugging and testing

8.1 Move validation and other non-AI features

1) Each possible move was individually checked.
2) Multiple games were executed to ensure that

8.2 AI test

1) The various parameters required for the AI were determined through trial and error.
2) Multiple players played against the computer to determine the performance of the Chess AI.
3) The AI functions were executed with very high depths to ascertain that there are no memory leaks and that the program can run properly under very high CPU loads.

9 Future work

1) A 3D variant of the GUI and graphics.
2) Enable network communications between front-ends through a network.
3) Analyze games of Chess Grandmasters and use Machine Learning to determine the values of the parameters used in our Artificial intelligence.