Course Project Documentation

CS101 Project 2015

Chess

Team Id. 181

Hritik Singh, 140010062

Sushrut Aniruddha Phadke, 140010002

Ronit Saha, 140010039

Adhish Tiwari, 140110022

Table of Contents

1. Introduction

Chess is a two-player strategy board game played on a checkered gameboard with 64 squares arranged in an eight-by-eight grid. It is one of the world's most popular games.

In chess, each player begins the game with 16 pieces: one king, one queen, two rooks, two knights, two bishops, and eight pawns. Each of the six piece have different type of moves. In chess, white starts first. The objective of game is to 'checkmate' the opponent's king by placing it under an inescapable threat of capture.

Our software displays an 8X8 grid on the screen with all the 16 pieces of both players. The white pieces are displayed by 'capital letters' and the blacks by 'small letters'.

2. Problem Statement

The aim of this project is to develop a two–player chess game that serves as an entertainment tool which carry out the following functions:

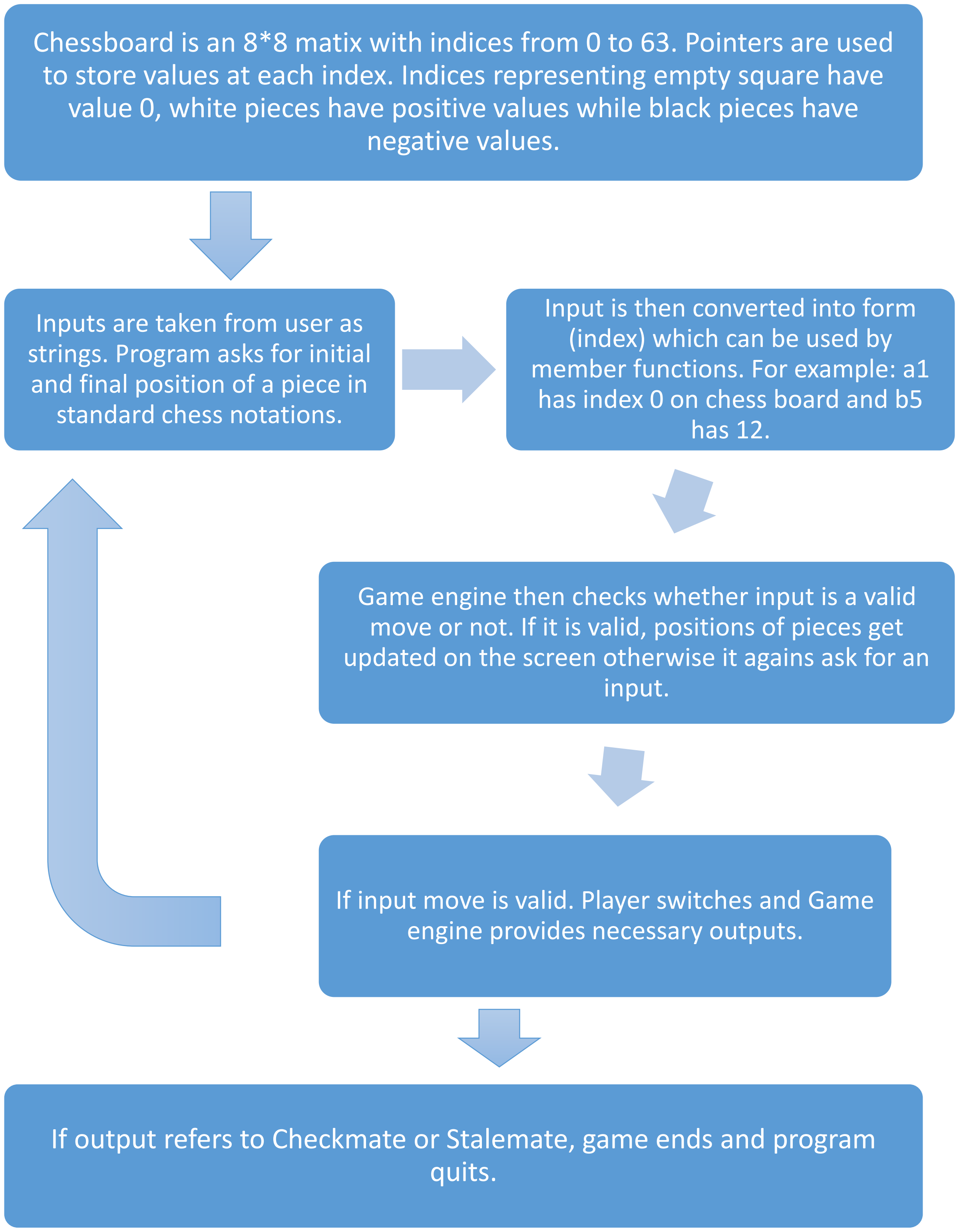
1. Take names of the two players playing the game as input and display them when they have their turn.
2. Check validity of moves of a selected piece.
3. Allow a move only when it is valid otherwise ask for correct input.
4. Keep record of pieces that each player killed and display them on the screen.
5. Display the name of winner of the game at the end of the game or tell the players if the game is a Draw.

3. Requirements

1) Code::Blocks or any other IDE that can compile and run C++ code
2) SDL 2.0.3: To integrate pictures that pops-up during gameplay.

4. Implementation

Game Flow



Chessboard is an 8*8 matix with indices from 0 to 63. Pointers are used to store values at each index. Indices representing empty square have value 0, white pieces have positive values while black pieces have negative values.

Inputs are taken from user as strings. Program asks for initial and final position of a piece in standard chess notations.

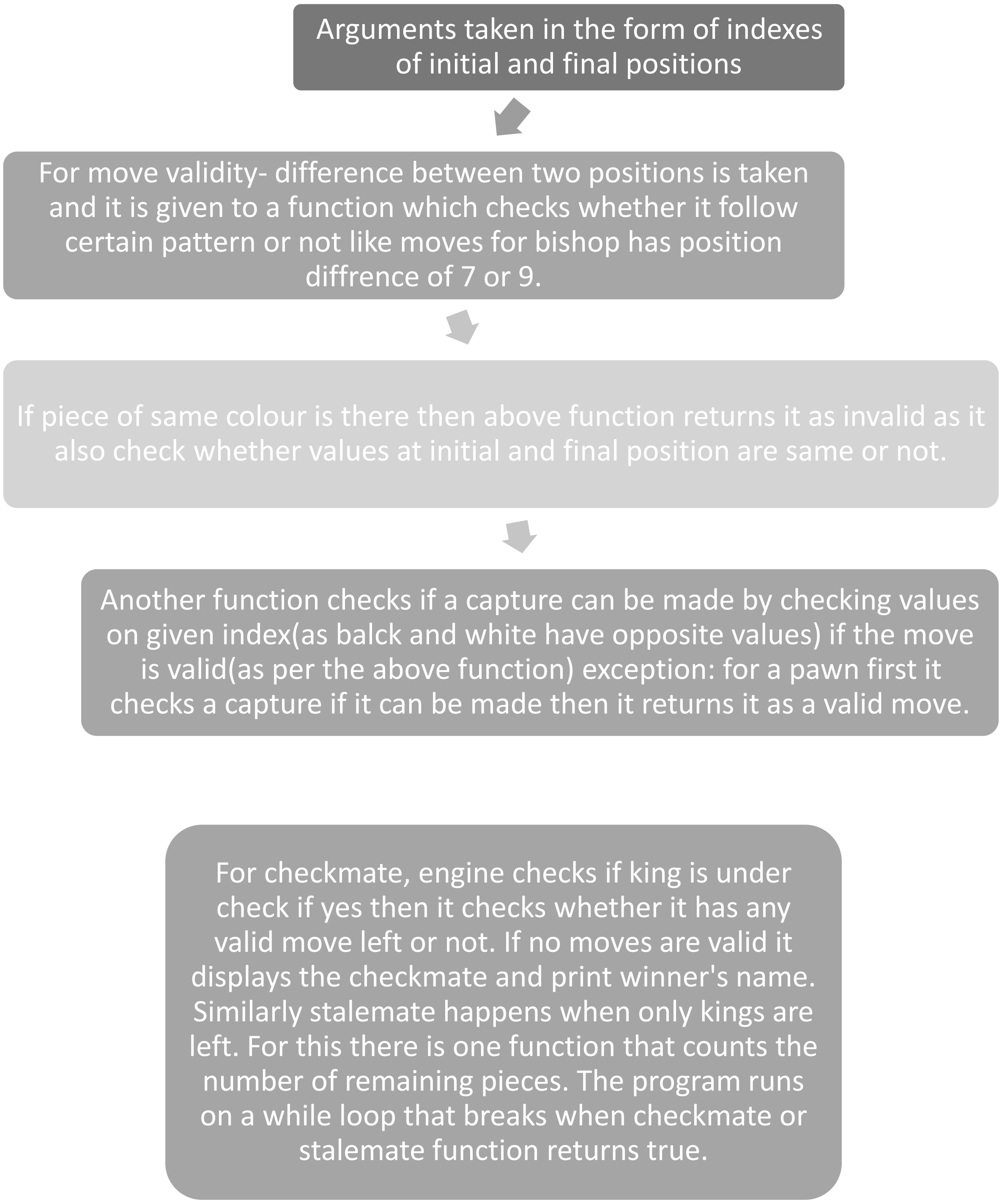
Input is then converted into form (index) which can be used by member functions. For example: a1 has index 0 on chess board and b5 has 12.

Game engine then checks whether input is a valid move or not. If it is valid, positions of pieces get updated on the screen otherwise it agains ask for an input.

If input move is valid. Player switches and Game engine provides necessary outputs.

If output refers to Checkmate or Stalemate, game ends and program quits.

Game Engine and Algorithm



Arguments taken in the form of indexes of initial and final positions

For move validity- difference between two positions is taken and it is given to a function which checks whether it follow certain pattern or not like moves for bishop has position diffrence of 7 or 9.

If piece of same colour is there then above function returns it as invalid as it also check whether values at initial and final position are same or not.

Another function checks if a capture can be made by checking values on given index(as balck and white have opposite values) if the move is valid(as per the above function) exception: for a pawn first it checks a capture if it can be made then it returns it as a valid move.

For checkmate, engine checks if king is under check if yes then it checks whether it has any valid move left or not. If no moves are valid it displays the checkmate and print winner's name. Similarly stalemate happens when only kings are left. For this there is one function that counts the number of remaining pieces. The program runs on a while loop that breaks when checkmate or stalemate function returns true.

For notifications like piece capture, check, stalemate or checkmate specific graphic windows pops-up displaying the respective results.

5. Testing and Data

A lot of testing has been done by us using different conditions and playing game with different strategies. Screenshots below show a basic moves of pieces, pawn promotion, piece capture and checkmate:

Moves Involved:

White: pawn-e2 to e4 Black: pawn-a7 to a5

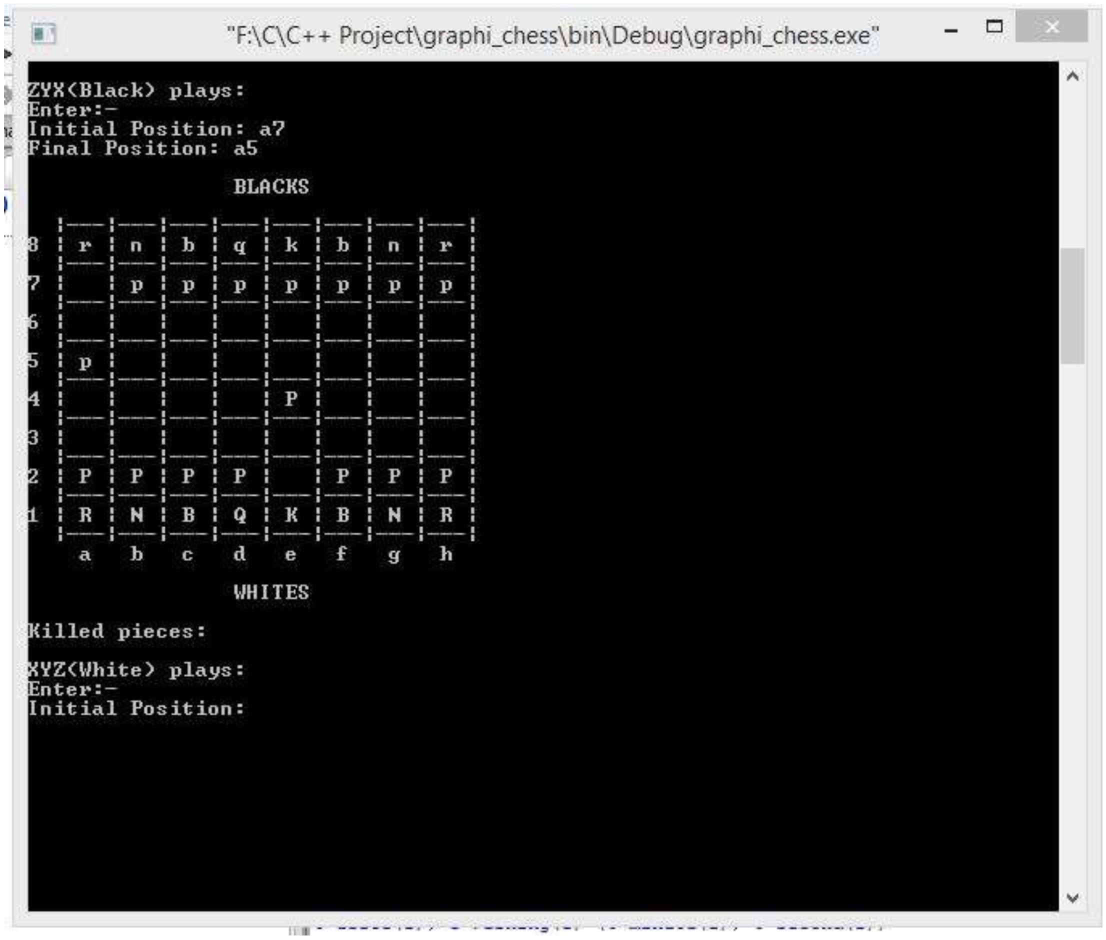
White: knight-g1 to f3 Black: pawn-a5 to a4

White: knight-f3 to g5 Black: pawn-a4 to a3

White: bishop-f1 to c4 Black: pawn x pawn- a3 to b2

White: queen-d1 to f3 Black: pawn x rook-b2 to a1, pawn promote to queen

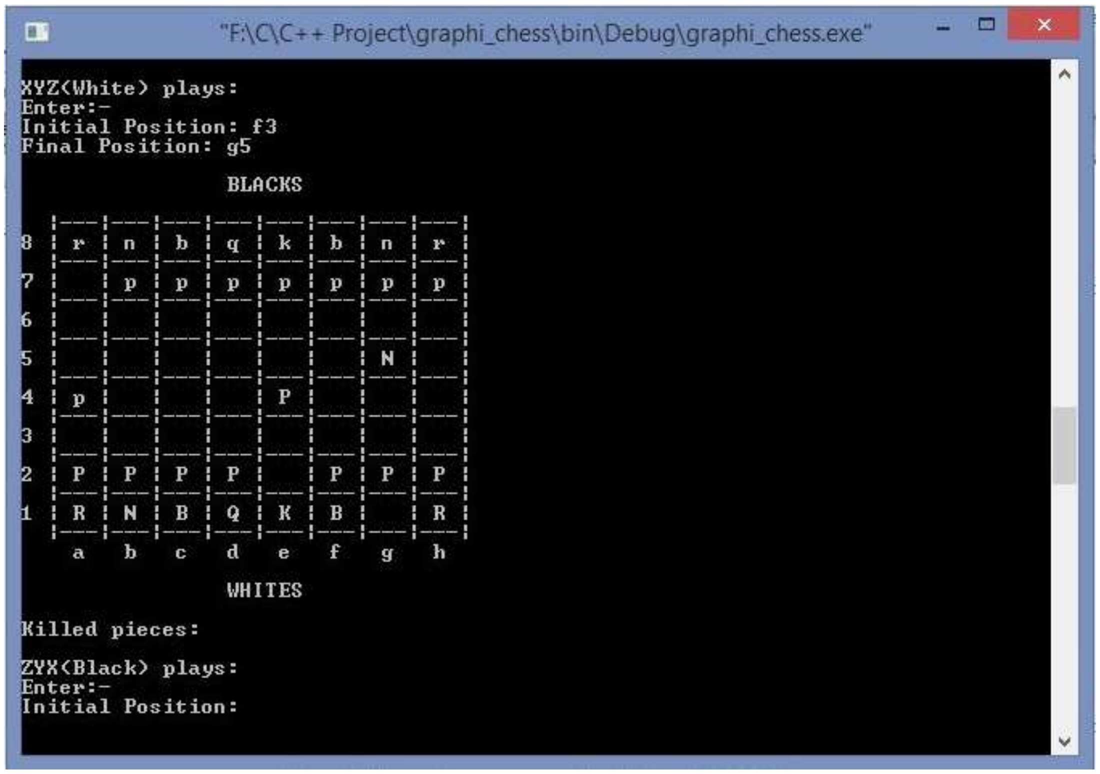
White: queen x pawn-f3 to f7, checkmate.



"F:\C\C++ Project\graphi_chess\bin\Debug\graphi_chess.exe"

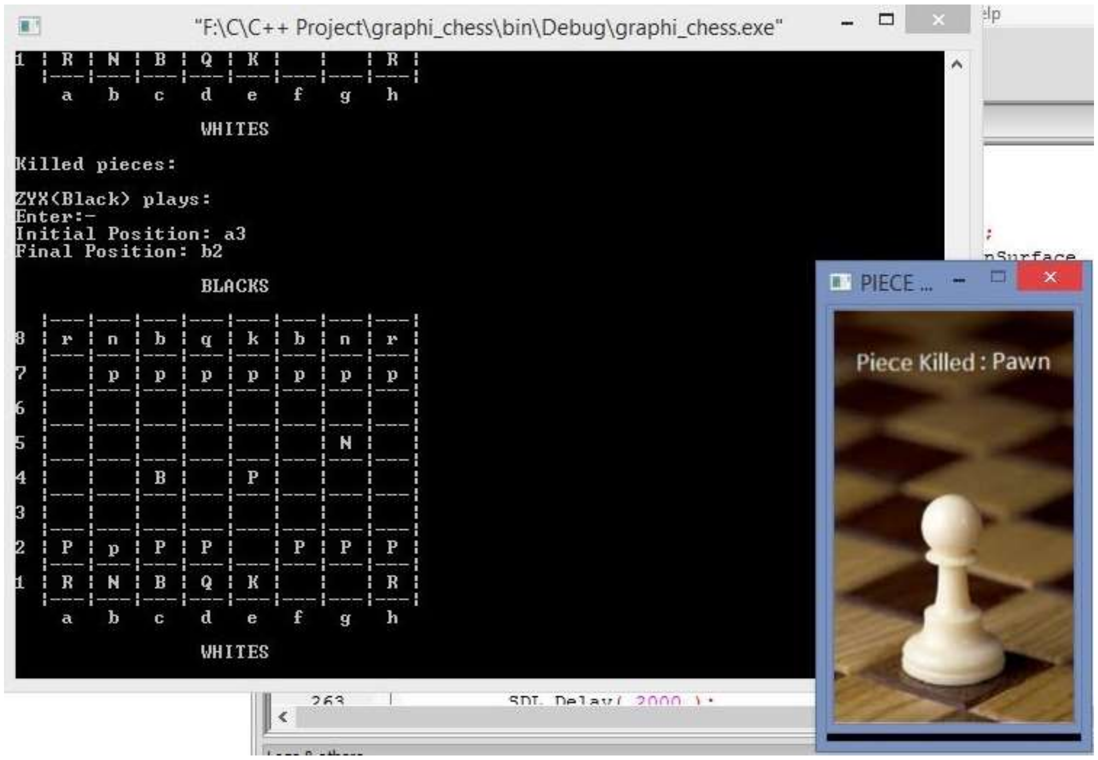
```
ZYX(Black) plays:
Enter:-
Initial Position: a7
Final Position: a5

                    BLACKS

  |----|----|----|----|----|----|----|----|
8 | r  | n  | b  | q  | k  | b  | n  | r  |
  |----|----|----|----|----|----|----|----|
7 |    | p  | p  | p  | p  | p  | p  | p  |
  |----|----|----|----|----|----|----|----|
6 |    |    |    |    |    |    |    |    |
  |----|----|----|----|----|----|----|----|
5 | p  |    |    |    |    |    |    |    |
  |----|----|----|----|----|----|----|----|
4 |    |    |    |    | P  |    |    |    |
  |----|----|----|----|----|----|----|----|
3 |    |    |    |    |    |    |    |    |
  |----|----|----|----|----|----|----|----|
2 | P  | P  | P  | P  |    | P  | P  | P  |
  |----|----|----|----|----|----|----|----|
1 | R  | N  | B  | Q  | K  | B  | N  | R  |
  |----|----|----|----|----|----|----|----|
    a    b    c    d    e    f    g    h

                    WHITES

Killed pieces:

XYZ(White) plays:
Enter:-
Initial Position:
```

Screenshot after first 2 moves.



```
XYZ(White) plays:
Enter:-
Initial Position: f3
Final Position: g5

                    BLACKS

  |---|---|---|---|---|---|---|---|
8 | r | n | b | q | k | b | n | r |
  |---|---|---|---|---|---|---|---|
7 |   | p | p | p | p | p | p | p |
  |---|---|---|---|---|---|---|---|
6 |   |   |   |   |   |   |   |   |
  |---|---|---|---|---|---|---|---|
5 |   |   |   |   |   |   | N |   |
  |---|---|---|---|---|---|---|---|
4 | p |   |   |   | P |   |   |   |
  |---|---|---|---|---|---|---|---|
3 |   |   |   |   |   |   |   |   |
  |---|---|---|---|---|---|---|---|
2 | P | P | P | P |   | P | P | P |
  |---|---|---|---|---|---|---|---|
1 | R | N | B | Q | K | B |   | R |
  |---|---|---|---|---|---|---|---|
    a   b   c   d   e   f   g   h

                    WHITES

Killed pieces:

ZYX(Black) plays:
Enter:-
Initial Position:
```

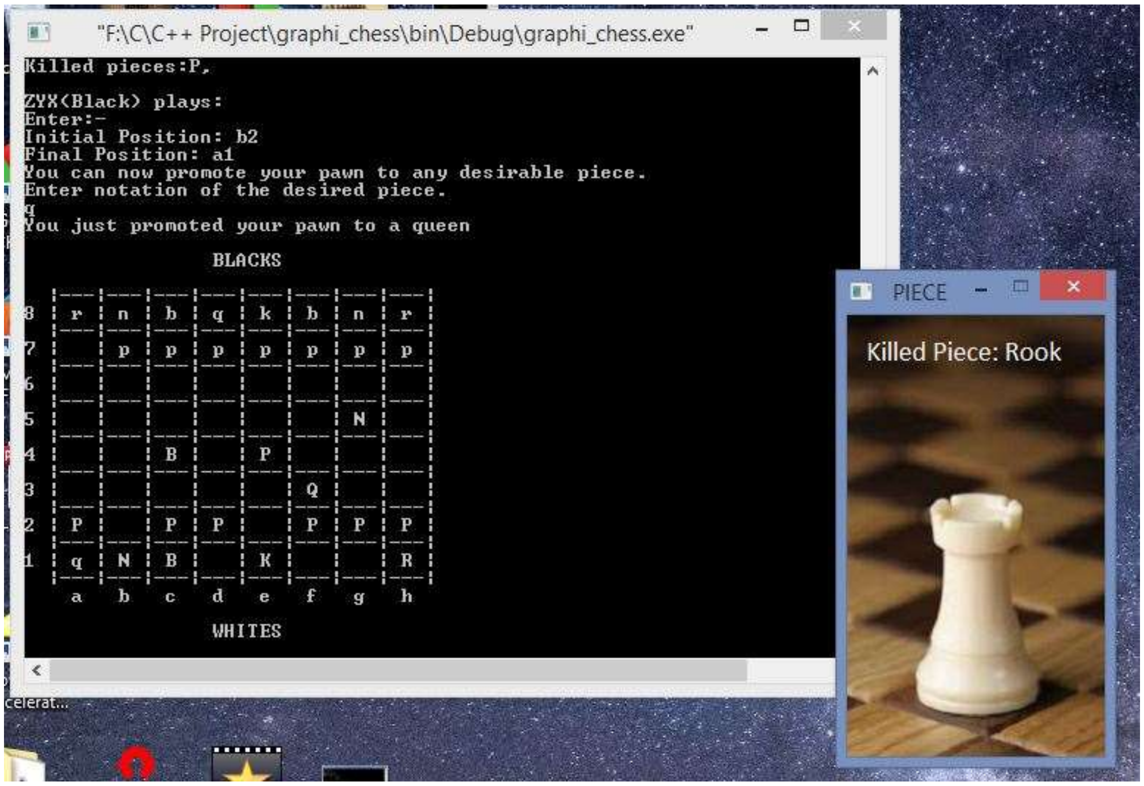
Screenshot when white moves knight from f3 to g5.



```
1 | R | N | B | Q | K |   |   | R |
  |---|---|---|---|---|---|---|---|
    a   b   c   d   e   f   g   h

                    WHITES

Killed pieces:

ZYX(Black) plays:
Enter:-
Initial Position: a3
Final Position: b2

                    BLACKS

  |---|---|---|---|---|---|---|---|
8 | r | n | b | q | k | b | n | r |
  |---|---|---|---|---|---|---|---|
7 |   | p | p | p | p | p | p | p |
  |---|---|---|---|---|---|---|---|
6 |   |   |   |   |   |   |   |   |
  |---|---|---|---|---|---|---|---|
5 |   |   |   |   |   |   | N |   |
  |---|---|---|---|---|---|---|---|
4 |   |   | B |   | P |   |   |   |
  |---|---|---|---|---|---|---|---|
3 |   |   |   |   |   |   |   |   |
  |---|---|---|---|---|---|---|---|
2 | P | p | P | P |   | P | P | P |
  |---|---|---|---|---|---|---|---|
1 | R | N | B | Q | K |   |   | R |
  |---|---|---|---|---|---|---|---|
    a   b   c   d   e   f   g   h

                    WHITES
```

```
Piece Killed : Pawn
```

Screenshot of black pawn capturing white pawn on b2.



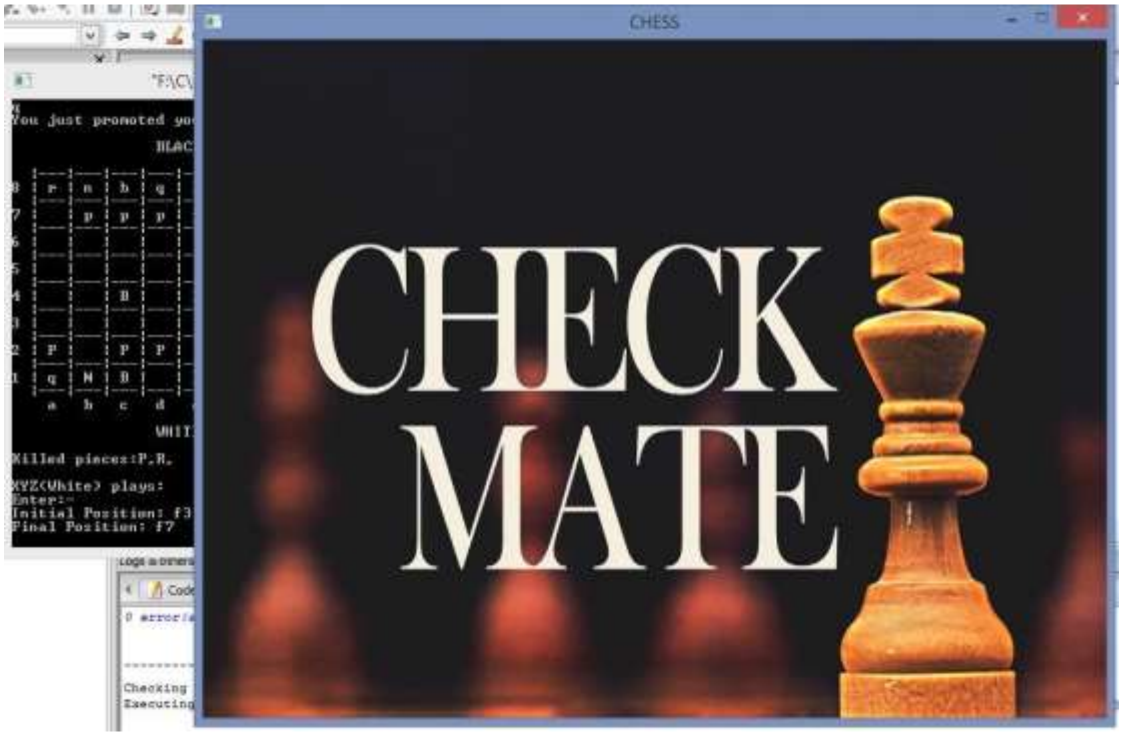
"F:\C\C++ Project\graphi_chess\bin\Debug\graphi_chess.exe"

```
Killed pieces:P,

ZYX(Black) plays:
Enter:-
Initial Position: b2
Final Position: a1
You can now promote your pawn to any desirable piece.
Enter notation of the desired piece.
q
You just promoted your pawn to a queen

                    BLACKS

   |---|---|---|---|---|---|---|---|
8  | r | n | b | q | k | b | n | r |
   |---|---|---|---|---|---|---|---|
7  |   | p | p | p | p | p | p | p |
   |---|---|---|---|---|---|---|---|
6  |   |   |   |   |   |   |   |   |
   |---|---|---|---|---|---|---|---|
5  |   |   |   |   |   |   | N |   |
   |---|---|---|---|---|---|---|---|
4  |   |   | B |   | P |   |   |   |
   |---|---|---|---|---|---|---|---|
3  |   |   |   |   |   | Q |   |   |
   |---|---|---|---|---|---|---|---|
2  | P |   | P | P |   | P | P | P |
   |---|---|---|---|---|---|---|---|
1  | q | N | B |   | K |   |   | R |
   |---|---|---|---|---|---|---|---|
     a   b   c   d   e   f   g   h

                    WHITES
```

PIECE

Killed Piece: Rook

Screenshot showing black pawn promotion to queen and rook being captured on a1.



```
q
You just promoted yo

                    BLAC

Killed pieces:P,R,

XYZ(White) plays:
Enter:-
Initial Position: f3
Final Position: f7
```

Screenshot of Checkmate when white queen moves from f3 to f7.

6. Challenges

The main challenges that we faced during the project are:

1. **The condition of checkmate:** The checkmate requires a lot of testing and checking of moves. As checkmate is one of the most difficult moves in chess it initially created some problems during implementation.

Solution: We made a function that checks all possible moves of king if it is under check. It also check if king kills the piece which is giving check to king then whether in turn it is being killed by other piece or not. If king has no move left after checking all such conditions, game engine decides it as checkmate and displays the winner.

2. **Memory Leaks and bad excess:** As most of the code is written using pointers, many errors related to pointers such as bad access, memory leaks, etc. occurred that caused the game to freeze and it displayed even valid moves as invalid.

Solution: Such errors were solved by regular googling of the errors. As many of such errors were faced previously, satisfactory solutions were found on websites like stackoverflow and other forums.

3. **Implementing the graphics:** Implementation of graphics was another challenge as none of us had previous experience with graphics and we didn’t use simplecpp due to its limited resources.

Solution: As the work on GUI was started pretty late (not a part of the main problem statement), sufficient justice couldn't be given to it. But we still implemented some basic graphics in form of pop-up windows that display notifications like image of captured piece, check, stalemate or checkmate.

7. Discussion of System

A) What worked as Planned?

1. Move validity check.

2. Record and Display of killed pieces.

3. Castling: a special move involving king and rook.

4. Declaration of winner after checkmate.

B) Changes made in plan

1. We have used SDL instead of OpenGL.

8. Bugs

There is a minor bug that we found in our program which is that King cannot move to e1 from e2 though it can move back from e3 to e2 and all other valid squares and also can move from f1 or d1 to e1.

9. Future Work

A) Improvements can be done in the code itself. A better, shorter and more efficient code can be written.

B) A very good GUI can be implemented. Mouse click can be used effectively in place of keyboard inputs.

C) A single player AI can be coded using minimax algorithm. We gave a try to it too, however we fell short of time in understanding the related theorem and then implementing it. The process was left half way by us, however sufficient justice can be given to it in the future.

10. References

1. Code::Blocks Installation Guide: http://www.codeblocks.org/downloads/binaries

2. SDL2.0.3 : https://www.libsdl.org/download-2.0.php

3. Rules of Chess: http://en.wikipedia.org/wiki/Rules_of_chess