Thomas STOKES

Chess board game project

Centre number: 66513 Candidate number: 2448

Contents

[Analysis 2](#_Toc40865074)

[Introduction 2](#_Toc40865075)

[Research 2](#_Toc40865076)

[Rules 2](#_Toc40865077)

[Bonus Rules 5](#_Toc40865078)

[Current software and ways of playing chess 6](#_Toc40865079)

[Questionnaire 8](#_Toc40865080)

[User requirements 9](#_Toc40865081)

[Objectives 9](#_Toc40865082)

[Acceptable limitations 10](#_Toc40865083)

[Data Sources and destination 10](#_Toc40865084)

[Object Analysis Diagrams 10](#_Toc40865085)

[Chosen Solution 10](#_Toc40865086)

[Pygame 11](#_Toc40865087)

[Design 11](#_Toc40865088)

[Process of design 11](#_Toc40865089)

[The user interface 11](#_Toc40865090)

[Modular design 12](#_Toc40865091)

[Pseudo code 12](#_Toc40865092)

[Algorithm 18](#_Toc40865093)

[Technology Solution 18](#_Toc40865094)

[Testing 18](#_Toc40865095)

[Evaluation 18](#_Toc40865096)

# Analysis

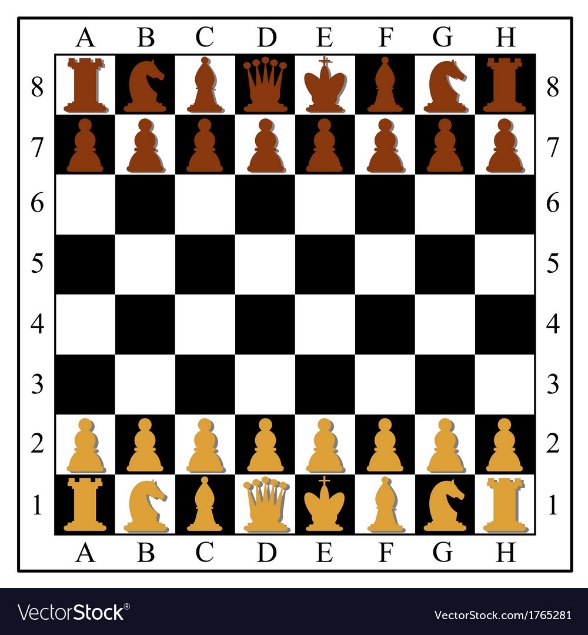
## Introduction

I am going to produce a chess computer game utilising the Pygame module. I have enjoyed playing chess from a young age, therefore I think this project would be perfect for me. The game will allow two players to play against one another on the same screen: each player will be able to select pieces, when it is there turn, and move them to the allowed squares my program will calculate. I aim for my program to detect check, checkmate and some of the stalemate conditions, possibly along with bonus rules like ‘En passant’. A lot of chess apps can be quite confusing and I hope to create a simple app.

## Research

### Rules

Chess is a two player game played on a chequered 8\*8 grid. Each player has different pieces which they can utilise, which can be seen below:



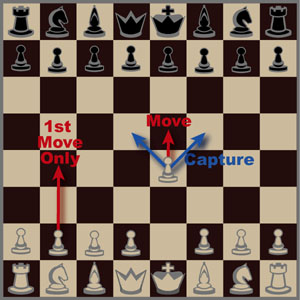
[1]-Chessboard

The aim of the game is to checkmate your opponent’s king (I will explain how to do this below). You can move one of your pieces per turn. To take a piece you move your piece to the square and remove the opponent’s piece from the board leaving your piece there, but it is not obligatory to take the opponents piece. You cannot move a piece to a square already occupied by one of your own pieces. However, to understand how to do this you must first understand how each piece moves.

#### Pawn

The front row are called pawns and if they are being moved for the first time they can move two spaces forward (or one), after this they can only move one forward, providing there is not another piece in the square. They can take an opponent’s piece if it is in any of the diagonal squares in front of it.

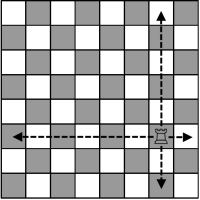
[1]-<https://www.vectorstock.com/royalty-free-vectors/chess-pieces-floor-vectors>- Simple picture of a chessboard.



[2]-Pawn

#### Rook

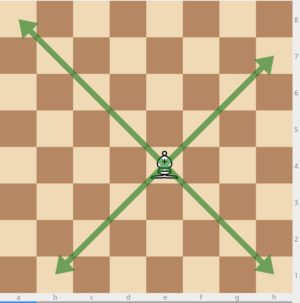
These can move in a straight line, horizontally or vertically, any number of squares(but cannot move past a piece in that row or column unless it is the piece they are trying to take). They can take the first piece in that path.



[3]-Castle

#### Bishop

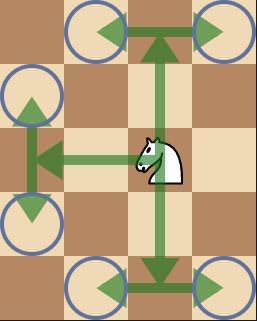
This piece can move diagonally forwards or backwards from their position any number of squares(but cannot move past a piece in that diagonal, unless that piece is the one they are trying to take). They can take the first piece in that path.



[4]-Bishop

#### Knight

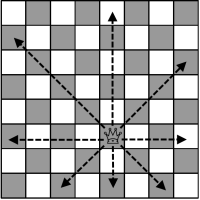
These pieces are the only piece that can jump over other pieces. They move in an L-shape moving two in a direction and one in either direction perpendicular to that. They can take pieces in any of these squares.



[5]-Knight

#### Queen

The queen is a very powerful piece and can move any direction: diagonally, horizontally or vertically any number of square (up to the first piece in that path). They can take the first piece in that path.



[6]-Queen

#### King

This is the piece you are trying to protect, and it can move in any direction but only one square in that direction. The king cannot move through check (see below) or into check.

#### Ending the game

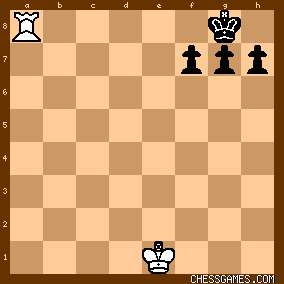
The aim of the game is to use the pieces to eliminate other pieces and checkmate the king by trapping it in a position where it cannot avoid being taken, also referred to as checkmate. Check is where a king is threatened by an opposing player’s piece. They must move the king out of this position or block it or take the piece threatening the king. If they can’t then it is checkmate, like below:

[2]- <https://chessforrookies.wordpress.com/2013/12/11/all-you-need-are-pawns/>

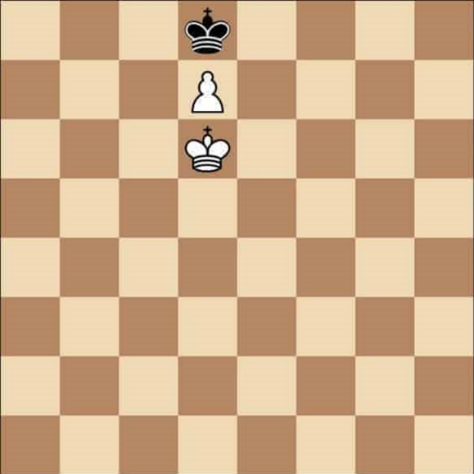
[3]- <https://www.google.com/url?sa=i&url=https%3A%2F%2Ftch775503.tch.www.quora.com%2FWhat-are-some-tips-and-techniques-to-teach-kids-chess&psig=AOvVaw3i5EAaiOdCCoRjCmJvqitc&ust=1582549949278000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCOCB34_g5-cCFQAAAAAdAAAAABAx>

[4]- <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.ichess.net%2Fblog%2Frules-of-chess%2F&psig=AOvVaw07CYJHd-l2rHaD9f8Kwqwf&ust=1582550369934000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCNisrdnh5-cCFQAAAAAdAAAAABAD>

[5]- <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.ichess.net%2Fblog%2Fchess-pieces-moves%2F&psig=AOvVaw2yyEXzW5tJRJ2T1IWMbQ26&ust=1582550425313000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCMCE7_Lh5-cCFQAAAAAdAAAAABAD>

  
[7]-In this case the white has won.

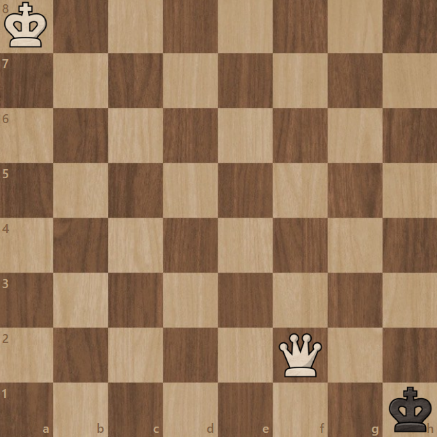
One other ending is a stalemate where the opposing king, usually being one of the last pieces on the opposing team, can’t move but is not in check. The way this works is that a king cannot move into check and if all the squares around them will result in check, then they can’t move. At this point neither team wins. Other ways of obtaining a draw are if either: the board is the exact same three times; no pieces are taken or no pawns have moved for 50 moves; or if there are insufficient pieces to reach a checkmate. Examples can be seen below:



[8]-Stalemate

[6]- <https://www.google.com/url?sa=i&url=https%3A%2F%2Fchessstarter.neocities.org%2F&psig=AOvVaw3Vm-WJ5r3eeJ2OrB8FD5mR&ust=1582550546690000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCIjPyqvi5-cCFQAAAAAdAAAAABAD>

[7]- <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.chessgames.com%2Fperl%2Fchesscollection%3Fcid%3D1037232&psig=AOvVaw2j4vXkyr6TsClnAbrLGwlD&ust=1582550586780000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCPCHzMHi5-cCFQAAAAAdAAAAABAD>



[9]-Stalemate

### Bonus Rules

#### Pawn upgrade

When a pawn reaches the opponents back row they can upgrade the pawn to any other piece.

#### En passant

If a pawn moves forward two on its opening turn and ends up next to an enemy pawn the enemy pawn can move to the position it would’ve been in had it moved one, taking it and removing the piece from the board.



[10]-En passant

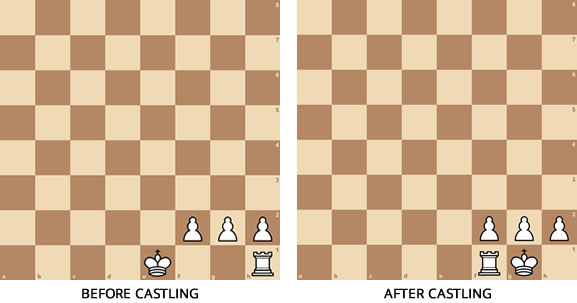
#### Castling

This is where on the back row a king can swap with castle, providing there are no pieces between them. This can only happen if the king and castle involved haven’t moved. It is the only time 2 pieces can move in the same turn. The king moves to squares towards the castle and the castle involved moves to the opposite side like below.

[8]- <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.ichess.net%2Fblog%2Fstalemate-in-chess%2F&psig=AOvVaw1OYZiN8a0wZvcviPCsUs8Q&ust=1582550911240000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCODF89nj5-cCFQAAAAAdAAAAABAD>

[9]- <https://www.google.com/url?sa=i&url=https%3A%2F%2Fsupport.chess.com%2Farticle%2F682-what-is-stalemate&psig=AOvVaw04DUp54sM1k84P__3rs38b&ust=1582550953684000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCKCrqO7j5-cCFQAAAAAdAAAAABAD>

[10]- <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.ichess.net%2Fblog%2Fchess-en-passant%2F&psig=AOvVaw1YlXYw__ceXaLijgofHchI&ust=1582551021730000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCLjvgo7k5-cCFQAAAAAdAAAAABAD>

 [11]-Castling

#### Timer

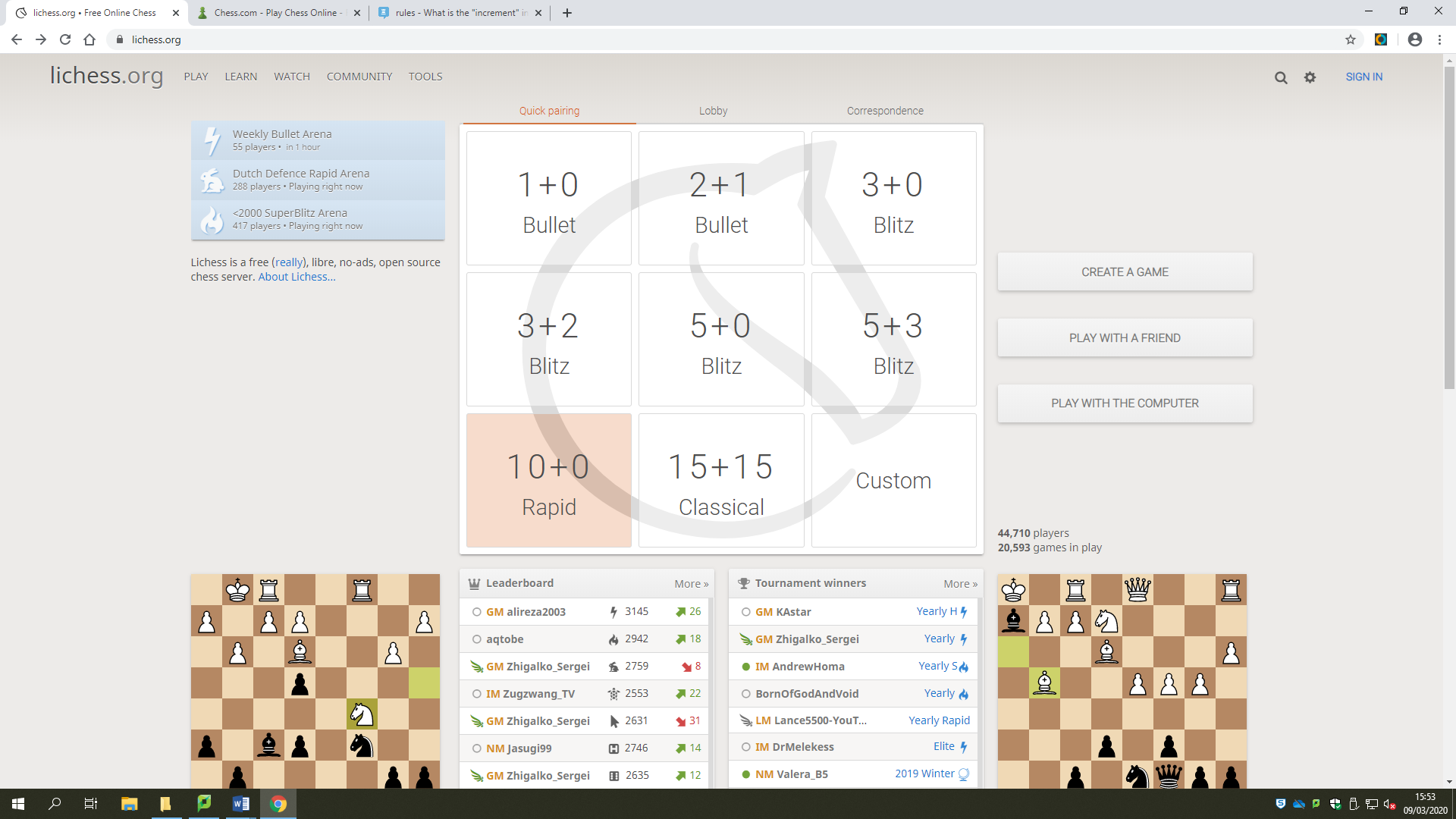
In chess each player has a countdown timer that lasts the entire game. For example, a common countdown is 15minutes with a 15 second increment. Each time a player starts their move the timer is swapped to the other player and continues counting down. The increment is to allow players to get ready and is added to the timer. If they do their move in this time they gain extra time. I will have a set increment to allow for players to swap seat or get a good view of the screen.

### Current software and ways of playing chess

There are various websites and apps that teach and allow the user to play chess such as: ‘Lichess.org’ and ‘chess.com’, but of course it was originally played on a physical board, with physical pieces. Using an app has its pros and cons: it is good because the computer can spot a checkmate, stalemate or draw that might have been missed in normal gameplay, however the device the game is running on, might run out of charge part way through. Furthermore, chess apps are useful for those wanting to play chess on the move and those wanting to play chess when they don’t have an opponent in real life. In addition, they can play against another computer or learn how to play using built in features. However, in my app I will only be implementing a two-player game, that is simple and understandable for the user. There will be one simple menu, a pause menu and a game over menu.

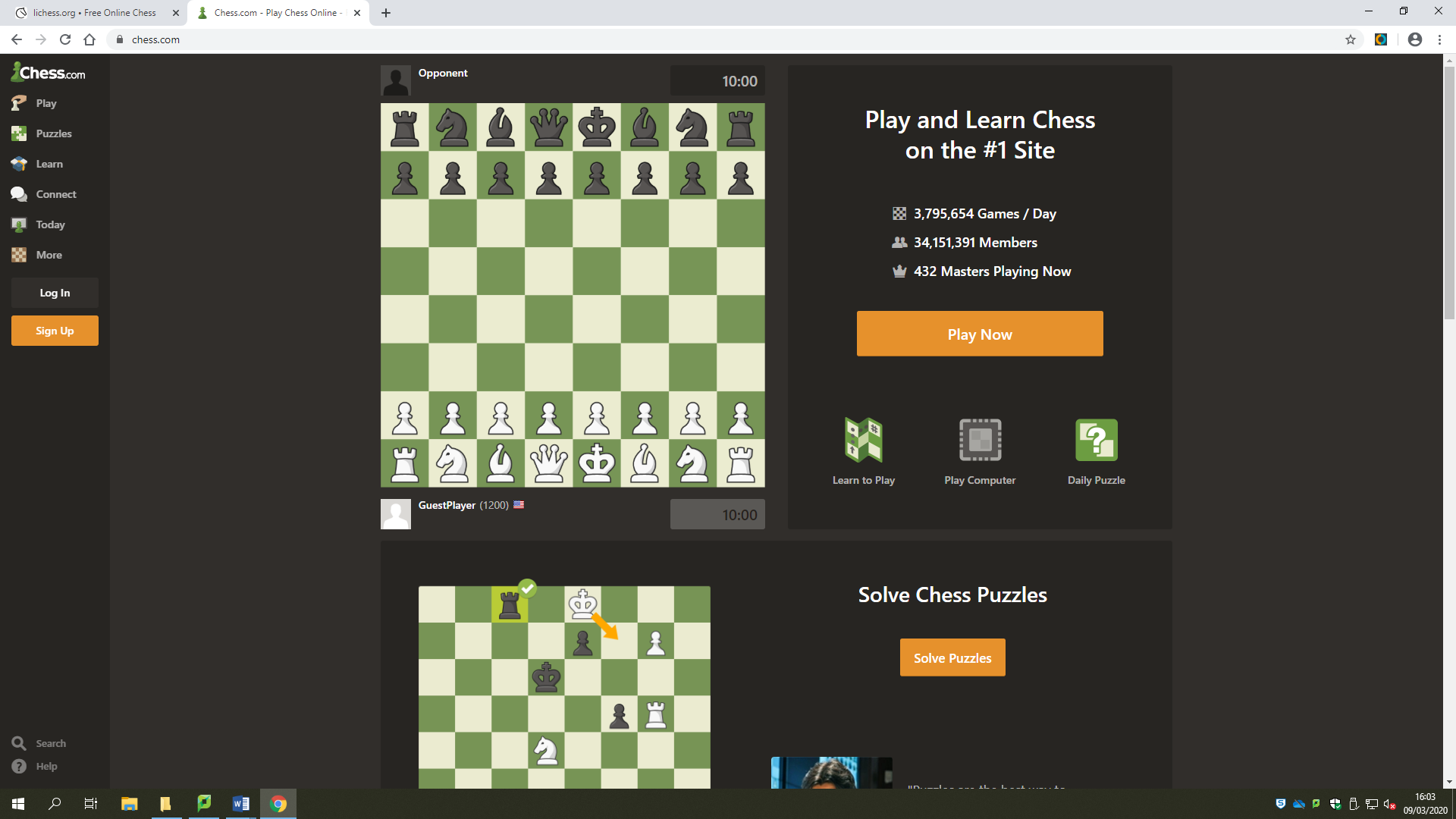
My project will allow two players of any age who understand the rules to engage in a game of chess. The app is designed for mobile chess, so you don’t need to take a chess board with you when travelling for example.

[11]- <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.ichess.net%2Fblog%2Fcastling-chess-complete-guide%2F&psig=AOvVaw1drzlVKTww5L277BC0EmdO&ust=1583405435429000&source=images&cd=vfe&ved=0CAkQjhxqFwoTCPiIu4bTgOgCFQAAAAAdAAAAABAD>



Lichess

This layout is smart, but a bit clustered. The first time I visited I was confused as to what the central panel meant, but I now know it is about the timer settings. However there are things, like the central panel and puzzles that draw your attention to these more important things and once you break it down, it is quite simple. Overall though, it is free and so all features are available and looks professional, albeit initially confusing.

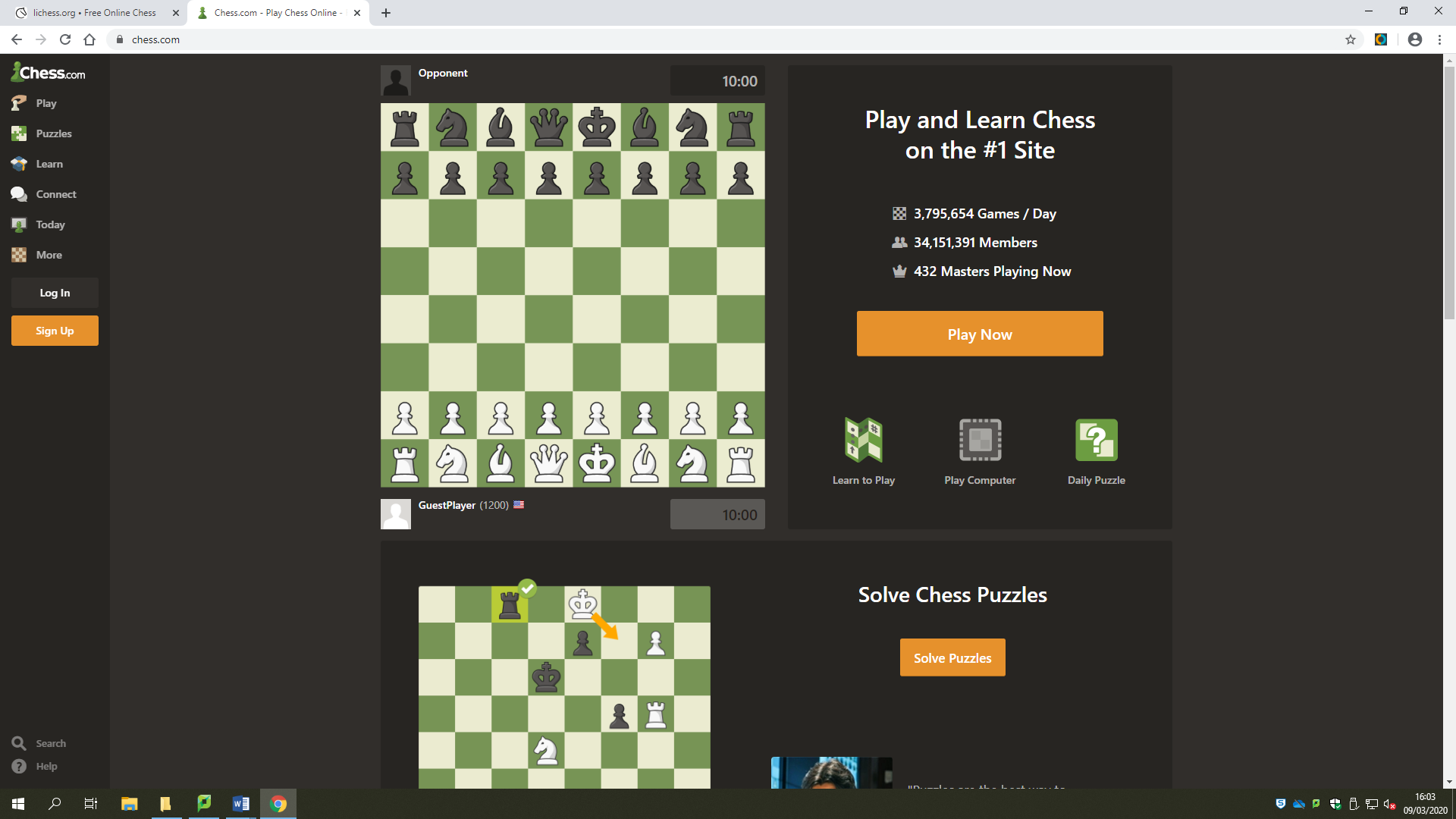
Chess.com

This is much simpler and has the main menu down the left, with no repeated menus. Both this and Lichess have submenus when hovered over. This website is kept simple and doesn’t have large swathes of text. Important buttons are highlighted orange for beginners, however chess.com does has made the menu quite small and not located the sign in button in the traditional top right corner.

### Questionnaire

I also need other people’s opinions on the websites as they might have differing views.

*Chess.com:*



***What are your first thoughts?***

1. The style of chessboard is relaxed with a nice design of chess pieces.
2. Nice and refined. Clean GUI.
3. Colourful. The cartoon style.

**How often to do you play chess?**

1. Every other month
2. From time to time
3. 3 times a year

***Does the website look nice?***

1. Yes.
2. Yes
3. Yes

***What is most important to you (what do you like to see) when you open an app or website?***

1. Aesthetically pleasing and simple.
2. Clear instructions and navigation.
3. It is colourful and there are lots of options

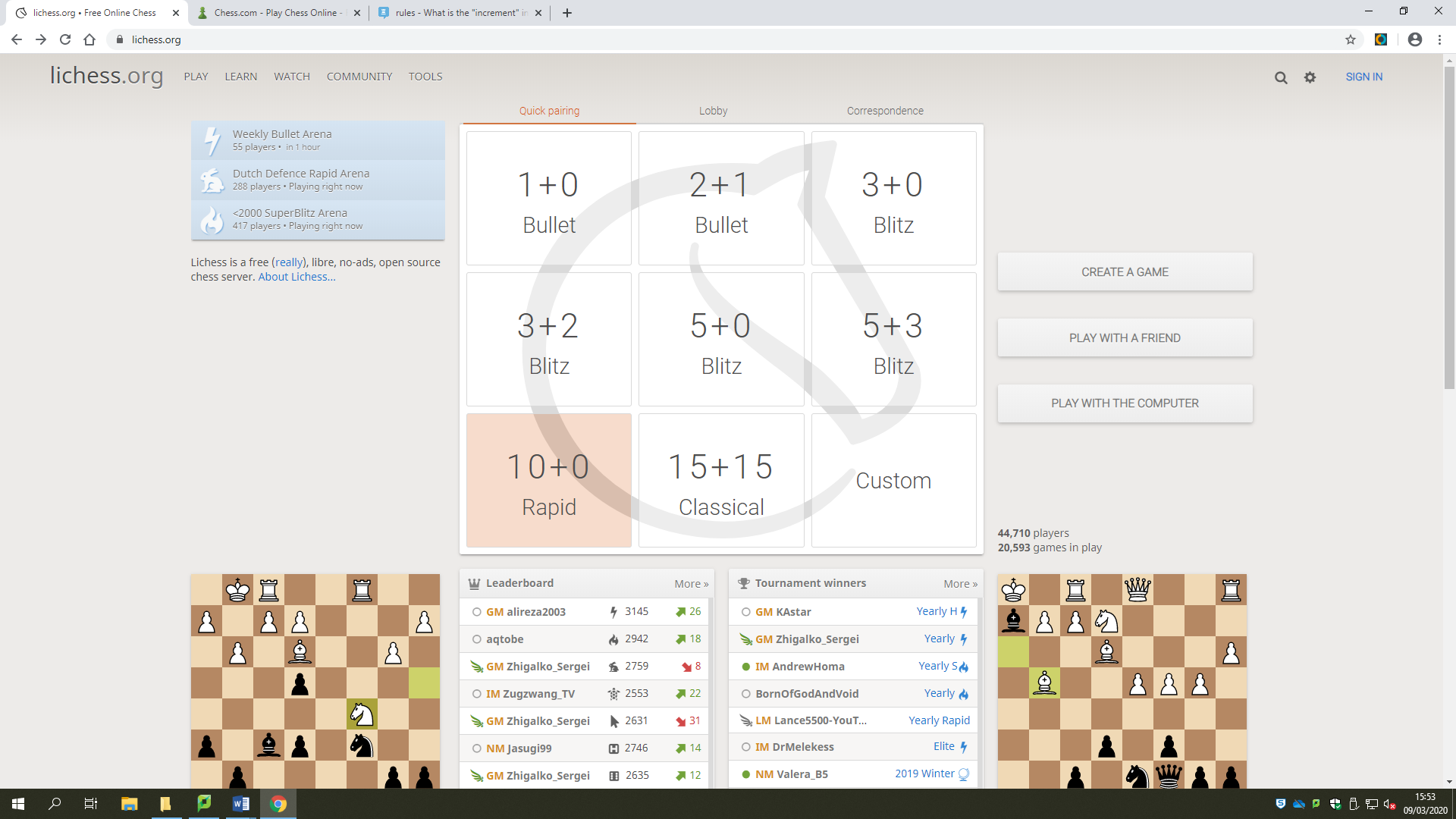
***What features stand out?***

1. The dots to show available moves and the highlighted square which show the last move.
2. Bright bold colours, like play now and chess board.
3. The chessboard.

***What features would you like to see or change?***

1. The menu being more visible and open, so I can see where to go.
2. A greater focus on the chessboard. Pieces move more slowly between squares.
3. Easy to see what’s going on.

*Lichess.org:*



***What are your first thoughts?***

1. Ugly colours and I don’t understand what is on the screen.
2. Cluttered and confused. What is going on?
3. Lots of options and submenus.

***Does the website look nice?***

1. No.
2. Dull colour scheme.
3. Yes

***What features stand out?***

1. 2 chess boards as an example of what’s going on. The central ‘thing’ stands out, but doesn’t make sense.
2. The larger buttons. The leader boards have unique status icons that grab your attention.
3. The central panel

***What features would you like to see or change?***

1. Better colours and an update of graphics. Also, while I don’t mind lots of text, I prefer to have a simpler home page that allows me to first get acquainted with site.
2. A more user friendly home menu for new players, with instructions to compliment them.
3. N/A

##### Analysis of Questionnaire

I will need to design a board that uses the right colour scheme. Ideally the green and yellowy-white from ‘chess.com’. Furthermore, I need to make a simple main menu that displays only things a first time player needs. Any other settings and options may be accessible by a little cog symbol. It may be worth having a ‘readme’ text document or a built in instructions. I want to make it as understandable as possible. I plan on using simple visual cues and colour to draw the attention to say the ‘play button’. Chess.com did this using a key colour of orange repeatedly to show it symbolised important links. However, the homepage pushed the menu to the left and to access more options the user had to hover over buttons. I want the user to have everything they need(be that a link to something that is more complicated) on the screen.

### User requirements

I hope to produce a game allowing the user to:

1. Play chess in a user-friendly GUI (Graphical User Interface)
2. Play against an opponent locally.
3. Play with the correct rules.
4. Give the user a menu and end message.

### Objectives

* A game of chess that two players can interact with
* A chess game which follows the rules of the game.
* There will be appropriate error handling so the user can only move the pieces to the allotted squares.
* Checkmate will be noticed
* The program will calculate allowed moves when a piece is clicked on.

#### Extended objectives

* Implement all additional rules.
* Save game feature
* Ability to change colour scheme at beginning of game.
* A game timer will be implemented ending the game if a player runs out of time.
* Add other versions of chess like killer chess, where the king can be taken (opposed to checkmate).

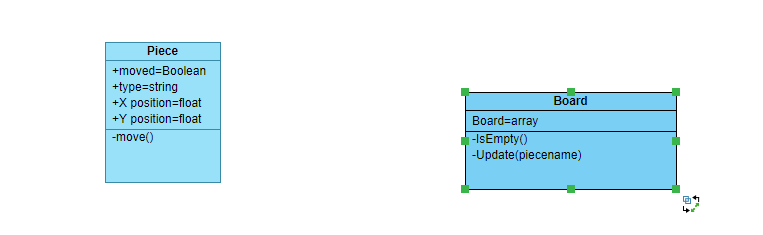
### Acceptable limitations

It is unlikely I will be able to implement the full set of rules, unless I code the whole project with them in mind. I aim to include as many of the basic rules as possible. I also do not plan to implement an online multiplayer as it would involve adding lots of extra code and modules. For the same reasons as above, I will not be designing a system to teach the players the rules, although I will tell them what moves they can make by highlighting the corresponding squares. Another function which I will not be implementing is an AI to act as an opponent.

### Data Sources and destination

The sources of data are the user inputs and clicks on the board. They will be stored as variables and are how the code will run. I will store board states in a text document and check it each turn to check for 3 board state repeats, which would result in a draw. This text document will be cleared at the end of each game or if the user quits. The final destination is the output and will be displayed on the user interface, using the Pygame module. My code will move all the pieces and once everything is done will update the screen and apply the new changes. A lot of the data stored will be done using classes. Each piece will be a separate class, with its own type and position. There will also be a class for the board which will be used to update the screen.

### Object Analysis Diagrams



### Chosen Solution

I will code my project with python. I plan to utilise the pygame module [12]. It allows me to capture inputs, mouse position and other variables and present a graphical user interface. I will use the python shell for testing and may use the tkinter module [13], which is a module that comes with python, to produce a more efficient menu. The game will start with a menu where the user can exit or play. The exit will close the game while play will take them to the chess screen. They can then click on pieces to bring up available moves. After the move the other player will be allowed to move in the same manner. The game will end when checkmate is reached and this will take them to the menu.

Pygame

[12]- <https://www.pygame.org/news>

[13]- <https://docs.python.org/2/library/tkinter.html>x

Pygame is a module that allows the user to create a game window and present and move images in that window. It is important to note that coordinates of pixels start in the top left of the window.

# Design

## Process of design

I think the best way to code this project is to use abstraction to remove details and reach set goals and waypoints. At each stage I will use decomposition to break up the problem into manageable tasks. I will have to ensure though that when making the program I do not abstract too far as to write code that cannot later be adapted so that all aims are incorporated. I will first do the below stages in pseudo code and ensure each stage works before progressing to the next one and will try to first outline the general structure and order of commands, functions and procedures. Then I will repeat the process for the actual code using the pseudo code.

#### Stage 1

First I will write the code to import all appropriate images, like the board and pieces. Their position will be manipulated to shift the pieces. Next I will write the classes and attributes. I will then implement the selection and movement of pieces using the methods of the classes, using the pygame module which can record mouse positions and buttons pressed on the mouse and keyboard. Whenever the player clicks on a piece the program will calculate all allowed moves and if they click elsewhere it will cancel. I will do this for both players by having a different ruleset for black and white pieces using an attribute called ‘Colour’.

#### Stage 2

I will then write the code to handle the taking of pieces using methods. Furthermore I will implement the check rule. Which will have to check after each move whether there is check state. This will work by checking all space the king can be taken from and check the type of the piece there to see if it matches with the corresponding move.

#### Stage 3

Next I will code the checkmate function, which will start with the check function but also check if the threatening piece can be taken or blocked or if the King can escape. This will all be done by checking from which squares the piece can be taken and checking if a piece is there, ensuring the move matches what the piece is allowed to do. I will also build the main menu and end menu.

#### Stage 4

I will add stalemate and draw rules. Furthermore, I will add extra rules like castling and En passant. Then, possibly a pawn upgrade feature. I will also add a pause feature.

#### Stage 5

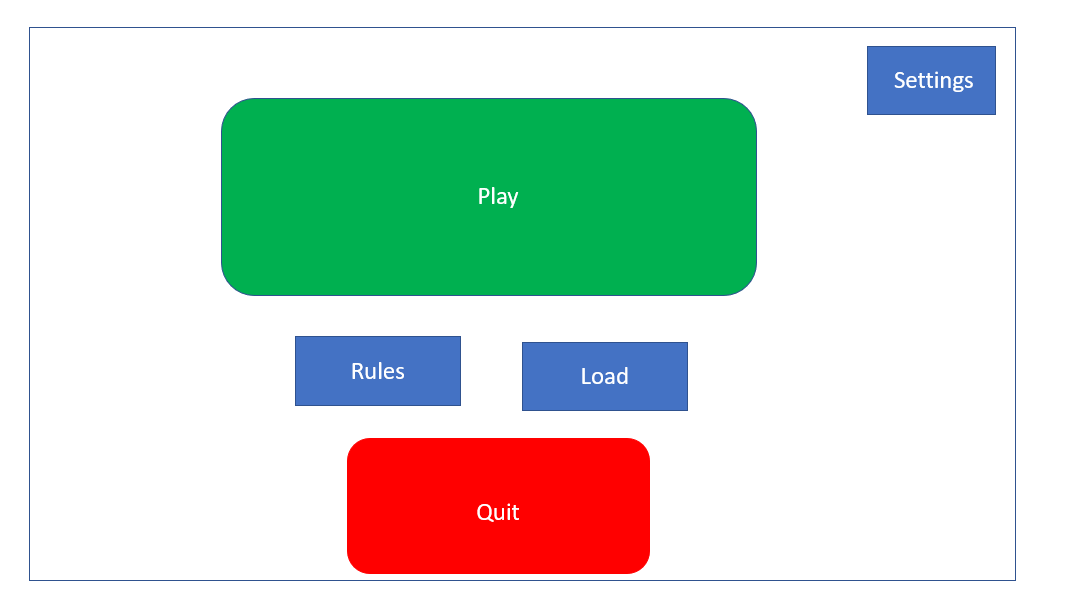
I will add save game and different timer lengths and increments.

## Input/Output

|  |  |  |
| --- | --- | --- |
| Data | Source | Destination |
| Click | User input via pygame | Method to display allowed squares to move to |
| Position of pieces | Program | Output to window |
| Board | Program | Output to winow |
| State of board(for board state repeat rule) | Program/text file | Program/ text file |

## The user interface

The user will be shown a pygame window containing the game and will first be presented with the main menu. Here they will have two options, play or quit(I may also implement a rules submenu). At any point the players will be able to click on the close cross in the corner to quit the application. I will also use the shell and possibly even tkinter to help with testing.



Home page



The chess board

## Basic function

There will be a game loop that runs and the pygame module will detect inputs and act accordingly.

## The maths behind animating the movement of pieces

Moving the pieces requires some maths and ensuring the pieces move to all squares at the same speed requires more maths. The first problem is deducing distance. I believe the best way is with Pythagoras. Let’s say the left corner of 2 squares have coordinates (1,1) and the other (2,3). I will do and . I will then use the abs() function to make both answers positive. The numbers *a* and *b* are then plugged into Pythagoras’ theorem: . This will be stored as distance. Next the code will need to do distance(which was just calculated)/speed. This will provide the time for moving the piece.

The next stage is to move the piece. I will be creating a method that runs in the game loop when the variable ‘moving’ is True. After the piece has moved the variable will be false until the user inputs the next move. This may be complicated but it allows the game loop to run, the board to update and the pieces to move without creating a mini loop that might produce frame jumps when the loop ends and the user returns to the main loop or cause errors where the window is unresponsive. Once the time is calculated the time is multiplied by 30(this is the frames per second I have set)to calculate the number of frames it will take. Then do distance/frames separately for x and y. Let’s say the number of frames is 45. In our example for the x coordinate the code will do 2/45=0.0444, So every frame the image will move 0.044444 closer to the point in the x axis. The same will occur in the y axis.

## Taking a piece

The piece that has been taken will be removed from the list of update files. So if they haven’t been placed.

## The idea of checkmate

Checkmate was devised because it was seen as dishonourable to kill the king. Therefore after every round check and checkmate must be checked for. The first step is checking whether or not the king can be taken by checking the squares the king could move to as if he could make the moves of every piece. If the colour of the piece in the square is the opposite of the colour of your pieces then the code must check if the piece is capable of taking the king.

If the king is found to be in check(an output to the user would be posted) then the the code must then decide if the king can escape or if the move can be blocked. This will take the most computing power. If the king can escape to a surrounding square all these squares must be checked as well. If it is a legal move then the loop will exit. For the king to be blocked all of your active (alive) pieces must be checked to see if they can take or block the enemy pieces. However the board state must be checked, prior to allowing the move, after each move has been calculated to see if that exposes the king.

Similar algorithms must be run when deciding the king’s available moves, but the above algorithm is checking if the game can continue. If there is a single allowed move, it will break from the loop so it doesn’t slow the program down. I will code it so the less likely outcomes that use lots of memory and checked for last.

As stated above the king’s legal moves will be calculated separately but will most likely require lots of computing power so the program may either calculate them automatically or if the king is clicked on once then the program will store the allowed moves in a text document or list so the user doesn’t have to wait if the king is deselected and reselected. A smaller calculation will be done after each piece moves checking the diagonals and horizontals for enemy pieces and the king as to stop illegal moves. If in check then the move moves must be checked to see if they alleviate the check.

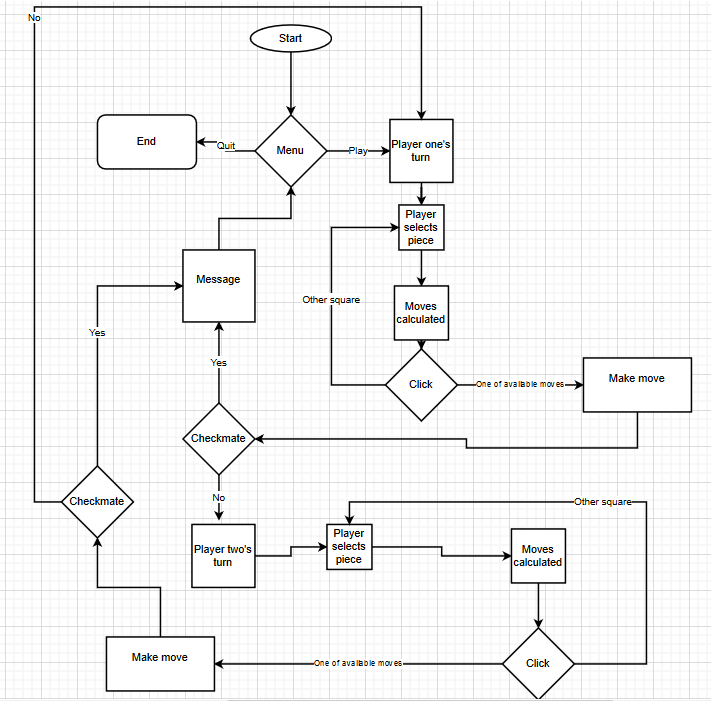
## Saving files

I will save details to a text file. I might need to find a way to compress

## Threefold repetition rules

A text file will be created that record the game state. Every round the new state will ideally be compared to the last one. There may be memory problems so I may only check the last 20.

## Flowchart



## Pseudo code

Pygame.init() #Starts pygame module.

displayWidth ← 600

displayHeight ←600

black ← 0,0,0)

white ← 255,255,255)#Here is an array of different colours which I might be using. This is in RGB the first number represents red intensity, the second green intensity and the last blue intensity.

red ← (255,0,0)

green ← 0,255,0)

blue ← 0,0,255)

chessDisplay ← pygame.display.set\_mode((displayWidth,displayHeight))#This creates the game window

pygame.display.set\_caption(“Chess”)#This sets the name of the window to chess

x ← 75

bottom ← x\*7

bottom2 ← \*6

These are used to make it easier to assign position of pieces based on row.

top ← 0

top2 ← x

RECORD board

ATTRIBUTES

board ← [['brook1', 'bknight1', 'bbishop1', 'bqueen', 'bking', 'bbishop2', 'bknight2', 'brook2'], ['bpawn1', 'bpawn2', 'bpawn3', 'bpawn4', 'bpawn5', 'bpawn6', 'bpawn7', 'bpawn8'], ['', '', '', '', '', '', '', ''],['', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', ''],['', '', '', '', '', '', '', ''], ['wpawn1', 'wpawn2', 'wpawn3', 'wpawn4', 'wpawn5', 'wpawn6', 'wpawn7', 'wpawn8'], ['wrook1', 'wknight1', 'wbishop1', 'wqueen', 'wking', 'wbishop2', 'wknight2', 'wrook2']] #This array stores the details of what is in each square

ENDATTRIBUTES

METHOD getPiece(mousex,mousey)

RETURN board[mousey][mousex]

ENDMETHOD

METHOD emptysquare(x,y)

IF board [y][x]= ‘’

RETURN True

ELSE

RETURN False

ENDMETHOD

METHOD enemysquare(colour, x,y)

TRY

character=board[y][x]

IF colour="black"

IF character[0]='w'

RETURN True

ELSE

IF character[0]='b'

RETURN True

CATCH

RETURN False

ENDMETHOD

ENDRECORD

RECORD piece#When the record or class is created parameters about the piece are passed

ATTRIBUTES(ptype,posy,posx,colour)

ptype ← ptype

posy ← posy

posx ← posx

moveforw ← 0

moveLorR ← 0

movedyet ← False

colour ← colour

ENDATTRIBUTES

ENDRECORD

RECORD King(piece)#The parent class or record is piece and any methods I write will be inherited by the individual piece classes

ATTRIBUTES(ptype,posy,posx,colour)

ptype ← ptype

posy ← posy

posx ← posx

moveforw ← 0

moveLorR ← 0

movedyet ← False

colour ← colour

IF colour ='black'

image ← pygame.image.load("blackking.png")#The way the GUI works is by manipulating the position of the images and this command imports the image for the black king and makes it an attribute of the class.

chessDisplay.blit(.image,(( posx-1)\*75,top))#This pygame command actually paints the image onto the window. It is important to do this in the right order otherwise the board image will paint over the pieces

ELSE

image ← pygame.image.load('whiteking.png')

chessDisplay.blit(image,((posx-1)\*75,bottom))

ENDIF

ENDATTRIBUTES

ENDRECORD

RECORD Queen(piece)

ATTRIBUTES(ptype,posy,posx,colour)

ptype ← ptype

posy ← posy

posx ← posx

moveforw ← 0

moveLorR ← 0

movedyet ← False

colour ← colour

IFcolour ='black'

image ← pygame.image.load("blackqueen.png")

chessDisplay.blit(.image,(( posx-1)\*75,top))

ELSE

image ← pygame.image.load(whitequeen.png')

chessDisplay.blit(image,((posx-1)\*75,bottom))

ENDIF

ENDATTRIBUTES

ENDRECORD

RECORD Knight(piece)

ATTRIBUTES(ptype,posy,posx,colour)

ptype←ptype

posy←posy

posx←posx

moveforw←0

moveLorR←0

movedyet←False

colour←colour

IF colour='black'

image←pygame.image.load("blackknight.png")

chessDisplay.blit(image,((posx-1)\*75,top))

ELSE

image←pygame.image.load('whiteknight.png')

chessDisplay.blit(image,((posx-1)\*75,bottom))

ENDATTRIBUTES

ENDRECORD

RECORD Rook(piece)

ATTRIBUTES(ptype,posy,posx,colour)

ptype←ptype

posy←posy

posx←posx

moveforw←0

moveLorR←0

movedyet←False

colour←colour

IFcolour←'black'

image←pygame.image.load("blackrook.png")

chessDisplay.blit(image,((posx-1)\*75,top))

ELSE

image←pygame.image.load('whiterook.png')

chessDisplay.blit(image,((posx-1)\*75,bottom))

ENDATTRIBUTES

ENDRECORD

RECORD Bishop(piece)

ATTRIBUTES(ptype,posy,posx,colour)

ptype←ptype

posy←posy

posx←posx

moveforw←0

moveLor←0

movedyet←False

colour←colour

IF colour='black'

image←pygame.image.load("blackbishop.png")

chessDisplay.blit(image,((posx-1)\*75,top))

ELSE

image←pygame.image.load('whitebishop.png')

chessDisplay.blit(image,((posx-1)\*75,bottom))

ENDATTRIBUTES

ENDRECORD

RECORD Pawn(piece)

ATTRIBUTES(ptype,posy,posx,colour)

ptype←ptype

posy←posy

posx=←posx

moveforw←0

moveLorR←0

movedyet←False

colour←colour

IF colour='black'

image←pygame.image.load("blackpawn.png")

chessDisplay.blit(image,((posx-1)\*75,top2))

ELSE

image←pygame.image.load('whitepawn.png')

chessDisplay.blit(image,((posx-1)\*75,bottom2))

ENDATTRIBUTES

ENDRECORD

## Algorithm

# Technology Solution

# Testing

# Evaluation

Touch control once touched

En passant

I can use recursive algorithm with pieces like bishop

Talk about backing up code.