40 Monaro Crescent Red Hill

2013

SDD Project Overview

Chess Game

Andrew M Hall

Contents

[User Section 2](#_Toc374081726)

[Program Overview 2](#_Toc374081727)

[Brief History 2](#_Toc374081728)

[Game Rules 2](#_Toc374081729)

[Screen Design 4](#_Toc374081730)

[Game Screen 4](#_Toc374081731)

[Main Menu 5](#_Toc374081732)

[Developer Section 6](#_Toc374081733)

[Data Structures 6](#_Toc374081734)

[Arrays 6](#_Toc374081735)

[Array of Records 7](#_Toc374081736)

[Structure Chart 8](#_Toc374081737)

[Chart 8](#_Toc374081738)

[Description 9](#_Toc374081739)

[Data Flow Diagram 12](#_Toc374081740)

[Diagram 12](#_Toc374081741)

[Description 13](#_Toc374081742)

# User Section

## Program Overview

### Brief History

Chess has been around in many forms since the 6th century, it’s early predecessors were the early East Indian game “chaturaṅga”. At around the 10th century, different versions of chess developed in Europe. These early versions of chess were the first to introduce the royal pieces and formalize the rules. In around 1475, the pieces gained the move range that they have today with the bishop being given it’s diagonal moves and the pawns being allowed to take two moves on their first turn.

### Game Rules

#### King

The Kink can move one square in any direction. When the king is threatened by one or more of the opponent’s pieces, It is said to be in check and only moves that remove the king from check are permitted. There is only one king per player and once the king is taken, the game is over

#### Rook

The rook moves any number of vacant squares vertically or horizontally.

#### Bishop

The bishop moves any number of vacant squares in any direction diagonally. Note that a bishop never changes square colour.

#### Queen

The queen can move any number of vacant squares diagonally, horizontally, or vertically. There is only one queen per player.

#### Knight

The knight can jump over occupied squares and moves two spaces horizontally and one space vertically or vice versa, making an "L" shape.

#### Pawns

Pawns can move directly forward one square, however, on their first turn, they can move two squares.

#### The Aim

The aim of chess is to force your opponents’ king into check mate, a position where it cannot escape being taken.

To do this, both players move their respective pieces to take those of their opponents and gain strategically advantage. Scoring in chess is done by assigning each piece a value, when that piece is taken, the score is added to the total of the person who took the piece.

The Values of each piece are below:

Pawn: 1 Point   
Rook: 5 Points   
Knight: 3 Points   
Bishop: 3 Points   
Queen: 9 Points   
King: 0 Points

## Screen Design

### BB.pngBK.pngBN.pngimagesCAHCMGXU.jpgimagesCAHCMGXU.jpgimagesCAHCMGXU.jpgimagesCAHCMGXU.jpgimagesCAHCMGXU.jpgBR.jpgBQ.jpgBB.pngimagesCAHCMGXU.jpgScreen Design gamescreen.pngWB.pngWB.pngWK.pngWN.pngWP.jpgWP.jpgWP.jpgWP.jpgWP.jpgWQ.pngWR.pngGame Screen

#### WP.jpgimagesCAHCMGXU.jpgBN.pngDesign

#### Description

Restart

Submit Scores

Exit

This screen is intended to allow the user to easily play the game. The list of pieces that have been lost by each player should help the players to easily judge who is winning and losing, a job that cannot be told by just the score. There is a button for each player to submit his/her score and this is to avoid accidental moves by the players by wayward clicks. The high scores are displayed in a list box because there could be potentially 100’s of high scores. There are three buttons at the bottom left hand corner of the screen that allow either player to restart, exit or submit the game.

### main menu Screen Design.pngMain Menu

#### Design

#### Description

This main menu screen allows the user to navigate through the program with access to help files, a list of rules and the actual game. The screen design is simple with four buttons that are clear in their intent and use. This plain design is meant to provide maximum functionality while not detracting from the game.

# Developer Section

## Data Structures

### Arrays

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Array Name | Dimensions | Index | Data Type | Description | Scope |
| high\_scores | 1 | 1 to ∞  (unlimited) | String | Holds the high scores so that they can be sorted | local |
| picturebox\_array | 1 | 1 to 64 | Picture box | Holds the picture boxes that will be used to display the board | global |
|  |  |  |  |  |  |

### Array of Records

Record Name: recPiece

Scope: Global

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| piece\_name | string | Name of the piece in question (e.g. “WK” (for white king)) |
| movement\_type | string | A code specific to the piece that defines how it can move.  Will be used to verify that the move is legal. |
| value | integer | will hold the game value of the piece in question.  (e.g. Pawn = 1) |
| location | string | holds location of the check on the board (e.g. (H, 3)) |

Array of record name: arrboard

Scope: Local

Dimension: 2

Index: 1 to 8, 1 to 8

Description: holds data about the board and each check on it including information about the pieces on it

Element 1

## Structure Chart

### Structure Chart.pngChart

### Description

The game is divided into three top-level modules, Initialise Board, Move and Scoring.

#### Chess

Chess:

The chess subroutine will use control the flow of logic in the game, it will allow the player to play the game and then submit their score.

Control Structure(s): it will use a While...EndWhile loops to allow the players to keep playing the game until they have finished. It will use several If...EndIf selections to validate the data that the user enters into the system. (names)

#### Initialise Board

Fill Picture Box Array:

This subroutine inputs the 64 picture boxes that are required to display the board into a 1D

For ease of manipulation

Control Structure(s): It will utilise two For...Next loops to enter data efficiently into the array

Fill Board (2D array):

This subroutine will input all the information about the chess pieces and the checks into a 2D array (arrBoard)

Control Structure(s): This will utilise at least one For...Next loop to enter data. More may be used for different data to be entered

Display Board:

This subroutine displays the board, the checks and the pieces onto the computer screen

Control Structure(s): This will use two For...Next loops to display the data in the appropriate picture box.

#### Move

Move:

This subroutine will control the actions of getting the players move, validating the move and executing the move.

Control Structure(s): It will use a Repeat...Until loop to get the move and validate it. It will exit the loop once the move is valid.

Get Move:

This subroutine is activated when the player wishes to submit their move, it then gets the move and all the information about the move.

Control Structure(s): This does not need any control structures as it is simply getting a move

Execute Move:

This subroutine will validate the move that is passed to it from “Get Move”, if the move is valid then it will execute the move by displaying it on the screen and, if needed, it will update the score. (if a piece is taken by the move)

Control Structure(s): This will use two For...Next loops to cycle through the board and find the appropriate squares.

Check Move:

This subroutine will control the validation of every move, it uses six functions to validate the move, these functions are not always used and are selected based on the piece that is passed to the subroutine.

Control Structure(s): This will use a multiway selection to select the appropriate function to call to validate the move.

Check King Move:

This function is used when the subroutine wishes to validate the move of a king piece.

Control Structure(s): This will use a two For...Next loops to check the cells that are surrounding the piece.

Check Queen Move:

This function is used when the subroutine wishes to validate the move of a queen piece.

Control Structure(s): This will use four, For...Next loops to validate the queens move, for loops are required due to the great range that the queen can move.

Check Rook Move:

This function is used when the subroutine wishes to validate the move of a rook piece.

Control Structure(s): This will use two, For...Next loops to validate the move.

Check Knight Move:

This function will be used whenever a knights move needs verifying. This is a slightly complicated function because of the knights complicated movement rules.

Control Structure(s): This will use a While...EndWhile loop to check squares that are at certain positions relative to the knight.

Check Bishop Move:

This function will be used whenever a bishops move needs verifying.

Control Structure(s): This will use two, For...Next loops to check squares that are in a diagonal position to it.

Update Board:

Once the move has been validated, this subroutine will update the board with the required changes. This will have access to the board and to the picture box array.

Control Structure(s): This will use three, For...Next loops, one to loop through the picturebox array, and two to loop through the board array.

Update Score:

This subroutine will update the score of the game whenever a piece is taken, it will then display it on the screen

Control Structure(s): This will use no control structures because it is merely updating a score

#### Scoring

Scoring:

This subroutine will control the flow of logic that is used to submit and display the scores.

Control Structure(s): This will use no control structures as it is not really manipulating anything that requires them.

Get High Scores:

This subroutine will read from an unordered, high score sequential text file that will contain the names and scores of previous players and put them into an array (high\_scores).

Control Structure(s): This will use a While...EndWhile loop to read the file contents into the high\_scores array

Sort Scores:

This function will use bubble sort method to sort the high\_scores array and return the sorted array.

Control Structure(s): This will use one While...EndWhile loop to loop through the array, and an If...EndIf selection to select the highest element out of a pair

Write to file:

This subroutine will append the new score to the end of the high scores file.

Control Structure(s): This will not use any control structures as it is a simple task that requires none

## Data Flow Diagram

### Level 1 DFD.pngDiagram

### Description

This data flow diagram shows all of the main data that is passed between the main subroutines. The subroutine “Initialise Board” has been left out because it does not take any data and could be considered a “black hole” if it were in the DFD. This DFD reflects the layout of the program by having most of the data passed to the “Chess” subroutine and then split up between the “move” subroutine and the “scoring” subroutine.

Chess:

The chess module controls the flow of data in the program and allows for the modules to be easily dealt with.

Move:

The move subroutine, as already stated, receives the players move and the board. It then returns the edited board and updated score.

Scoring:

The scoring subroutine receives the players score and gets the high scores from the file (datastore). It will then sort all the scores and return them to the player.