Software Engineering Group 17

User Interface

|  |  |
| --- | --- |
| Author: | Tate Moore [tam41] and Adrian Enache [ade12] |
| Date: | 14th February 2023 |
| Version: | 1.0 |
| Status: | Draft |

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Copyright © Aberystwyth University 2021

CONTENTS

CONTENTS 2

1. Introduction 3

1.1 Purpose of this Document 3

1.2 Scope 3

1.3 Objectives 3

2. Possible users 4

2.1 Summary 4

2.2 In-depth 4

3. Guidance 5

4. Different Scenes 6

5. Use Cases 7

6. Paper UI Prototype 9

7. Program Prototypes 12

REFERENCES 13

DOCUMENT HISTORY 14

# Introduction

The user interface is an important part of software engineering.

## Purpose of this Document

This document is to describe how the user interface will work including current design ideas.

## Scope

This supports the standards laid down for CS22120 Group Projects [1].

## Objectives

The objective of this document is to show how the user interface will work including the design ideas for the user interface. This is an important part for the design of the project.

# Potential users

## Summary

* **Novices –**
  + Do not really know how to play chess.
  + Needs lots of guidance.
* **Occasional –**
  + Knows the rules of chess and has some experience.
  + May forget some moves over time so may need a little reminding.
* **Expert** –
  + Knows how to play chess.
  + Does not need any help at all.
  + Just wants to play the game without hindrance.

## In-depth

### Novices

A novice user has little to no knowledge of chess, upon playing the game they will be pondering many questions such as: what does this piece do? Can I select this piece? Is it my turn? How does the game end? From this we can conclude they need to be delivered a great deal of guidance.

### Occasional

The occasional user has played chess to some extent where they know the basic rules of chess however, they may forget some moves over time for example it may slip their mind castling is possible. Accordingly, they will need a little guidance.

### Expert

The expert user has vast knowledge and experience of chess, they know the best move to make and want to transfer that onto the virtual chessboard and have it carried it out as quickly as possible. In other words, the game must not slow them down or hinder them in anyway.

### General

This is what all 3 of the previously mentioned users have in common, we are assuming they can use a computer; they know their way round a basic, typical interface and can use a mouse and keyboard decently. For example, “How do I load an old game? Well, I am guessing I go the ‘File’ tab then click ‘Load’.”

Virtual chess has very simple controls, you click on a piece, you then click somewhere else on the board so we will be relying on our assumption of our users’ ability to use a computer.

### Conclusion

Many of our users (occasional and novice types) will need guidance however it is important for our expert users that this guidance does not slow or hinder them. This is significant for how we need to create and tailor the guidance.

## Examples of users

### Novices

Rahim is a 9-year-old in primary school and has been encouraged by a friend to play chess with him. They do not have access to a chessboard and enjoy using computers. As a result, they are searching for an application online.

### Occasional

John is a young adult; he has recently started working after graduating university and is looking to spruce up his life with some hobbies and reconnect with some university friends. He previously played chess with his father when he was a child. He cannot remember some of the rules, but he is thinking about playing chess with friends on the computer.

### Expert

Kaylie is a 40-year-old mother who previously played in secondary school chess clubs and even in competitions (winning most of the matches). She is extremely confident and skilled at the game. She commutes by train to work and is interested in using a chess application on her laptop whilst on the train.

# Guidance

## Introduction

We need a system that delivers guidance without slowing down or hindering an experienced user, this section goes over different bits of guidance which obey this.

## Valid Piece Selection

A valid piece includes one that is an actual chess piece (not off the board or an empty square) and is of the right colour. The currently selected piece will be shown with a yellow glowing outline around the chess piece. Novice users will then know if they have successfully selected a valid piece.

## Valid Moves

When a piece is selected, a novice user will need to know which moves it can make. You can either move to an empty square, take an opposing piece.

* Guidance includes:
  + Valid piece selection – Such as a glowing light outline around the chess piece to show it has been selected.
  + Valid moves –
    - Such as a small glowing circle to show the moves.
    - Potential pieces that can be taken shown with a red glowing outline or marks around it.
  + Whose turn it is –
    - JavaFX has rotation animations, we were thinking that we may rotate the board and the piece around, so the bottom side indicates whose turn it is.
    - Or we could display a quickly disappearing text saying e.g. “White’s [or player name’s] turn” which does not hinder experienced players (i.e. stop the game to display the message) so you can still do stuff while it is displayed.
    - The latter is easier and I am sure it is possible, for it is harder and I am not sure about being able to rotate pieces. If they are assembled in a [Grid Pane](https://jenkov.com/tutorials/javafx/gridpane.html), we may be able to rotate the grid pane and with the centre/point of rotation being the centre of the board/grid pane.
    - Or we could rotate the positions and keep the graphics the same so that the pieces are still facing the correct direction.
* In JavaFX you can create a [menu bar](https://jenkov.com/tutorials/javafx/menubar.html) at the top of the window, we are thinking about what we are doing inside this:

* When a king is in check or in checkmate
  + Check – at the start of the round, quickly disappearing text says ‘Check! – [colour]’s king is in check’
  + Checkmate – brings up the end scene.

* An incorrect move or selection.
  + Quickly disappearing text saying ‘Incorrect move: [error]’

# Guidance

* Need a system that delivers guidance without slowing down/hindering experienced users.
* Guidance includes:
  + Valid piece selection – such as a glowing light outline around the chess piece to show it has been selected.
  + Valid moves –
    - Such as a small glowing circle to show the moves.
    - Potential pieces that can be taken shown with a red glowing outline or marks around it.
  + Whose turn it is –
    - JavaFX has rotation animations, was thinking we could rotate the board and the piece around, so the bottom side indicates whose turn it is.
    - Or we could just make a quick appearing and disappearing text saying e.g. “White’s [or player name’s] turn” which does not hinder experienced players (i.e. stop the game to display the message) so you can still do stuff while it is displayed.
    - The latter is easier and I am sure that it is possible, former is harder and not sure about being able to rotate pieces. If they are assembled in a [Grid Pane](https://jenkov.com/tutorials/javafx/gridpane.html), we may be able to rotate the grid pane and with the centre/point of rotation being the centre of the board/grid pane.
* In JavaFX you can create a [menu bar](https://jenkov.com/tutorials/javafx/menubar.html) at the top of the window
* When a king is in check or in checkmate
  + Check – at the start of the round, quickly disappearing text says ‘Check! – [colour]’s king is in check’
  + Checkmate – brings up the end scene.

* An incorrect move or selection.
  + Quickly disappearing text saying ‘Incorrect move: [error]’

# Different Scenes

* **Start-up/Menu**
  + Start new game
    - Prompt for names and whether black or white.
  + Load new game –
    - [File chooser](https://jenkov.com/tutorials/javafx/filechooser.html) (File handling required)??
  + Exit program (but we can add an exit button to the top right)

* **End**
  + Displays who has won due to… [checkmate/ player surrenders / stalemate / quitting] or if they mutually agreed to end the game.
  + Save the game?
  + Exit to start-up scene?

* **Board**
  + Pieces – in a grid pane
  + Board (Image? Or [grid pane](https://jenkov.com/tutorials/javafx/gridpane.html))
    - *Can we have two grid panes in a root node?*
  + Event message text label (e.g. invalid move, check etc)
  + Menu bar
    - Assisted gameplay
      * Piece moves
      * Graphical effects
    - File
      * Load
      * Save
      * Save & exit to start up menu
    - End game
      * [player] Resign
      * Draw
    - Round
      * Undo – goes back a round
      * Redo – goes forward a round

# Use Cases

## Summary

**User tasks** – All users, Novice users

|  |  |  |  |
| --- | --- | --- | --- |
| **Tasks** | **Information Needed** | **Information Presentation** | **User Choices Needed** |
| Select piece | All available pieces. | Chess board with pieces. | Click on a piece. |
| Whose turn it is. | Rotation of board and text event message to show whose turn it is. |
| Which piece is currently selection. | Yellow glowing outline around chess piece. |
| Move to empty square | Available empty squares (potential moves) | Small glowing yellow icon for each empty square. | Select a piece then click on empty board square. |
| Take enemy piece | Available captures (potential moves) | Red glowing outline around enemy piece. | Select a piece then click on an enemy piece. |
| Castling | If castling is possible. | Red glowing outline around king or castle. | Select a king (or castle) then click on a castle (or king). |
| Load | The users’ files. | JavaFX FileChooser class. | On start-up menu scene, click load. |
| Un/Redo | Where to click | JavaFX MenuBar class item. | Click on the MenuBar item. |
| Start | Player names for white and black. | Basic example:   * White: [input name] * Black: [input name] | On start-up, click start, enter the player names for both teams. |

## In-depth

### Select piece

To select a piece on the board, the user must know all the available pieces that can be selected, this will be shown through a display of chessboard with pieces. Secondly, what if a player is unsure whether it is their turn or not? Suitable presentation will be rotation of the board (bottom shows the acting team) and a text event message showing whose turn it is. Lastly, the user will need confirmation that they have successfully selected a valid piece which will appear as a glowing yellow outline around it. The way you will select a piece is by clicking it with mouse button 1.

### Moving a piece

Each piece type has a different set of available moves they can make, for a piece this set is restricted by other pieces and the edges of the board. They will be moving to either an empty square or take a piece of the opponent. Castle and king pieces can perform castling under several conditions, most novice and occasional users will forget about this move. Therefore, we can conclude it is important to show the possible moves. Valid empty squares are presented by small glowing icons and valid pieces (includes both taking opposing pieces and castling) shown by red outlines. Before this can be all be done, a valid piece will need to have been selected. Once done, you can move to a valid square or piece by clicking upon it with mouse button 1.

### Loading an old game

Two players may wish to resume a game later, upon that time they will need to load an autosave file, the only information they will need is the file’s name and folders in which it is stored. This will be presented on a scene with JavaFX’s FileChooser class. To get to that, the user will either click a button on the start-up menu or a JavaFX MenuBar class item. The user will not need to manually save the file as the game will autosave every round.

### Undoing and redoing moves

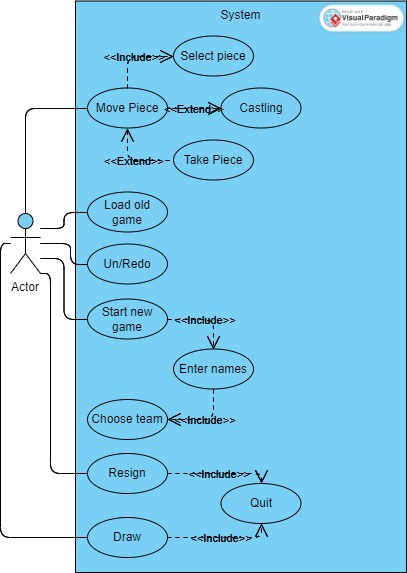
The current/most recent state of the game will be loaded, players may wish to resume an earlier point. To do so they will need to go forwards (redo) and backwards (undo) throughout the game. They can do this on the chessboard scene by clicking one of two JavaFX MenuBar class items – one for redo and the other undo.

### Starting a new game

In order to do all the previous tasks, they will have needed to have created a new game in the first place. This can be done by clicking on a ‘start new game’ button on the start-up menu which will lead to a new scene where they will input

**User tasks** – All users, Novice users

|  |  |  |  |
| --- | --- | --- | --- |
| **Tasks** | **Info needed** | **Info presentation** | **User choices needed** |
| Select piece | All available pieces | Chess board /w pieces | -Click on a piece – grid pane click detection |
| Whose turn is it | Rotation of board or text message to show whose turn it is. |
| Which piece is currently selection | Yellow glowing outline around chess piece |
| Move | Available moves | Small glowing yellow circle for each move | [Select a piece](https://docs.google.com/document/d/1crrGssqi4JDSVN9pkLXcBBjZapfDJ5UgKEphQ3WnhOQ/edit#bookmark=id.ewj4iowzjqvk)  Click on empty board square. |
| Take | Available captures | Red glowing outline around enemy piece | [Select a piece](https://docs.google.com/document/d/1crrGssqi4JDSVN9pkLXcBBjZapfDJ5UgKEphQ3WnhOQ/edit#bookmark=id.ewj4iowzjqvk)  Click on an enemy piece. |
| [~~Help~~](https://docs.google.com/document/d/1crrGssqi4JDSVN9pkLXcBBjZapfDJ5UgKEphQ3WnhOQ/edit#bookmark=id.9en5fw9rwtt1) | ~~Help information~~ | ~~Separate scene~~ | ~~Click on Menu Bar.~~ |
| Load | File name? | javaFX [File Chooser](https://jenkov.com/tutorials/javafx/filechooser.html) | On Start-up/Menu, click Load |
| Un/Redo | Button? Current round? | Menu bar option under round |  |
| Start | Player names. | E.g. White: [textbox] | On start-up menu, click start, type name |
| Quitting | Should a user wish to stop the game and save it for later | Button in the top corner | Quit button which is clicked on your round to end the game and saves the game for later |



**Potential Errors**

|  |  |
| --- | --- |
| Tasks | Errors |
| Select | * Select outside the map * Select a piece that is not yours. |
|  |
|  |
| Move | * Outside of map. * Friendly piece * Enemy piece outside of range. |  |
| Take | Available captures   * Outside of map * Friendly piece. * Enemy piece outside of range. |  |
| Load | * Not the right file – show txt only * Wrong format of txt |  |
| Un/redo | Undoing when it is the first round, redoing when it is the last. |  |
| Start | Name too long or invalid characters |  |

# Paper UI Prototype

A piece of paper with writing on it

Description automatically generated with medium confidence  
  
  
  
  
  
A piece of paper with writing on it

Description automatically generatedA drawing on a piece of paper

Description automatically generated with medium confidence

# Program Prototypes

The image below is a prototype of the start screen of the application.

# Diagram Description automatically generated

# A picture containing text, checker, tiled Description automatically generated

The above image is a prototype of how the game board may look in the last version of the game.

REFERENCES

[1] Software Engineering Group Projects: General Documentation Standards. C.W. Loftus. SE.QA.03. 2.5 Release

DOCUMENT HISTORY

| *Version* | *Issue No.* | *Date* | *Changes made to document* | *Changed by* |
| --- | --- | --- | --- | --- |
| 1.0 | N/A | 12/09/08 | N/A - original version | Ade12 and Tam41 |