Jake McDonagh [jam93]

SE Group 07  Aberystwyth University

user Interactions between UI and UMl diagrams

Contents

[Brief Introduction 2](#_Toc474756509)

[The scenes within the application 2](#_Toc474756510)

[Main Menu 3](#_Toc474756511)

[Game Map 4](#_Toc474756512)

[Game Rules 6](#_Toc474756513)

[Game Information 7](#_Toc474756514)

# Brief Introduction

The following pages within this document contain how the application is going to call the functions needed in order to update the game in an appropriate manner via the use of user interaction. Essentially, this document will go into more detail on what functions are called when certain actions by the users are committed.

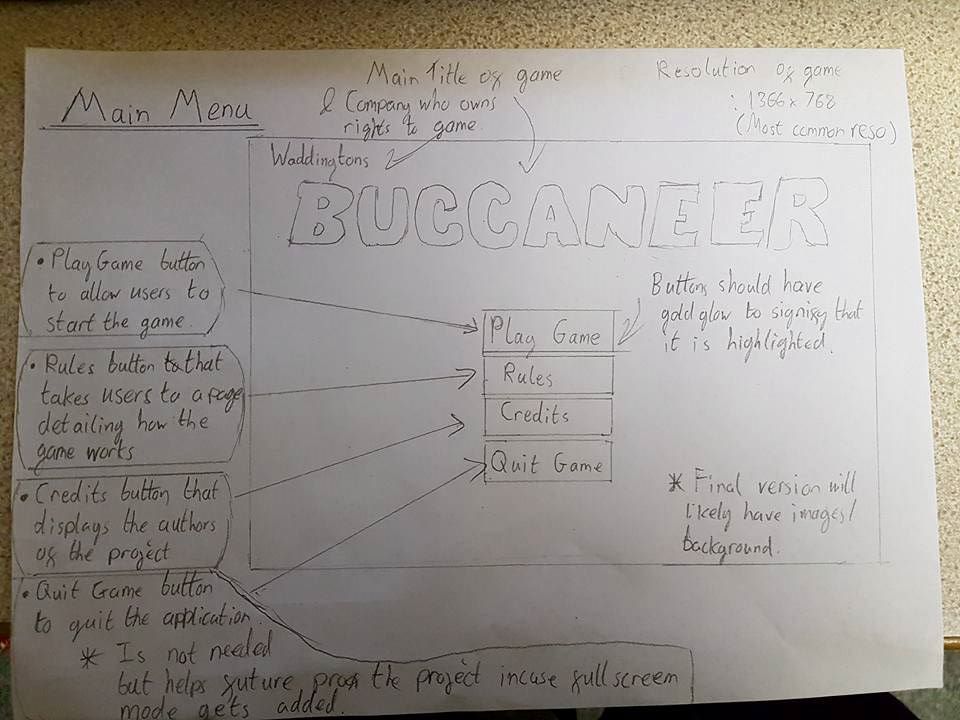
Be aware that this document is a draft in its entirety and is therefore informal. This does not accurately reflect the documentation of the UI which will be produced at a later date.

# The scenes within the application

During the design of the UI, there are several screens (in terms of JavaFX, scenes) that will need to be created in order to allow the users of this program to seamlessly move from one stage of the game to the next. The following scenes that will need to be created and thus, will be discussed in terms of functionality are as follows:

* Main Menu
* Game Map (The Main part of the game)
* Rules Page
* Game Information
* Credits Page
* Several overlaying scenes throughout player turns (popups)

## Main Menu



As we can see from the above draft, the main menu screen contains only buttons with the intention of taking us away from the main menu and onto new screens entirely.

**User Input** -> *Function Called (Brief summary of user interaction) (if Function Called underlined, needs to be discussed with group/added to UI UML diagram)*

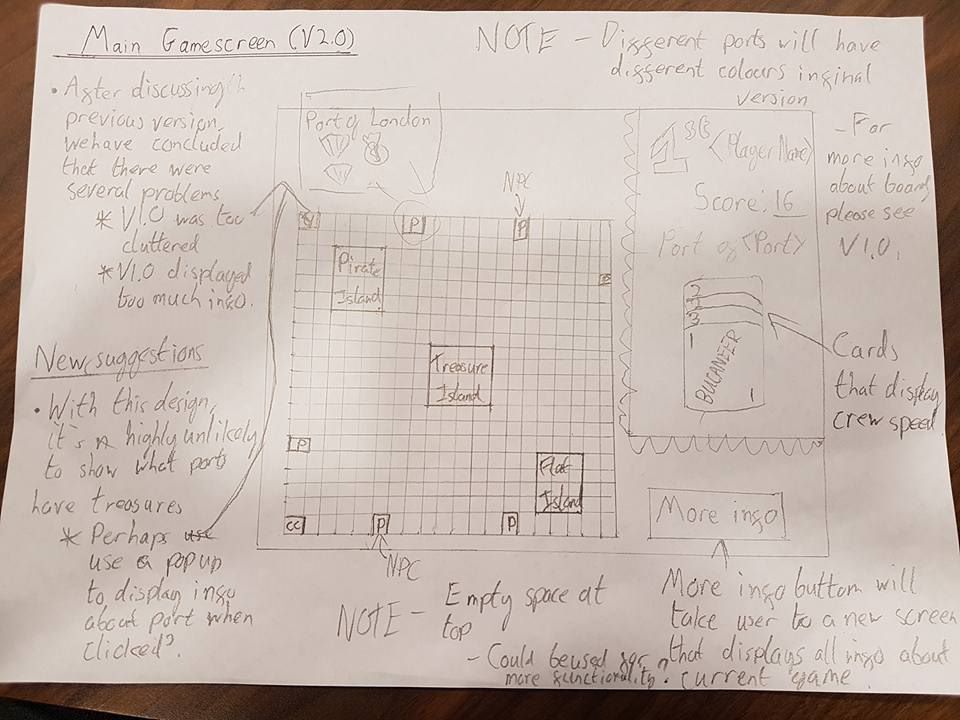
**Play Game Button Clicked** -> *+StateMachine setupGame() (When the user clicks the button, the method is called, which will prepare the game board, allow the users to enter their names(?) and move onto the Game Map to proceed with the game)*

**Rules Button Clicked** -> *+Application displayRules() (When this button is clicked, the rules page is displayed for the user. The rules page is a static page that briefly explains the rules)*

**Credits Button Clicked** -> *+Application displayCredits() (When this button is clicked, the credits page is displayed to the user, which gives credit to the people involved within the project)*

**Quit Game Button Clicked** -> *+Application quitGame() (When this button is clicked, the game closes, along with all of the information about the game as well as the state that the game is in)*

## Game Map



From this draft, we can see that there is a lot more going on. Along with several buttons that will take the user to new scenes, the UI also features an interact-able map, which allows the user to click on the coordinates that they wish to go to (when it’s their turn) and on the islands and ports in order to gain more information about them.

**User Input/UI Element** -> *Function Called (Brief summary of user interaction) (if Function Called underlined, needs to be discussed with group/added to UI UML diagram)*

**More Info Button Clicked** -> *+Application displayMoreInfo* *(When this button is clicked, the user is taken to a new screen which will display all of the information about the current game)*

**Top Right Corner displaying info about current player** -> *+Application updateCurrentPlayer (Uses* *Player getCurrentPlayer) (The top right of the scene will display all of the information about the current player, including score, which port they belong to as well as their crew cards, treasure cards and the current position that they’re in. Using the getCurrentPlayer method allows us to display all of the information that is needed for the user. This function will be called whenever a change in data occurs to the current player)*

**Clicking on port/island** -> *+Application displayInfo(Might be Board displayInfo?) (When the user clicks an island/port, all of the information regarding that space should be displayed as an overlapped interface like above)*

**When a player ship lands on another ship** -> *+Application displayFightPrompt (When two ships meet, it should display a prompt for the attacker player, then followed by the defending player stating a true/false case of if they wish to attack. If attacking player OR defending player wishes to attack, call the performFight method already in the UML diagram)*

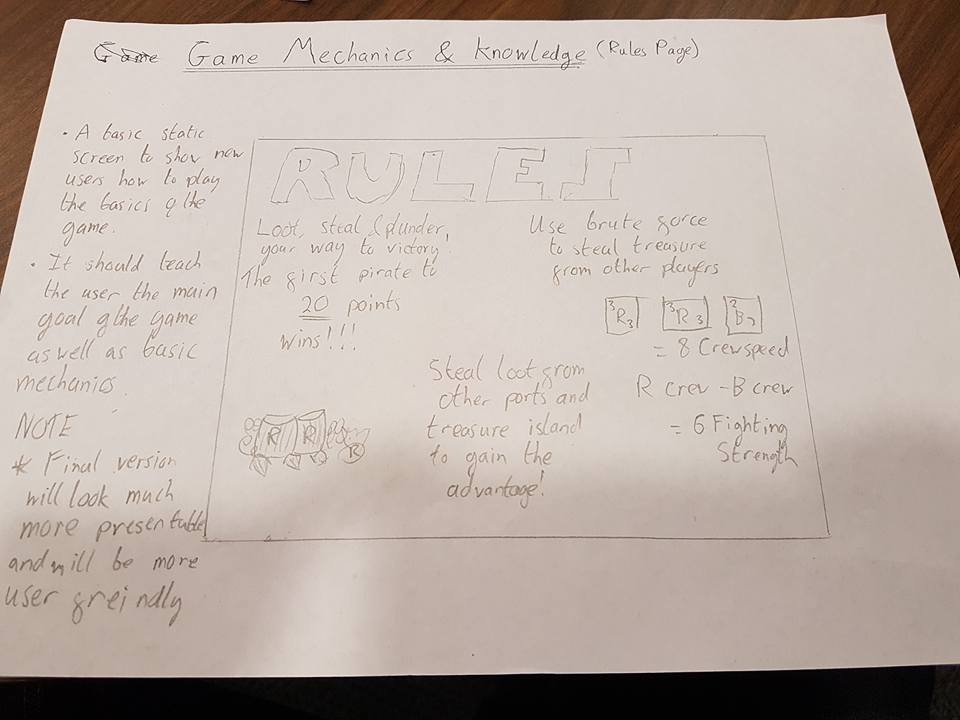
**When a player moves to their own port** -> *+void despositTreasure (The current player deposits the current treasure that they’re carrying)*

**When a player moves to an opponent’s port** -> *+void stealTreasure / +void stealCards / +void performTrade (The current player may do several actions, such as stealing treasures/cards or performing a trade with them)*

**When a player has chosen the direction that they want to face for next turn (via compass)** -> *+int serOrientation (The player ship’s orientation faces to whichever direction the player has chosen. The program uses ENUMS for this) NOTE: The ship will likely physically change direction on the map, will need to create a new function in application for this.*

**When a player picks up a Chance Card** -> *+Application displayChanceCard (The program will display an overlay showing the card and what the chance card does to the user)*

## Game Rules

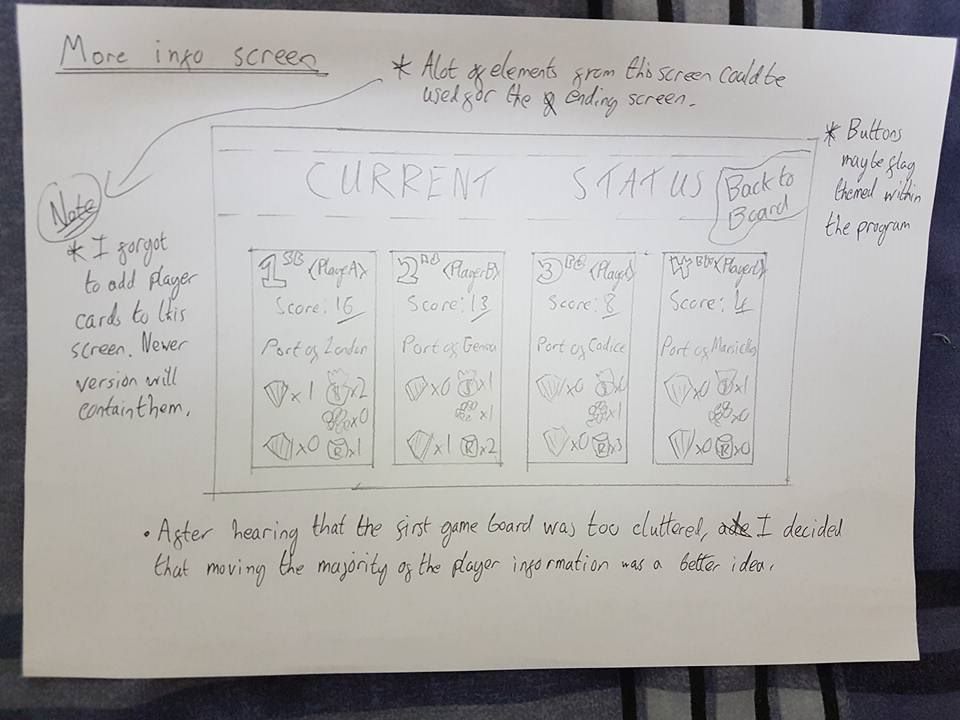


From this picture, we can see that the rules page for the most part is completely static. Although it doesn’t correctly display it here, there will be a button in the top right which will prompt the user to return to the main menu once done

**Rules Button Clicked** -> *+Application displayMainMenu() (When this button is clicked, the user is returned to the main menu. The rules page is a static page that briefly explains the rules)*

NOTE: This case is also the same for the Credits Page. To save time to spend on other parts of the project, I have chosen not to do this for the credits page as the user interaction between the credits and rules pages are exactly identical.

## Game Information



This scene displays all of the available information to the players at any given point. This is accessed by clicking the “More Info” button during the game scene.

**Information in each player box** -> *+Application displayPlayerData (When this scene loads, the program should use the data that is held to each player in order to display the current positions, scores, belonging ports and treasures and anything else to the user)*

**Back to Board Button Clicked** -> *+Application displayGameBoard() (When this button is clicked, the user is returned to the game board)*