CO2001 HCI & User Interfaces

Benjamin Meysner
Department of Informatics
University of Leicester
Student ID:159039533

January 3, 2017

1 Introduction

This is a report which discusses the various Human Computer Interaction and User Interface decisions made when designing, developing and testing the Mini Project part of this module. I will discuss these concepts in corresponding order.

1.1 Software & Tools

Eclipse, SceneBuilder, Java, JavaFX, JUnit, Hamcrest, FXML, Adobe Photoshop, CSS.

2 Design

The main argument of this project is angled around usability, hence the module title. When incorporating user centered philosophy into my project design I wanted my decisions to produce a completely playable experience for a person who has no previous understanding of the underlying code. Personally, a good user experience in this sort of capacity will depend on a collection of aspects including easy-on-the-eye graphical components, errors (if exists) which are handled and will not break the run time and longevity.

2.1 Graphical & Layout

Firstly, I first began researching various online versions of "Hang Man" to refresh my memory and collect new ideas. I drafted a "pseudo-code" adaptation of the HangMan game logic inside a temporary .java file. Once I was happy with the way I would approach the game logic I began to fish for free-to-use graphics which I hoped would bring the element of "in-game experience" to the user rather than depressing blocks of colour. I eventually settled for a free cartoon-style background and appropriate images which I thought would appeal to both younger and older generations. I then went on to personally create the games

logo, the hang man images and the general graphical theme for my project - "The FangMan".
I chose to begin the application with an introductory page "Intro.FXML" where the player would be able to enter their name before continuing into the next scene. I thought that this would incorporate a personal touch and make the player feel welcomed, thus improving longevity. Inside "Main.FXML" (the main game window), I divided the pane using various containers to clearly separate the areas of the window. The panes would be filled black, set to reduced opacity (so the background can be partially viewed) and font colour set to white. These settings would aid the application aesthetically. Grid panes were used to divide statistical information and buttons were used for user functionality (also heavily modified with CSS).
I decided to use different fonts for certain components inside the game, for example the alphabetic game pad used a custom font which I felt wouldn't sustain the same character as if used with a basic system font.

3 Development

The practical part of this project presented many new challenges for me. This would be my first experience using SceneBuilder, JavaFX and FXML therefore required a lot of effort familiarising myself with its features. I found that various different versions of the game show different methods of implementing actionable scenarios. For instance, When a player would like to make a letter selection they would either be required to type into a dialog box, typing into the application itself or by clicking the letter on screen (with the letter disappearing from view after). I felt it necessary to give the users of program the best possible experience with minimal effort (or room for error) so I created a "click-able" alphabet which removed letters "on-action". However, I noticed that there was a problem with the custom font and Labels containing them overlapped with the boundaries of other Labels, causing wrongly selected letters and difficulty for the user. To resolve this I used a CSS ":hover" styling (to a different colour) to make it clear to the user which letter is being selected. As an extra feature I decided to use mnemonic parsing to the alphabet so the player could make his selection from his keyboard instead of a mouse-click. I felt this gave less restriction to the player who may enjoy using a keyboard over a mouse.

When a player makes an illegal attempt or action, again, some variations of the game required windows to pop up, caused errors on the screen which halt the game play and cause frustration. Instead, I decided to add an information box which displays text information regarding your last action attempt. Again, I thought this was a less intrusive method of communicating to the user.

I thought it was a nice idea to include sound effects relating to each game event (correct guess, incorrect guess, game win, game loss, error). Some users who suffer from visual impairment may benefit from extra sound cues over visual

ones. Either way I believe it adds character to the experience. Most good games also have longevity, something which users find reward in. For this reason, I decided to include a statistics box where he/she can track their current progress and an ability to select between different difficulty levels with an appropriate point reward for each win at that level. Users can also save the current game state and load the previous save in with either a mouse click or by using F5 or F9 respectively, thus updating all the statistics to that current load point. Again, this gives the interface more flexibility for different user preferences.

It was important to me and my idea of a good user interface that I used visual cues for the users performance over text or console output. I included an area of the Pane dedicated to showing images referring to the current state of the game to achieve this. As an additional function I decided to implement a user help menu for game instructions and information. Should the user become confused by the mechanisms of the game or by the points system then there are resources available for guidance. Finally, I exported the project as a jar file so it can be ran with a single click by the user without using any command-line. This is important as the user doesn't need to understand any compiling or run commands.

4 Testing

I tested the application continuously throughout development using a TDD (Test Driven Development) methodology which I believed gave me better direction and allowed me to make changes without too much cost. This was achieved using JUnit tests combined with Hamcrest matchers, and using them to cover numerous possible situations. This essentially allowed me to create a robust application in which contained little to no bugs and being extremely user friendly.

For such a project, an equally important way for me to test the application for general usability is to let various users of different ages test it and provide me with constructive feedback. I provided the application to 4 individuals, all of which possess different personal interests, I.T experience and ages. The reason for this was that I could have a greater chance of correcting different types of poor interface characteristics, or game logic before release. I was given some feedback regarding font's, the type and the size and that it may not be suitable to an older person. I corrected this early on in the development phase. The ability to switch font is something I would introduce given more time.

I was also made aware of some fairly important game characteristics which I had overlooked. When the user completes the word or fails to complete the word, practically, it should be of interest to show the hidden word at this time. Something which I initially failed to add but later did.

5 Improvements

Given I had more time to complete this project, or that this was a project needing to be of a professional standard I would be looking to make the following revisions and additions:

- A help menu containing more detail, and structured in a way in which the user can quickly have access to what they want to find.
- The ability to toggle the sounds on and off. I can understand for some users the sounds may cause annoyance or distractions.
- The ability for the user to switch between screen sizes to suit the resolutions of their monitors. Although the current window is set to a standard size, having it larger may help those with visual impairments. Font sizes would also increase.
- Changeable fonts set to the users preference. This setting will also be saved inside the class.
- Introduction of a timer for the competitive side of the game, and for each round. This will give users incentives to play again.
- The ability to create multiple save points and then be able to select which load point you wish to use. This would use the FileChooser classes of Java.
- I would re-factor the code in areas to keep the code maintainable for future development or expansion. The use of loops when toggling label visibility would also be preferable (not sure if possible however).
- Explore other areas of GUI and HCI components and experiment with it. I may find there are better solutions to problems then those which I have chosen.