VoxSpell Final Project Report

SoftEng 206

Max Griffith
Software Engineering
University of Auckland
Auckland, New Zealand
Mgri334@aucklanduni.ac.nz

I. Summary

The VoxSpell application is a spelling aid aimed at educating younger children, from approximately 8 – 12 years of age, who are at an age where spelling is very important to master. The application features four exciting game modes that allow for new and creative ways to coax children to learn how to spell. Using a level based spelling system, which is not unfamiliar from a school child already, there is much progression to be had in the standard quiz. The review quiz targets the process of mastering those tricky words that a child has spelt incorrectly in the past. Then there is fun to be had with the one life and three lives game modes, where users must compete for high scores before they run out of lives from spelling incorrectly. Designing an application for children involved creating a bright and colourful GUI with lots of entertaining features so that they would not grow tired of it quickly with their short attention span.

II. Design

A. Choice of Programming Language and Packages Used

As an object-oriented language, Java specializes in representing a detailed application with many differing features, all varying in complexity. With very flexible features and relations between classes, it was relatively easy to design and implement the VoxSpell application using Java. Many classes could be used for multiple occasions, for example the "Enter Word Scene" used to submit the spelling of the current quiz word could be used for any 4 of the game modes. This meant it was easier to maintain if new features were to be implemented and was much simpler to comprehend as well for the programmer and any onlookers, as well as having the scene itself look familiar for user so the GUI doesn't appear too confusing and jumbled. In regards to building the GUI, JavaFX packages were solely used. This was done because of the simplicity of the JavaFX packages, ease of handling action events and switching between scenes seamlessly. On top of that JavaFX is much more customizable in terms of aesthetic appeal than Swing. Using CSS to reskin any buttons, labels or text with one CSS style file meant that the entire GUI could have a global visual style and at very little extra implementation cost.

B. Colour Consideration and Display Layout

For a target audience of children with creative, expressive and young the application needs to look colourful and appealing. Hence the warm red and orange hues for all the buttons and heading fonts. This colour resembles warmth and the white font that goes alongside it in the text within the buttons is clear and informative. As well as the sunny, natural background of a green landscape helps paint the picture of this desirable, safe place for children to relax and comfortably learn. The idea of safety is very significant as some kids can be bullied or feel down and VoxSpell is always there to try and lift their spirits.

C. Presentation of Information

Throughout the application, font colour, size and weight are all intended to be clear to understand while not appearing cluttered or confusing. This is so the user knows what they can do while running the application, regardless of which scene they are on. There is a theme of warm oranges, reds and yellows (summer colours) coloured fonts with a black stroke outline that makes it clear to read against the blue and green background. Another useful feature to inform the user is in the Main Menu scene, if the mouse hovers over one of the six Main Menu buttons a description appears in the bottom center giving help for what happens if that button is clicked. This feature helps the user know the features of each button.

III. Functionality

A. Motivation for Selected Functionality

With children's short attention spans, especially these days where iPads and smart phones are commonplace replacements for toys, kids get bored very quickly. VoxSpell aims to minimise boredom by implementing many different game modes and features that will keep kids entertained and learning for longer. The standard quiz is fascinating enough as it is due to that goal of succeeding in a quiz and unlocking a new level. The review quiz is there for kids to target their weaknesses and make them their strengths by testing on the words they've spelt incorrectly in the past. Finally, we have the one life and three lives game modes, where kids can compete for high scores.

This is since once a "live" quiz has been completed, the user can submit their score with their name so children can have competitions between themselves for who can get higher scores – adding another reason to replay VoxSpell, seeing how competitive kids can get.

B. Userability Decisions

I decided to maintain a standard quiz mode as the functionality was already implemented before it came to work on the project. Review mode was also incorporated however it needed slight adjustments to incorporate the GUI style. One Life and Three Lives game modes were added to appeal to the short attention span of children and adding a level of competitiveness to the application. The application is intended for classroom use and it would help spur more use if children were trying to beat each other in getting a higher score. The actual quiz interface is very intuitive and streamlined, designed to be fast for users to pick up and use. The user can submit a word by pressing enter once they have finished typing into the input text field and then continue onto the next scene using enter as well. This leads to a quick, responsive GUI that is intuitive for the user.

IV. Code Design and Development

A. Documentation of Software Design

As mentioned above in II.A, I believe Java was an appropriate programming language to use with this project due to its flexibility and maintainability as a language. I am also most competent with Java as a coding language which means I can get the most out of it than any other language. Other libraries used include JavaFX for all the GUI elements, ProcessBuilder for the festival calls and HashMap to store all the file information.

B. Development Process

From the start of the semester, all SoftEng206 projects were working towards the final project. A 4-month development process however most of the implementation that is seen in the final version is from only assignment 3 and the time between then and final submission.

For Assignment 3, working together with Fraser McIntosh was very beneficial as we mapped out the plan for the GUI before any code was implemented. We came up with the concept that there would be a top-level controller, AppModel, that would house the GUI and each separate scene to be displayed would be responsible for building itself and displaying itself by hitching onto the AppModel when it needed to display.

This paved the way for the entire development of assignment 3 and past that, as it is also how VoxSpell is implemented.

Using a journal and empty paper to map out the entire application helped to get a visual idea for how each class and scene related to each other. From there, it was necessary to assign every task/class to one of us two and to schedule in order of importance. AppModel and main menu scenes were implemented first and so on. GitHub was very useful at this stage in simultaneously coding implementation of the GUI and then being able to upload it to the online repository for the other person to update their own local repo.

Once the assignment 3 was submitted, it came time to go back to the journal and list off more features to be implemented for the final version. Afterwards, plan how they could be incorporated into the existing GUI and what modifications needed to be made for the final submission. From there it was just a matter of scheduling again and implementing. I used GitHub once again, by myself this time, in case something went wrong and I had a backup or I wanted to refer to past implementations.

C. Innovation in Implementation

The AppModel and the numerous scenes that build and display themselves is a foolproof system that allows for efficient maintainability, ease of implementation and flexibility. If anything is to happen to the AppModel window, it is already reflected within all of the built scenes that already have a link to the AppModel.

Many scenes are reused for multiple features. As mentioned above in II.A, EnterWordScene is used for all four game modes where the title is changed depending on mode and if lives are used, they are displayed as well.

For statistics, an ingenious implementation is used by reusing the ScoreScene for all four game mode statistics as well:

Statistics are done for Standard Quiz by reading the stats from a file and generating word statistics per word, the word itself, and the times it's been spelt correctly or incorrectly.

For the Lives game modes, it generates the word statistics by reading from a file the name of the person and their score. It then overrides the implementation of the word statistic and parses the user name as the word, the score as the times spelt correctly and the lives used as the times spelt incorrectly. It can then display this in the statistics scene as if it was a standard quiz using word statistics but it is using alternative word statistics for the lives mode.

V. Evaluation and Testing

A. Evaluation and Testing by Ones Self

My time spent evaluating my peers' submissions was well spent. It was intriguing seeing how other people had approached the same project and subconsciously comparing my own to theirs. Providing feedback to their projects also helped me to consider the things I was analyzing in my own project when I went back to it.

B. Results of Evaluation and Testing by Peers

Peer evaluations were mostly helpful, one was not particularly as it gave me next to no information to go on. Main issues regarding my own beta project were to allow

the pressing of the enter button to submit a word in the EnterWordScene and to fix font that was messing up with the background. These fixes were quite easy to rectify and made sense to allow user to use the GUI with more clarity and speed.

Other features that were commented on was the lack of a file choosing option, to select custom lists which was implemented as well.

The last thing to mention is that I was previously planning on implementing another game mode called Time Attack. Comments were on the lack of a timer with that game mode which is true, I however opted to switch that mode out with review mode (which I had previously got rid of) as it has more emphasis on learning how to spell as we desire.

VI. Conclusion

Working on VoxSpell has taught me a lot on the effectiveness of planning a project before any implementation takes place. As well as the importance of journals and version control in maintaining a healthy, software, work environment that enables easy managing of a software project. I have also learnt the difference a creative, colourful GUI makes on appealing to the user and how to implement such a thing with good maintainability and flexibility.

Aiming for the target audience of young children aged 8-12, the VoxSpell application succeeds in engaging users with a bright, interesting and colourful style of GUI. With 4 varying Quiz Modes, children will be able to spend lots of time learning how to spell without every growing bored. On top of learning how to spell new words and mastering old ones, users will also be able to compete with friends for the highest score in the One Life and Three Lives game mode. VoxSpell is highly rewarding and hopes to have a long, successful lifetime as a spelling aid application.

VII. References (For media in Code)

Audio Reward

Games, B. (2013, June 28). *Nameless: the Hackers RPG Soundtrack*. Retrieved from Free Music Archive: http://freemusicarchive.org/music/BoxCat_Games/Nameless_the_Hackers_RPG_Soundtrack/BoxCat_Games_-_Nameless-_the_Hackers_RPG_Soundtrack_-_10_Epic_Song

Background Image

Larisa-K. (2013, July 19). *Road-Field-Summer-Trees.* Retrieved from PixaBay: https://pixabay.com/en/road-field-summer-design-trees-163551/

Lives image

OpenClipart-authors. (2013, October 18). *heart-love-red-valentine*. Retrieved from PixaBay: https://pixabay.com/en/heart-love-red-valentine-romantic-157895/