VOXSpell Project Tecnical Report

Michael Kemp
Software Engineering
Department of Electrical and Computer Engineering
University of Auckland
Auckland, New Zealand 1010
Email: mkem114@aucklanduni.ac.nz

***Abstract*—VOXSpell is a spelling quiz game developed for computers which uses speech synthesis to convey the word and is specifically targeted at children between the ages of 7-10. This influenced the design of interface, the supporting material (user manual and readme file) and the game mechanics. The interface keeps kids engaged by being easier to use as they have less motor control. The supporting material; specifically the user manual does away with large words and complex sentences so the kids can read it themselves. The mechanics engages kids in learning through gamification with points systems.**
Some of it's inovative features were; homophone detection, quiz word smart selection and game extension. Homophone detection is the automatic exclusion of any words that the user might be quizzed that could sound the same as other words which can lead to quite a frustrating experience when the only to get the word to spell from the game is through speech synthesis. I wanted to ensure kids were being thoroughly tested and tailored for; any words that weren't yet attempted were more likely to be in quizzes, likewise with words that have been incorrectly spelt more times. Kids like to interact with what they're given and fiddle so I tried to make resources as extendable and swapable as possible without compromising the userability and eduction.

I. General User Interface Design

A. Choice of Packages

1) Java: Being very highly used at the time of development means that it is highly documented by the Java developer community and there are many GUI frameworks to choose from.
Cross-platform mobility makes Java a good candidate especially with Android's primary developer language being it, thus leaving the door open for a future mobile application. All that would be needed is for a mobile frontend with modified controllers.
Java is intended to be cross-platform which leads to developers of frameworks and libraries trying to retain that; this makes for developing across all PCs much easier.

2) JavaFX: Gluon scenebuilder is a drag'n'drop GUI builder with a high level of refinement which speeds up development and reduces errors. It also presents options during creation leading to more new ideas.
Swing (the predecessor) will run out of support sooner which means the application's life is extended by being created with JavaFx. It also brings new features out they may be very useful during development.
JavaFX when correctly done enforces the MVC pattern because all view information is put in the ".fxml" files and the controller classses are auto-generated thus increasing the maintainability and flexibility of the interface.

3) Festival: The client uses a version of Ubuntu as their operating system and Festival is easy to install and maintain with Ubuntu as it's probably the most widely used speech synthesis program there is on Linux.

4) Iconic: Iconic is open source so there is no legal issues in using it.
It is aesthetically pleasing while being simple, clear and inoffensive.
It is comprhensive and generic so there is little concern of not having an appropriate icon.

B. Colours and Layout

Subsubsection text here.
ueaob

C. Information Presentation

ee
ee

D. Other Challenges

ee
ee

II. Functionality

A. Motivation

B. Userability Decisions

III. Code Design and Development

A. Software Design

java??
other libraries

B. Development Process

C. Innovation

D. Shortcut keys

motivations
implementation

E. Other Challenges

IV. Evaluation and Testing

A. Individual Results

B. Peer Results

- Background music button: A mute button was added to the main menu and it no longer plays on anything else to avoid clashing.
- Back button in quizzes: Added but saves progress thus far as the user did attempt to spell those words.
- Explanation of statistics: Included in the user manual.
- Voice change while playing: Decided against due to cluttering the UI.
- Plain main menu: Intentional to avoid UI clutter and easy navigation for children due to their lesser motor skills.
- Custom word list: Added in the ”Custom” mode
- Level difficulty feedback: Done in a different way with the words in it being previewed (with speech) so the user knows the difficulty before hand.
- Start button for quiz: Not implemented as user can replay the word at no penalty so it would only serve to annoy power users.
- Repeating voice message: This is a feature of the player pressing buttons too many times and will stop, kids will find this comical so it was decided not to remove.
- All words shown in stats: Intentional to keep UI consistency.
- New method of displaying stats: Implemented stats as a table which is a much more natural way of reading.
- Disabling submit button: Not implemented to show to markers that my UI doesn’t freeze and if the player is experienced enough they might know the word from the beginning sound.
- Quiet music: Intentionally not loud so as not annoy caregivers and it is background not foreground music.
- Hidden savegame file: Implemented but not fully so power users can do things like back up savegames while avoiding novice users from accidentally deleting.
- Non-alphabet characters: Implemented, non-alphabet characters stop the player from submitting and children have less motor control than adults.
- Reset method: Intentionally this way as it restores the game to as if it is started for the first time.
- Review-mode bug: Not applicable as the mode was removed.
- Voices not working bug: Not reproduceable, it was tested on UG4 with all three voices.
- Music custimsation: Implemented, if the readme or user manual is read.
- More gamification: Implemented, a new experience system was introduced.
- File searching: Implemented under ”Custom” mode.

V. Future Work

VI. Conclusion

The conclusion goes here.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.