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

The colour scheme was based on a set of calming green hues because it's a naturally calming colour which keeps kids focused on the game and avoids built up frustration with the game. Green is also neutral or positive in almost all cultures. [1]. The Layout was intentionally kept minimal to avoid confusing and frustrating children who have a shorter temper than adults, I also tried to follow design conventions such as grouping related items together for the same reason.
In addition the smallest font size is 14pt so that the children can read everything without having to strain as they are still learning to read.

C. Information Presentation

Icons went allong with text as much as possible to present the options to the player both ways so whether they are a textual or visual persion they will be able to navigate with ease and they allowed for grouping of similar buttons to convey meaning quicker.
Extensive tooltips were implemented instead of a dedicated help function as a traditional help function requires reading and relating to the program which is okay for adults but for younger kids who have a shorter attention span it's impossible. It's also more intiutive for a user to let their mouse hover over something while they figure out what it is. Tooltips give a much stronger connection to the function of UI items than a manual but the manual was also included for things that can't possibly be included in tooltips.
The spelling of a failed word was implemented because

allowed kids to associate the auditory information with the textual.
UI Items intentionally had text to further exercise the reading abilities of the children and thus helping their spelling.

D. Other Challenges

An insignificant portion of the population is red-green colour blind [2] so a design decision was made to keep the UI mostly all green so that colour blind kids would not be affected. This required the right selection of greens to keep the UI interesting and readable.

II. Functionality

A. Motivation

1) Words to be quizzed were picked on what the player needed to work on the most so that they wouldn't miss out on words while they progress through the game and to target their weak words insteading hiding behind their strong words.
2) Words that sound the same are not quizzed to the user to avoid confusion and frustration, this is a limitation of using speech to deliver the word to spell.
3) Game extension primarialy through adding more words provided by the user allows them to focus on what they need to that the game no longer targets. The default list is based on the New Zealand Goverment's education department for that age group but avid readers or players would soon outgrow it.

B. Userability Decisions

1) The user doesn't know any better unless they read the user manual, it is entirely hidden. The only issue would be if it ended up repeating the same words forever but that is avoided by weighting in words less attempted.
2) This is presented to the user in the manual and readme if they wish to alter it so they can be tested on those words if they wish to be but with it being hidden in the "VOXSpell" folder a novice user needn't be concerned about it.
3) The decision to a new wordlist incorporate with the existing list was so that it wouldn't have to be reloaded on every start up of the game and also allows for all the features available to the default wordlist such as stats and level progression. This was trading off with robustness as it's hard to control what the user adds; to combat this instruction was given in the user manual.

III. Code Design and Development

A. Software Design

1) Language Choice: Java may not have been the best language as some others allow greater cross platform abilities by using web technology and it is not as good as them. However the developers already knew Java so development cost was saved. It also offers an extensive set of inbuilt frameworks and libraries such as JavaFX, MediaPlayer and ProcessBuilder which saves time hunting libraries and normally guarantees they are maintained and of high quality.

2) Libraries Used: Festival is the primary library used not included in Java and it was picked because there it's the most cost-effective solution for Linux as it's free. Mac OSX's in built "say" command is much nicer; it manages to pronounce every word and is much easier to understand because it sounds nicer.

B. Development Process

The development process used was XP (eXtreme Programming) but a modified form for a development team of one. The schedule was semi-relaxed to meet with the Client's times. It was quite effective as the clients further refined their specificaiton as time went on and weekly meetings helped nurture this. It was also an issue because it caused unpredictable workload due to it being dependant on what the clients felt like.

C. Innovation

The primary implementation innovation was that of a ResourceLoader class. Java requires a resources to be in a subdirectory of a class to be accessible which leads to messy packaging. By having resources in a seperate package mess was organised and with ResourceLoader implementing the singleton pattern it made resources accessible to any class which needed to load it. Duplicate code also fell as the code to get a resource only needed to be in ResourceLoader.

D. Shortcut keys

Using the arrow keys and space to navigate the UI was implemented because kids have better control with a keyboard and because expert users will prefer it due to being faster. Space instead of enter as a key is much larger and allows the hand to stay on the home row of the keyboard better. The arrow keys engage the kids on a game level. Not much work needed to be done and JavaFX did most of the work.

IV. Evaluation and Testing

A. Individual Results

Unit tests were written for particularly hard or complex classes and methods but for the most part integration revealed the large issues and was quite time effective as the application is quite small with one developer who knows the entire codebase. The mindset of a child was adopted because they think outside of the box and push the boundaries of what they are suppposed to do which is very effective for finding issues and it covers most of the things they being the target audience will do. A lot of evaluation was based on the developer's experience in the field of gaming and from there what would make a fun game was tested and implemented.

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

If any future work is to be done the focal point should be on homophones as excluding them from the word list doesn't give the player the chance to learn the word through the game. Possible solutions include giving a dictionary definition which can be a problem for words with more than one definition or ones that have a poor definition. Another possible solution is to retrieve sentences using the word but once again curating for a good sentence is difficult with expandable word lists. The user could provide the definitions or sentences but the time involved would likely deter them. Probably the best option would be to get the speech synthesis to say the word in a specific way that doesn't sound the same but this also has the issue of english accents and quality of computer speakers.

VI. Conclusion

VOXSpell has innovative features which engage the target audience and is well polished but has a major flaw in curating user generated content and homophone detection. This further work is outside the scope of a single developer team but if VOXSpell is successful then this could be investigated.

The specifications of the clients were met but they were vague and customer acceptance testing is yet to be conducted.

References

[1] J. Scott-Kemmis. (2016, 09 28). The Color Green [Online]. Available: http://www.empower-yourself-with-color-psychology.com/color-green.html

[2] Colour Blind Awareness. (2016, 10 23). Types of Colour Blindness [Online]. Available: http://www.colourblindawareness.org/colour-blindness/types-of-colour-blindness/