VOXSPELL Spelling Aid for Second Language Learners

Mohan Cao
Department of Software Engineering
University of Auckland
Auckland, New Zealand
mcao024@aucklanduni.ac.nz

***Abstract*— the use of games and text to speech synthesis assists in the improvement of spelling for second language users. By ensuring the proper design, the user is provided the optimal experience to learn words at their own pace.**

Keywords—text-to-speech, spelling aid, second language, ESOL, education

I. Introduction

The use of spelling aids in the past have been solely for educational use. With the popularity of video games and the possibility of educational games, VOXSPELL aims to provide a professional, educational approach along with the entertainment focus of games to improve English spelling among adult second language learners.

We use our own motif in the spelling aid VOXSPELL by implementing an assortment of user-tailored features to improve spelling habits and change the perception of spelling as a stressful task to that of an enjoyable activity.

II. Intended Audience

The audience is targeted towards second-language learners aged 18-25 years old. The purpose is to provide an intuitively easy, stress free environment for the users to learn English at their own pace. While the application is in English, the menus are simple and easy to navigate.

For adult learners, features and aesthetic appeal matter more than the bright colours and images in children's' spelling games, and the high contrast readability in elderly spelling aids. We set to improve upon existing spelling aids with our own VOXSPELL Blue motif, focusing on intuitive design and ease of use rather than heavyweight functionality. Also, the application focuses on accuracy of spelling over the speed to cover different styles of learning that users may have.

This application may also be used by native English learners of the same age group who are looking to improve their own spelling, as the application does not distinguish between different age groups, only the difficulty of words being spelt.

III. Implementation And Dev Process

We chose to use a Java-based framework for our spelling aid, as it easily integrated with the free Festival text-to-speech synthesiser. It was also the most suitable language to use, as it is freely available across all Linux distributions on which Festival runs.

The software development process model used was primarily eXtreme Programming. By fulfilling user stories and constantly changing to meet client needs, the full specification of the final product could be met.

The instabilities associated with the use of agile processes such as XP have surfaced in the code and best practices used in this project. There was increasing difficulty in maintenance and addition of features as the project scope changed with each iteration, although it was also due to the lack of test-driven development that was much needed for such a project.

However, overall more time was dedicated towards coding new features rather than fixing bugs in code, so the process can be considered a success.

IV. General Features

The spelling game utilises multiple scoring types for the tracking of different types of statistics.

The main game itself uses a "high-score" system where points are allocated to the spelling of the words. It is necessary to obtain as many points as possible as part of the game. Naturally, points are given to correct words, and none are given to incorrect words. This system allows for rewards to be more justified, as you can obtain a good score for the right number of words that rewards the players simply playing the game. It is feasible to have no penalty for poor performance in the spelling game as to meet our objective of providing a stress-free environment for learning.

The statistics are stored using the mastery system using mastered, faulted, and failed words. Individual words can have mastery points, so a single word can both have mastered and failed points, which contribute towards the statistics. Intuitively, this represents the individual statistics for the completion of each word in the quiz, and is easily represented in the statistics menu in each type for each word.

- Mastered words are previously spelt words that have been completed in the first try.
- Faulted words are words that have been completed in the second try (i.e. the word was incorrect the first time, then correct the second time).
- Failed words are words that have not been completed (i.e. incorrect twice).
- Forfeited words are words that are incomplete as a result of quitting the spelling quiz while it is in progress. It counts as a failed word in all respects.

A. *Colour scheme*

A 'cool blue' colour scheme was used to emphasise the peaceful and easy-going aspect of the spelling quiz. The visual cues of a blue colour scheme are almost universally peaceful, therefore we have chosen this colour to represent the VOXSPELL application.

The majority of the GUI is in a low saturation, but varying brightness of colour. The high contrast allows items to be easily distinguished by colour-blind users and those with visual impairments. No support for screen-readers are available at this time, but future designs may include this (see Future Changes)

B. *Resizability*

In general, the application is resizable, and efforts made to ensure that there is scaling of GUI features when resizing. The application is ideal when windowed full-screen, i.e. maximised, allowing users to focus on the application instead of distractions elsewhere.

C. *Auto-saving*

Additionally, VOXSPELL automatically saves the statistics file when the application closes. Session data is merged with global data and the serialized statistics object written to an appropriate statistics file (corresponding to the name of the list).

V. First-time run menu

The run-once menu opens upon first ever use of the application. The user is given the choice to unlock up to any of the given number of levels in the default list. This feature provides some ease of use in giving a chance to unlock up to whatever level they feel confident in. If they have previously played this game elsewhere and wish to continue their progress, they can simply unlock to the level they wish.

However, after the initial run, the user cannot change their mind without tampering with the statistics files. This prevents a cheating mind set, which ruins the aspect of the game. Nevertheless, we cannot fully prevent cheating with the initial unlocking of the levels, as if the users are knowledgeable enough, can simply delete the statistics files and start over.

VI. Main menu

The main menu is used for navigation for the spelling aid. There is a main menu, in accordance with the video game industry standard of a centralised menu for every main feature

Buttons were styled blue-white, and when hovered over, change colour to blue to indicate the mouse 'hovering' status. When pressing down the mouse over a button, it turns darker blue to indicate selection. This gives visual feedback over a choice made.

There is a help button in the top right hand corner, which when clicked, opens up a new window with the README. This is convenient when the user wants to open up help without actually navigating through folders to reach the README file.

Selection of various options is easy:

1) New Quiz
Opens up the level selection menu for the new quiz mode.

2) Review
Opens up the level selection menu for the review mode. Identical in appearance to that of new quiz except some slight functionality changes.

3) Statistics
Opens up the statistics menu for historic statistics and statistics for the current session (unsaved).

4) Settings
Opens up the settings menu where you can tweak different aspects of the application and game

B. *Dynamic background*

There is a dynamic background feature which utilises an online open-source API (unsplash.it) to display scenic, natural backgrounds to fit the theme of a relaxing spelling quiz.

This fades in when loaded to give the illusion of having preloaded images, overall ensuring a smoother user experience.

Care has been taken to ensure that any scene will work with the colour scheme, even those of different colours. A blue overlay filter has been applied over the background image to colourise the background image.

The interface was purely web-based, so dynamic loading (on-the-fly) with multithreading was done to minimise GUI lag.

C. *Background music*

Thematic background music accompanies the main menu screen allowing the user to feel more relaxed and comfortable doing spelling, which is considered a stressful task for second language learners. The users are given the option to toggle the background music in the settings menu, in the case that they feel better focused without it.

The theme of the music is "village".

VII. Statistics menu

The statistics menu displays the statistics of the previously spelt words.

A bar chart comparison of the mastered words to the faulted and failed words is displayed on the left of a resizable split view and detailed statistics on the right (WYSIWYG editor style).

The bar chart is for a general summary of what statistics the user obtained in current and previous sessions and a useful measure of how they're progressing in terms of progress through a spelling list. The bar chart also resizes upon resizing the split window.

A simple but effective colour scheme was used in the bar chart. Red represents the failed words, orange represents the faulted words, and green represents the mastered words.

The style of chart was chosen to allow the user to quantitatively analyse their statistics side by side. The bars show progress in improving spelling; as mastered words increase, the green bar dwarfs that of the others, the chart itself being a visual reward for learners.

On the right of the split window, a table of all words completed in the current session or previously in the past are displayed by level, word and mastery (mastered, faulted, failed, and percentage mastery). This detailed statistics panel allows users to better gauge their ability to spell certain words.

The table is filterable by level and incomplete word. An example is filtering by level 1 words only by typing **"1"** and filtering by words containing **"ate"** by typing it into the toolbar textbox. This becomes particularly useful for searching up a particular word that the user wants to improve.

In standardising our GUI, buttons for navigating back to the main menu, as well as clearing stats are available on the toolbar. There is a dropdown list allowing selection of either "global" all-time statistics, or "session" statistics tracking the unsaved statistics from the current session.

VIII. Level selection menu

The level select menu allows you to choose the level of words in the spelling list loaded. There are 11 dynamically generated selection buttons corresponding to 11 levels in the default NZ Curriculum spelling list. These buttons, when clicked, will take you to the quiz game.

When placing your mouse within the boundaries of the button, there are summarised scores for each individual level displayed above.

There is a button in the toolbar at the top which leads back to the main menu, in case the user wants to go to the settings to quickly change a setting.

We decided not to have a background image on this menu because of poor contrast between the buttons and the background; this is visually stressful for the user.

A. Background music

Thematic background music accompanies the menu. It was selected to fit the mood of the selection, being slightly suspenseful and darker than the main menu background music. The purpose of the music is to focus the users' thoughts before the actual game and hopefully improving their performance.

As with the main menu music, it can be toggled on and off in the settings.

The theme of the music is "desert".

IX. Settings menu

The settings menu contains drop-down selectable lists (combobox) and toggleable buttons for configuration of the game and application settings. Drop-down lists were selected for their simplicity and ease of use, and toggleable buttons were used as feature 'switches'.

The first setting, "spelling list", allows you to select a custom formatted spelling word list (in the NZCER word list format), and load it into the system, which will by default only unlock level 1 when loaded. This is to allow the users to actually experience the list before sending it to others.

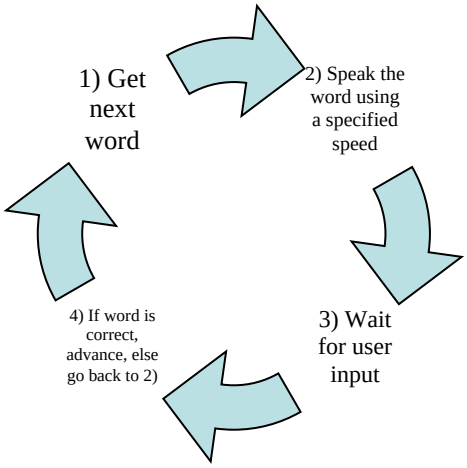
A file chooser is opened when the "custom word list" option is selected. It accepts text files utilising the "%Level #" format. No support for other types of file is guaranteed in order to standardise the format of the spelling list for future extensibility.

The second setting, "festival voice", will allow you to select a preferred voice from all Festival text-to-speech voices installed on your system. As of right now, the voices are displayed using their simple names (akl_nz_jdt_diphone, etc.) due to limitations in Festival voice synthesiser for Java.

The last setting, "background music (on/off)" is a toggleable button which allows you to turn on and off background music (that of the main menu and level selection menu). It does not turn off speech audio, as that is required for the quiz, and audio rewards at the end of the level, as it is not background music.

X. GAME

The game is a multiword spelling quiz, similar in format to the spelling bee tournaments held at schools. The test is carried out sequentially, with each word being said, the user inputting an answer, checking for correctness, repeating the word if necessary then advancing the test. Below is a cycle showing the necessary implementation of the game:



As it is a very demanding task, background music is cut from this mode to avoid distractions, and to prevent the text-to-speech system from being misheard.

The words are located from the spelling list file specified in the settings file, and compiled into a ten word spelling list consisting of random words. The game will assume the spelling list is always correct. It load words into the list as-is, so if invalid characters (numbers and non-English symbols) are present in the words, they will cause the spelling list to fail silently (they cannot be spelt correctly).

The system will not differentiate between upper and lowercase entries of words.

e.g. **WaTERmeLON** and **watermelon** are the same

The game automatically starts when a level is selected. The spoken prompt "Please spell the spoken words…" is given, then the first word spoken out loud at a regular speed. It is then possible to type the attempted word into the text box in the lower section of the screen.

The textbox at the bottom of the screen occupies a proportion large enough for the user to clearly see what word they have typed in. There is a tooltip displayed if the text they entered does not meet the validation requirements (given in the user manual), such as being too long or not entering anything.

All feedback text is displayed in the centre, to convey proper feedback to the user.

The words in the spelling list are never revealed to the user if they have not completed or forfeited the words previously. Thus, it is justified that the user is not permitted to exit a currently running game, as to prevent revealing words during a level. If they do forcefully wish to close the game (after a dialog confirming it), then they concede their high-score points and the current word is marked as incorrect. Words done before the cancelled word are unaffected as they are technically "completed".

After a game is finished, options given are to return to the main menu, remain on the same level, or to safely close the application.

A. Shortcuts

Pressing enter in the textbox allows you to submit the word without manually pressing the confirm button. This is much simpler to do than reach over to the mouse, which was the primary reason for adding in such a feature.

B. Review mode

Review mode is a selectable mode from the main menu. It selects from all previously incorrect words and allows the user to convert their faulted/failed words into mastered words. For each word reviewed and mastered, all failed/faulted statistics are erased and 1 mastered point added to the word.

The user cannot start a review mode for a level that contains no incorrect words, which results in an alert dialog box popping up to remind the user.

C. Points system

Allocation of points interlinks the two systems when trying a new spelling quiz. In the game, you can obtain points which, if beaten, give you a personal best (your highest score), and the maximum total score you can obtain from the level.

The number of points reflects the total achievement of each completed word and follows intuitiveness of design. Using base points of 1000 is a smart choice for a points system as it is a round number and is sufficiently great to feel 'rewarding' when achieved.

The other scores are based off the base score, with a faulted word being $1/5^{th}$ the base, and a failed word not incrementing the score counter. In review, a word being faulted twice is worth half that of a word faulted once.

There are additional rewards with the **word streak** system. After obtaining mastery of 2 or more words consecutively, additional 500 points are added per word. This allows for more accurate representation of points (a user should be aiming for as few faults as possible, with the majority of words spelt perfectly on the first try). Hence, if a user gets a word incorrect halfway through the game, it should be accounted for in the final calculations.

Shown below: a table relating different scoring systems

Table 1 Allocation of points

Type	**Points**
Mastered	1000 + 500 per word streak bonus
Faulted	200
Faulted twice (review only)	100
Failed	0

D. Spoken examples

Present is an option to speak out the words in the list. The definitions are fetched from an online source, the Oxford online API. The interface is RESTful, which uses JSON-based data. A specific design decision was to use Maven with JEE to obtain the plugin library minimal-json for use to parse the data object.

E. Repetition of spoken words

Being able to listen to repeated instances of the word is useful in many cases that the text-to-speech synthesiser may speak a word too quietly or too quickly. The button with a cycled arrow (by intuition, "repeat") when pressed repeats the word in the sentence "The word is … [word]." This feature supports the learning of the user by avoiding frustration over not hearing the words correctly the first time.

The button with a music speaker symbol (slightly less intuitive, with a tooltip saying "Changes TTS voice") represents the "change voice" button in the game mode. This allows for the Festival text-to-speech voices to be changed, swapping out the current voice for another. The button cycles through all possible Festival voices, with a speech sample given

e.g. "Now using New Zealand voice" – akl_nz_jdt_diphone

"Now using British voice" – rab_diphone

"Now using American voice" – kal_diphone

"Now using cmu lex etc" – undefined alternate voice

XI. Rewards and Level advancement

Level unlocking and rewards are provided at the end of each normal quiz level (not review mode), given that the user passes the requirement to advance to the next level. The requirement is 9 out of 10 words mastered or above 48% of the maximum points attainable on the level.

48% of the maximum points was obtained to be a reasonable number, as the points increase quadratically as you get a streak. 48% is the minimum number of points obtained by getting 1 word incorrect out of 10 words in the middle of the quiz.

A. In-game audio rewards

A ding sound is played back to users when words are spelt correctly, regardless of mastery. This reinforces the reward of spelling a word correctly.

When the level ends, a "ta-da" synthesised sound clip is played back to the user. The audio track is in harmony with the ding sound, which was qualitatively determined to fit the needs of the software. The volume was tuned to avoid being aggressively loud, and should not drown out the text-to-speech synthesiser.

B. Dynamic rewards

The reward itself is partly audio and partly an optional graphic reward of a cat image, which may or may not be animated. This uses a free Mashape API, which allows images of cats/kittens to be fetched from anywhere on the internet. There is no quality assurance made towards checking if the images are suitable for reward at the current time, but all images should be friendly towards the user.

Occasionally, it allows you to find your own animated graphics to search using the website Giphy.

These features allow for a brief break between games, allowing the user to take their minds off spelling. While it is slightly unorthodox to have a cat image as a reward, users are not persuaded to click the rewards button for any gain, allowing users who dislike cats to still enjoy the game.

At the end of all the levels in a wordlist, the user is shown a victory screen, where the game congratulates the user for mastering every level in the list. The user still has the option of going back through the levels at any time; this is persistent across multiple sessions. It was considered at one time to not have a victory screen, but it was more difficult to implement with the system, and it was better to allow the user to feel satisfied with their achievement.

XII. Other features

The software system self-diagnoses and corrects itself when a system anomaly is detected.

Outdated statistics and settings files (as a result of incremental versioning) are automatically cleared and upgraded to the newest version. This hides some internal representations that may be exposed as a result of incorrect configuration.

XIII. Evaluations of the software

A. Self evaluation

The code quality of the project was overall good, with improvements to the overall structure of the project needed (possibly an overhaul) to better utilise JavaFX design principles such as using Binders (→observer pattern) in the game without having to recreate the same functionality manually.

Some improvements to the GUI needs to be done as layout is sometimes error-prone (in very strange edge-cases). Most issues have been fixed in the development process, however as we did not use test-driven development, absolute certainty of component quality is not assured.

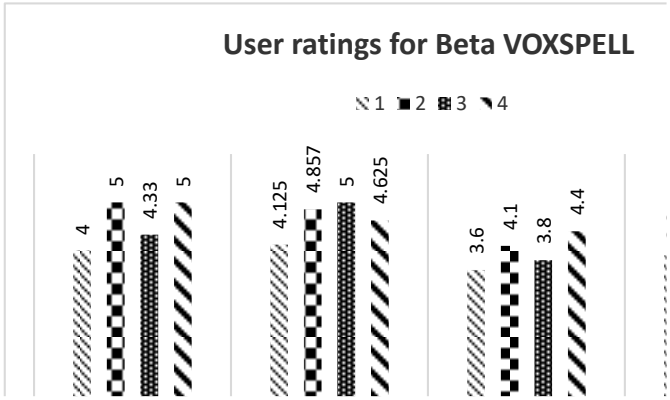
Although debug logging was used, debugging was difficult and more test-driven development would have been needed at the start of the project to manage the complexity of the project; use of newer JavaFX features such as the testing Robot in TestFX would encourage a more robust GUI and help maintain quality code.

Ideally, more transitions and sliding side panels would have resulted in a sleeker, more modern interface. We added transitions to backgrounds to mock pre-loading of backgrounds, and attempts at adding transitions to the game failed due to concurrency issues.

B. User evaluations

Sampling from 4 random evaluations from beta users among a cohort, we obtained reasonable evidence that user satisfaction was generally very high. Data was obtained from users that were familiar enough with Java to evaluate the beta source code, as well as the functionality of the program.

Below is a summary of the results obtained



The code quality review was consistently high, scoring above 4 from all sources. In our code, this was reflected by the use of appropriate design patterns and package hierarchy, as well as management through build tools.

We received overwhelmingly positive comments such as the code being "well documented" and "the Javadoc comments" providing "information about the desired abstraction a class has" (2016). This is ideal as we wish in future to allow the game to be extensible by other players in the form of modifying the game by using the base game as a framework. We continued to use the existing code from the beta, but with mind to the maintenance of the code in the future.

There were some good comments about functionality, with the functionality also being one of the driving points of the VOXSPELL application. However there were some concerns about the voice overlapping in the review mode; this was not caught in the normal development process and could not be replicated, therefore the bug report is marked as "won't fix". We did, however, fix a major bug crashing the application when a certain operation was carried out.

We received lower marks for the interface design as being rather 'boring', in which the target audience and reasons for doing so (to meet design goals) were unfortunately not communicated to the testers during the beta phase. Eventually, it came to adding some drop-shadows to give the GUI more substance, but the overall simple colour scheme was untouched as part of the design goals. We feel as though this may be a weak point in the overall design, and as such, have added it to Future Changes to identify a focus in the future.

XIV. Future changes

There are still some very minor bugs with the layout that are occasionally present with use of the application.

The software marketability could be improved through adding support for screen-readers (accessibility functions), and navigation cues.

Additionally, features observed in competitors, such as a login system, level editor, 'experience points', and additional rewards such as soundtracks and downloadable backgrounds could be implemented with respect to the overall theme in the future given enough time.

Having a more diverse range of colours in the application colour palette may assist in delivery of content that is bland and uninteresting. By having a contrasting colour set, excitement through differing colours may be achieved. This is a goal for future iterations and may need considerable change to implement.

A point of improvement in the future would be to carry out small iterations of useful features through test-driven development. This would ensure faster, more rapid releases of software that is feature-rich and thoroughly tested.

XV. Conclusions

It was determined that the software was ultimately successful in providing spelling aid to secondary learners.

Using Java and JavaFX, we have created a spelling aid tool that successfully utilises the free text-to-speech synthesiser Festival, and provides high quality spelling education to the 18-25 year old second language learner audience. By using intuitive design and the necessary features for a spelling aid, we created a spelling system that can have extensible support in the future, whilst having innovative features such as a spelling examples feature, scenic backgrounds, and individual filterable spelling statistics.

Implementation of a concrete software development process such as XP allowed rapid software development, at the cost of specification and maintenance costs in the future, with suggestions that test-driven development may solve issues relating to the overall software quality.