# Voxspell - Game Based Spelling Learning

En-Yu Mike Lee
Department of Electrical and Computer Engineering
University of Auckland
elee353@aucklanduni.ac.nz

***Abstract*****—** Game-based learning encourages learning of a certain subject or acquiring certain skills by means of playing. Educational games must fulfil both recreational and didactic goals in order to succeed. This paper introduces a spelling game called Voxspell, designed especially for ESL students to learn English orthography and vocabulary. A brief description of the game is provided, along with the software development process applied.

***Keywords***—Game-based learning, orthography, spelling, vocabulary, English learning, software development process

## I. Introduction

Game-based learning is an attractive alternative option particularly for a generation who has grown up digital. The success of game-based learning experiences relies on two key characteristics: an effective pedagogical background and a sound entertaining support. Learning effectiveness of the game depends directly on the former, but the latter also deeply affects student's motivation.

In this paper, an application for game-based orthography learning aimed at second language learners between eighteen and twenty five years old is presented. Traditional spelling exercises tend to be monotonous and tedious, which represents a critical handicap for maintaining the student's interest and motivation. Memorising the English orthography also raises similar problems.

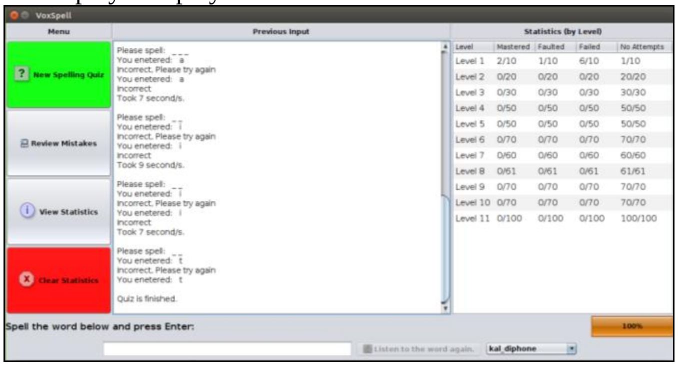
The main advantage of game-based learning is to maintain student's motivation by its recreational characteristics. It allows the development of repetitive learning tasks. Thus, it constitutes an ideal scenario for practicing orthography and vocabulary learning, by completing simple spelling exercises of incrementing difficulty.

The Extreme Programming (XP) technique is applied throughout the development of the program to ensure the quality is satisfying and to adopt changing requirements. This includes techniques such as client meeting with the lectures, peer evaluation, and pair programming.

## II. Description of the Program

The objective of any educational games is achieving a didactic goal while maintaining the entertaining value expected. When designing this application, the easiest way of ensuring its entertaining objective was to adapt an existing amusing game and a media award feature. Nevertheless, the didactic factor had to be carefully introduced to integrate both the didactic and the recreational goals without jeopardising each other.

The objective of Voxspell consists of spelling a maximum of ten words after pronunciation and receiving a number of entertaining awards. Once each answer is submitted, the correctness of spelling is checked and the result recorded. If a satisfactory level of correct spelling is achieved, a number of awards are provided such as a Tic-Tac-Toe game and a media player to play music and videos.



```
Please spell:  ___
You enetered:  a
Incorrect, Please try again
You enetered:  a
Incorrect
Took 7 second/s.

Please spell:  _ _
You enetered:  i
Incorrect, Please try again
You enetered:  i
Incorrect
Took 9 second/s.

Please spell:  _ _
You enetered:  i
Incorrect, Please try again
You enetered:  i
Incorrect
Took 7 second/s.

Please spell:  _ _
You enetered:  t
Incorrect, Please try again
You enetered:  t

Quiz is finished.
```

| Level | Mastered | Faulted | Failed | No Attempts |
|---|---|---|---|---|
| Level 1 | 2/10 | 1/10 | 6/10 | 1/10 |
| Level 2 | 0/20 | 0/20 | 0/20 | 20/20 |
| Level 3 | 0/30 | 0/30 | 0/30 | 30/30 |
| Level 4 | 0/50 | 0/50 | 0/50 | 50/50 |
| Level 5 | 0/50 | 0/50 | 0/50 | 50/50 |
| Level 6 | 0/70 | 0/70 | 0/70 | 70/70 |
| Level 7 | 0/60 | 0/60 | 0/60 | 60/60 |
| Level 8 | 0/61 | 0/61 | 0/61 | 61/61 |
| Level 9 | 0/70 | 0/70 | 0/70 | 70/70 |
| Level 10 | 0/70 | 0/70 | 0/70 | 70/70 |
| Level 11 | 0/100 | 0/100 | 0/100 | 100/100 |

Fig. 1. Screen Capture of the Program

## III. GUI Design of the Program

Java is utilised to construct the program as a result of its cross-platform capability. The Swing Framework is utilised to construct the Graphical User Interface (GUI). In addition, the VLCJ package is chosen to construct the media player as an award option.

"Colour is one of the most effective visual attributes for coding information in displays, and is capable of achieving powerful and memorable effects when used correctly." [1]

The classical green and red contrast is utilised throughout the design of the GUI. The 'New Spelling Quiz' and 'Clear Statistics' buttons contains this colour contrast to encourage selection of the former using green colour, while discouraging selecting the latter using red colour. This is also applied in the media player to distinguish the difference between replay and stop. Additionally, the hues are mainly the default grey and white colours because they make visually significant contrasts to the black text colour and avoid visual fatigue of the users. The hues and the black text colour also tend to be neutral colours in most cultures. For example, the Chinese culture dislikes red as a text colour as the names of the dead were previously written in red. The Nimbus Look and Feel is chosen to increase the overall brightness of the Swing components, therefore increasing the contrast between the components and the black colour of the text. It also increases the colour saturation to make the colours look rich and full. Icons are also applied to most Swing buttons to address the colour vision deficiency issue by providing an alternative way to distinguish the buttons visually.

A solid effort has been made to maintain the simplicity of the GUI. The five main buttons relating to the five main functionalities are grouped to the left, while the Swing components relating to the quiz functionalities are placed at the bottom section. The grouping helps to minimize eye and hand movements since the GUI components with similar functionalities are adjacent to each other and the user can slightly move the cursor to select another option. The middle section contains a text field for outputting quiz feedback and instructions. The middle position is prioritised to represent the most important information because the user has a natural tendency to focus at the middle of a window. On the right side of the main screen, a table displays the live progress of the current quiz including the numbers of total words as well as the mastered and failed frequencies.

The description on each button has been condensed to suite the targeted user group. The second language learners are assumed to have mediate to advance computer proficiency and no physical limitations. Concise descriptions such as ‘New Quiz’ are used to describe the functionality of each button, in order to address the vocabulary problem. The vocabulary problem means “the users may know what they are looking for, but lack the knowledge needed to articulate the problem in terms and abstractions used by the information system.” [2] Additional tool tips have also been implemented for several buttons to further clarify the functionalities when the cursor hovers over them.

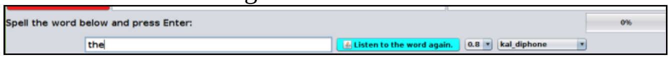
## IV. FUNCTIONALITY

Voxspell has four main functionalities: New quiz, Review Quiz, View Statistics, and Clear Statistics. The user is given the flexibility to load his or her own media and word list files for their specific needs.

Both New Quiz and Review Quiz are based on the traditional question answering to remind formal educational evaluations. The New Quiz functionality provides the user a quiz with ten words from the current level to spell. With each word tested, a maximum of two attempts is provided. Similarly, the alternative Review Quiz provides a quiz with a maximum of ten words that are incorrectly spelled previously. This functionality is included because second language users may be likely to make spelling mistakes, and one of the best learning strategies is to learn from one’s own mistakes. The feature that a failed word can be reviewed again after the completion of a quiz also encourages learning by spaced repetition. Spaced repetition states “…with any considerable number of repetitions a suitable distribution of them over a space of time is decidedly more advantageous than the massing of them at a single time.” [3]

The user submits the answer by pressing the enter key after each word is pronounced. Instant response will be received for the correctness of spelling. The enter key is used for submission because most users have a natural tendency to submit inputs by pressing it. An error checking logic has been implemented that shows an error message when it detects non English characters and ignores empty inputs. Hence, this contributes to the overall robustness to the program against various possible textual inputs.

The voice/accent and speaking rate of the Festival Synthesis System utilised can be adjusted in both Quiz functionalities. English as a widely spoken language has many distinctive regional accents. They should not become a barrier for understanding the word pronounced in order to complete the quiz. The rate is adjustable since the English listening proficiency is highly likely to differ between individuals for second language users. Additionally, the user can heard the pronunciation of each word multiple times without penalties. This encourages repeated learning by listening to the same pronunciation multiple times. Because the user group is assumed to mostly dislike animation and sound, no sound effects such as cheering is included in the instant feedback.


Spell the word below and press Enter:

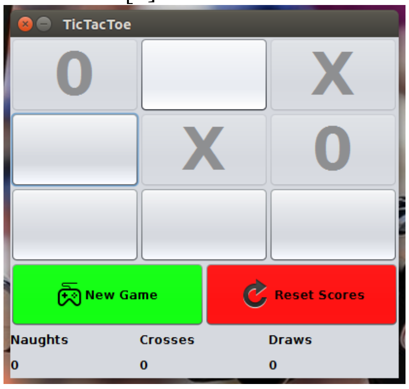
0%

the

Fig. 2. Screen Capture of the Basic Operation Buttons

Once the user starts a quiz, other functionalities will be disabled and he or she must finish the current quiz before using another feature. This strategy enforces the user to concentrate on the current quiz in progress and prevents cheating by selecting another functionality half way through a quiz.

If the user achieves a distinctive level of correctness, one of the following award options can be selected: adjust the spelling level, play a reward video with or without echo effects, or play the Tic-Tac-Toe game. The spelling level can be maintained or incremented by one. The Tic-Tac-Toe game is a single player game with a voice synthesis feature, while the media player includes controls such as pausing and replaying a video for the user. Both reward options achieve the entertaining goal and motivate the user to improve the spelling accuracy. “Motivation can precede learning and promote memory formation, independent of stimulus features or feedback.” [4]



| Naughts | Crosses | Draws |
|---|---|---|
| 0 | 0 | 0 |

Fig. 3. Screen Capture of Tic-Tac-Toe

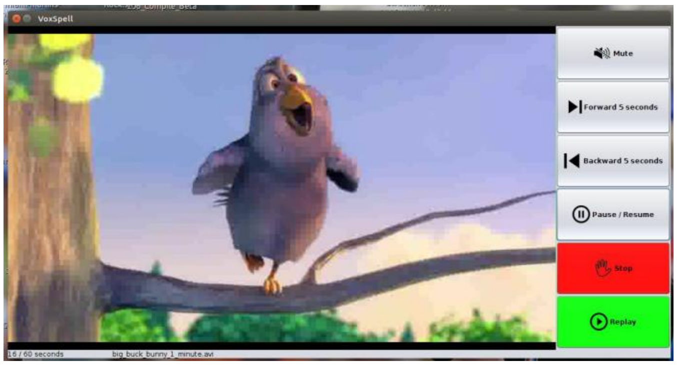


Fig. 4. Screen Capture of the Media Player

The View Statistics functionality presents the numbers of times each word is mastered, faulted, and failed individually. This is implemented because my assumption is that second language users frequently monitor the words misspelled and mastered. On the other hand, the Clear Statistics functionality clears the hidden files associated with the statistics after confirmation.

After each quiz, the user chooses one of the four main functionalities again or close the application. This strategy is intended to reinforce learning by means of repetition. When restarting after closing the application, this strategy does not apply and the words are loaded and selected randomly again.

A Help button is also provided in the main menu to assist the user. Upon selection, it opens up the user manual.

## V. Code Design and Development of the Program

Java was nearly the best choice of language for this project as a result of its cross-platform capability. In comparison, C# as another popular object-oriented language is only cross-platform if used along with the third-party Mono project. The cross-platform capability is crucial as the program relies heavily on the Festival Voice Synthesis System and FFMPEG installed in Linux, making the Microsoft based C# unsuitable.

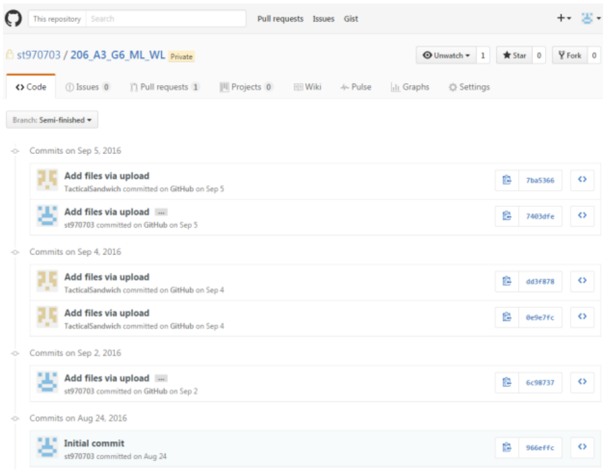
However, Java does have several limitations and disadvantages. It is generally considered to run slower than the fastest third generation strongly typed languages such as C and C++. Java's file manipulation abilities are also less effective compared to the native Linux Shell scripts.

Other additional libraries utilised in this project include the Swing framework, the VLCJ framework by CAPRICA, the FFMPEG software, and the Festival Voice Synthesis System. The Swing framework is utilised to construct the GUI, while the VLCJ framework by CAPRICA is chosen to construct the media player of Voxspell. In addition, the FFMPEG software processes the reward video with special audio and video effects in the background. Lastly, the Festival Voice Synthesis System pronounces the words to be spelled and the instructions in the background.

In the prototype stage of the project, the project was shared between a pair of two students and the Extreme Programming (XP) technique applied. The application of various XP values and practices utilised fell into the following three categories: fine scale feedback, shared understanding, continuous process, and programmer welfare.

Pair programming with an agreed coding standard was utilised to develop several implementations to balance coding efficiency with quality. During the development of the prototype, both partners had a collective code ownership, as well as maintained a simple design and an understandable system metaphor to ensure solid shared understanding. A sustainable developing pace has been maintained by means of adequate planning and scheduling to avoid deadline stress.

Continuous integration was also applied with version control aided by Git and GitHub. This ensured both partners work on the latest version while keeping a reliable record of all previous versions. The GitHub repository consisted of many experimental branches that were used for newly introduced features. Individual changes were committed locally using Git and then pushed to the remote GitHub repository. Lastly, successful experimental branches were merged with the master branch.



st970703 / 206_A3_G6_ML_WL Private

Branch: Semi-finished

Commits on Sep 5, 2016

Add files via upload
TacticalSandwich committed on GitHub on Sep 5 — 7ba5366

Add files via upload
st970703 committed on GitHub on Sep 5 — 7403dfe

Commits on Sep 4, 2016

Add files via upload
TacticalSandwich committed on GitHub on Sep 4 — dd3f878

Add files via upload
TacticalSandwich committed on GitHub on Sep 4 — 0e9e7fc

Commits on Sep 2, 2016

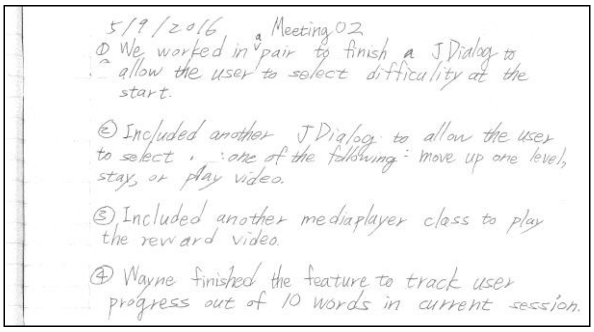
Add files via upload
st970703 committed on GitHub on Sep 2 — 6c98737

Commits on Aug 24, 2016

Initial commit
st970703 committed on Aug 24 — 966effc

Fig. 5. Screen Capture of the GitHub Repository

Apart from the commit messages, all important design decisions and implementation difficulties were recorded in a logbook to ensure flawless communication.



5/9/2016 Meeting 02

① We worked in a pair to finish a JDialog to allow the user to select difficulty at the start.

② Included another JDialog to allow the user to select : one of the following : move up one level, stay, or play video.

③ Included another mediaplayer class to play the reward video.

④ Wayne finished the feature to track user progress out of 10 words in current session.

Fig. 6. A Page from the Logbook

After the prototype stage, the project progressed to the final product stage and became a single person project. This stage involved the client meetings where the lecturers participated as clients to clarify project planning and specifications. It enabled the XP practice of whole team, meaning that the project is teamwork of both the clients and developers. After each meeting, new user stories were introduced, requiring refactoring of current code and new implementations. Hence, frequent iterations between the release planning and iteration phases of XP were promoted. A peer review process was also involved to measure the quality of the prototype submitted to encourage iterations between the test and iteration phases of XP.

There are several innovations in the implementation. The factory and singleton design patterns are applied to the Statistics class in order to ensure that at most one instance of the class can be created. In addition, only one JFrame is created in the program instead of multiple ones. The Cardlayout is also applied as an alternative to assist switching between various JPanels. This ensures the ease of maintenance and user friendliness of seeing the single frame icon as expected. The Statistics class extends the AbstractTableModel to update the statistics table in the main screen timely as the user progresses through the quizzes. Dedicated logic has been implemented to hint the length of each word tested to assist the spelling. The overall implementation has been condensed to follow the XP value of simplicity to ensure ease of communication and to avoid unnecessary implementations. It is demonstrated in the UML diagram in Figure 5.

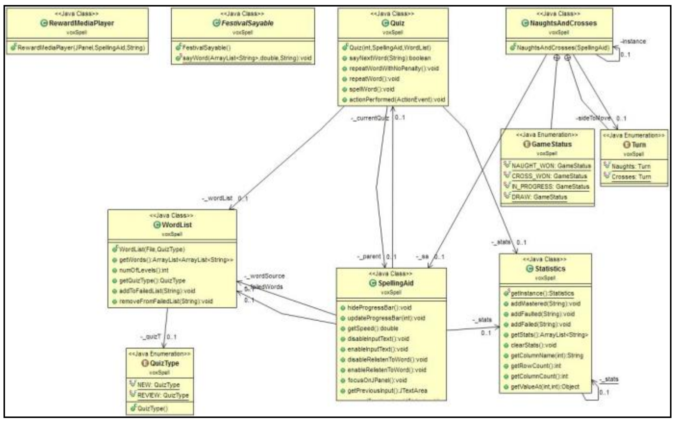


Fig. 7. UML Diagram of the Implementation

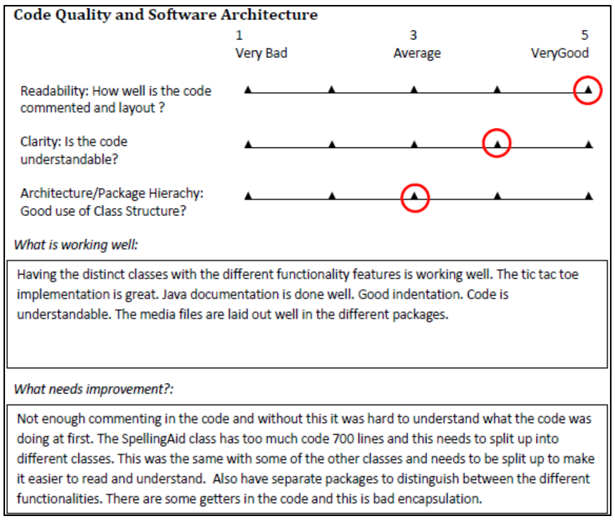
There is no specific shortcut keys implemented for this program apart from pressing the enter key to submit answers. Since the main program functions are concise that involve only a small amount of typing and mouse clicking, the motivation would be to encourage the users to concentrate on the quizzes at hand instead of getting distracted trying to memorise shortcut keys.

Various developmental issues were faced during the development of the program. The most difficult issue was to retrieve the Festival voice list of all the voice packages available in the current computer in order to process the voice changing functionality. Unfortunately no alternatives was found to retrieve the list from a bash environment process, and the voice packages had to be hard-coded. An assumption was made that the computer running this program would have a similar set of voices as the computers in the laboratory.

Other issues include the compatibility of different versions of Java such as 1.7 and 1.8 as well as the following proposed coding standard. Java 1.7 was chosen as it was the most widely used version in 2015. [5] Consequently, the implementation was refactored to Java 1.7 standards. Additionally, the issue of following the proposed coding standard was resolved by consulting my partner before the naming of new implementations and code refactoring to ensure the shared understanding was maintained.

## VI. Evaluation and Testing

The prototype developed has been actively self-tested before the prototype submission. The testing involved manually running the program and code review with my partner. However, self-testing could be biased and the peer review process has been introduced to minimise the biases. The peer review form was designed in a client meeting by the class as demonstrated in Figure 6.



**Code Quality and Software Architecture**

| | 1 Very Bad | 2 | 3 Average | 4 | 5 VeryGood |
|---|---|---|---|---|---|
| Readability: How well is the code commented and layout ? | | | | | ✓ |
| Clarity: Is the code understandable? | | | | ✓ | |
| Architecture/Package Hierachy: Good use of Class Structure? | | | ✓ | | |

*What is working well:*

Having the distinct classes with the different functionality features is working well. The tic tac toe implementation is great. Java documentation is done well. Good indentation. Code is understandable. The media files are laid out well in the different packages.

*What needs improvement?:*

Not enough commenting in the code and without this it was hard to understand what the code was doing at first. The SpellingAid class has too much code 700 lines and this needs to split up into different classes. This was the same with some of the other classes and needs to be split up to make it easier to read and understand. Also have separate packages to distinguish between the different functionalities. There are some getters in the code and this is bad encapsulation.

Fig. 8. Screen Capture of the Peer Evaluation Form

The submitted prototype was distributed to four anonymous students. On the other hand, each student also received four prototypes to evaluate.

### A. *Overall Peer Review Feedback*

Overall the peer reviewers were satisfied by the indentation and formatting of my code. However, appropriate packages should be applied to organize classes and prevent class name collisions. More commenting was also required for the GUI code and the number of getter methods should be reduced to thoroughly imply encapsulation.

In addition, the video/music player was very easy to use and the dashes showing amount of letters to spell was a great feature. The progress bar showing the percentage on the side was very indicative of the quiz process. The large buttons of the menu on the left with symbols makes it easy to see and click.

With regards to functionality, the main criticisms for the functionality were the followings:

1. A warning should be given before blocking out the other buttons when doing a new spelling.
2. After choosing to play the Tic-Tac-Toe, the main VoxSpell GUI disappeared. The Tic-Tac-Toe option was a great feature but was a multiplayer game. This would be an issue if the user didn’t have someone to play with.
3. The speaking pace was too slow, and it caused impatience or made the word harder to understand. An option to choose the speed of Festival should be included.
4. When choosing the file if an incorrect format file was chosen, the error was shown but then went onto choosing the video option. It should have an option to select another file.
5. Requiring the user to select the video and the music before entering the main application was a bit confusing if they didn't know it was for the reward functionality.
6. There was a lag between the initial file choosing and the please pick level message. It was a bit inconvenient that the main GUI had to wait until the video finished rendering before appearing.
7. The text in the text area was small and the formatting should be changed to see the text with ease.
8. Also there should be a feature to exit out of a quiz if required by the user.

## B. *Justification of Changes*

According to the criticisms above, the following issues were addressed by refactoring the code:

1. The method temporarily disabling the buttons were changed from actually disabling the buttons to removing Action Listeners. The buttons would still be clickable but nothing would happen. This prevented the user from misleadingly thinking the program was crashing.
2. After choosing to play the Tic-Tac-Toe, the main VoxSpell GUI would still disappear. This feature was maintained to prevent the user from multitasking, therefore concentrating on the task at hand. The Tic-Tac-Toe was still a single player game without Artificial Intelligence (AI) due to the lack of time.
3. A combo box was added to adjust the speaking pace of Festival easily at runtime to suit the user's personal preference.
4. As a result of the lack of time, I was unable to develop reliable logic to detect if the custom word list exactly matches the original format. Therefore the user himself or herself would be responsible for checking the correctness of the format.
5. A new message was introduce at the beginning of the program specifying the reasons and instructions for loading user’s own word list and media files to avoid confusion.
6. The level picking message still showed up after the initial file choosing. This was maintained because the GUI should not show up unless the files have been loaded appropriately.
7. The font size was increased and the font bolded in the GUI to increase the readability.
8. The user was still prohibited to quit a quiz in progress. The purpose of this feature was to encourage completing a quiz thoroughly and not quitting halfway, because learning a second language could be challenging.

## C. *Additional Changes*

To improve the user experience, some additional changes were added apart from the ones listed above:

1. Voice synthesising was added to many functionalities such as saying the label of a button upon pressing. It was assumed that the second language user would appreciate the opportunities to enhance the listening practices and memorisation of pronunciation.
2. A confirmation message was added before delete the statistics. This prevented the user from accidently deleting the statistics.
3. The occasional voice overlapping of Festival was solved by storing the previous Java process object. If the logic implemented detected there was an unfinished previous process, current request to say words would be ignored.

## VII. Future Work

As for future work, it is planned to increase the platform support for the program, such as Microsoft Windows and even a web-based platform. However, the dependencies on some of the Linux based packages such as FFMPEG will need to be resolved in order to adopt another platform. A more intelligent and sophisticated speech synthesis system can also be integrated into this program to produce more humanlike and smoother voices. This is because Festival is not good at pronouncing words of very short lengths and sometime static noises appear when a very slow speaking pace is used. The mispronounced words may mislead the second language learners. In addition, a more sophisticated GUI framework for Java can be adapted to support automated testing of the GUI. Last but not least, the program can be extended to support orthography and spelling learning of other languages such as Spanish.

## VIII. Conclusion

Electronic educational games are learning and recreational environments that embed pedagogical activities in highly enjoyable interactions to increase learner motivation.

This kind of applications is particularly interesting for learning activities involving routine and repetitive tasks, because students need an extra motivation to successfully complete them. Orthography and spelling learning, as well as memorizing new vocabulary, can be classified in this group. [6]

In this paper, an educational game, Voxspell for reinforcing orthography and vocabulary learning has been introduced. The recreational goal is achieved by means of game and media-playing awards for distinction in spelling attempts. On the other hand, the didactic goal is achieved via the traditional question answering in the spelling quizzes to remind formal educational evaluations. The XP values and practices have been applied during the development of the program in both the prototype and final product stages to ensure the final product will be of high quality. This includes both self-testing and peer-evaluations. The peer-evaluations helps to avoid the potential biases that may exist in the previous self-testing. Both the client meetings and the peer-evaluations have encouraged iterations between various XP phases.

As for future work, extended platform and language support can be implemented to fulfil various user needs. AI and more intelligent speech synthesis systems can be integrated to improve user experience and user friendliness. These have not been implemented due to the lack of time.

## IX. Ackonwledgement

Special credits must be given to my partner, Wayne Liang who kindly assisted me in peer programming and code review in the prototype stage of the project. Furthermore, I would like to express my appreciation for the four peer reviewers who gave me honest and helpful feedbacks for my prototype submission, as well as Darius Au who sponsored the Tic-Tac-Toe template.

## X. References

[1] MacDonald, Lindsay W. "Using color effectively in computer graphics." *IEEE Computer Graphics and Applications* 19, no. 4 (1999): 20-35.

[2] Henninger, Scott, and Nicholas J. Belkin. "Interface issues and interaction strategies for information retrieval systems." In *Conference Companion on Human Factors in Computing Systems*, pp. 352-353. ACM, 1996.

[3] Ebbinghaus, H. (1885). Über das Gedchtnis. Untersuchungen zur experimentellen Psychologie. Leipzig: Duncker & Humblot; the English edition is Ebbinghaus, H. (1913). Memory. A Contribution to Experimental Psychology. New York: Teachers College, Columbia University (Reprinted Bristol: Thoemmes Press, 1999).

[4] Adcock, R. Alison, Arul Thangavel, Susan Whitfield-Gabrieli, Brian Knutson, and John DE Gabrieli. "Reward-motivated learning: mesolimbic activation precedes memory formation." Neuron 50, no. 3 (2006): 507-517.

[5] "Java version statistics: 2015", last modified April 8, 2015, https://plumbr.eu/blog/java/java-version-statistics-2015-edition.

[6] García, Raquel M. Crespo, Carlos Delgado Kloos, and Manuel Castro Gil. "Game based spelling learning." In *2008 38th Annual Frontiers in Education Conference*, pp. S3B-11. IEEE, 2008.