­­­­­­­­­­­­­­­

**Gamifying the Japanese writing system**

**Final Kanji Quest**

**Final Kanji Quest**

Author: Firas Altayeb

Supervisor: Mr. Andrew Holyer

Student ID: 1469210

April 16, 2017

Author: Firas Altayeb

Supervisor: Mr. Andrew Holyer

Student ID: 1469210

April 16, 2017

**Abstract**

As Japanese entertainment products (e.g. animation, comics & video games) became more accessible, more and more people seek to study the Japanese language. But comparing the number of students in beginner classes with advanced classes, it can be clearly seen that the number of dropouts is very high compared to most other languages. This is most likely occurring due to the Japanese writing system requiring students to learn an extensive list of logographic kanji[A] characters. The aim of this project is to teach Japanese language students how to read and understand the meaning of each kanji by transforming (gamifying) traditional kanji learning steps into game elements and subsequently, evaluate the application's effectiveness in regards to this issue.

**Abstract**

As Japanese entertainment products (e.g. animation, comics & video games) became more accessible, more and more people seek to study the Japanese language. But comparing the number of students in beginner classes with advanced classes, it can be clearly seen that the number of dropouts is very high compared to most other languages. This is most likely occurring due to the Japanese writing system requiring students to learn an extensive list of logographic kanji[A] characters. The aim of this project is to teach Japanese language students how to read and understand the meaning of each kanji by transforming (gamifying) traditional kanji learning steps into game elements and subsequently, evaluate the application's effectiveness in regards to this issue.

**Originality Avowal**

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King’s College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

**Firas Altayeb April 16, 2017**

**Originality Avowal**

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King’s College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

**Firas Altayeb April 16, 2017**

**Acknowledgements**

I would like to thank my supervisor, Mr. Andrew Holyer for always promptly answering all my questions. The support and guidance provided by him had an enormous impact on the project’s quality and completion. His time and assistance are much appreciated. Thanks to Department of Informatics staff and IT services at King’s College London.

**Acknowledgements**

I would like to thank my supervisor, Mr. Andrew Holyer for always promptly answering all my questions. The support and guidance provided by him had an enormous impact on the project’s quality and completion. His time and assistance are much appreciated. Thanks to Department of Informatics staff and IT services at King’s College London.

Table of Contents

[Introduction 6](#_Toc92453467)

[Motivation 6](#_Toc92453468)

[Project Aims 7](#_Toc92453469)

[Project Aims 7](#_Toc92453470)

[Report Structure 7](#_Toc92453471)

[Research Methodology 8](#_Toc92453472)

[Setting and Population 8](#_Toc92453473)

[Instrument and Measurement 8](#_Toc92453474)

[Limitation of the study 9](#_Toc92453475)

[Ethical considerations 9](#_Toc92453476)

[Background 10](#_Toc92453477)

[Platform 10](#_Toc92453478)

[Constraints 10](#_Toc92453479)

[Existing Solutions 10](#_Toc92453480)

[Requirements Specification 12](#_Toc92453481)

[Functional Requirements 12](#_Toc92453482)

[Non-Functional Requirements 12](#_Toc92453483)

[Implementation 13](#_Toc92453484)

[Player Controller 13](#_Toc92453485)

[Player interface 14](#_Toc92453486)

[Progress UI 16](#_Toc92453487)

[Inventory UI 16](#_Toc92453488)

[JSON Scripts 20](#_Toc92453489)

[Texture region 20](#_Toc92453490)

[Hiragana and Katakana UI 21](#_Toc92453491)

[Kanji UI 23](#_Toc92453492)

[Mnemonics UI 25](#_Toc92453493)

[Resize Action 26](#_Toc92453494)

[Battle System 27](#_Toc92453495)

[Design 31](#_Toc92453496)

[Evaluation 33](#_Toc92453497)

[Testing 34](#_Toc92453498)

[Conclusion and Future Work 34](#_Toc92453499)

[Definitions 35](#_Toc92453500)

[Bibliography 36](#_Toc92453501)

# Introduction

As Japanese entertainment products are becoming more famous, the number of people seeking to study the Japanese language is increasing. But comparing the number of students in beginner classes with advanced classes, it can be clearly seen that the number of dropouts is very high compared to most other languages. This is most likely occurring due to the Japanese writing system.

The difficulty of the Japanese writing system can often be attributed to the extensive list of logographic kanji characters that each have two types of pronunciations, the ON pronunciations and the KUN pronunciations. The ON pronunciation, also known as Onyomi, is used when kanji characters are used with other kanji characters while the KUN pronunciation, also known as Kunyomi, is used when kanji characters are alone and not part of a compound.

Alongside kanji, syllabic kana [B] which consists of two-character types, hiragana [C] and katakana [D], is used to construct almost every sentence in the Japanese language. Hiragana is used primarily for grammatical elements such as particles that connect words to make sentences. Katakana is used mainly for foreign words, names, and sometimes to emphasise a point. The Japanese writing system also makes use of Latin script to Romanize [E] kana and kanji character and make them more readable for foreigners.

Putting all of the above into consideration, it can be clearly understood why the Japanese writing system is deemed to be the most complicated in use anywhere in the world. However, current implementations of Japanese teaching applications are either made with too many unnecessary features or made with far too few while also not paying attention to how to keep the user engaged and wanting to come back for more.

Therefore, this project will display a significant amount of focus on how to teach users the Japanese language while not only maintaining, but increasing their motivation and interest in the language.

## Motivation

As a person who undertook three years’ worth of Japanese courses, I have come to understand the difficulty of the Japanese writing and reading systems that require you to learn and memorise over 3500 kanji characters. By working on this project, not only will I work towards solving this problem, but I will also work with two topics that I am most passionate about, game development and the Japanese language. This project will most certainly prove helpful for gaining knowledge on both topics and developing skills in the fields of game development, Android development, Java development and object-oriented development.

## Project Aims

### Project Aims

The aim of this project is to teach students of the Japanese language how to read and understand the meaning of each kanji by transforming (gamifying) traditional kanji learning into a game. Unlike conventional kanji learning, the game will not only focus on teaching Japanese but also have engaging battle and progression systems that support the learning process by adapting according to the student’s level of understanding, and a vast graphical world to explore that will help the students stay interested and motivated. Two good examples of game series with captivating worlds and scenery are the Final Fantasy and Dragon Quest series where the story-driven aspect of the games is what keeps the players coming back for more. The name Final Kanji Quest (FKQ) is a homage to these two series.

## Report Structure

This report will begin with a chapter discussing the most critical aspects of the research study that will be done using this project as an instrument - the type of the research study, the underpinning practices, and the procedures for conducting and replicating the study. The research study section will be followed by a chapter describing the background content of the project, constraints that are associated with the project, and existing solutions (books and apps) that have attempted to aid and help learners of the Japanese language. Following this, the requirements section will outline the functional and non-functional specification of the project with justifications on decisions made. Next, the implementation section and the testing section will provide an in-depth view of all of the application features and components. In the design section, we will discuss the design philosophy and explain how good design decisions can aid with language learning. After that, the evaluation section will show the result of the research study and an evaluation of the final implementation. Finally, we will discuss the conclusion of the project and introduce improvements for future work.

# Research Methodology

Alongside developing the Japanese teaching game, a quantitative research study has been set out to understand the impact of gamification on motivation and interest (in the context of studying the Japanese writing system) by using the game as an instrument. The result can potentially provide Japanese language teachers evidence on how effective gamification can be in the context of the Japanese language learning and help to design a system that teaches kanji effectively to students.

## Setting and Population

For the proposed study, participants were recruited from Kings College London Modern Language Centre’s (MLC) Japanese False beginner[[1]](#footnote-1) and Japanese Level 2 evening classes. Unlike the other classes, the evening classes accept King's local professionals, the residential community, and undergraduate and graduate students. As a result, this provided participants from various age groups, backgrounds and genders. As the MLC limits the number of students for each class to 16, the total number of participants will be 32.

Students from Japanese level 1 were not recruited as their course syllabus focuses on hiragana and katakana and does not cover Kanji. Furthermore, the number of students progressing from Japanese level 1 to level 2 is high which indicates that at their current level, the students have not started to lose motivation and interest - which this project is trying to tackle.

Students from Japanese False Beginner and Japanese Level 2 classes were the most suitable participants for the study as their course materials introduce kanji that, as discussed earlier, cause most students to lose motivation and stop learning the Japanese language.

## Instrument and Measurement

Questionnaires were chosen as the data collection method because they allowed the collection of information from a large number of people in a short period. Moreover, the results from questionnaires can also be analysed more objectively than results obtained through other means.

A journal published by the Association for Medical Education in Europe showed how high-quality questionnaires with particular emphasis on survey scales, can be used to improve the probabilities that survey designers will accurately measure what they intend to measure [H].

For the proposed study Test-For-Mean (TFM) was used as the statistical data analysis method. TFM was used to determine the overall trend of data very easily. Furthermore, TFM allowed the result to be calculated very quickly. In order to prevent outliers and use TFM more efficiently, Rating Scale Questions was used in the questionnaire [I]. The result of the analysis will determine whether the hypothesis - Gamification increases motivation and interest in regards to studying the Japanese writing system - is rejected or accepted.

## Limitation of the study

Due to time constraints (project deadline), participants were only emailed two questionnaires. Furthermore, not enough time was given to the participants to try the game as the questionnaires were emailed one week after the delivery of the game. Moreover, due to a limitation of resources, participants were only taught a limited number of kanji characters.

## Ethical considerations[[2]](#footnote-2)

During design and implementation of the project, great care has been taken to abide by the Code of Conduct which is issued by the British Computer Society.

The evening classes were chosen over the morning assessed classes as the MLC does not allow students/applicants under the age of 18 to undertake any evening class.

Students under the age of eighteen were not recruited for the research study as they are not able to give consent and sign for themselves.

Before the participants for the study were recruited, the following was clearly communicated to them:

1 - The UK Data Protection Act 1998 will apply to all information gathered.

2 - All data collected by the participants is confidential and anonymous.

3 - The un-anonymous data collected by the participants will only be accessed by the researcher and an agent of the university whom will verify the authenticity of the participation.

The risks and benefits the participants might accrue were also clearly identified to the participants;

Benefits: The kanji learning steps that will be gamified in the app will be from the book Genki-I, which Japanese False beginner and level 2 students use for their course. As a result, the app will greatly help the participants with their course.

Risks: The only disadvantage to taking part in this project is that students will be sparing a small amount of time to use the app/project and complete the questionnaire.

# Background

## Platform

In order to give Japanese language students more time to study and the ability to study at any place, a native mobile app will be developed. Android will be chosen over IOS as the platform of choice as according to the latest data gathered; the Android OS has sold over 290,000 units in 2016 against the 44,000 IOS sold units. Furthermore, Android development uses the Java programming language which offers automatic garbage collection features. Using Java also enables development with Libgdx. Libgdx is a cross-platform game development framework that provides low-level bookkeeping APIs that will help cut development time and allow for more time to be focused on developing the battle and player progression algorithms. The reason Libgdx will be chosen over game engines, such as Unity and Unreal, is because Libgdx does not cage the developer around the scaffolding of the engine and instead offers control of the main game loop to the developer so that he or she could call into the libraries as needed throughout the game lifetime.

## Constraints

The first significant constraint for this project is the performance/specifications of the user’s phone or tablet. Because of the vast variety of phones and tablets currently in the market, there will be some devices that are old and slow and will not be able to load larger files promptly for the purposes of a game such as this. Another related constraint is Android API level. If the project targets old API versions to support the highest number of devices, the project will have to sacrifice some functionality to do so - more on this in the evaluation section.

Another constraint that is related to the above is testing. Because the project is very concerned with sprites and user experience, testing the project on multiple screen sizes is crucial, but because of the many available screen sizes currently in the market, that would not be feasible. However, testing the project on what is considered a standard screen size according to the Android generalised size chart, will cover 87% of the currently used Android devices.

## Existing Solutions

Some of the most prominent solutions to the kanji writing and reading problem are Remembering the Kanji book series and the Spaced Repetition Software, Anki-Flashcards.

Remembering the Kanji (RTK) is a series of books by James Heisig, intended to teach the 3007 most common kanji to students of the Japanese language. Unlike most courses that employ rote-memorization, RTK teaches students how to write and read kanji by associating kanji and kanji radicals with keywords and then connecting the meaning of each kanji with the meanings of all the radicals used to write that kanji using mnemonic devices [G].

However, a study [J] done over a period of a year on university students studying kanji through the use of mnemonic strategies found that while mnemonics are useful to memorise kanji and kanji radicals when applied in a meaningful way, an over-reliance on this approach can have contrary effects for the learner.

The study highlighted numerous accounts of the meaning of a kanji being lost in overly complicated mnemonic strategies.

The Spaced Repetition algorithm is a learning technique that incorporates increasing intervals of time between subsequent review of previously learned material to exploit a psychological effect that helps retain items in memory.

A study [K] done on Japanese language students over a period of 10 months found a strong relationship between extended studying using spaced repetition software (SRS) and second language (L2) proficiency test score gains. The result of the study showed that out of the 72 participants, 60 participants who continued using an SRS, acquired an overall 11% score increase compared to their pre-SRS test scores. The increase is deemed to be very impressive when considering the many other variables which may have influenced these scores.

The Anki flashcards software utilises a slightly modified version of the Spaced Repetition algorithm which allows for priorities on cards (Kanji in this context) and shows cards in order of their urgency. However, the previously mentioned study shows that out of the 72 original participants that used SRS, 12 participants did not continue using the software due to a lack of motivation.

A study that was done [L] by the US Air Force Academy on undergraduate level Computer Science students showed that by transforming a traditional classroom experience into a competitive multiplayer game, motivation among students remarkably rose. When asked about their overall satisfaction level, many students stated that they were more than content with the gamified experience. An examination of student usage logs further confirms that the gamified tool is compelling to students; although the use of the software is entirely voluntary, 85% of the students used the tool during every classroom session.

# Requirements Specification

This project aims to create a game that can teach the Japanese writing system to students while keeping the students motivated and interested by immersing them in a game. Achieving that will encourage students to keep returning to the game and study more.

Initially, the plan was to implement the game only for the Android OS, but the majority of the recruited students did not possess Android devices, and so a desktop version was developed alongside the Android version. Developing the desktop version proved to save time for testing and debugging as the desktop version runs and is complies faster.

Thanks to the Libgdx cross-platform framework, apart from a few platform-specific considerations that had to be taken, the majority of the code stayed the same for both versions.

Taking both the achievements and missteps of the existing solutions into consideration, this project will include aspects of both the spaced repetition algorithm and the mnemonic strategies as they have been proven to be useful for learning how to write and read kanji.

## Functional Requirements

F1 – Have a battle system that adapts to the spaced repetition algorithm.

F2 – Have a progression system that adapts to the spaced repetition algorithm.

F3 – Determine which letter the user will encounter according to user’s knowledge of the language.

F4 – Have an on-screen controller that does not hide backgrounds and visuals.

F5 – Have an easily navigable user interface.

F6 – Track and save the user’s progress.

F7 – Implement a list where the mnemonic strategies can be fully utilised.

F8 – Have sounds and music that aid learning and immersion in the graphical game world.

F9 – Have a user interface that can identify the screen size and change accordingly.

F10 – Implement all the necessary collision, spawn and portal layer for traversing the graphical world.

## Non-Functional Requirements

NF1 – Teach the user how to read and write Romaji [F].

NF2 – Teach the user how to Romanize kana characters.

NF3 – Use Romaji characters to teach Hiragana characters.

NF4 – Use Romaji characters to teach Katakana characters.

NF5 - Teach the user the meaning and writing of each Kanji.

NF6 – Use mnemonics devices to teach Kanji.

NF7 – Use Hiragana characters to teach Kanji.

NF8 – Use Katakana characters to teach Kanji.

NF9 – Teach Kanji only when it is determined that the user has already learned Hiragana and Katakana.

# Implementation

In the early stages of the development cycle, object-oriented programming was used to create the base classes. However, soon after that, the project ended up having large base classes that tried to do everything. Furthermore, the classes became exceedingly difficult to manage as it was easy for one small change to have disastrous side effects. To avoid that, a solution brought forward by Patrick Hoey's Mastering Libgdx Game Development book [J] was used as the basis for all the previous and ongoing work. The solution entailed decoupling the work into component parts - containing core logic - that notify each other when changes are made.

## Player Controller

In order not to violate requirement F4, on-screen arrow-buttons and on-screen analogue stick, were not added as they would take away from the user's immersion in the game world. Instead, an implementation where the player character moves according to where the user clicks/touches the screen was used. Initially, four parts of the screen were given listeners that moved the player character up, down, left and right according to where the user clicks/touches. After evaluating the project with the recruited participants, it was determined that it took the player character too long to reach certain areas. Therefore, four more listeners were included on the screen that allowed the player character to move diagonally up-left, up-right, down-left and down-right, see figure 1.

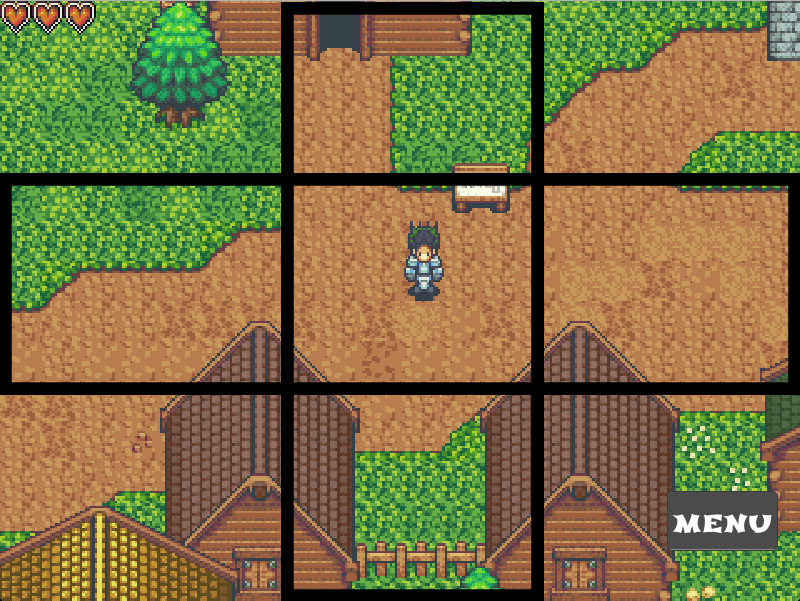


Figure 1 – Player Controller[[3]](#footnote-3)

When one of the screen parts is touched/clicked, the player character input component will notify the player character physics component to move in the direction of the tapped box. One drawback of implementing the input controllers in this way occurs when the user uses both of his thumbs (when using an Android device) to move the character. This is because the player input component will have a hard time sending the correct notification to the player physics component when more than one area is clicked at the same time. To avoid that, an If-Statement was used in combination with a Boolean variable to stop this from occurring, the code snippet in figure 2 shows how the screen is divided and how the code checks to see if a direction has been already clicked.



Figure 2 – Player input handler

## Player interface

When the player interface is first initialised, a "MENU" button, a list and five windows are created. Each window is initialised to show a different aspect of the game such as the progress UI that shows the progress of the user and the Inventory UI that shows the current items the player holds. Alongside that, ten heart shaped sprites are created and then added to an image array. A variable called *numberOfHearts -* manages the current number of hearts to display - is also created and initialised.

After the player interface has initialised, the menu list and five windows are hidden so that the interface won’t be cluttered and confusing. Furthermore, some of the hearts will be hidden based on the *numberOfHearts* variable - more on this later. Clearing out the interface as specified will also satisfy requirement F5 by helping to maintain the user's immersion in the game world, see figure 3.



Figure 3 – Main game interface.

When the menu button is clicked, the list - which was initially set as hidden - will be pop-up for the user, see figure 4. The character moving component is paused when the menu button is clicked. This is done so to allow the program to distinguish between the user wanting to move the character and wanting to click one of the menu list items.

  
Figure 4 – Menu List.

Pressing the "MENU" button again will hide the list - this functionality also applies to all the buttons - and then reactivate the character moving component. Clicking a popped-up window will allow the user to scroll up and down - this is so the interface component will know which window the user wishes to scroll up or down.

## Progress UI

Pressing the "PROGRESS" button in the "MENU" list, will pop-up the progress window which was initially hidden. Using the progress window, the user can see his/her Hiragana, Katakana and Kanji progress, see figure 5. The Katakana and Kanji progress can be seen by clicking the progress window and then scrolling down, see figure 9 and 10 of Appendix B.

  
Figure 5 – progress

## Inventory UI

Similarly, pressing the "INVENTORY" button will pop-up the inventory window which was also initially hidden. Using the inventory window, the user can view his/her current acquired items, see figure 6.

  
Figure 6 – Inventory Window

A click listener was added to each slot of the inventory by using a for loop that has the same number of cycles as the inventory has slots. A special click listener that performs differently to the other slots is added to the top slot (more on this later). The top slot is added to the top left of the window in order indicate that it differs from the other slots. When the user clicks any of the slots apart from the top slot, an If-Statement first checks if the clicked slot contains any item. If the clicked slot contains an item, the item will appear/load in the top slot. If the users click an item while the top slot is filled, the previously loaded item will be replaced with the newly clicked item, see figure 6 for the code snippet.

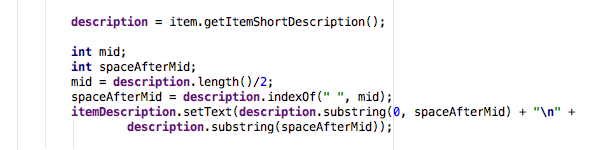
Alongside adding the item to the top slot, a description of the clicked item will be displayed next to the item's sprite, see figure 7 for example.



Figure 7 – Inventory slot listeners

  
Figure 8 – Item in top slot

Initially, the user had to scroll to the right to see what remains of the description if it was long. As this was time-consuming and unpleasant looking, a combination of *String.length()* and *String.index()* methods were used to split the text. The description text was divided in half using the first space located after the text midpoint. This allowed us to avoid having words cut in half, see the code snippet in figure 9.

  
Figure 9 – Long text split code

Clicking the top slot while an item is loaded will activate the effect of the item and then remove both the copy of the item added to the top slot and the item that was initially used to fill the top slot. In this instance, see figure 8, the Small Potion was used. When an Item is used, the inventory observer component will notify the character interface component of the name of the clicked item. After receiving the notification, the player interface will do one of the following based on the name of the item:

All Health Potions- Recovers lost health – increases the value of the *numberOfHearts* variable.

Extra Heart - Increase the maximum number of heart the user can have - this item was made available in case users found the game hard and liked to change the difficulty.

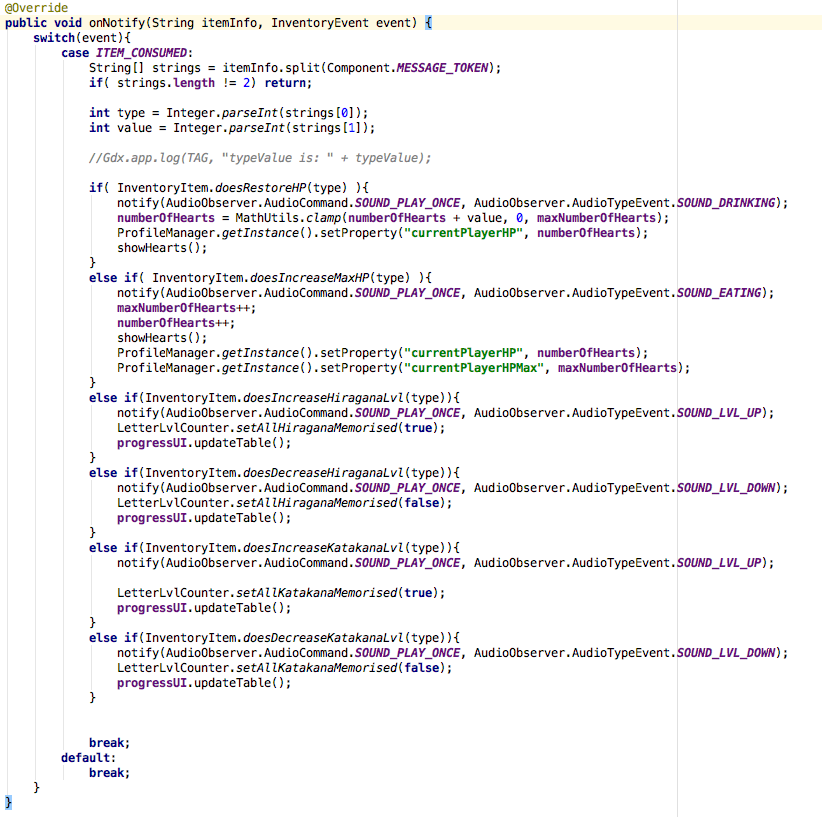
Red H-labelled stone - Stop hiragana character from showing up

Red K-labelled stone - Stop katakana character from showing up

Blue H-labelled stone - Reverse effect of H red stone

Blue K-labelled stone - Reverse effect of K red stone

The red stone items were implemented in case the user has already memorised Kana and does not wish to encounter them while traversing the game world. The blue coloured stone items were implemented to reverse the effect of the red stones, in case the user changes his/her mind or if the user wants to refresh his memory. Implementing the four stones mentioned above satisfies F3. See figure 9 for the code snippet.

  
Figure 9 – Inventory notification handler.

## JSON Scripts

JavaScript Object Notation (JSON) is a standard human-readable data format for transmitting data objects as attribute-value pairs. For this project, JSON scripts alongside factory classes will be used to load information from scripts instead of using hardcoded values in the source code. Moreover, JSON scripts will remove the need to recompile the entire project every time a straightforward change is made as only the scripts will need to change.

Thanks to Libgdx, all complexities surrounding JSON, such as serialising and deserialising JSON objects, are handled automatically without requiring customised readers and writers. That being said, Libgdx does, however, need the user to create a plain old Java object (POJO) for serialising and deserializing the JSON objects.

According to an article posted to Microsoft developer community [N], data formatted according to the JSON standard is lightweight and can be parsed with incredible ease making it an ideal data exchange format. Moreover, unlike popular belief, JSON is not just limited to web applications, but can also be used in virtually any scenario where applications need to exchange or store structured information as text.

## Texture region

Usually, most games tend to render multiple images to the screen at once. This process involves turning each model into a texture, uploading the texture into the graphics processing unit (GPU) and then rendering it - usually using the by OpenGL API. However, doing this for multiple images at once is very expensive on the GPU as the texture binding, and switching process is hugely taxing on the GPU.

By using a texture packer, multiple images can be packed together into a single image. And using a TextureRegion object, the packed image can be divided into regions that can be render separately. This combination solves the above GPU issue as only a single image will need to be bind and render.

The following screenshot visually outlines how the TextureRegion class can be used. A single image is used for the whole texture thus avoiding the expensive texture switching [O].



Figure 10 – Example showing how the TextureRegion class can be used

## Hiragana and Katakana UI

Clicking either the "HIRAGANA" or "KATAKANA" buttons from the menu list will display a window with the same name. When either of the windows is first initialised, an array list of Kana letters is created using a factory class. A factory class is class that creates objects using a POJO that is linked with a JSON script, see figure 11 – in our case, the Kana objects are initialised with three strings holding the Hiragana, Katakana and Romaji equivalents of each letter.

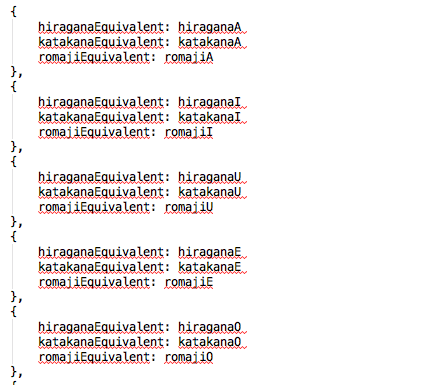


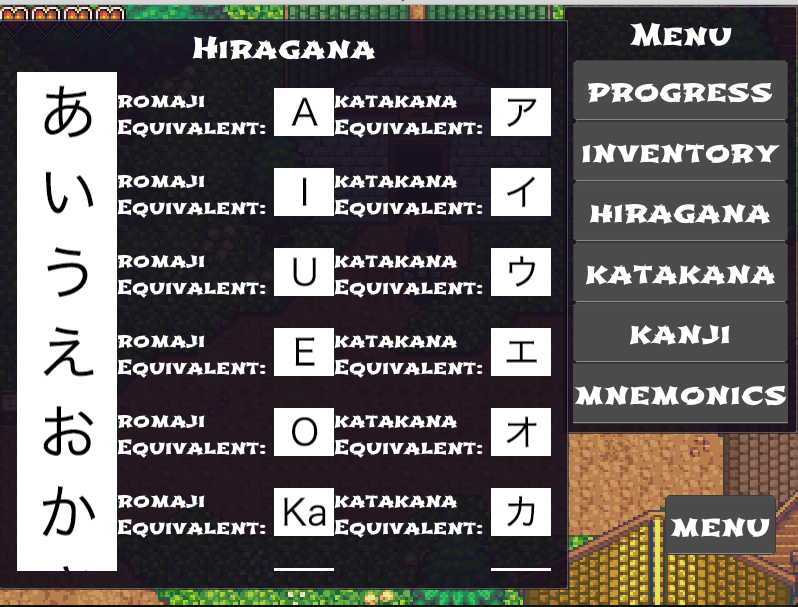
Figure 11 – JSON script snippet

After the Kana array list is created - due to "HIRAGANA" or "KATAKANA" being clicked - the back-end code will pass the info regarding "which menu button was clicked" to the window component. The window component will reveal data accordingly using a for loop that - each cycle - creates three images using texture objects, found in the Utility class, and the Kana array list. If "HIRAGANA" was clicked, the first image set will hold all the Hiragana characters. The second and third image sets will hold the Romaji and Katakana equivalents of the first image set, see figures 12.

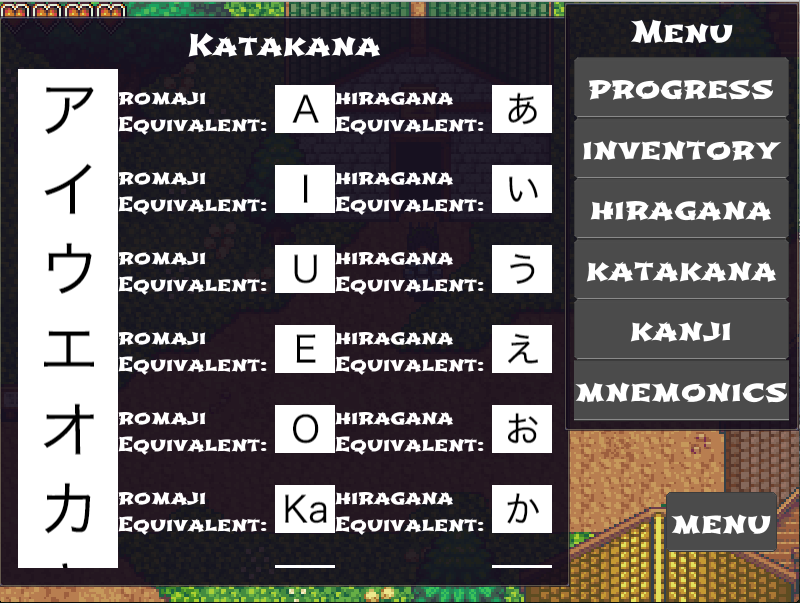
If KATAKANA" was clicked, the first image set will contain Katakana characters, the second will still contain the Romaji equivalent and the third will contain the Hiragana equivalent. To clarify which button was clicked, the first image set will contain larger images, see figure 13 & 14.



Figure 12 – Kana UI’s code snippet

  
Figure 13 - Hiragana list

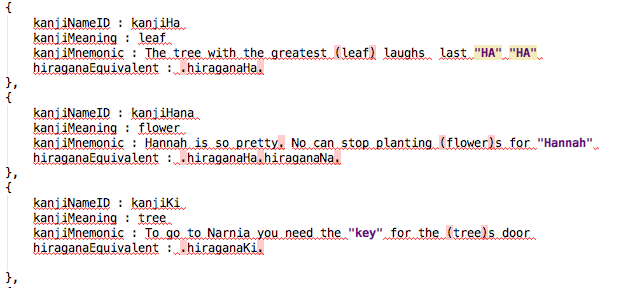
Implementing the Hiragana and Katakana list in conjunction with the battle system – more on that later – satisfies the non-functional requirements NF2, NF3 and NF4.

  
Figure 14­­ – Katakana list

## Kanji UI

By pressing the “KANJI” button in the “MENU” list, a window with the title “KANJI”, which was initially hidden, will be displayed. Using the “KANJI” window, the user can view all the kanji character available in this game with their meaning and Hiragana equivalents.

When the Kanji window is first initialised, an array list of Kanji letters is created using a factory class similar to how the Kana objects are created, see figure 16.

  
Figure 16 – kanji JSON script snippet

The kanji objects are initialised with four strings holding each Kanji’s keyword, meaning, mnemonic and Hiragana equivalent.

After the Kanji array list is created - due to the "KANJI" button being clicked - the back-end code will pass control to the window component. The window component will then reveal the data for all Kanji in the game using a for loop that - each cycle - creates Kanji and Hiragana sprites using texture objects, found in the Utility class. In order to show each Kanji’s unique composition of Hiragana characters, full stops were added to the *hiraganaEquivalent* variable in the kanji JSON script, see figure 16.

During each cycle of the for loop, an inner for loop is run with the number of the Hiragana characters the current Kanji object has – this data is obtained from the *hiraganaEquivalent* variable. The inner for loop will cycle through each Char in the *hiraganaEquivalent* String, and whenever a full stop is located, the substring method will be used to create a new String. And, each time a new String is created, a new image will be created by retrieving a texture region object with the same value as the new String from the Utility class, see figure 17 for the code snippet and figure 18 for the result.

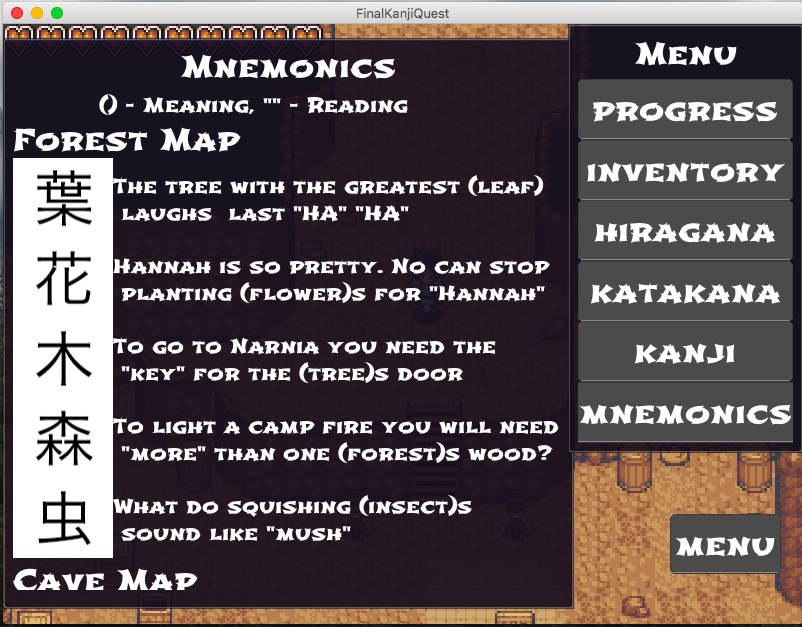
  
Figure 17 – Kanji UI’s inner for loop

Furthermore, the meaning of each Kanji is added alongside the Hiragana equivalent in order to satisfy the non-functional requirement NF5.

Figure 18 – Kanji List.

## Mnemonics UI

By pressing the Mnemonics Button in the Menu list, the Mnemonics window that was initially set as hidden will be displayed. Using the Mnemonics window, the user can see the available mnemonics devices that are used to help aid learning each kanji’s reading and writing, see figure 19. The mnemonics window makes use of the already created kanji letters array list to instantiate itself similarly to how the kanji window does but with the exception that in this case mnemonics are used instead of the hiragana equivalents.

  
Figure 19 – Mnemonics List

As can be seen in the key beneath the “Mnemonic” title, each mnemonic story connects the kanji reading with its meaning using () and “”. Implementing the mnemonics list satisfies both non-functional requirement NF6 and the functional requirement F7.

## Resize Action

While using the desktop version of the game, whenever the screen size is changed, the “*onResize*” super[[4]](#footnote-4) method is called. The “*onResize*” super method will call all the children resize methods and among these methods is the Player interfaces resize method. Whenever the Player interfaces resize method is called, the menu button, menu list and all the windows will reposition to their original relative position. A for loop is also run with the highest number of hearts, and in each cycle, the health heart in the corresponding index in the image array is repositioned. The aforementioned is done so to maintain the positions of the interface items at all times. Once the windows have been repositioned, the update size method for each window will be called. Each window will resize to the same proportion they originally had and then update their content accordingly, see figure 20 for the code snippet.



Figure 20 – resize method

Battle SystemWhile traversing the world of FKQ, the player character will encounter enemies that have to be defeated/answered in order to progress, see figure 21 for example.

****Figure 21 – Enemy encounter

When the player interface is first initiated, the battle stage is created, set to false and then set to the back of the screen to be hidden. When the player character starts moving, the player graphics component notifies the player interface of the event. If the player is moving in an Enemy-Encounter zone - more on this later, the player will check whether it has been at least three seconds since the last time an enemy counter happened. If three second has passed, the battle stage then creates a random number between one and hundred and if this random number is higher than the pre-set chance of encounter (65%), the set current opponent method is called, and the battle stage is set to visible and pushed to the front of the screen.

The set current opponent method will first check if all hiragana characters have been memorised, if they are already memorised, the method will check if all the katakana character have been memorized, if they are already memorised, the method will choose a kanji letter as the current opponent. This is done because it is not possible for a Japanese student to learn kanji before learning hiragana and katakana. This also satisfies the non-functional requirement NF9, see figure 22.

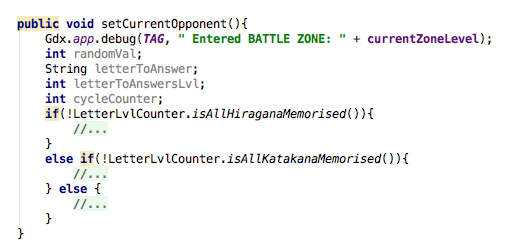


Figure 22 – Only show kanji if hiragana and katakana are memorised – 1

The set current opponent method uses the letter level counter class which keep tracks of all letter’s level to check if a certain letter or letters have been answered.

If either the hiragana or katakana letters have not been memorised, the set current opponent method will get a random kana character and then check if the selected character’s level is below three. If the selected character’s level is below three, then that means that it has not been memorised yet. After that, the method will check if the selected character appeared in the last encounter. If the selected character appeared in the last encounter, a different character is selected. Once a character that satisfies both conditions is found, below level 3 and did not appear in the last encounter, the battle stage will be notified about the selected character so to allow the battle stage to add the selected character alongside its correct equivalent and three random incorrect equivalents to the stage. The aforementioned satisfy the functional requirement F2, see figure 23 for code snippet.

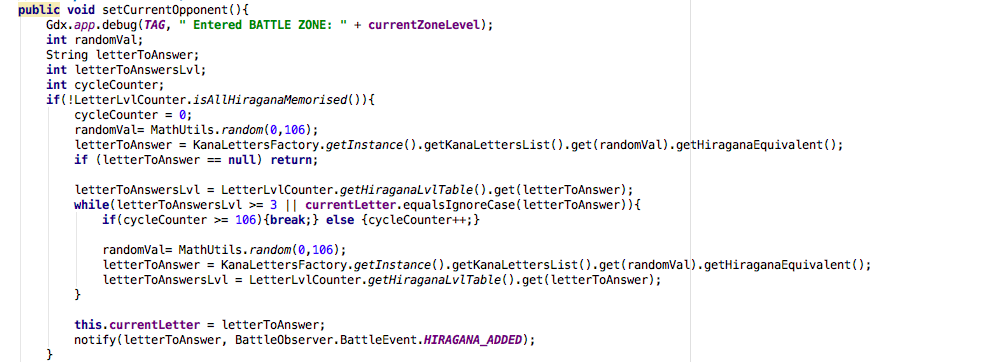


Figure 23 – set current opponent double check before choosing a letter

If both hiragana and katakana letters have been memorised, the set current opponent method will call the encounter factory class that will create an array list of kanji letters using an encounter zone POJO that is linked with an encounter zone JSON script, see figure 24. And then get a random kanji from the encounter factory kanji array list.

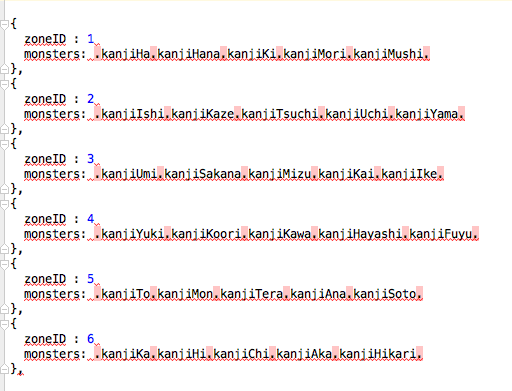


Figure 24 – Encounter JSON, more on ZoneID in next section

The encounter factory class will create the array list using a method that utilizes the full stops in the JSON script and the “indexOf” and substring methods, similar to how kanji’s hiragana equivalent are obtained,

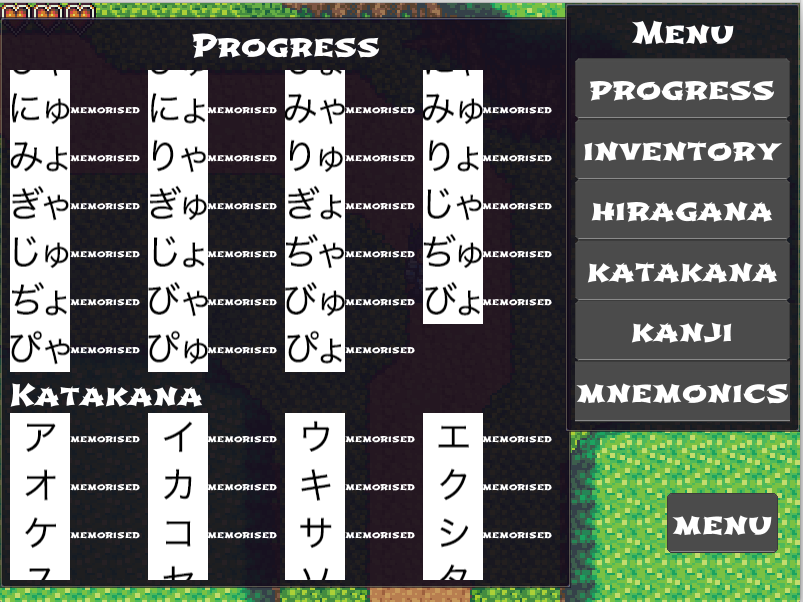
see figure 18. After the kanji is obtained from the encounter factory class, the set current opponent method will conduct the same double check, below level 5 and did not appear in the last encounter, done for the kana letter but with exception that kanji letters have to have a lower level then 5 – more on this later.

In order to defeat the enemy letter, the user has to choose the correct equivalent in another character type, in this case, the correct Romaji equivalent of the encountered hiragana character. Answering the letter correctly will increase the user’s answered letter level, in this case (figure 21), the hiragana character しょlevel will increase. After any successful, correct answer, the battle observer component will notify the player interface of the event. The player interface will then update the letter level counter class followed by the progress UI to display that the hiragana character しょhas been answered correctly.

The player interface will also play a sound that indicates that the character has been answered correctly. A different sound is played when the player chooses the incorrect answer. This is implemented in order to help the user to understand whether he answered the character correctly or not. This also satisfies the functional requirement F8.

After three consecutive, correct answers for hiragana and katakana or 5 consecutive correct answers for kanji the specific letter that reached the aforementioned level will stop appearing. The kanji letters require more consecutive correct answers because unlike hiragana and katakana, each kanji contains a meaning to memorise alongside how to read and write it. If the player clicks the incorrect answer/equivalent, the battle observer sends a failed answer attempt event to the player interface. The player interface then deducts/set to hidden one health heart. When the player’s health hearts are all set as hidden, the user is sent to a Game over screen. From there the user can click continue to return to the last place a save occurred or choice to load a different profile/user file.

If the user does not know the answer to the encountered character, the user can either open the list where encountered character is in and learn the answer or escape the encounter by clicking the Run button. In order to discourage the user from clicking the run button every time, there is a 15% chance that after clicking run, the escape will fail. For each failed escape, the player character loses one heart. The fastest way to mark all hiragana and katakana characters as memorised is to use the previously mention red stones. After using both stones, the progress UI will show that all the characters are memorised, see figure 24.

  
Figure 24 – all characters memorised.

## Design

A study was done at the Charles Sturt University on students studying veterinary science examined the effect of visual learning using an interactive application. Before the students were given the application, the average class mark for a poisonous plant recognition test was 11.9%. After using the visual learning application for eight weeks, the students were re-tested, and the average class mark was 93%. This clearly showed that visual learning could be very effective if implemented correctly [P].

Elements of visual learning have been implemented into the game because the aforementioned study have already proven that visual learning could be very effective

As can be seen from figure 18, above each five kanji letters a label with the name of a map is added. The label indicates where the following kanji can be found. From figure 11, the kanji meaning insect can be found when traversing the forest map, see figure 25 and 26. The remaining five maps can be seen in Appendix A.

Each kanji has been placed in a map that matches its meaning, e.g. the fire kanji was placed in a lava map and the kanji for ice has been placed in an ice map, in order to give the student a visual aid that will help him/her associate the kanji with its meaning better.

When the player character’s collision box collides with an encounter zone collision box, a number associated with the encounter zone collision box will be sent by the player physics component to the player interface. Using the number sent by the physics component the player interface will set the battle encounter background to a map associated with the number, e.g. zone 1 is the forest map and zone 2 is the cave map.

See figure 26 for the code snippet.

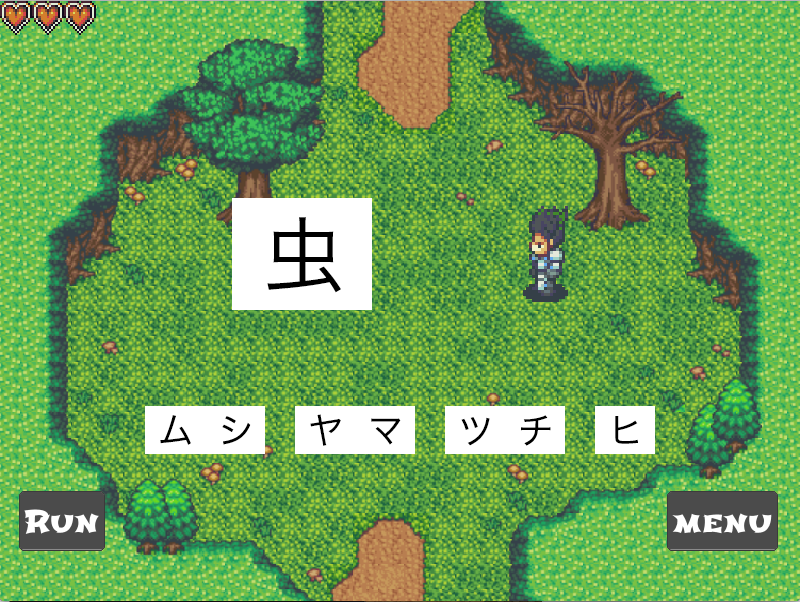


Figure 25 – encounter with the kanji for insect

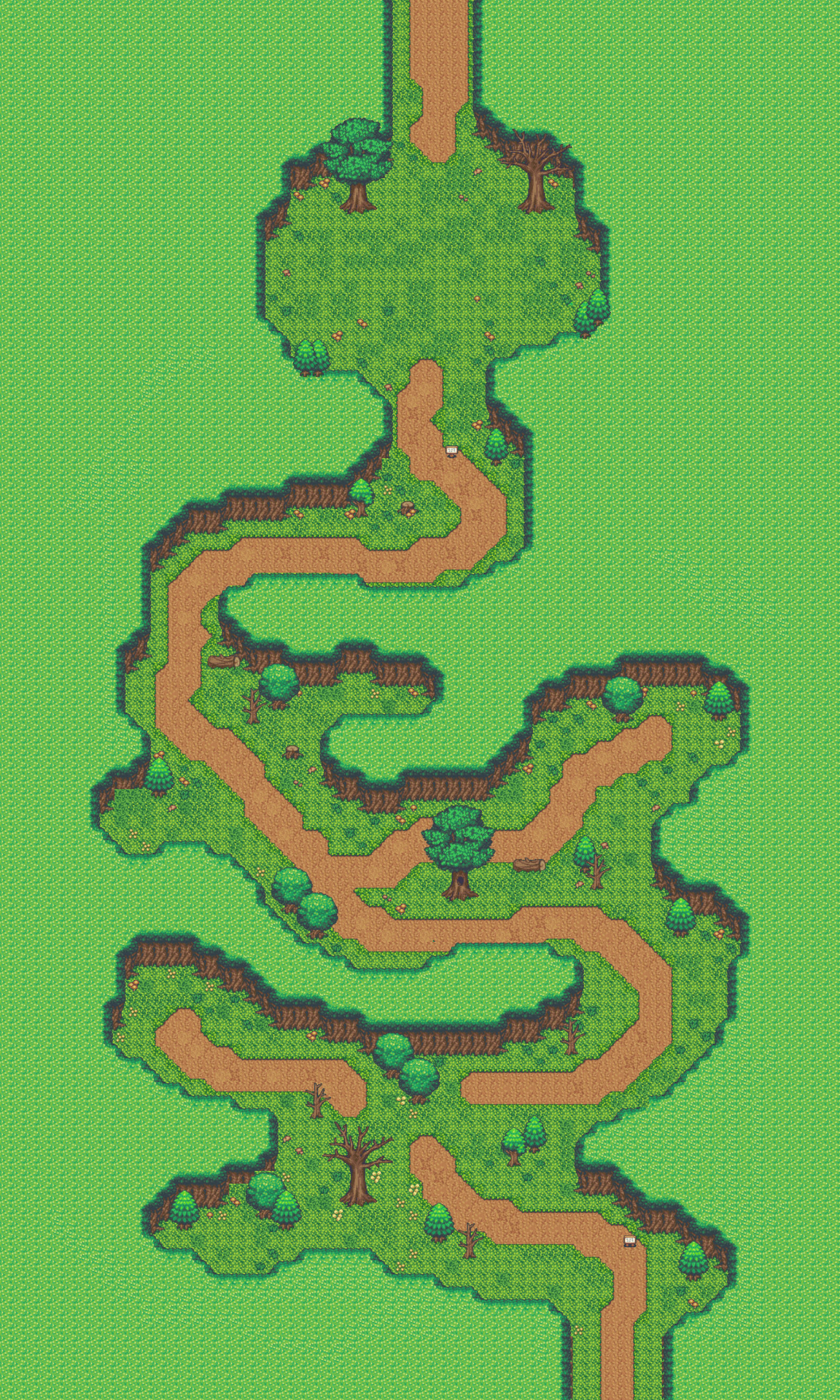
  
Figure 26 – forest map



Figure 27 – battle background selector

# Evaluation

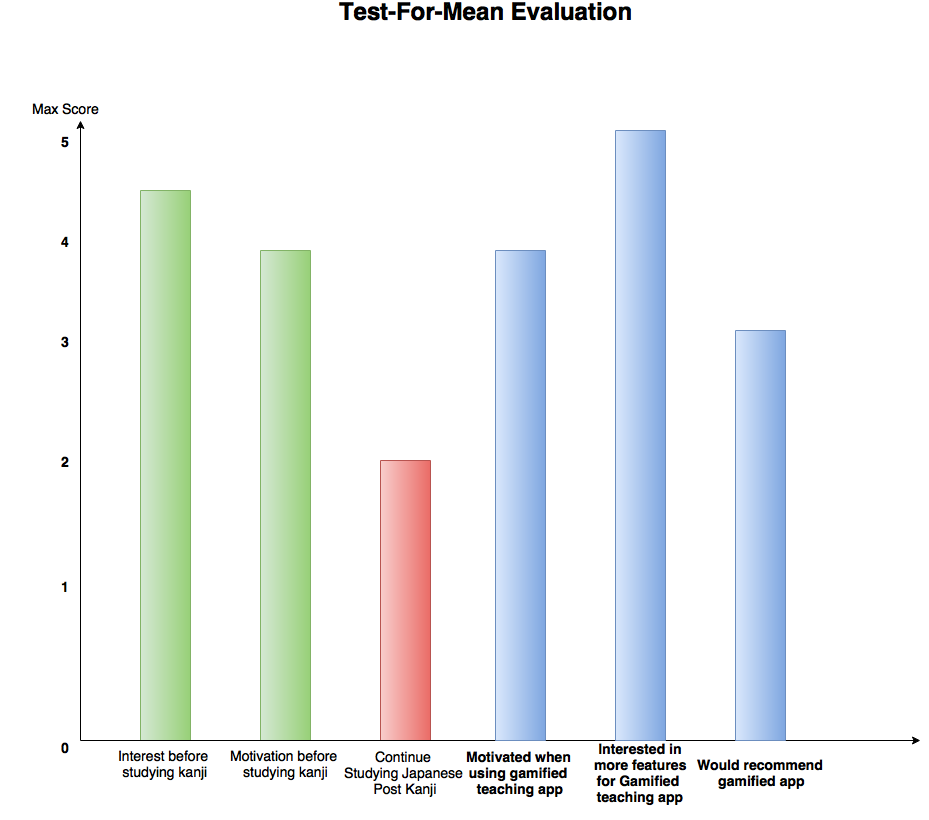


Figure 28 – Data analysis

After collecting all the completed questionnaires from the recruited students and using the TFM statistical data analysis method to analyse the result, the above figure was created. The mathematical equation for TFM analysis is total score divide by number of scores.

Looking at the three blue bars in the figure, it can be clearly seen that the hypothesis, Gamification increases motivation and interest in regard to studying the Japanese writing system, it is **accepted** as the total result for each bar exceeds the mid score, 2.5.

As expected, it was clear that students lost motivation and interest once kanji was introduced in class. It was also clear that the students with higher level of Japanese language proficiency, did not find the app very helpful. This was not a surprise, because due to time constraints only 30 kanji characters out of 2000 currently used are included

Almost all the students agree that if more elements and kanji characters were included in the game, they would be highly interested in using it. All the students also mostly agree that the traditional (using books) kanji learning method is not as preferable as using a gamified teaching application.

## Testing

To ensure the software works correctly, it was essential to test all the features defined in the requirement specification section. Besides functionality, the performance of the system was also a significant concern that needed to be tested.

Thanks to the recruited participants, the software was tested on many different systems. Furthermore, by offering rewards, such as Amazon gift cards, the participants promptly emailed whenever they detected an issue with the software. Receiving prompt feedback from the participants allowed some of the significant concerns with the software to be tested and fixed. Some of the concerns were;

1 - Whether the software performs the functions as expected.

2 - Whether the software wastes resources.

3 - Whether the software can be installed on Android devices with different screens and versions.

4 - Whether the software is easy to use.

5 - Whether the software clearly shows each sprite accurately.

Conclusion and Future WorkExperience with complex Java properties such as how to read and write to JSON files using Plain Old Java Objects was obtained from this project. Using both the OpenGL API for graphics and OpenAL API for music and sound extensively for the project provided very valuable insight, as OpenAL and OpenGL are very commonly used with game engines, game frameworks and graphical desktop applications.

Implementing the Android version of the game followed by the Desktop version provided more insight about the Android life cycle as the Libgdx life cycle is based on the Android life cycle. Because the Android OS does not allow the application to be saved by writing to files in the internal memory for security concerns, I was not able to implement the save and load functionality for the Android version. Unlike the desktop version, the Android version of the game lacks a save and load feature. Whenever a user clicks his or her Android phone’s home button or back button, the user will lose all his or her progress in the game and will have to start over with a new save file. As future work, it would be necessary for the Android version to have the save and load functionality if it is to be useful to language students and if it is to be released in the play store.

As the questionnaire’s evaluation has revealed, the students eventually reached a point where they lost their motivation to continue using the application. One way to turn this around is to include an engaging story to the game that will captive the users and bring them back for more.

This project could also be extended by abstracting the code in a manner that allows the game to run with any other language such as Korean. A solution to the first constraints and a possible future extension of the game is to have multiple versions of the game, each with assets of different sizes, uploaded to a server. This extension will have to include a feature that checks the user's phone or tablet specifications the first time the app is run and if necessary, it replaces the game with the most compatible version of the game from an online server to offer the best performance.

# Definitions[[5]](#footnote-5)

A – Kanji: A system of Japanese writing using Chinese characters, used primarily for content words.

B – Kana: The system of syllabic writing used for Japanese, having two forms, hiragana and katakana.

Compare with kanji

C – Hiragana: The more cursive form of kana (syllabic writing) used in Japanese, primarily used for function words and inflections.

D – Katakana: The more angular form of kana (syllabic writing) used in Japanese, primarily used for words of foreign origin.

E – Romanize: Put (text) into the Roman alphabet or into roman type.

F – Romaji: A system of Romanized spelling used to transliterate Japanese.

G – Mnemonic: A system such as a pattern of letters, ideas, or associations which assists in remembering something.

# Bibliography

H - Andrew W. Phillips, Shalini Reddy, Steven J. Durning. (2016) [Improving response rates and evaluating nonresponse bias in surveys: AMEE Guide No. 102](https://www.tandfonline.com/doi/abs/10.3109/0142159X.2015.1105945). Medical Teacher 38:3, pages 217-228.

I - Statistical methods for survival data analysis. --3rd ed./Elisa T. Lee and John Wenyu Wang.

p. cm.--(Wiley series in probability and statistics)

J - Rose, Heath. 2013. L2 learners’ attitudes toward, and use of, mnemonic strategies when

learning Japanese kanji. Modern Language Journal97(4). 981–992.

K - Bower, V., Rutson-Griffiths, A., 2016. The relationship between the use of spaced repetition software with a TOEIC word list and TOEIC score gains. Computer Assisted Language Learning 29, 1238–1248.

L - De Freitas, A., de Freitas, M., 2013. Classroom Live: a software-assisted gamification tool. Computer Science Education 23, 186–206.

M - https://www.packtpub.com/game-development/mastering-libgdx-game-development

N - https://msdn.microsoft.com/en-us/library/bb299886.aspx#intro\_to\_json\_topic7

O - https://github.com/libgdx/libgdx/wiki/Textures,-textureregion-and-spritebatch

p - Geoffrey E. Burrows, Gaye L. Krebs & Bruce K. Kirchoff (2015): ‘Visual Learning – Agricultural Plants of the Riverina’ – A New Application for Helping Veterinary Students Recognise Poisonous Plants, Bioscience Education

1. The Japanese False beginner course is aimed at students who took Japanese GCSE [↑](#footnote-ref-1)
2. The Research Ethics Office granted the Ethical approval. [↑](#footnote-ref-2)
3. The black borders are used only for visual clarification and are not included in the game [↑](#footnote-ref-3)
4. http://stackoverflow.com/questions/3767365/super-in-java [↑](#footnote-ref-4)
5. https://en.oxforddictionaries.com/search?utf8=✓&filter=dictionary&query=dicorton [↑](#footnote-ref-5)