



**ESCUELA SUPERIOR DE INGENIERÍA**  
**INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS**

Sistema de apoyo al aprendizaje del idioma japonés para  
consola portátil

Diego Barrios Romero

12 de octubre de 2009





## ESCUELA SUPERIOR DE INGENIERÍA

### INGENIERO TÉCNICO EN INFORMÁTICA DE SISTEMAS

Sistema de apoyo al aprendizaje del idioma japonés para consola portátil

- Departamento: Lenguajes y sistemas informáticos
- Director del proyecto: Manuel Palomo Duarte
- Autor del proyecto: Diego Barrios Romero

Cádiz, 12 de octubre de 2009

Fdo: Diego Barrios Romero



# Agradecimientos

Este proyecto significa la culminación de mi carrera, por lo que me gustaría dedicárselo a todas las personas que me han ayudado a conseguir acabarla.

En primer lugar me gustaría agradecerle a mi familia el apoyarme y ayudarme durante estos años, y el esfuerzo que han hecho para que yo haya podido llegar a este momento.

También me gustaría agradecérselo a mis compañeros, con los que tantos ratos inolvidables he pasado, y que tanto me han ayudado.

Por último quiero dedicarle este proyecto a todos los estudiantes de japonés y a todos los amantes de esta fascinante cultura, a los que espero que les sirva bastante.



## Licencia

Este documento ha sido liberado bajo Licencia GFDL 1.3 (GNU Free Documentation License). Se incluyen los términos de la licencia en inglés al final del mismo.

Copyright (c) 2009 Diego Barrios Romero.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".



## Notación y formato

- La videoconsola Nintendo DS dispone de dos pantallas, la superior y la inferior, que proporciona además una interfaz táctil. De aquí en adelante se denominarán como *pantalla principal o superior* y *pantalla inferior o sub pantalla*.
- Un ideograma es un símbolo que corresponde a un concepto, en japonés se dice *kanji*. Los usaremos indistintamente, prefiriendo *kanji* por ser más específico.



# Índice general

<b>Índice general</b>	<b>VII</b>
<b>Indice de figuras</b>	<b>XI</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Sobre este documento . . . . .	3
<b>2. Conceptos básicos</b>	<b>5</b>
2.1. Sistema de escritura japonés . . . . .	5
2.1.1. Historia . . . . .	5
2.1.2. Ideogramas japoneses . . . . .	6
2.1.3. Silabarios hiragana y katakana . . . . .	6
2.1.4. Romaji . . . . .	7
2.2. Programación . . . . .	7
<b>3. Organización temporal</b>	<b>9</b>
<b>4. Análisis del sistema</b>	<b>13</b>
4.1. Especificación de requisitos del sistema . . . . .	13
4.1.1. Requisitos de interfaces externas . . . . .	13
4.1.2. Requisitos funcionales . . . . .	13
4.1.3. Restricciones de rendimiento . . . . .	15
4.1.4. Restricciones de diseño . . . . .	15
4.1.5. Atributos del sistema software . . . . .	15
4.2. Análisis . . . . .	16
4.2.1. Modelo de Casos de Uso . . . . .	16
4.2.2. Modelo conceptual de datos en UML . . . . .	25
4.2.2.1. Diagrama de clases conceptuales . . . . .	25
4.2.2.2. Modelo de comportamiento del sistema . . . . .	26

<b>5. Diseño del sistema</b>	<b>33</b>
5.1. Capa de gestión de datos . . . . .	33
5.1.1. Clase Card . . . . .	34
5.1.2. Clase XMLParser . . . . .	37
5.2. Capa de dominio . . . . .	42
5.2.1. Diagrama de clases de diseño . . . . .	42
5.2.2. Clase DSMemorizer . . . . .	44
5.2.3. Clase TextBoxHandler . . . . .	46
5.2.4. Clase TextBox . . . . .	49
5.3. Capa de presentación . . . . .	53
5.3.1. Detalle de las clases en UML . . . . .	53
5.3.2. Clase MainScreenHandler . . . . .	53
5.3.3. Clase SubScreenHandler . . . . .	58
5.3.4. Clase ScreensHandler . . . . .	63
5.3.5. Clase SoundHandler . . . . .	64
<b>6. Implementación</b>	<b>67</b>
6.1. El problema de los colores . . . . .	67
6.2. Renderizado de texto . . . . .	71
6.3. Alineado de las cajas de texto . . . . .	71
6.4. Namespace Graphics . . . . .	73
6.4.1. Funciones . . . . .	73
6.4.2. Documentación de las funciones . . . . .	74
<b>7. Pruebas</b>	<b>79</b>
<b>8. Conclusiones</b>	<b>81</b>
8.1. Mejoras futuras . . . . .	81
8.1.1. Funcionalidad . . . . .	81
8.1.2. Interfaz . . . . .	82
8.1.3. Bases de datos . . . . .	82
8.2. Herramientas utilizadas . . . . .	83

<b>Apéndice Formato de los ficheros XML</b>	<b>85</b>
.1. Descriptor de paquete . . . . .	85
.2. Bases de datos para kanjis . . . . .	85
.3. Bases de datos para vocabulario . . . . .	86
<b>Bibliografía</b>	<b>87</b>
<b>GNU Free Documentation License</b>	<b>89</b>
1. APPLICABILITY AND DEFINITIONS . . . . .	89
2. VERBATIM COPYING . . . . .	91
3. COPYING IN QUANTITY . . . . .	91
4. MODIFICATIONS . . . . .	92
5. COMBINING DOCUMENTS . . . . .	94
6. COLLECTIONS OF DOCUMENTS . . . . .	94
7. AGGREGATION WITH INDEPENDENT WORKS . . . . .	94
8. TRANSLATION . . . . .	95
9. TERMINATION . . . . .	95
10. FUTURE REVISIONS OF THIS LICENSE . . . . .	96
11. RELICENSING . . . . .	96
ADDENDUM: How to use this License for your documents . . . . .	96



# Índice de figuras

1.1. Imagen de una Nintendo DS Lite . . . . .	1
1.2. Logo DSMemorizer . . . . .	2
3.1. Diagrama de Gannt 1. Desarrollo del proyecto . . . . .	11
3.2. Diagrama de Gannt 2. Desarrollo del proyecto . . . . .	12
4.1. Diagrama de casos de uso . . . . .	16
4.2. Diagrama de clases conceptuales . . . . .	25
4.3. Diagrama de secuencia para el caso de uso Menú principal . . . . .	26
4.4. Diagrama de secuencia para el caso de uso Memorización de kanjis . . . . .	27
4.5. Diagrama de secuencia para el caso de uso Preguntas de kanjis . . . . .	28
4.6. Diagrama de secuencia para el caso de uso Memorización de vocabulario . . . . .	29
4.7. Diagrama de secuencia para el caso de uso Preguntas de vocabulario . . . . .	30
4.8. Diagrama de secuencia para el caso de uso Opciones de kanjis . . . . .	31
4.9. Diagrama de secuencia para el caso de uso Opciones de romanización . . . . .	32
5.1. Diagrama de la capa de datos . . . . .	34
5.2. Diagrama de colaboración de la clase Card . . . . .	35
5.3. Diagrama de colaboración de la clase XMLParser . . . . .	38
5.4. Diagrama de clases de diseño . . . . .	43
5.5. Detalle de la clase DSMemorizer en UML . . . . .	44
5.6. Diagrama de colaboración de la clase TextBoxHandler . . . . .	47
5.7. Detalle de la clase TextBox en UML . . . . .	49
5.8. Detalle de la clase mainScreenHandler en UML . . . . .	54
5.9. Detalle de la clase SubScreenHandler en UML . . . . .	59
5.10. Detalle de la clase ScreensHandler en UML . . . . .	64
5.11. Detalle de la clase SoundHandler en UML . . . . .	65

6.1. Ejemplo de imagen con paleta y “tick” y “cross” incrustados . . . . .	68
6.2. Paleta de la imagen generada por GRIT . . . . .	69
6.3. Ejemplo de paleta generada por GRIT en hexadecimal . . . . .	70
6.4. Ejemplo de cajas de texto no alineadas . . . . .	72
6.5. Ejemplo de cajas de texto alineadas . . . . .	73

# Capítulo 1

## Introducción

Japón, la segunda economía mundial está de moda. Cada vez hay más personas que se interesan por esta cultura y se animan a aprender su lengua, con lo que están aumentando la cantidad de academias en las que se enseña este idioma.

Por otra parte, las videoconsolas portátiles están en auge, la Nintendo DS ha vendido 107,75 millones de unidades hasta ahora <sup>1</sup>. Su competidora directa en el mundo portátil, la Sony PSP, también cuenta con bastante unidades vendidas, sumando un total de 55,9 millones de unidades vendidas en todo el mundo <sup>2</sup>.

Este auge no se debe sólo a los jóvenes sino que con títulos como “Brain Training” o la serie “Touch Generations”, Nintendo ofrece juegos destinados a personas de cualquier género y edad, desde los más pequeños a los más mayores, aprovechando la intuitividad de su pantalla táctil.



Figura 1.1: Imagen de una Nintendo DS Lite

A todo esto, siempre me ha atraído la cultura japonesa y hace unos años empecé a aprender japonés, aprobé el JLPT<sup>3</sup> 4, y seguí estudiando. De pronto se me ocurrió

---

<sup>1</sup>Unidades vendidas en todo el mundo hasta el 30 de Junio de 2009 incluyendo Nintendo DS, Nintendo DS Lite y Nintendo DSi. Fuente: Consolidated Financial Highlights, Nintendo, July 30, 2009 <http://www.nintendo.co.jp/ir/pdf/2009/090730e.pdf>

<sup>2</sup>Unidades vendidas en todo el mundo hasta el 3 de Agosto de 2009. Fuente: Tor Thorsen (Gamespot) [http://www.gamespot.com/news/6214693.html?om\\_act=convert&om\\_clk=newstop&tag=newstop;title;5](http://www.gamespot.com/news/6214693.html?om_act=convert&om_clk=newstop&tag=newstop;title;5)

<sup>3</sup>Japanese Language Proficiency Test. <http://www.jlpt.jp/e/>

escribir un curso de japonés, con las lecciones que me dieron a mí, y empecé a escribir un curso de japonés<sup>4</sup> para que las personas que no dispusiesen de medios, pudiesen aprender esta apasionante lengua.

Al aprender un idioma, todos sabemos que lo más tedioso es memorizar el vocabulario, y si a ésto le sumamos la necesidad de memorizar ideogramas como en el caso del japonés donde es necesario conocer los aproximadamente 2000 kanjis de uso normal (necesarios para leer el periódico, por ejemplo) la tarea se vuelve mucho más ardua y con el tiempo termina en abandono.

Sumando todos estos factores, me propuse hacer como proyecto fin de carrera un juego para la Nintendo DS, dedicado a completar el curso facilitando la tarea de la memorización. Y que permitiese al jugador aprender de forma sencilla los ideogramas y el vocabulario necesario.

Así nació DSMemorizer.

DSMemorizer (que será el nombre del sistema resultante) será un juego desarrollado para la videoconsola portátil Nintendo DS dedicado a ayudar a la memorización de ideogramas y vocabulario, centrado principalmente en el idioma japonés.

Para ello dispondrá de diversos modos de juego, algunos dedicados a la memorización donde se presentan los ideogramas y palabras con sus versiones escritas en kanji, sus lecturas, traducciones y ejemplos (también escritos en kanji, sus lecturas en kana y sus traducciones). También habrá otros en los que se preguntará por alguno de estos elementos, que el jugador deberá elegir correctamente.

DSMemorizer permite facilitar la tarea de memorizar vocabulario y kanjis para los estudiantes de japonés. De esta forma se pueden aprovechar los tiempos muertos, como en los transportes, para entretenerse a la vez que se aprende.

DSMemorizer puede ser ampliado en sus modos de vocabulario para estudiar cualquier idioma simplemente creando un archivo con un determinado formato con las palabras que se deseen sin necesidad de modificar nada.

Además, la interfaz para los modos dedicados a ideogramas podría ser fácilmente modificada para adaptarse a otras lenguas que contengan ideogramas, ya que el juego se desarrollará para que pueda ser usado con cualquier idioma, manipulando siempre cadenas de caracteres UTF-8 y usando para el renderizado la universal librería freetype y fuentes vectoriales.



Figura 1.2: Logo DSMemorizer

---

<sup>4</sup>Curso de japonés por libre. <http://www.japonesporlibre.com>

## 1.1. Sobre este documento

Este documento se organiza de la siguiente forma:

- En el capítulo 1 se hace una pequeña introducción sobre el proyecto, explicando la motivación para hacer este proyecto.
- En el capítulo 2 se explican los conceptos básicos que hacen del japonés un idioma más peculiar a la hora de aprenderlo.
- En el capítulo 3 se muestra la organización temporal del proyecto, su planificación.
- En el capítulo 4 se hace un análisis sobre las funcionalidades que deberá tener el juego, incluyendo los casos de uso y el análisis de los requisitos de sistema.
- En el capítulo 5 se muestra el diseño del sistema, estructurado en tres capas, capa de gestión de datos, de dominio y de presentación.
- En el capítulo 6, se explican los problemas que han surgido a la hora de implementar el juego y cómo se han resuelto los problemas más importantes.
- En el capítulo 7 se describen las pruebas ejecutadas en el sistema.
- En el capítulo 8 se muestran las conclusiones obtenidas en el desarrollo del proyecto, incluyendo diversas mejoras posibles.
- Se incluye un **apéndice adicional** para explicar el formato de los ficheros XML para las bases de datos.



# Capítulo 2

## Conceptos básicos

A la hora de aprender un idioma todos sabemos que lo más tedioso es memorizar el vocabulario. Y si a ésto le sumamos la necesidad de memorizar ideogramas como en el caso del japonés donde es necesario conocer los aproximadamente 2000 kanjis de uso normal (necesarios para leer el periódico, por ejemplo) la tarea se vuelve mucho más ardua y con el tiempo termina en abandono.

### 2.1. Sistema de escritura japonés

La escritura japonesa combina 3 tipos de escritura, dos silabarios, que son como nuestro abecedario pero con sílabas y un conjunto de símbolos o ideogramas importados de la cultura china <sup>1</sup>.

#### 2.1.1. Historia

Los japoneses no conocían la escritura antes de la llegada de los chinos a sus costas. Por lo que su lengua no se había desarrollado demasiado y su vocabulario era limitado.

Los chinos trajeron sus ideogramas y los japoneses empezaron a usarlos para escribir. Éstos se adaptaron de varias formas:

- Se usaron ideogramas chinos para representar conceptos japoneses, y se le asignaron por tanto una correspondiente palabra japonesa, que corresponde a la lectura japonesa o en japonés, “*kunyomi*”.

Éstos fueron los ideogramas que se tomaron por semántica.

Hubo veces que se tomó el mismo ideograma para diferentes conceptos y por eso existen ideogramas que tienen varias lecturas “*kun*”.

---

<sup>1</sup>Estos conceptos pueden verse ampliados en [5]

- Había palabras japonesas que no podían ser representadas por los ideogramas chinos por lo que se tomaron ideogramas por su pronunciación china adaptada a la japonesa para unir varios y escribir palabras japonesas.

Éstos fueron los ideogramas que se tomaron por fonética.

Éstos ideogramas se fueron estilizando con el tiempo en dos corrientes separadas, el estilo de escritura de las mujeres y el estilo de escritura de los hombres, y se convirtieron en los actuales silabarios “*hiragana*” y “*katakana*” respectivamente.

- El vocabulario japonés no era muy rico por aquella época así que con la llegada de la cultura china se enriqueció enormemente con palabras chinas. Éstas palabras se tomaron con sus símbolos y su pronunciación se adaptó a la pronunciación japonesa.

De esta forma, los ideogramas se repiten entre las palabras pero se pronuncian de forma diferente. Dividiendo la pronunciación en sílabas entre los kanjis que formaban las palabras se obtuvo la lectura china de los ideogramas, o en japonés, “*onyomi*”.

Por ésto los kanjis suelen tener varias lecturas “*on*”.

### 2.1.2. Ideogramas japoneses

Los ideogramas japoneses o “*kanjis*”, como ya se ha dicho, son símbolos que representan un concepto. Tienen dos tipos de lecturas, la lectura “*on*” y la lectura “*kun*” siendo cada una de ellas la adaptada de la pronunciación china, que se usa normalmente cuando el kanji está acompañado de otros, formando una palabra, y la lectura japonesa correspondiente a la pronunciación del concepto en japonés original, usada cuando el kanji está solo.

Por esto, al estudiar un kanji se hace necesario memorizar, además de su significado, sus lecturas (de cada tipo puede, y suele, haber varias).

También es necesario conocer el orden de escritura de los trazos del kanji para poder escribirlo correctamente, aunque existen unas normas generales que suelen adecuarse bien salvo algunas excepciones.

Los ideogramas se usan para escribir los sustantivos y raíces de los adjetivos, verbos y adverbios.

### 2.1.3. Silabarios hiragana y katakana

- El silabario hiragana se usa para escribir las desinencias para las conjugaciones de los adjetivos y verbos, las partículas, etc.

Es el sistema de escritura por defecto. Los sustantivos y palabras que deberían escribirse en kanji también pueden ser escritas en hiragana, no sin una fuerte pérdida de precisión, por lo que se mantienen los kanjis en el japonés actual.

- El silabario katakana se usa para escribir palabras extranjeras en japonés. Para escribir una palabra extranjera se adapta su pronunciación a la fonética japonesa y luego se escribe con los símbolos de este silabario.

Ambos silabarios son equivalentes y sólo difieren entre ellos la grafía empleada. Los símbolos de ambos se conocen como “kanas”. Ambos silabarios tienen 46 símbolos base.

Algunos de ellos pueden además “impurizarse” o “semiimpurizarse”, con lo que se consiguen representar 25 sílabas más.

Otros cuantos símbolos pueden combinarse con otros tres, “ya”, “yu” y “yo”, con una grafía más pequeña con lo que se consiguen representar 33 sílabas más.

En total cada silabario representa 104 sílabas.

#### 2.1.4. Romaji

“Romaji” es la denominación para la grafía occidental. Acrónimo de “Roman” y “ji” (*letra, símbolo*). Si se usa es sólo porque queda exótico, así que puede verse a veces en carteles publicitarios.

También se usa en la educación para los extranjeros.

## 2.2. Programación

La plataforma Nintendo DS es realmente muy potente, aunque debido a su naturaleza cerrada no sea posible aprovechar sus muchísimas posibilidades. Aun así, existe un “toolchain” y unas utilidades libres muy buenas para programar. Además, existen varias librerías que proporcionan el acceso a los recursos de la consola. Las dos más importantes son libnds[8] y PAlib (que trabaja sobre libnds).

En mi caso usaré libnds.

Adicionalmente a estas librerías básicas existen otras como EFS Lib[11], que uso para manejar los ficheros y que proporciona una interfaz muy buena y consistente, o la librería UTF8-CPP[10], para manejar cadenas de caracteres UTF-8 en C++ fácilmente.

Para el renderizado de las fuentes uso la librería FreeType[2] del FreeType Project. Una extraordinaria librería que no necesita más que la librería de C y que es capaz de renderizar cualquier tipo de fuente.

Como lenguaje de programación podía escoger C o C++, optando por este último debido a su mayor nivel de abstracción, mi afición por la orientación a objetos y porque me encanta este lenguaje.

El código está escrito en inglés, dado que el propósito del juego es que sea internacionalizable, y la documentación también, siguiendo la sintaxis de doxygen [3].



# Capítulo 3

## Organización temporal

Para la elaboración de este proyecto me ha sido necesario estudiar el funcionamiento de la plataforma Nintendo DS. La programación de consolas portátiles no se estudia como tal en la carrera, y presenta una serie de peculiaridades hardware que suponen un esfuerzo de aprendizaje considerable:

**Escasa cantidad de RAM** La cantidad de memoria RAM disponible es muy escasa, sólo 4 MB, por lo que se hace imposible la suposición universal durante la carrera de que los datos son accesibles desde la RAM. Es necesario ahora cargar muchas cosas de ficheros, no se pueden cargar muchas cosas a la vez y hay que estar siempre liberando memoria.

**Dos procesadores independientes** Esta plataforma dispone de dos procesadores independientes, un ARM9 y un ARM7 que es necesario controlar.

Además, por ejemplo, el touchpad sólo puede acceder a él el procesador ARM7, pero la programación principal se hace en el ARM9 que es más potente.

**Relativamente baja velocidad de procesamiento** Aunque disponga de dos procesadores y tenga una gran potencia, la velocidad de procesamiento no es comparable a la de los procesadores de ordenadores de escritorio a los que estamos acostumbrados, por lo que es necesario extremar la eficiencia algorítmica.

Por ello he tenido que dedicar un tiempo a mi propia formación a la vez que iba perfilando los detalles del análisis y del diseño.

A partir de ahí ya he podido empezar a implementar cosas.

Además, he tenido muy poco tiempo, contando sólo con 2 meses para hacerlo todo, desde el análisis del sistema a las pruebas finales y la elaboración de este documento. En la planificación inicial he tenido que modificar muchas cosas porque la integración de las librerías no ha sido fácil, en particular la librería FreeType me ha sido bastante difícil de entender y más de integrar en el juego.

También he tenido que cambiar el tiempo que estimé para implementar la clase de procesado de ficheros XML ya que ha sido necesario implementar un sistema de indexado para los ficheros, así como para los kanjis, manejándolos según su grado de dificultad

y número de trazos para que el usuario pueda seleccionar específicamente un rango de ellos.

La planificación queda por tanto de la siguiente forma:

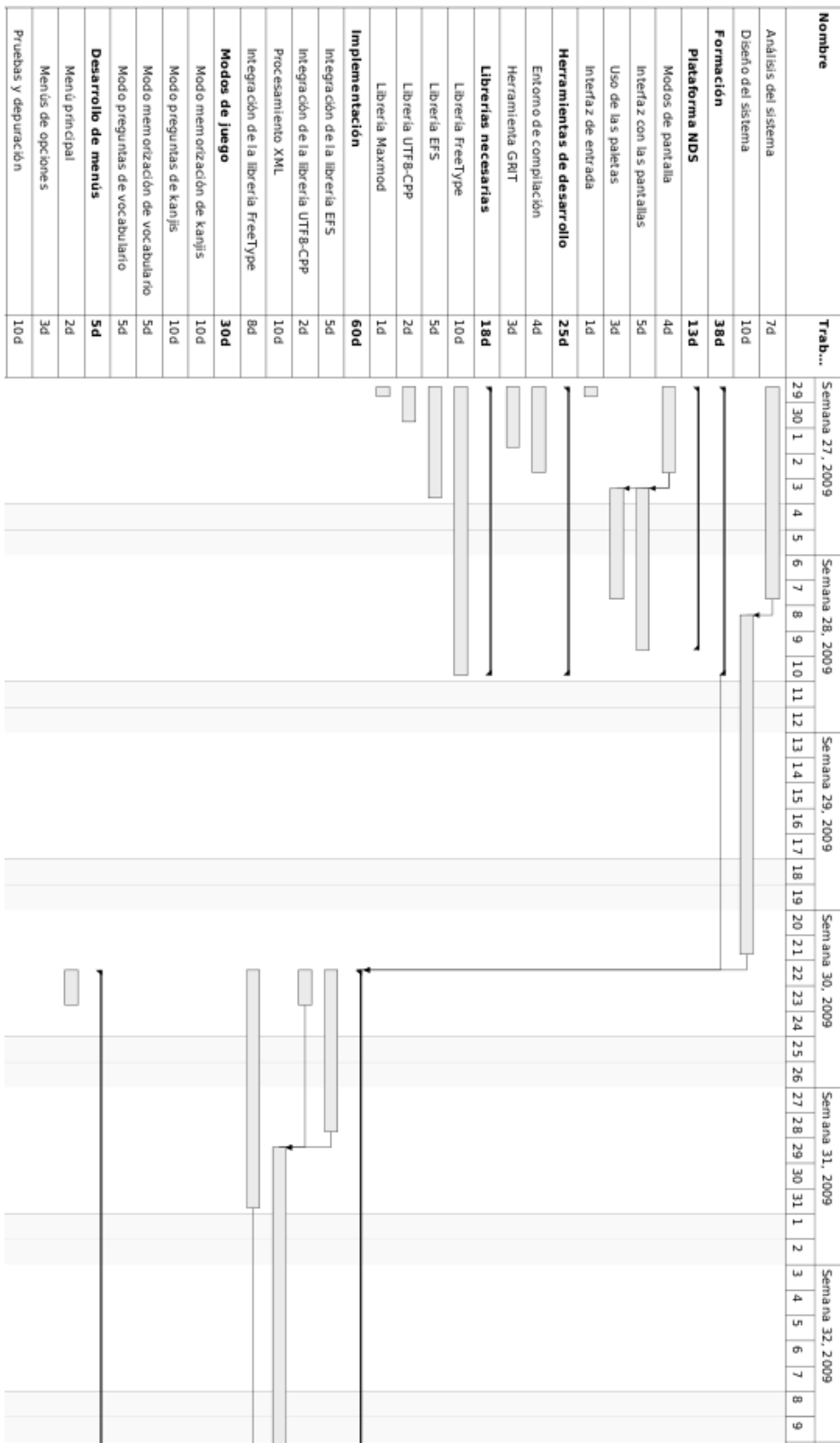


Figura 3.1: Diagrama de Gannt 1. Desarrollo del proyecto

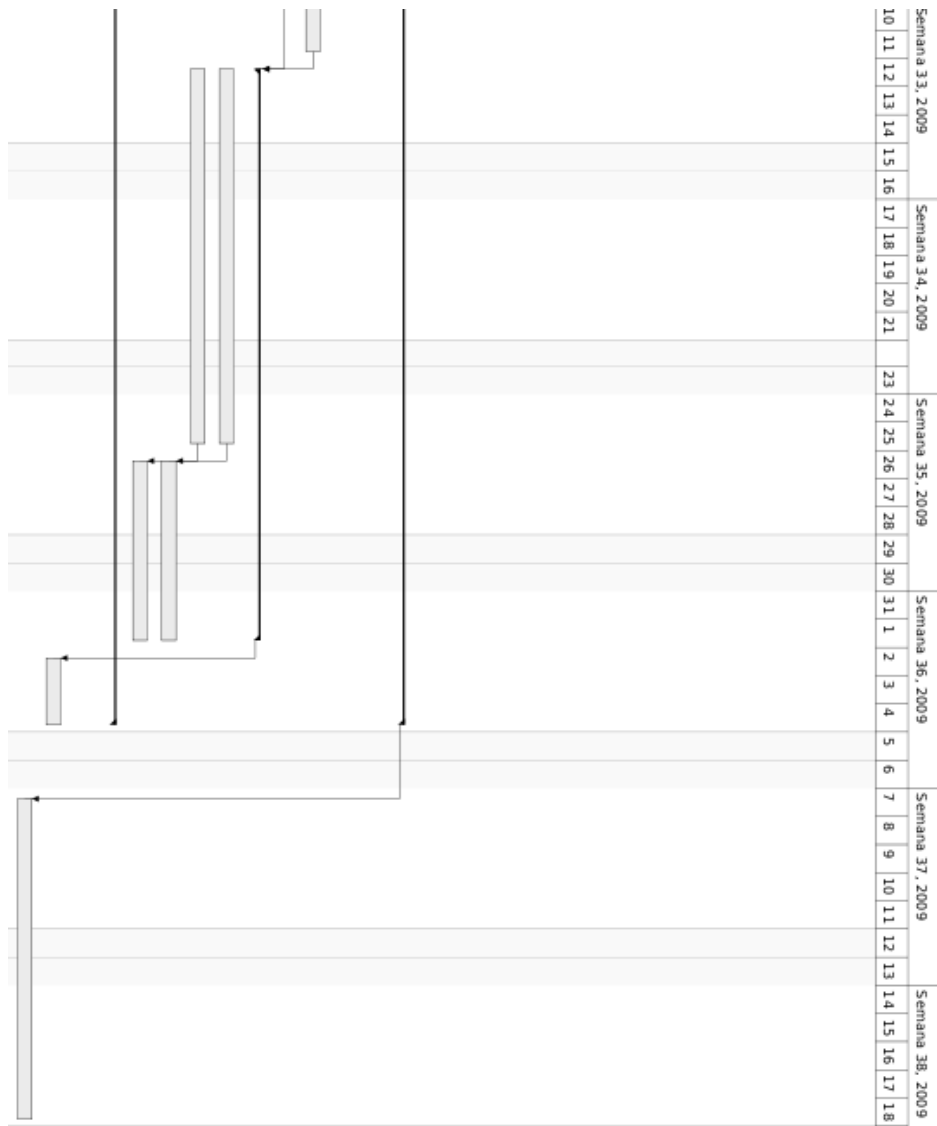


Figura 3.2: Diagrama de Gannt 2. Desarrollo del proyecto

# Capítulo 4

## Análisis del sistema

### 4.1. Especificación de requisitos del sistema

#### 4.1.1. Requisitos de interfaces externas

La interacción con el hardware de la Nintendo DS la realizaremos a través de la librería libnds, que proporciona formas de interactuar tanto con las pantallas como con los botones, la pantalla táctil y demás dispositivos.

Para la interacción con el usuario se usarán las dos pantallas. Dedicando la superior sólo a mostrar y la inferior además dedicada a proporcionar la interacción con el usuario de forma táctil.

La imagen en la pantalla superior será mayor que la de la pantalla y cuando sea necesario el usuario podrá desplazarse por ella con los botones de dirección para verla completa. Esto se hace necesario para poder mostrar toda la información disponible de un kanji, por ejemplo.

El jugador siempre podrá volver al menú principal haciendo clic en la imagen de la flecha a la izquierda en la esquina inferior izquierda de la pantalla de abajo o bien pulsando el botón “B”.

#### 4.1.2. Requisitos funcionales

- Debe mostrar las tarjetas con todos sus campos en su forma adecuada según el modo de juego que el usuario seleccione.
- Debe permitir al usuario acceder a los distintos modos de juego iniciándolos cuando el usuario los seleccione.
- Debe permitir salir del modo de juego actual de forma fácil.
- El juego dispondrá de 4 modos de juego diferentes. Cada uno de los modos será:

**Modo de memorización de kanjis** Este modo servirá para ayudar al jugador a memorizar ideogramas, de esta forma, se mostrará primero el kanji solo, sin más datos, para que el usuario recuerde cuáles eran sus lecturas y su traducción.

En ese momento el jugador podrá ir descubriendo los distintos campos, las lecturas “on”, las lecturas “kun”, y su traducción. Esto permitirá al usuario comprobar si ha memorizado el kanji, y refuerza el aprendizaje una vez que se muestra, corrigiendo o corroborando al jugador.

Adicionalmente también se mostrará un ejemplo del kanji en una frase, de la que se dará su forma en kanji, su lectura y su traducción, de forma que se permita al jugador ver cómo se usa el kanji en una frase real.

También existirá un botón de opciones donde se podrá seleccionar un rango de grado y de número de trazos de los kanjis que se quieran aprender. También será posible seleccionar si se desea convertir las lecturas escritas en kana a romaji.

**Modo de preguntas de kanjis** En este modo se mostrará en la pantalla de arriba toda la información de un kanji, sus lecturas “on” y “kun” y su traducción, y en la pantalla de abajo se mostrará su kanji, mezclado con otros 3 aleatorios de la base de datos.

El jugador deberá escoger cuál es el kanji correcto al cual corresponde la información de la pantalla de arriba.

Cada vez que se seleccione un kanji se actualizarán los marcadores de porcentaje de acierto, y la cantidad de puntos, que se irá incrementando cada vez que el usuario acierte. También se informará al jugador dibujando una pequeña marca de acierto debajo del kanji seleccionado si se eligió correctamente.

En caso de que el usuario se haya equivocado se dibujará una marca de fallo debajo del kanji seleccionado y una de acierto sobre el kanji correcto.

También existirá un botón de opciones donde se podrá seleccionar un rango de grado y de número de trazos de los kanjis que se quieran aprender, igual que en el método de memorización, y también será posible seleccionar si se desea convertir las lecturas escritas en kana a romaji.

**Modo de memorización de vocabulario** Este modo es análogo al de memorización de kanjis, servirá al jugador para memorizar vocabulario.

En la pantalla superior se mostrará una palabra en kanjis de la base de datos pero sin más datos, para que el usuario intente recordar cómo se leía y cuál era su traducción.

El jugador podrá ahora ir desvelando la lectura y traducción de la palabra y así corregir o corroborar lo que pensaba que era.

Existirá también un botón de opciones donde se podrá seleccionar si se quiere convertir la lectura de la palabra romaji. Esta vez no tendrá sentido seleccionar el número de trazos ni el grado de los kanjis ya que una misma palabra puede llevar varios kanjis de diversa dificultad.

**Modo de preguntas de vocabulario** Este modo es análogo al de preguntas de kanjis. Servirá para comprobar el aprendizaje del usuario.

En la pantalla superior se mostrará la lectura y traducción de la palabra y en la pantalla de abajo se mostrará la palabra escrita en kanjis mezclada con otras 3 escogidas aleatoriamente de la base de datos.

Cada vez que se seleccione una palabra, como en el modo de preguntas de kanjis, se mostrará una pequeña marca de acierto sobre la palabra seleccionada si el jugador ha acertado. En caso de haber fallado se mostrará una marca de fallo sobre la palabra seleccionada y una marca de acierto sobre la palabra correcta.

También se actualizará el marcador de porcentaje de acierto del jugador y se incrementará el marcador de puntos en caso de que el usuario haya acertado.

Al igual que en el modo de memorización existirá un botón de opciones donde será posible seleccionar si se desea convertir las lecturas a romaji.

### **4.1.3. Restricciones de rendimiento**

Esta aplicación no requiere un procesamiento masivo por lo que los requisitos de rendimientos son bajos. Además, dada la naturaleza de la aplicación, disponemos de un amplio margen.

### **4.1.4. Restricciones de diseño**

La mayor restricción es que la memoria RAM disponible es muy limitada, por lo que no será posible cargar fuentes, archivos ni imágenes muy grandes directamente en memoria.

### **4.1.5. Atributos del sistema software**

La aplicación debe ser jugable tanto en la videoconsola física como en un emulador para que sea accesible a la mayor cantidad de público.

La aplicación deberá hacer buen uso de la interfaz táctil, que resulta más intuitiva.

La aplicación deberá tener una interfaz limpia que no distraiga al jugador.

## 4.2. Análisis

### 4.2.1. Modelo de Casos de Uso



Figura 4.1: Diagrama de casos de uso

A toda acción del usuario le seguirá un corto sonido a modo de respuesta del sistema. No se incluye en las descripciones por redundante.

#### DESCRIPCIÓN CASO DE USO: Menú principal

- Caso de uso: Menú principal
- Descripción: Muestra botones para acceder a los diferentes modos.
- Actores: Usuario, Sistema.
- Precondiciones: Ninguna.
- Postcondiciones: Ninguna.
- Escenario Principal
  1. El sistema muestra el logo del juego en la pantalla mientras suena una música de introducción.

2. El sistema muestra el menú principal en la pantalla inferior.
  3. El usuario toca el botón de memorización de kanjis.
  4. El sistema inicia el modo de memorización de kanjis.
- Extensiones (Flujo alternativo):
    - 3a. El usuario toca el botón de preguntas de kanjis.
      1. El sistema inicia el modo de preguntas de kanjis.
    - 3b. El usuario toca el botón de memorización de vocabulario.
      1. El sistema inicia el modo de memorización de vocabulario.
    - 3c. El usuario toca el botón de preguntas de vocabulario.
      1. El sistema inicia el modo de preguntas de vocabulario.

### **DESCRIPCIÓN CASO DE USO: Memorización de kanjis**

- Caso de uso: Memorización de kanjis
- Descripción: Muestra los kanjis en la pantalla y su información.
- Actores: Usuario, Sistema.
- Precondiciones: Ninguna.
- Postcondiciones: Ninguna.
- Escenario Principal:
  1. El sistema carga la primera tarjeta
  2. El sistema muestra sólo el kanji de la tarjeta.
  3. El usuario presiona el botón "A".
  4. El sistema muestra la lectura *on*.
  5. El usuario presiona el botón "A".
  6. El sistema muestra la lectura *kun*.
  7. El usuario presiona el botón "A".
  8. El sistema muestra la traducción.
  9. El usuario presiona el botón "A".
  10. El sistema muestra un ejemplo.
  11. El usuario presiona el botón "A".
  12. El sistema carga la siguiente tarjeta y vuelve al paso 2.
- Extensiones (Flujo alternativo):
  - \*a. El usuario presiona el botón "B".

1. El sistema sale del modo actual.
- \*b. El usuario toca la flecha para volver en la pantalla inferior.
  1. El sistema sale del modo actual.
- \*c. El usuario presiona el botón de dirección a la izquierda.
  1. El sistema carga la tarjeta anterior y vuelve al paso 2.
    - 1a. Si la tarjeta actual es la primera no se hace nada.
- \*d. El usuario presiona el botón de dirección a la derecha.
  1. El sistema carga la siguiente tarjeta y vuelve al paso 2.
    - 1a. Si la tarjeta actual es la última no se hace nada.
- \*e. El usuario toca la pantalla para ver la carta anterior.
  1. El sistema carga la tarjeta anterior y vuelve al paso 2.
    - 1a. Si la tarjeta actual es la primera no se hace nada.
- \*f. El usuario toca la pantalla para ver la siguiente carta.
  1. El sistema carga la siguiente tarjeta y vuelve al paso 2.
    - 1a. Si la tarjeta actual es la última no se hace nada.
- \*g. El usuario presiona el botón de dirección arriba.
  1. El sistema desliza el contenido de la pantalla superior hacia arriba.
    - 1a. Si se llega al máximo no se hace nada.
- \*h. El usuario presiona el botón de dirección abajo.
  1. El sistema desliza el contenido de la pantalla superior hacia abajo.
    - 1a. Si se llega al máximo no se hace nada.
- \*i. El usuario toca el botón de opciones.
  1. El sistema inicia el modo de opciones de kanjis.
- \*j. El usuario pulsa el botón Y.
  1. El sistema inicia el modo de opciones de kanjis.
- 12a. La tarjeta actual es la última.
  1. No se hace nada.

### **DESCRIPCIÓN CASO DE USO: Preguntas de kanjis**

- Caso de uso: Preguntas de kanjis
- Descripción: Muestra la traducción y lecturas de un kanji y pregunta por el ideograma.
- Actores: Usuario, Sistema.
- Precondiciones: Ninguna.
- Postcondiciones: Ninguna.

- Escenario Principal:
  1. El sistema carga la primera tarjeta
  2. El sistema muestra la traducción y las lecturas *on* y *kun* del kanji.
  3. El sistema muestra 4 kanjis en la pantalla inferior, siendo uno el correcto y los otros 3, aleatorios de la base de datos.
  4. El usuario toca el kanji correcto.
  5. El sistema muestra un “tick” para indicar que es correcto y actualiza los marcadores.
  6. El usuario presiona algún botón o toca la pantalla.
  7. El sistema carga la siguiente tarjeta y vuelve al paso 2.
- Extensiones (Flujo alternativo):
  - \*a. El usuario presiona el botón “B”.
    1. El sistema sale del modo actual.
  - \*b. El usuario toca la flecha para volver en la pantalla inferior.
    1. El sistema sale del modo actual.
  - \*c. El usuario presiona el botón de dirección a la izquierda.
    1. El sistema carga la tarjeta anterior y vuelve al paso 2.
      - 1a. Si la tarjeta actual es la primera no se hace nada.
  - \*d. El usuario presiona el botón de dirección a la derecha.
    1. El sistema carga la siguiente tarjeta y vuelve al paso 2.
      - 1a. Si la tarjeta actual es la última no se hace nada.
  - \*e. El usuario presiona el botón de dirección arriba.
    1. El sistema desliza el contenido de la pantalla superior hacia arriba.
      - 1a. Si se llega al máximo no se hace nada.
  - \*f. El usuario presiona el botón de dirección abajo.
    1. El sistema desliza el contenido de la pantalla superior hacia abajo.
      - 1a. Si se llega al máximo no se hace nada.
  - \*g. El usuario toca el botón de opciones.
    1. El sistema inicia el modo de opciones de kanjis.
  - \*h. El usuario pulsa el botón Y.
    1. El sistema inicia el modo de opciones de kanjis.
- 5a. El usuario toca un kanji incorrecto.
  1. El sistema muestra una cruz sobre el kanji tocado y un “tick” sobre el kanji correcto.
  2. El sistema actualiza el marcador de porcentaje de aciertos.

**7a.** La tarjeta actual es la última.

1. No se hace nada.

#### **DESCRIPCIÓN CASO DE USO: Memorización de vocabulario**

- Caso de uso: Memorización de vocabulario
- Descripción: Muestra las palabras en la pantalla y su información.
- Actores: Usuario, Sistema.
- Precondiciones: Ninguna.
- Postcondiciones: Ninguna.
- Escenario Principal:
  1. El sistema carga la primera tarjeta
  2. El sistema muestra sólo la palabra en kanjis.
  3. El usuario presiona el botón “A”.
  4. El sistema muestra la lectura de la palabra.
  5. El usuario presiona el botón “A”.
  6. El sistema muestra la traducción.
  7. El usuario presiona el botón “A”.
  8. El sistema carga la siguiente tarjeta y vuelve al paso 2.
- Extensiones (Flujo alternativo):
  - \*a. El usuario presiona el botón “B”.
    1. El sistema sale del modo actual.
  - \*b. El usuario toca la flecha para volver en la pantalla inferior.
    1. El sistema sale del modo actual.
  - \*c. El usuario presiona el botón de dirección a la izquierda.
    1. El sistema carga la tarjeta anterior y vuelve al paso 2.
      - 1a. Si la tarjeta actual es la primera no se hace nada.
  - \*d. El usuario presiona el botón de dirección a la derecha.
    1. El sistema carga la siguiente tarjeta y vuelve al paso 2.
      - 1a. Si la tarjeta actual es la última no se hace nada.
  - \*e. El usuario toca la pantalla para ver la carta anterior.
    1. El sistema carga la tarjeta anterior y vuelve al paso 2.
      - 1a. Si la tarjeta actual es la primera no se hace nada.
  - \*f. El usuario toca la pantalla para ver la siguiente carta.

1. El sistema carga la siguiente tarjeta y vuelve al paso 2.
  - 1a. Si la tarjeta actual es la última no se hace nada.
- \*g. El usuario presiona el botón de dirección arriba.
  1. El sistema desliza el contenido de la pantalla superior hacia arriba.
    - 1a. Si se llega al máximo no se hace nada.
- \*h. El usuario presiona el botón de dirección abajo.
  1. El sistema desliza el contenido de la pantalla superior hacia abajo.
    - 1a. Si se llega al máximo no se hace nada.
- \*i. El usuario toca el botón de opciones.
  1. El sistema inicia el modo de opciones de romanización.
- \*j. El usuario pulsa el botón Y.
  1. El sistema inicia el modo de opciones de romanización.
- 12a. La tarjeta actual es la última.
  1. No se hace nada.

#### **DESCRIPCIÓN CASO DE USO: Preguntas de vocabulario**

- Caso de uso: Preguntas de vocabulario
- Descripción: Muestra la traducción y lectura de una palabra y pregunta por su versión escrita en kanjis.
- Actores: Usuario, Sistema.
- Precondiciones: Ninguna.
- Postcondiciones: Ninguna.
- Escenario Principal:
  1. El sistema carga la primera tarjeta
  2. El sistema muestra la traducción y la lectura de la palabra.
  3. El sistema muestra 4 palabras escritas en kanjis en la pantalla inferior, siendo una la correcta y las otras 3, aleatorias de la base de datos.
  4. El usuario toca la palabra en kanjis correcta.
  5. El sistema muestra un “tick” para indicar que es correcta y actualiza los marcadores.
  6. El usuario presiona algún botón o toca la pantalla.
  7. El sistema carga la siguiente tarjeta y vuelve al paso 2.
- Extensiones (Flujo alternativo):
  - \*a. El usuario presiona el botón “B”.

1. El sistema sale del modo actual.
- \*b. El usuario toca la flecha para volver en la pantalla inferior.
  1. El sistema sale del modo actual.
- \*c. El usuario presiona el botón de dirección a la izquierda.
  1. El sistema carga la tarjeta anterior y vuelve al paso 2.
    - 1a. Si la tarjeta actual es la primera no se hace nada.
- \*d. El usuario presiona el botón de dirección a la derecha.
  1. El sistema carga la siguiente tarjeta y vuelve al paso 2.
    - 1a. Si la tarjeta actual es la última no se hace nada.
- \*e. El usuario presiona el botón de dirección arriba.
  1. El sistema desliza el contenido de la pantalla superior hacia arriba.
    - 1a. Si se llega al máximo no se hace nada.
- \*f. El usuario presiona el botón de dirección abajo.
  1. El sistema desliza el contenido de la pantalla superior hacia abajo.
    - 1a. Si se llega al máximo no se hace nada.
- \*g. El usuario toca el botón de opciones.
  1. El sistema inicia el modo de opciones de romanización.
- \*h. El usuario pulsa el botón Y.
  1. El sistema inicia el modo de opciones de romanización.
- 5a. El usuario toca una palabra en kanjis incorrecta.
  1. El sistema muestra una cruz sobre la palabra tocada y un “tick” sobre la correcta.
  2. El sistema actualiza el marcador de porcentaje de aciertos.
- 7a. La tarjeta actual es la última.
  1. No se hace nada.

## **DESCRIPCIÓN CASO DE USO: Cargar datos**

- Caso de uso: Cargar datos.
- Descripción: Carga la información de una tarjeta de un archivo.
- Actores: Sistema.
- Precondiciones: El archivo existe y tiene una sintaxis adecuada.
- Postcondiciones: Ninguna.
- Escenario Principal:
  1. El sistema los datos generales del paquete.

2. El usuario pide una tarjeta con unos requisitos dados en el caso de que sea un kanji.
  3. El sistema carga la carta deseada.
- Extensiones (Flujo alternativo):
    - 1a. El fichero no existe.
      1. El sistema informa del error.
    - 1a, 3a. El fichero no tiene una sintaxis adecuada.
      1. El sistema informa del error.
    - 3b. La carta seleccionada no existe.
      1. El sistema no hace nada.

### DESCRIPCIÓN CASO DE USO: Opciones de kanjis

- Caso de uso: Opciones de kanjis.
- Descripción: Muestra opciones para seleccionar la romanización y para seleccionar la dificultad de los kanjis a mostrar.
- Actores: Sistema.
- Precondiciones: Ninguna.
- Postcondiciones: Ninguna.
- Escenario Principal:
  1. El sistema muestra un recuadro para seleccionar si se desea romanizar las lecturas y las opciones de grado y trazos mínimos y máximos.
  2. El usuario selecciona si desea romanización tocando sobre el recuadro de romanización.
  3. El sistema muestra un a X en el recuadro de romanización si se ha seleccionado.
  4. El usuario selecciona el grado mínimo de los kanjis a mostrar tocando los botones “-” y “+” que aparecen alrededor.
  5. El sistema va actualizando el valor del recuadro.
  6. El usuario selecciona el grado máximo de los kanjis a mostrar tocando los botones “-” y “+” que aparecen alrededor.
  7. El sistema va actualizando el valor del recuadro.
  8. El usuario selecciona el número de trazos mínimo de los kanjis a mostrar tocando los botones “-” y “+” que aparecen alrededor.
  9. El sistema va actualizando el valor del recuadro.

10. El usuario selecciona el número de trazos máximo de los kanjis a mostrar tocando los botones “-” y “+” que aparecen alrededor.
  11. El usuario pulsa sobre la V en la esquina inferior derecha para volver.
  12. El sistema guarda las opciones y vuelve.
- Extensiones (Flujo alternativo):
    - \*a. El usuario pulsa el botón “B”.
      1. El sistema guarda las opciones y vuelve.

#### **DESCRIPCIÓN CASO DE USO: Opciones de romanización**

- Caso de uso: Opciones de romanización.
- Descripción: Muestra opciones para seleccionar la romanización.
- Actores: Sistema.
- Precondiciones: Ninguna.
- Postcondiciones: Ninguna.
- Escenario Principal:
  1. El sistema muestra un recuadro para seleccionar si se desea romanizar las lecturas.
  2. El usuario selecciona si desea romanización tocando sobre el recuadro de romanización.
  3. El sistema muestra un a X en el recuadro de romanización si se ha seleccionado.
  4. El usuario pulsa sobre la V en la esquina inferior derecha para volver.
  5. El sistema guarda las opciones y vuelve.
- Extensiones (Flujo alternativo):
  - \*a. El usuario pulsa el botón “B”.
    1. El sistema guarda las opciones y vuelve.

## 4.2.2. Modelo conceptual de datos en UML

### 4.2.2.1. Diagrama de clases conceptuales

Las clases que vamos a necesitar son:

- **Tarjeta:** Tarjeta conteniendo los datos necesarios.
- **Lector de ficheros:** Intérprete para extraer los datos de los ficheros.
- **Manejador de pantalla superior:** Interfaz con la pantalla superior.
- **Manejador de pantalla inferior:** Interfaz con la pantalla inferior.
- **Manejador de sonido:** Clase para reproducir sonidos.
- **Texto:** Clase para imprimir texto.
- **Juego:** Clase del juego en sí.

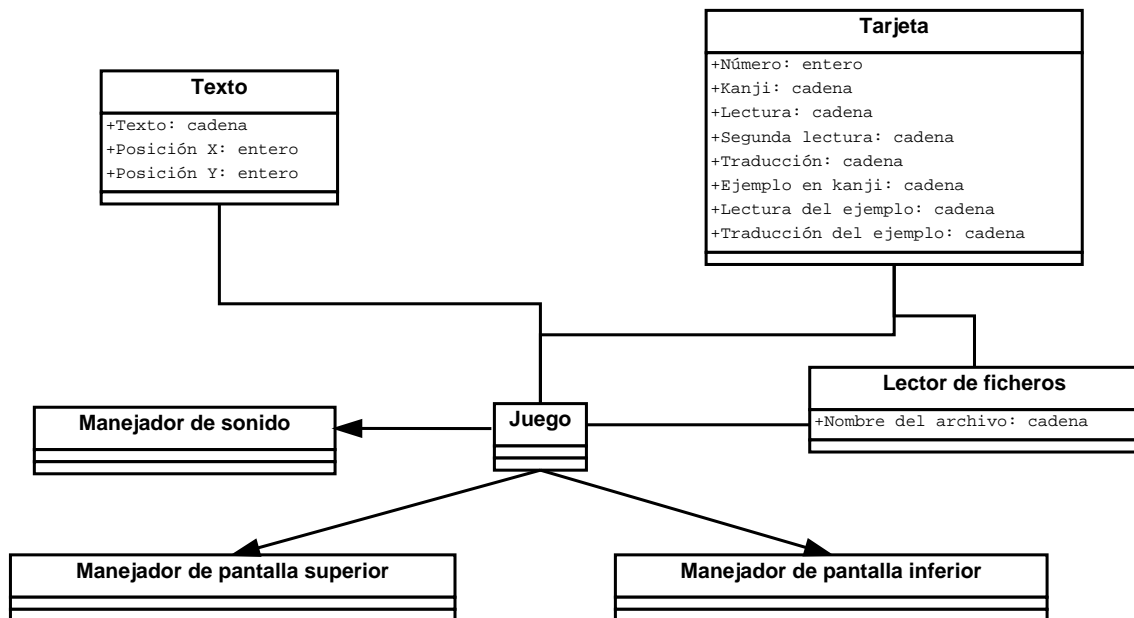


Figura 4.2: Diagrama de clases conceptuales

#### 4.2.2.2. Modelo de comportamiento del sistema

### Diagrama de secuencia de sistema y contratos de las operaciones del sistema

#### DIAGRAMA DE SECUENCIA: Menú principal

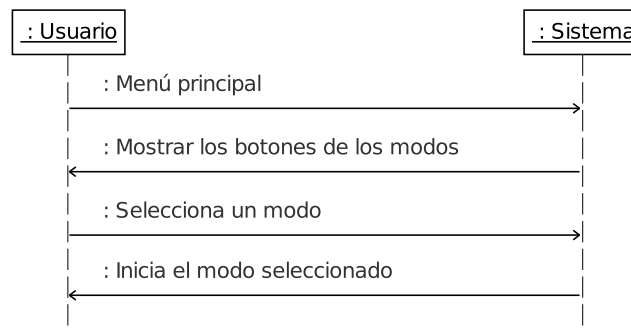


Figura 4.3: Diagrama de secuencia para el caso de uso Menú principal

### Contrato de las operaciones

- **Operación:** Menú principal
- **Responsabilidades:** Muestra al usuario 4 botones para acceder al modo que desee jugar.
- **Precondiciones:** Ninguna.
- **Postcondiciones:** Ninguna.

## DIAGRAMA DE SECUENCIA: Memorización de kanjis

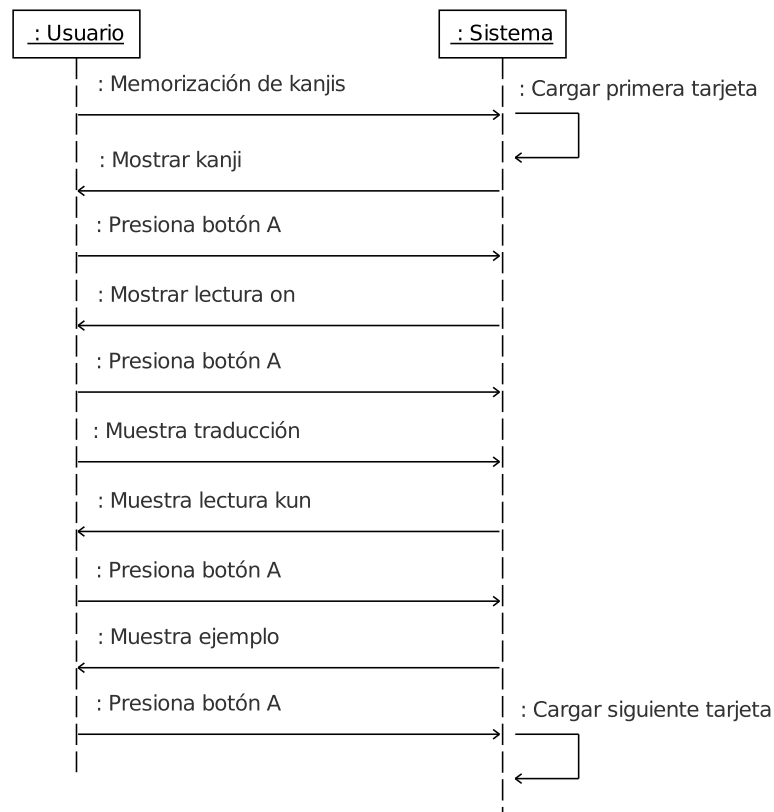


Figura 4.4: Diagrama de secuencia para el caso de uso Memorización de kanjis

### Contrato de las operaciones

- **Operación:** Memorización de kanjis
- **Responsabilidades:** Muestra un kanji, sus lecturas, traducción y ejemplo conforme el usuario va presionando el botón "A".
- **Precondiciones:** Se deben cumplir las restricciones del caso de uso cargar datos.
- **Postcondiciones:** Ninguna.

## DIAGRAMA DE SECUENCIA: Preguntas de kanjis

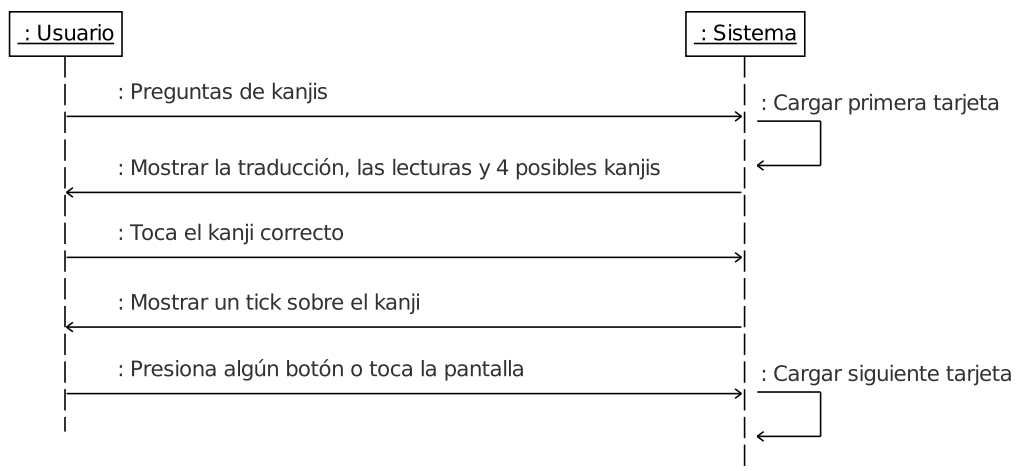


Figura 4.5: Diagrama de secuencia para el caso de uso Preguntas de kanjis

### Contrato de las operaciones

- **Operación:** Preguntas de kanjis
- **Responsabilidades:** Muestra la traducción y lecturas de un kanji y 4 posibles kanjis, uno de los cuales será el correcto y los otros 3 serán aleatorios de la base de datos.
- **Precondiciones:** Se deben cumplir las restricciones del caso de uso cargar datos.
- **Postcondiciones:** Ninguna.

## DIAGRAMA DE SECUENCIA: Memorización de vocabulario

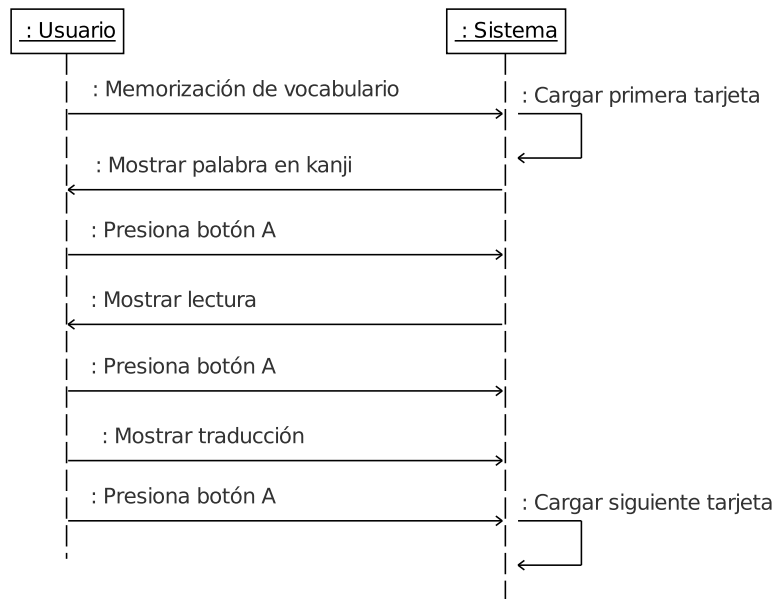


Figura 4.6: Diagrama de secuencia para el caso de uso Memorización de vocabulario

### Contrato de las operaciones

- **Operación:** Memorización de vocabulario
- **Responsabilidades:** Muestra una palabra en kanji, su lectura, y su traducción conforme el usuario va presionando el botón "A".
- **Precondiciones:** Se deben cumplir las restricciones del caso de uso cargar datos.
- **Postcondiciones:** Ninguna.

## DIAGRAMA DE SECUENCIA: Preguntas de vocabulario

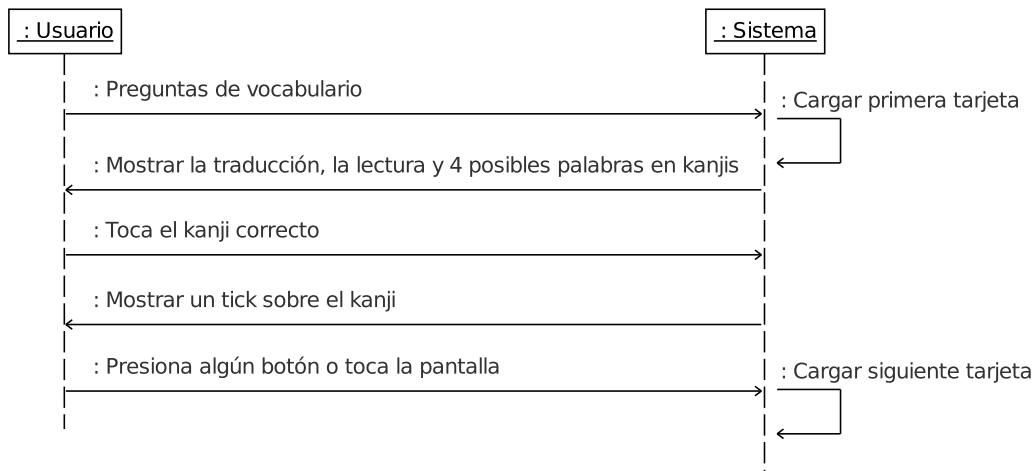


Figura 4.7: Diagrama de secuencia para el caso de uso Preguntas de vocabulario

### Contrato de las operaciones

- **Operación:** Preguntas de vocabulario
  
- **Responsabilidades:** Muestra la traducción y lectura de una palabra y 4 posibles palabras escritas en kanji, una de las cuales será la correcta y las otras 3 serán aleatorias de la base de datos.
  
- **Precondiciones:** Se deben cumplir las restricciones del caso de uso cargar datos.
  
- **Postcondiciones:** Ninguna.

### DIAGRAMA DE SECUENCIA: Opciones de kanjis

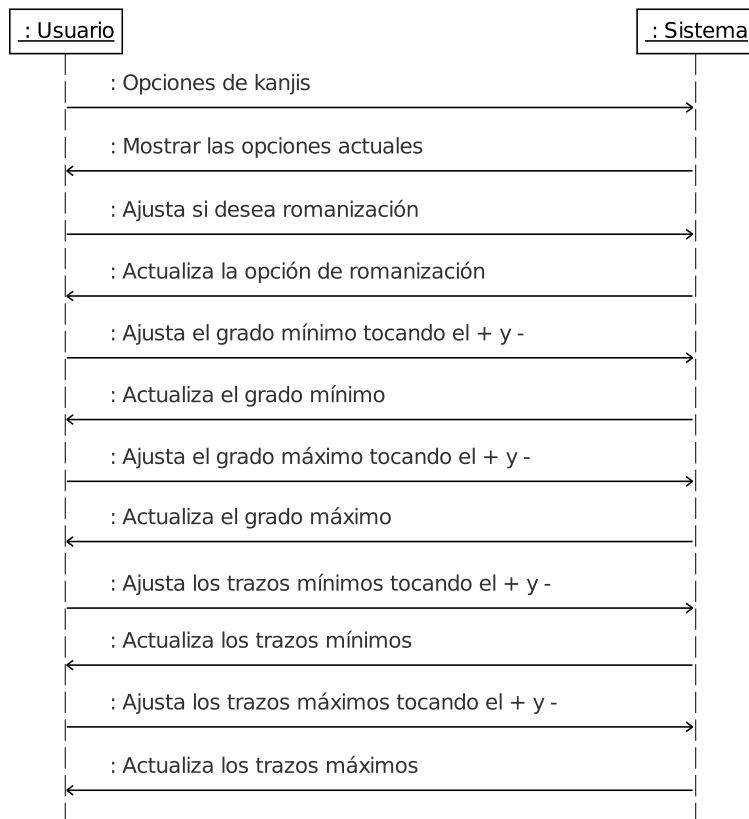


Figura 4.8: Diagrama de secuencia para el caso de uso Opciones de kanjis

### Contrato de las operaciones

- **Operación:** Opciones de kanjis
- **Responsabilidades:** Muestra las opciones al usuario y permite seleccionar un nuevo rango de grados y trazos de los kanjis así como seleccionar si se desea conversión a romaji.
- **Precondiciones:** Ninguna.
- **Postcondiciones:** Ninguna.

### DIAGRAMA DE SECUENCIA: Opciones de romanización

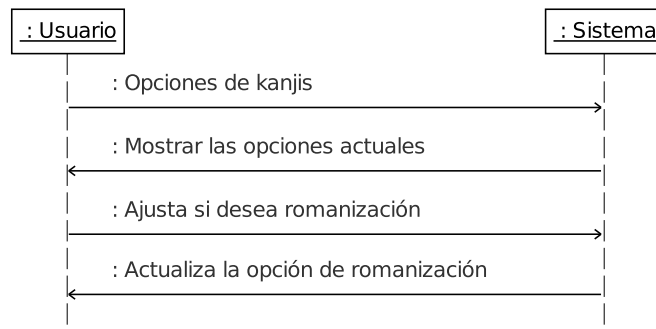


Figura 4.9: Diagrama de secuencia para el caso de uso Opciones de romanización

### Contrato de las operaciones

- **Operación:** Opciones de romanización
- **Responsabilidades:** Muestra la opción de romanización al usuario y le permite seleccionar si se desea conversión a romaji.
- **Precondiciones:** Ninguna.
- **Postcondiciones:** Ninguna.

# Capítulo 5

## Diseño del sistema

Seguiremos una arquitectura en 3 capas, la capa de datos que será la encargada de manejar la interfaz con los ficheros, la capa de dominio que será la encargada de implementar la funcionalidad del sistema y la capa de presentación que será la encargada de implementar la interfaz del juego, incluyendo el manejo de las pantallas.

**Nota:** La documentación puede extraerse del código fuente a través de doxygen (eso sí, en inglés que es como he escrito el código). También puede consultarse desde la página del proyecto en: <http://blog.japonesporlibre.com/dsmemorizer>

### 5.1. Capa de gestión de datos

Las palabras o ideogramas del juego necesitan ser guardadas de alguna forma por lo que nos hace falta una estructura para los paquetes de tarjetas.

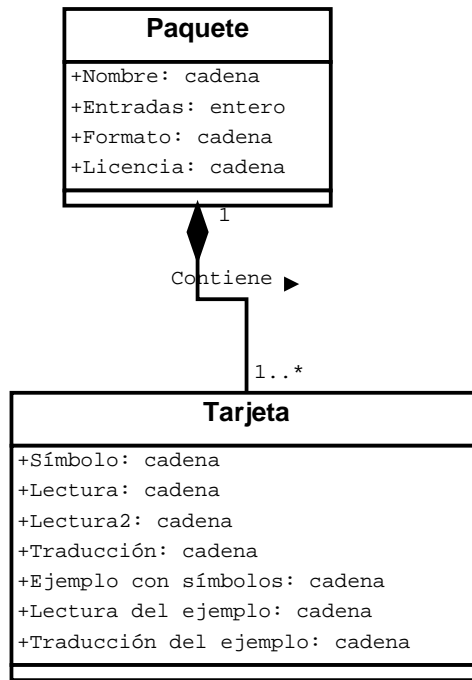


Figura 5.1: Diagrama de la capa de datos

Los paquetes de tarjetas se guardarán en ficheros XML, acordes al formato descrito en el apéndice [Formato de los ficheros XML](#).

Para leer estos ficheros será necesaria una clase que será XMLParser. Ésta clase se encargará de leer del fichero las tarjetas solicitadas y leer la información de los paquetes.

La composición de los paquetes a base de cartas no se implementa con la clase “Paquete” porque si se implementase, para que fuese coherente habría que cargar todas las cartas asociándolas al paquete en la RAM, pero las bases de datos están pensadas para que sean muy grandes por lo que esta opción no es practicable.

Para conocer la información del paquete actualmente activo preguntaremos directamente a la clase XMLParser.

### 5.1.1. Clase Card

Esta clase almacena los atributos de una tarjeta.

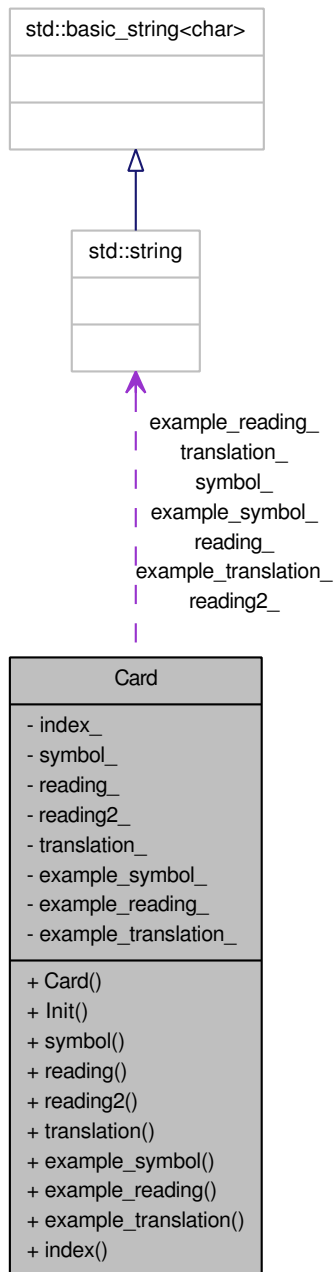


Figura 5.2: Diagrama de colaboración de la clase Card

### Funciones miembro públicas

- `Card ()`  
*Constructor por defecto.*
- `void Init (int index, std::string symbol, std::string reading, std::string reading2, std::string translation, std::string example_symbol, std::string example_reading, std::string example_translation)`

*Inicializador.*

- `std::string symbol () const`  
*Símbolo/s de la tarjeta.*
- `std::string reading () const`  
*(Primera) lectura del símbolo/s*
- `std::string reading2 () const`  
*Segunda lectura del símbolo.*
- `std::string translation () const`  
*Traducción del símbolo/s.*
- `std::string example_symbol () const`  
*Ejemplo escrito con símbolos.*
- `std::string example_reading () const`  
*Lectura del ejemplo.*
- `std::string example_translation () const`  
*Traducción del ejemplo.*
- `int index () const`  
*Índice de la tarjeta en el paquete. Está en el rango [1 - records].*

### **Atributos privados**

- `int index_`  
*Índice de la tarjeta en el paquete. Está en el rango [1 - records].*
- `std::string symbol_`  
*Símbolo/s de la tarjeta.*
- `std::string reading_`  
*(Primera) lectura del símbolo/s*
- `std::string reading2_`  
*Segunda lectura del símbolo.*

- `std::string translation_`

*Traducción del símbolo/s.*

- `std::string example_symbol_`

*Ejemplo escrito con símbolos.*

- `std::string example_reading_`

*Lectura del ejemplo.*

- `std::string example_translation_`

*Traducción del ejemplo.*

### 5.1.2. Clase XMLParser

Esta clase se encarga de implementar la interfaz con los ficheros tal como se explica en [5.1](#). Esta clase se inicializa (se puede hacer más de una vez para cambiar de fichero) apuntando a un fichero que respete el [Formato de los ficheros XML](#), lee el descriptor de paquete, comprueba que el paquete es adecuado y cuando se le pregunta por una tarjeta (ver [5.1.1](#)), la lee del fichero y la devuelve.

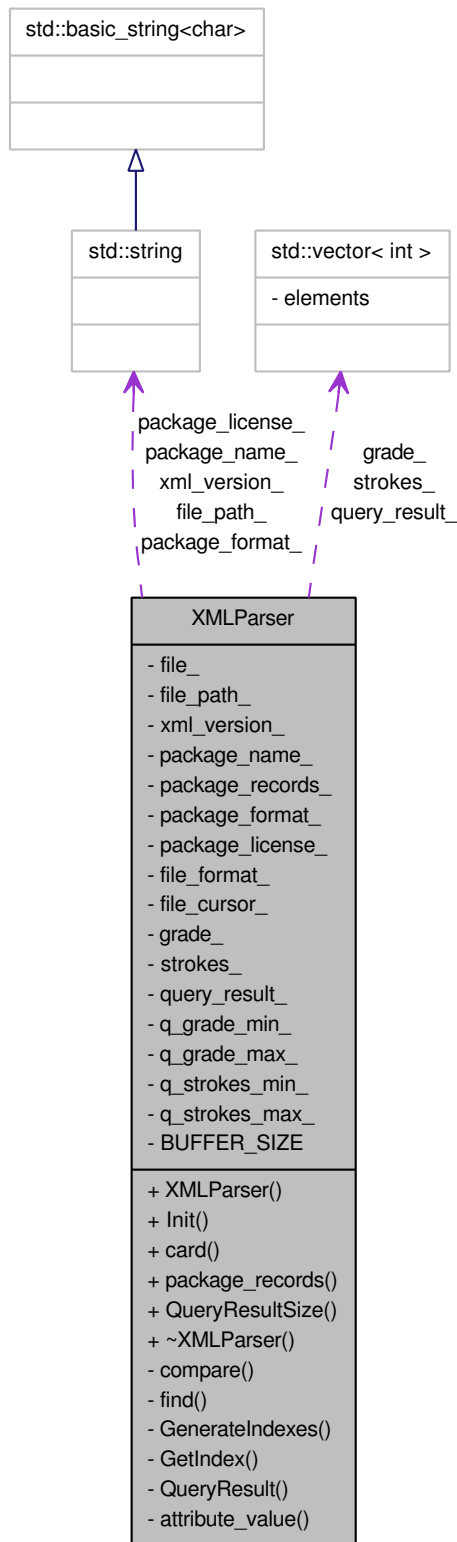


Figura 5.3: Diagrama de colaboración de la clase XMLParser

## Funciones miembro públicas

- XMLParser ()  
*Constructor por defecto.*
- void Init (const std::string &file\_path)
- Card card (unsigned index, unsigned grade\_min=0, unsigned grade\_max=0, unsigned strokes\_min=0, unsigned strokes\_max=0)  
*Devuelve una carta determinada por su índice.*
- int package\_records () const  
*Devuelve el número de cartas en el paquete.*
- int QueryResultSize (unsigned grade\_min, unsigned grade\_max, unsigned strokes\_min, unsigned strokes\_max)  
*Devuelve el número de kanjis que hay con las restricciones dadas de rango de grados y trazos.*

## Tipos privados

- enum mode { XML\_KANJI, XML\_VOCABULARY }  
*Tipos de ficheros XML.*

## Funciones miembro privadas

- bool compare (const char \*s1, const char \*s2)  
*Función simple de comparación de igualdad entre cadenas.*
- bool find (const char \*s, const char \*buffer, int &position)  
*Función simple de búsqueda.*
- void GenerateIndexes ()  
*Rellenar los índices file\_cursor\_, grade\_ y strokes\_ (dependiendo del tipo de archivo).*
- unsigned GetIndex (unsigned index, unsigned grade\_min, unsigned grade\_max, unsigned strokes\_min, unsigned strokes\_max)
- void QueryResult (unsigned grade\_min, unsigned grade\_max, unsigned strokes\_min, unsigned strokes\_max)  
*Generar el vector de resultados para los rangos de grados y trazos dados.*
- std::string attribute\_value (const char \*name, const char \*buffer, int &position)

## Atributos privados

- FILE \* file\_  
*Archivo XML.*
- std::string file\_path\_  
*Ruta del archivo.*
- std::string xml\_version\_  
*Versión del archivo XML.*
- std::string package\_name\_  
*Nombre del paquete en el archivo (sólo se permite un paquete por archivo).*
- int package\_records\_  
*Número de cartas en el paquete.*
- std::string package\_format\_  
*Formato del paquete.*
- std::string package\_license\_  
*Licencia del paquete.*
- mode file\_format\_  
*Formato del archivo.*
- fpos\_t \* file\_cursor\_  
*Vector de posiciones de las cartas en el archivo.*
- std::vector< int > \* grade\_  
*Vector de número de tarjetas organizadas por grado.*
- std::vector< int > \* strokes\_  
*Vector de número de tarjetas organizadas por trazos.*
- std::vector< int > query\_result\_  
■ unsigned q\_grade\_min\_  
*Grado mínimo en la consulta actual.*
- unsigned q\_grade\_max\_  
*Grado máximo en la consulta actual.*

- unsigned q\_strokes\_min\_  
*Mínimo número de trazos en la consulta actual.*
- unsigned q\_strokes\_max\_  
*Máximo número de trazos en la consulta actual.*

### Atributos privados estáticos

- static const int BUFFER\_SIZE = 512  
*Tamaño del buffer temporal de lectura.*

### Funciones miembro

- string XMLParser::attribute\_value (const char \* *name*, const char \* *buffer*, int & *position*) [*privado*]

Devuelve el valor de un atributo en una línea en el XML

Parámetros: *name* Nombre del atributo

*buffer* Línea del XML donde buscar

*position* Posición donde empezar a buscar. Se actualiza a la posición del último carácter leído

- unsigned XMLParser::GetIndex (unsigned *index*, unsigned *grade\_min*, unsigned *grade\_max*, unsigned *strokes\_min*, unsigned *strokes\_max*) [*privado*]

Devuelve el índice de la tarjeta en el paquete en relación a los grados y trazos dados si es el caso.

Parámetros: *index* Número de tarjeta a devolver en relación a los grados y trazos dados

*grade\_min* Grado mínimo del kanji a devolver

*grade\_max* Grado máximo del kanji a devolver

*strokes\_min* Mínimo número de trazos del kanji a devolver

*strokes\_max* Máximo número de trazos del kanji a devolver

Devuelve: Un número en [1 - package\_records\_] si se pudo encontrar el kanji. 0 si no se encontró ningún kanji con esos parámetros.

- void XMLParser::Init (const std::string & *file\_path*)

Initializer

Parámetros: *file\_path* Ruta del archivo a leer

## Miembros de datos

- `std::vector<int> XMLParser::query_result_` [privado]

Vector de resultados de la consulta a la base de datos para el rango de grados y número de trazos dados.

## 5.2. Capa de dominio

### 5.2.1. Diagrama de clases de diseño

Este es el diagrama de todas las clases del sistema para su implementación. Se ha omitido todo menos los nombres y sólo se muestra la colaboración entre las clases.

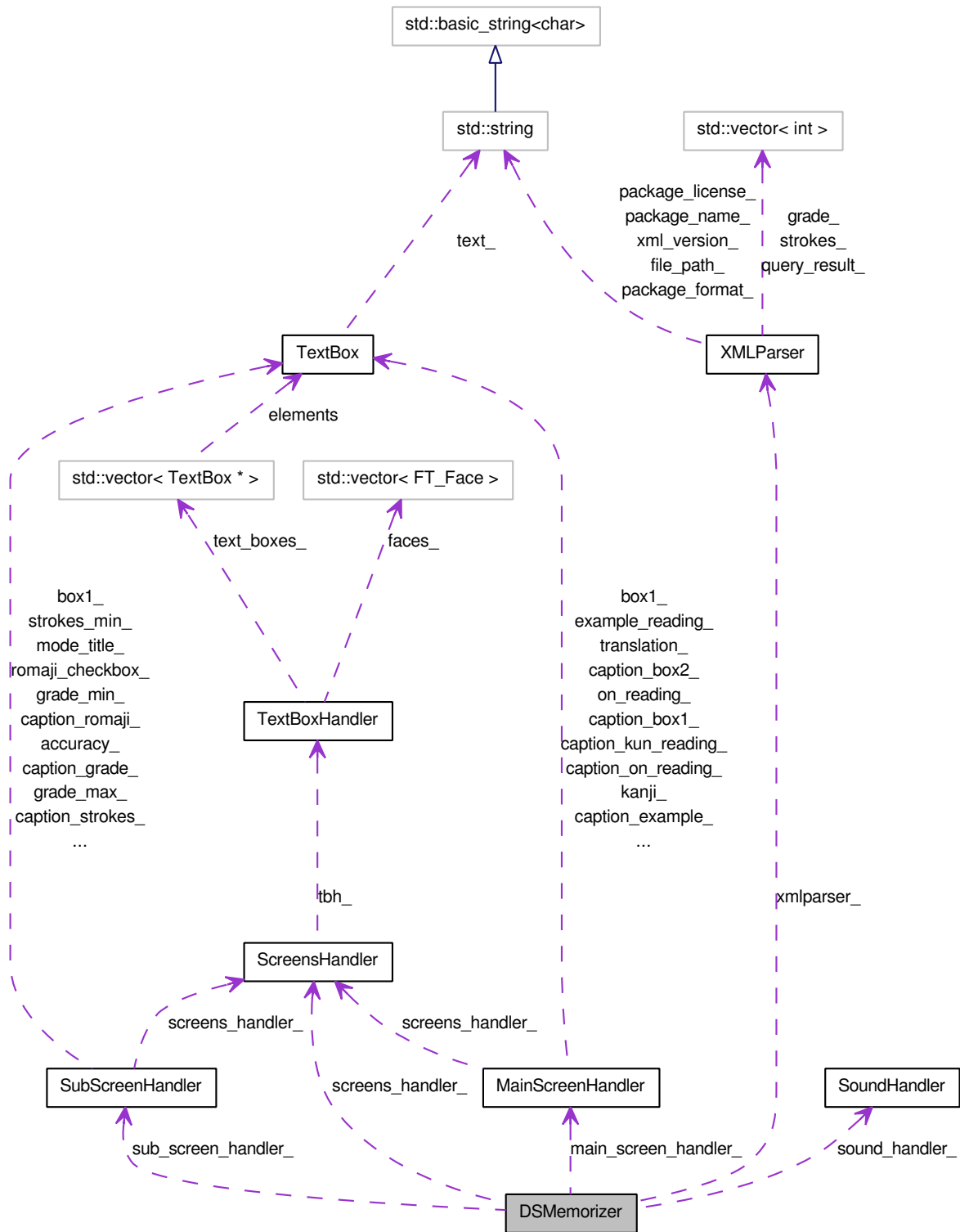


Figura 5.4: Diagrama de clases de diseño

A continuación se detallan las clases con sus miembros en notación UML.

## 5.2.2. Clase DSMemorizer

Esta es la clase principal. Contiene el bucle principal del juego y los diferentes modos de juego.

Se inicia el juego al llamar a la función `Init()`. En este momento se mostrará el logo del juego en la pantalla superior y el menú principal en la inferior.

Cuando el jugador toque sobre el botón de alguno de los modos en la pantalla inferior, se iniciará ese modo llamando a la función correspondiente.

Las funciones que implementan los modos son funciones privadas de esta clase y cada una de ellas incluye un bucle de juego que se ejecutará mientras el usuario no decida regresar al menú principal.

Cuando se cambia de modo de juego, se cambia el modo de las dos pantallas y se muestra en la pantalla lo que sea conveniente respecto al modo y las acciones del usuario.

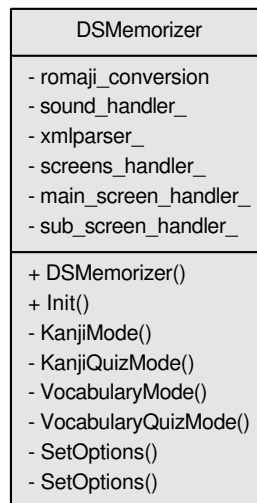


Figura 5.5: Detalle de la clase DSMemorizer en UML

El diagrama de colaboración es el de la subsección anterior.

### Funciones miembro públicas

- `DSMemorizer()`  
*Constructor por defecto.*
- `void Init()`  
*Inicializar y comenzar el juego.*

## Funciones miembro privadas

- void KanjiMode ()
- void KanjiQuizMode ()
- void VocabularyMode ()
- void VocabularyQuizMode ()
- void SetOptions (unsigned &grade\_min, unsigned &grade\_max, unsigned &strokes\_min, unsigned &strokes\_max)  
*Ajustar las opciones de grados, trazos y conversión a romaji.*
- void SetOptions ()  
*Ajustar la opción de conversión a romaji.*

## Atributos privados

- bool romaji\_conversion  
*Opción de conversión a romaji.*
- SoundHandler sound\_handler\_  
*Manejador de sonido.*
- XMLParser xmlparser\_  
*Lector de archivos XML.*
- ScreensHandler screens\_handler\_  
*Manejador para cosas comunes a las dos pantallas.*
- MainScreenHandler main\_screen\_handler\_  
*Manejador de la pantalla principal.*
- SubScreenHandler sub\_screen\_handler\_  
*Manejador de la pantalla inferior.*

## Funciones miembro

- void DSMemorizer::KanjiMode () [privado]  
Modo de memorización de kanjis, muestra un kanji con sus lecturas, traducción y un ejemplo. (ver [4.1.2](#))

- `void DSMemorizer::KanjiQuizMode () [privado]`

Modo de preguntas de kanjis. Muestra la traducción y lecturas de un kanji en la pantalla principal y el kanji mezclado con otros 3 aleatorios de la base de datos en la pantalla inferior. El usuario tiene que elegir el kanji correcto. (ver [4.1.2](#))

- `void DSMemorizer::VocabularyMode () [privado]`

Modo de memorización de vocabulario. Muestra palabras con su lectura y traducción. (ver [4.1.2](#))

- `void DSMemorizer::VocabularyQuizMode () [privado]`

Modo de preguntas de vocabulario. Muestra la traducción de una palabra y su lectura y pregunta su versión en kanji. (ver [4.1.2](#))

### 5.2.3. Clase `TextBoxHandler`

Esta clase se encarga de crear y destruir cajas de texto y cargar las distintas fuentes.

Es necesario crear esta clase y no manejar las cajas de texto (ver [5.2.4](#)) directamente para no cargar varias veces la misma fuente en memoria ya que la memoria RAM disponible es muy limitada.

Usa la librería `FreeType 2`.

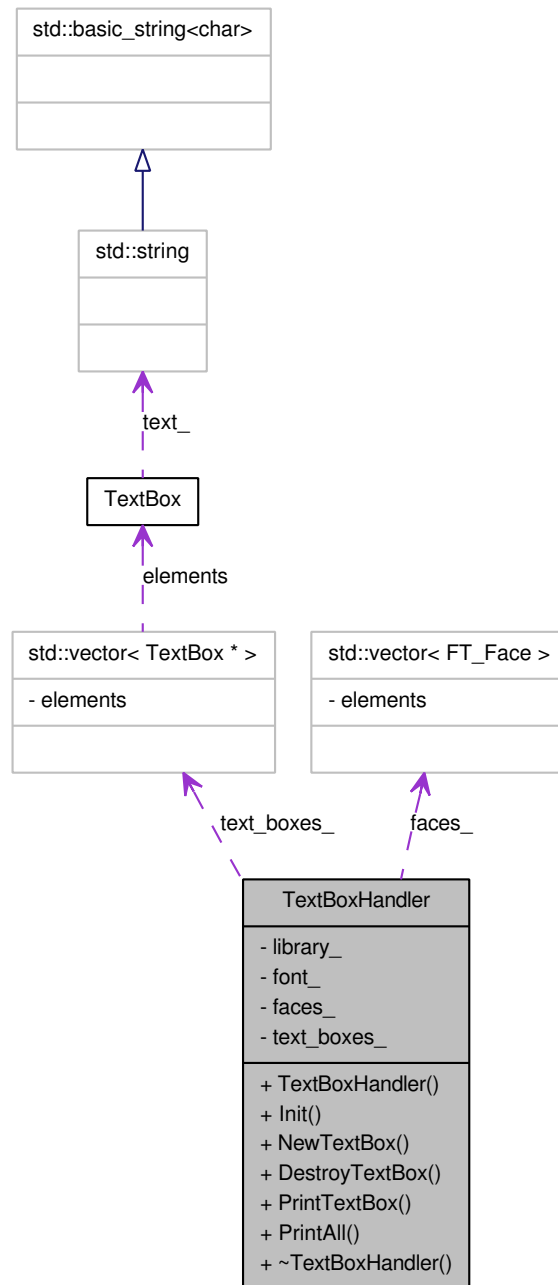


Figura 5.6: Diagrama de colaboración de la clase TextBoxHandler

### Funciones miembro públicas

- `TextBoxHandler ()`  
*Constructor por defecto.*
- `void Init ()`  
*Inicializador.*

- `TextBox * NewTextBox (Types::Screen::selector screen, int bgid, Types::Font font, int size, int x, int y, int width=0, int height=0)`
- `void DestroyTextBox (TextBox *tb)`  
*Destruye una caja de texto.*
- `void PrintTextBox (TextBox *tb)`  
*Imprime una caja de texto ajustándose a las superiores si es necesario.*
- `void PrintAll (Types::Screen::selector)`  
*Imprime todas las cajas de texto de una pantalla ajustándose unas a otras si es necesario.*
- `~TextBoxHandler ()`  
*Destructor.*

### Atributos privados

- `FT_Library library_`  
*Biblioteca de fuentes.*
- `Types::Font font_`  
*Fuente cargada.*
- `std::vector< FT_Face > faces_`  
*Caché de “faces” de fuentes cargadas.*
- `std::vector< TextBox * > text_boxes_`  
*Punteros a las cajas de texto creadas.*

### Member Function Documentation

**TextBox \* TextBoxHandler::NewTextBox (Types::Screen::selector screen, int bgid, Types::Font font, int size, int x, int y, int width = 0, int height = 0)**

Crea una nueva caja de texto en la clase

Parámetros: *screen* Pantalla donde el la caja de texto estará.

*bgid* ID del background

*font* Nombre de la fuente

*size* Tamaño de la fuente

*x* Posición X de la esquina superior izquierda

*y* Posición Y de la esquina superior izquierda

*width* Anchura de la caja en píxeles

*height* Altura máxima de la caja. Por defecto será 0 que significa infinito

Devuelve: Puntero a la caja de texto creada

## 5.2.4. Clase TextBox

Esta clase se encarga de renderizar el texto de la fuente usando la librería FreeType [2] y mantener el texto dentro de la caja, controlando la anchura y altura de la caja.

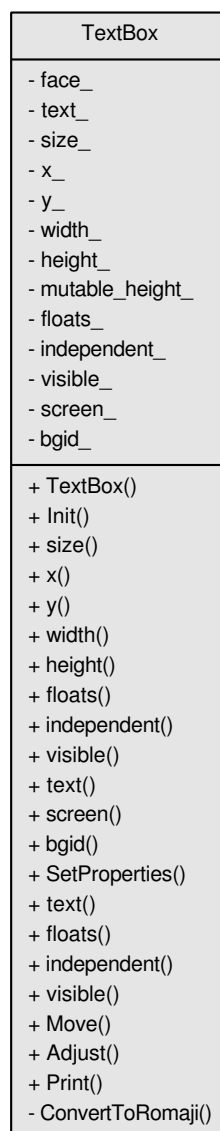


Figura 5.7: Detalle de la clase TextBox en UML

## Funciones miembro públicas

- **TextBox ()**

*Constructor por defecto.*

- void **Init** (**Types::Screen::selector** screen, int bgid, FT\_Face face, int size, int x, int y, int width=0, int height=0)

- int **size** () const

*Devuelve el tamaño de la fuente.*

- int **x** () const

*Devuelve la posición X de la esquina superior izquierda.*

- int **y** () const

*Devuelve la posición Y de la esquina superior izquierda.*

- int **width** () const

*Devuelve la anchura de la caja en píxeles.*

- int **height** () const

*Devuelve la altura de la caja en píxeles.*

- bool **floats** () const

*Devuelve si la caja de texto flota o no.*

- bool **independent** () const

*Devuelve si la caja de texto es independiente o no.*

- bool **visible** () const

*Devuelve si la caja de texto será visible o no.*

- std::string **text** () const

*Devuelve el texto.*

- **Types::Screen::selector** **screen** () const

*Devuelve la pantalla donde está la caja de texto.*

- int **bgid** () const

*Devuelve el ID del background.*

- void **SetProperties** (int x, int y, int width, int height)

*Ajusta la posición y dimensiones de una caja de texto.*

- void **text** (const std::string &str, bool convert\_to\_romaji=false)  
*Ajusta el texto.*
- void **floats** (bool f)  
*Selecciona si la caja de texto flota o no.*
- void **independent** (bool i)  
*Selecciona si la caja de texto es independiente o no.*
- void **visible** (bool v)  
*Selecciona si la caja de texto será visible o no.*
- void **Move** (int x, int y)  
*Mueve la caja de texto a otra posición.*
- void **Adjust** (int y)  
*Ajusta la caja de texto verticalmente si la caja de encima se superpone.*
- void **Print** ()  
*Imprime la caja de texto en su posición.*

## Funciones miembro privadas

- std::string **ConvertToRomaji** (std::string str)  
*Convertir una cadena en kana a romaji.*

## Atributos privados

- FT\_Face **face\_**  
*“Face” de la fuente, no es necesario que tenga el tamaño correcto.*
- std::string **text\_**  
*Cadena de texto a imprimir.*
- int **size\_**  
*Tamaño de la fuente.*
- int **x\_**

*Posición X de la esquina superior izquierda.*

- **int y\_**  
*Posición Y de la esquina superior izquierda.*
- **int width\_**  
*Anchura de la caja de texto en píxeles.*
- **int height\_**  
*Altura de la caja de texto en píxeles.*
- **bool mutable\_height\_**  
*Si la altura es variable o no.*
- **bool floats\_**
- **bool independent\_**
- **bool visible\_**
- **Types::Screen::selector screen\_**  
*Pantalla donde está la caja de texto.*
- **int bgid\_**

## Funciones miembro

- **void TextBox::floats (bool f)**  
Selecciona si el texto flota o no.
- **bool TextBox::floats () const**  
Devuelve si el texto flota o no.
- **void TextBox::independent (bool i)**  
Selecciona si el texto es independiente o no.
- **bool TextBox::independent () const**  
Devuelve si la caja de texto es independiente o no.
- **void TextBox::Init (Types::Screen::selector screen, int bgid, FT\_Face face, int size, int x, int width = 0, int height = 0)**  
Inicializador

Parámetros: **screen** Pantalla donde el la caja de texto estará.

**bgid** ID del background

**face** Objeto FT\_Face de la clase TextBoxHandler

*size* Tamaño de la fuente

*x* Posición X de la esquina superior izquierda

*y* Posición Y de la esquina superior izquierda

*width* Anchura de la caja en píxeles

*height* Altura máxima de la caja. Por defecto será 0 que significa infinito

- void `TextBox::visible` (bool *v*)  
Selecciona si el texto será visible o no.
- bool `TextBox::visible` () const  
Devuelve si el texto será visible o no.

## Miembros de datos

- int `TextBox::bgid_` [privado]  
ID del background
- bool `TextBox::floats_` [privado]  
Una caja flotante no se ajusta a su predecesora ni ajusta a la siguiente. Por defecto: `false`
- bool `TextBox::independent_` [privado]  
Una caja independiente no se imprimirá por la función “PrintAll” de `TextBox-Handler` y necesita imprimirse manualmente. Por defecto: `false`
- bool `TextBox::visible_` [privado]  
En una caja no visible se renderiza el texto con la librería FreeType (se necesita para obtener los datos de las dimensiones) pero no se imprime en la pantalla. Por defecto: `true`

## 5.3. Capa de presentación

### 5.3.1. Detalle de las clases en UML

### 5.3.2. Clase `MainScreenHandler`

Esta clase se encarga de manejar la pantalla principal. La interfaz de la pantalla depende del modo de juego y del modo para la pantalla seleccionado.

Al iniciar este manejador se crean las cajas de texto necesarias en su posición deseada para el modo seleccionado.

Esta clase se encarga también de insertar el texto en las cajas, hacer scroll, etc.

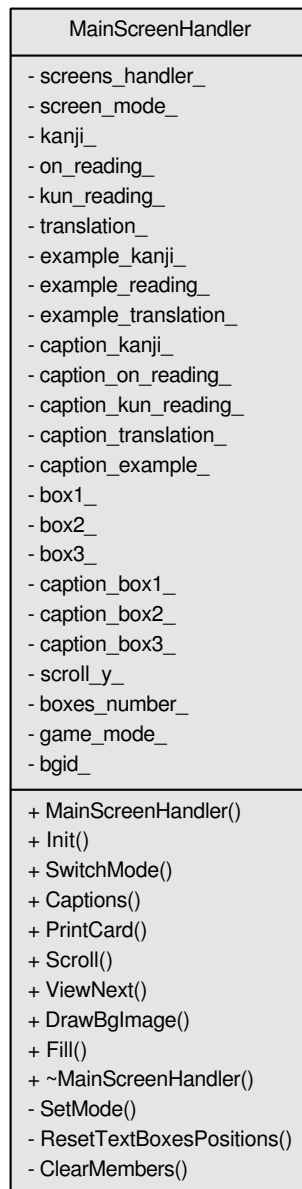


Figura 5.8: Detalle de la clase MainScreenHandler en UML

## Funciones miembro pública

### ■ MainScreenHandler ()

*Constructor.*

- void Init (Types::MainScreenMode::mode screen\_mode, GameMode::mode game\_mode, ScreensHandler \*screens\_handler, int boxes\_number=0)

*Inicializa la pantalla con un modo.*

- void **SwitchMode** (**Types::MainScreenMode::mode** screen\_mode, **GameMode::mode** game\_mode, int boxes\_number=0)  
*Cambiar de modo de pantalla.*
- void **Captions** (std::string box1, std::string box2, std::string box3)  
*Introducir el texto de los títulos.*
- void **PrintCard** (const **Card** &card, bool convert\_to\_romaji=false)  
*Imprimir una carta.*
- void **Scroll** (int sx, int sy)  
*Desplaza la pantalla.*
- bool **ViewNext** ()
- void **DrawBgImage** ()  
*Dibuja la imagen de fondo de la pantalla en el modo actual.*
- void **Fill** (unsigned short color)  
*Rellena la pantalla con un color.*
- ~**MainScreenHandler** ()  
*Destructor.*

### Funciones miembro privadas

- void **SetMode** (**Types::MainScreenMode::mode** screen\_mode, **GameMode::mode** game\_mode, **ScreensHandler** \*screens\_handler, int boxes\_number)  
*Selecciona el modo de pantalla.*
- void **ResetTextBoxesPositions** ()  
*Resetea las posiciones de todas las cajas de texto.*
- void **ClearMembers** ()  
*Elimina todos los miembros.*

### Atributos privados

- **ScreensHandler** \* **screens\_handler\_**  
*Manejador de pantallas base.*

- **Types::MainScreenMode::mode screen\_mode\_**  
*Modo de la pantalla.*
- **TextBox \* kanji\_**  
*Kanji.*
- **TextBox \* on\_reading\_**  
*Lectura “on”.*
- **TextBox \* kun\_reading\_**  
*Lectura “kun”.*
- **TextBox \* translation\_**  
*Traducción.*
- **TextBox \* example\_kanji\_**  
*Ejemplo en kanji.*
- **TextBox \* example\_reading\_**  
*Lectura del ejemplo.*
- **TextBox \* example\_translation\_**  
*Traducción del ejemplo.*
- **TextBox \* caption\_kanji\_**  
*Título del kanji.*
- **TextBox \* caption\_on\_reading\_**  
*Título de la lectura “on”.*
- **TextBox \* caption\_kun\_reading\_**  
*Título de la lectura “kun”.*
- **TextBox \* caption\_translation\_**  
*Título de la traducción.*
- **TextBox \* caption\_example\_**  
*Título del ejemplo.*
- **TextBox \* box1\_**

*Caja 1.*

- **TextBox \* box2\_**

*Caja 2.*

- **TextBox \* box3\_**

*Caja 3.*

- **TextBox \* caption\_box1\_**

*Título de la caja 1.*

- **TextBox \* caption\_box2\_**

*Título de la caja 2.*

- **TextBox \* caption\_box3\_**

*Título de la caja 3.*

- **int scroll\_y\_**

*Desplazamiento vertical.*

- **int boxes\_number\_**

*Número de cajas: [0-3]*

- **GameMode::mode game\_mode\_**

*Modo de juego.*

- **int bgid\_**

*ID del background.*

## **Funciones miembro**

- **void MainScreenHandler::Init (Types::MainScreenMode::mode screen\_mode, GameMode::mode game\_mode, ScreensHandler \* screens\_handler, int boxes\_number = 0)**

Inicializa la pantalla con un modo.

Parámetros: *screen\_mode* Modo de la pantalla

*game\_mode* Modo de juego

*screens\_handler* Manejador de pantallas ya creado e inicializado

*boxes\_number* Número de cajas de texto para ser usadas: [0-3]

- void MainScreenHandler::SetMode (**Types::MainScreenMode::mode** *screen\_mode*, **GameMode::mode** *game\_mode*, **ScreensHandler** \* *screens\_handler*, int *boxes\_number*) [private]

Selecciona el modo.

Parameters: *screen\_mode* Modo de la pantalla

*game\_mode* Modo de juego

*screens\_handler* Manejador de pantallas ya creado e inicializado

*boxes\_number* Número de cajas de texto para ser usadas: [0-3]

- void MainScreenHandler::SwitchMode (**Types::MainScreenMode::mode** *screen\_mode*, **GameMode::mode** *game\_mode*, int *boxes\_number* = 0)

Cambiar el modo de pantalla.

Parámetros: *screen\_mode* Modo de la pantalla

*game\_mode* Modo de juego

*boxes\_number* Número de cajas de texto para ser usadas: [0-3].

- bool MainScreenHandler::ViewNext ()

Muestra la siguiente caja invisible

Devuelve: true si todas eran ya visibles

### 5.3.3. Clase SubScreenHandler

Esta clase se encarga de manejar la pantalla inferior. Al igual que la clase MainScreenHandler (ver 5.3.2), esta clase se encarga de manejar la pantalla inferior. La interfaz dependerá del modo de juego y el modo de pantalla seleccionado.

Al iniciar este manejador se crean las cajas de texto necesarias en su posición para el modo seleccionado.

Esta clase se encarga también de insertar el texto en las cajas, etc.

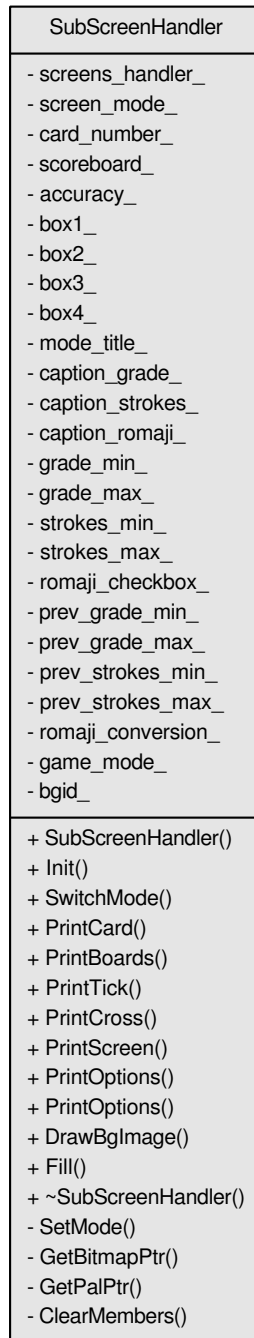


Figura 5.9: Detalle de la clase SubScreenHandler en UML

## Public Member Functions

### ■ SubScreenHandler ()

*Constructor.*

- **void Init** (**Types::SubScreenMode::mode** screen\_mode, **GameMode::mode** game\_mode, **ScreensHandler** \*screens\_handler)  
*Inicializa la pantalla con un modo.*
- **void SwitchMode** (**Types::SubScreenMode::mode** screen\_mode, **GameMode::mode** game\_mode)  
*Switch mode.*
- **void PrintCard** (const **Card** &card)  
*Imprime una carta.*
- **void PrintBoards** (int score, int accuracy)  
*Imprime el porcentaje de aciertos y los puntos.*
- **void PrintTick** (int position)  
*Imprime un “tick”.*
- **void PrintCross** (int position)  
*Imprime una cruz.*
- **void PrintScreen** (std::string kanji1, std::string kanji2, std::string kanji3, std::string kanji4, int score, int answers)  
*Imprime los kanjis.*
- **void PrintOptions** (unsigned grade\_min, unsigned grade\_max, unsigned strokes\_min, unsigned strokes\_max, bool romaji\_conversion)  
*Imprime el rango de grados, trazos y la conversión a romaji en el menú de opciones.*
- **void PrintOptions** (bool romaji\_conversion)  
*Imprime la conversión a romaji en el menú de opciones.*
- **void DrawBgImage** ()  
*Dibuja la imagen de fondo de la pantalla en el modo actual.*
- **void Fill** (unsigned short color)  
*Rellena la pantalla con un color.*
- **~SubScreenHandler** ()  
*Destructor.*

## Funciones miembro privadas

- void **SetMode** (**Types::SubScreenMode::mode** screen\_mode, **GameMode::mode** game\_mode, **ScreensHandler** \*screens\_handler)  
*Selecciona el modo de pantalla.*
- const unsigned int \* **GetBitmapPtr** ()  
*Devuelve un puntero al bitmap de la imagen del modo actual.*
- const unsigned short \* **GetPalPtr** ()  
*Devuelve un puntero a la paleta de la imagen del modo actual.*
- void **ClearMembers** ()  
*Elimina todos los miembros.*

## Atributos privados

- **ScreensHandler** \* **screens\_handler\_**  
*Manejador de pantallas base.*
- **Types::SubScreenMode::mode** **screen\_mode\_**  
*Modo de pantalla.*
- **TextBox** \* **card\_number\_**  
*Número de carta.*
- **TextBox** \* **scoreboard\_**  
*Puntuación.*
- **TextBox** \* **accuracy\_**  
*Porcentaje de acierto.*
- **TextBox** \* **box1\_**  
*Caja 1.*
- **TextBox** \* **box2\_**  
*Caja 2.*
- **TextBox** \* **box3\_**  
*Caja 3.*

- **TextBox \* box4\_**  
*Caja 4.*
- **TextBox \* mode\_title\_**  
*Título del modo.*
- **TextBox \* caption\_grade\_**  
*Título de grados en opciones.*
- **TextBox \* caption\_strokes\_**  
*Título de trazos en opciones.*
- **TextBox \* caption\_romaji\_**  
*Título de conversión a romaji en opciones.*
- **TextBox \* grade\_min\_**  
*Mínimo grado en opciones.*
- **TextBox \* grade\_max\_**  
*Máximo grado en opciones.*
- **TextBox \* strokes\_min\_**  
*Mínimo número de trazos en opciones.*
- **TextBox \* strokes\_max\_**  
*Máximo número de trazos en opciones.*
- **TextBox \* romaji\_checkbox\_**  
*Casilla de conversión a romaji.*
- **unsigned prev\_grade\_min\_**  
*Mínimo grado anterior.*
- **unsigned prev\_grade\_max\_**  
*Máximo grado anterior.*
- **unsigned prev\_strokes\_min\_**  
*Mínimo número de trazos anterior.*
- **unsigned prev\_strokes\_max\_**  
*Máximo número de trazos anterior.*

- **bool romaji\_conversion\_**  
*Conversión a romaji.*
- **GameMode::mode game\_mode\_**  
*Modo de juego.*
- **int bgid\_**  
*ID del background.*

### Funciones miembro

- **void SubScreenHandler::Init (Types::SubScreenMode::mode screen\_mode, GameMode::mode game\_mode, ScreensHandler \* screens\_handler)**

Inicializa la pantalla con un modo.

Parámetros: *screen\_mode* Modo de la pantalla

*game\_mode* Modo de juego

*screens\_handler* Manejador de pantallas ya creado e inicializado

- **void SubScreenHandler::SetMode (Types::SubScreenMode::mode screen\_mode, GameMode::mode game\_mode, ScreensHandler \* screens\_handler) [private]**

Selecciona el modo de pantalla.

Parámetros: *screen\_mode* Modo de la pantalla

*game\_mode* Modo de juego

*screens\_handler* Manejador de pantallas ya creado e inicializado

- **void SubScreenHandler::SwitchMode (Types::SubScreenMode::mode screen\_mode, GameMode::mode game\_mode)**

Cambiar el modo de pantalla.

Parámetros: *screen\_mode* Modo de la pantalla

*game\_mode* Modo de juego

### 5.3.4. Clase ScreensHandler

Esta clase se encarga de manejar las cosas comunes a las dos pantallas. De momento sólo maneja el manejador de cajas de texto (ver 5.2.3), que ha de ser común a las dos cajas para que no se carguen dos veces las fuentes.

Esta clase se creará al principio en la clase DSMemorizer y se pasará un puntero a ella a los diferentes modos y manejadores de pantallas.

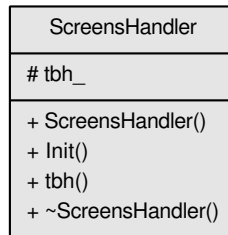


Figura 5.10: Detalle de la clase ScreensHandler en UML

### Funciones miembro públicas

- **ScreensHandler ()**

*Constructor.*

- **void Init ()**

*Inicializador.*

- **TextBoxHandler \* tbh () const**

*Devuelve un puntero al manejador de cajas de texto. (ver 5.2.3)*

- **~ScreensHandler ()**

*Destructor.*

### Atributos protegidos

- **TextBoxHandler \* tbh\_**

*Manejador de cajas de texto.*

### 5.3.5. Clase SoundHandler

Esta clase se encarga de manejar los sonidos del juego. Los carga, descarga y reproduce o interrumpe según se le pida. Usa la librería maxmod [9]

Esta clase se creará al principio en la clase DSMemorizer y se pasará un puntero a ella a los diferentes modos.

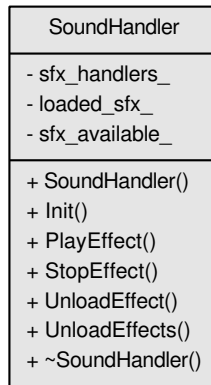


Figura 5.11: Detalle de la clase SoundHandler en UML

### Tipos públicos

- enum **SFX** { **ACTION**, **THEME** }  
*Efectos de sonido disponibles.*

### Funciones miembro públicas

- **SoundHandler ()**  
*Constructor.*
- void **Init ()**  
*Inicializador.*
- void **PlayEffect (SFX sfx)**  
*Cargar un efecto en memoria si no está cargado y entonces lo reproduce.*
- void **StopEffect (SFX sfx)**  
*Detener la reproducción de un efecto.*
- void **UnloadEffect (SFX sfx)**
- void **UnloadEffects ()**
- **~SoundHandler ()**  
*Destructor.*

## Atributos privados

- `mm_sfxhand * sfx_handlers_`  
*Vector de manejadores de efectos de sonido.*
- `bool * loaded_sfx_`  
*Vector de efectos de sonido cargados o no.*
- `mm_sound_effect * sfx_available_`  
*Estructuras para todos los efectos de sonido disponibles en el banco de sonidos.*

## Enumeraciones miembro

- `enum SoundHandler::SFX`  
Efectos de sonido disponibles.  
  
Enumerados: ***ACTION*** Sonido reproducido al hacer una acción.  
***THEME*** Tema para el menú principal.

## Funciones miembro

- `void SoundHandler::UnloadEffect (SFX sfx)`  
Descargar un efecto específico de la memoria. Se detendrá su reproducción antes en caso de que se estuviese reproduciendo.
- `void SoundHandler::UnloadEffects ()`  
Descargar todos los efectos de la memoria. Se detendrá su reproducción antes en caso de que alguno se estuviese reproduciendo.

# Capítulo 6

## Implementación

Para implementar este proyecto he encontrado bastantes dificultades, sobre todo debidas a que la plataforma no la conocía en absoluto, es más, me compré la videoconsola para desarrollar el proyecto.

En primer lugar, la librería libnds es de bastante bajo nivel y excepto cuando he podido usar DMA, el resto he tenido que pintarlo en la pantalla píxel a píxel, por lo que me ha sido necesario implementar el espacio de nombres “Graphics” (ver 6.4) donde he implementado diversas funciones para imprimir bitmaps y diversas funcionalidades que he ido necesitando a medida que lo implementaba.

Los 4 modos implementados son análogos entre sí de forma que las interfaces de las pantallas son parecidas. De esta forma he encapsulado todas las cosas relativas a la interfaz de cada pantalla en dos clases, una para cada pantalla. (ver 5.3.2 y 5.3.3)

De esta forma, los modos de juego llaman a los manejadores de las pantallas y simplemente le dicen el modo que quieren usar y no tienen que hacer nada más. Luego simplemente se le envían los datos y éstas clases hacen el resto para mostrarlo al usuario.

De esta forma es como he conseguido aislar la capa de presentación de la capa de dominio.

### 6.1. El problema de los colores

La videoconsola Nintendo DS, para las pantallas se usa un modo de color indexado, de forma que en el buffer de vídeo para cada capa, lo que hay son índices de los colores de los píxeles en la paleta de colores.

Las imágenes se encuentran en “png”, pero para incluirlas en el juego, es necesario pasarlas a vectores de valores que serán los índices de los colores y la paletas correspondientes. Ésta conversión se hace con la utilidad GRIT [6].

Para imprimir texto se necesitan los colores negro y gris, pero para imprimirlos hace falta que estén en la paleta, porque al imprimirlos lo único que se hace es, en el píxel adecuado, escribir la posición de la paleta donde está el color de ese píxel.

Cuando la imagen de fondo es simple no hay problema porque los colores no van a llenar toda la paleta, así que basta con añadirlos al final a la paleta, pero cuando la imagen de fondo es compleja, la paleta se llenará, y no será posible sobrescribir ningún color con el negro y el gris porque los píxeles de ese color, ahora se imprimirán en negro y gris.

También existe el mismo problema con las imágenes de “tick” y “cross”, que son más complejas que dos colores para el texto y que precisan varios colores cada una.

Para esto, he ideado una solución consistente en añadir una línea de píxeles extra a las imágenes sobre las que hará falta imprimir texto, que contenga dos píxeles, uno negro y otro gris, de forma que se incluyan en la paleta de la imagen. Luego una función que buscará los colores en las paletas y los guardará (ver función `SetColor`).

Para solucionar el problema de los “tick” y “cross” he adoptado una solución similar que consiste en incluir estas dos imágenes debajo de la imagen de fondo, de forma que sus colores ya se incluyan en la paleta. Luego para poder imprimirlas he creado la función `PrintBitmapRegion` que permite imprimir un trozo de una imagen en una determinada posición de un background de una pantalla.

Para no consumir colores extras, como fondo de estas imágenes y colores extras presentes en la imagen se usará algún color ya presente en la imagen original.

Nada de esto interfiere en la visualización de las imágenes, ya que sólo se imprime la zona de la pantalla y lo que queda por debajo no se ve. El único inconveniente es que la utilidad GRIT tendrá que simplificar algo más los colores de la imagen para poder incluir en la paleta los colores de estas imágenes extra.

Un ejemplo de esto puede verse en la imagen “kanji\_choose\_options\_sub\_bg.png”:

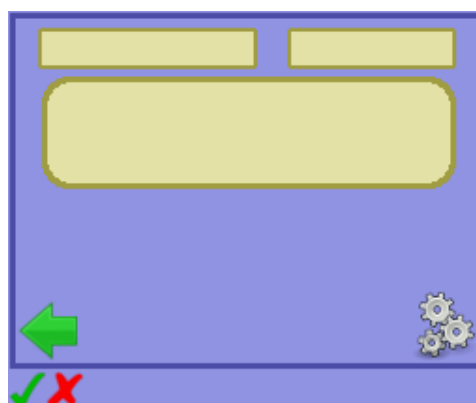


Figura 6.1: Ejemplo de imagen con paleta y “tick” y “cross” incrustados

En esta imagen se han metido debajo de la imagen que se va a imprimir, las imágenes del “tick” y el “cross”, así como dos píxeles, uno negro y uno gris en la última fila de la imagen.

De esta forma, al pasar la imagen a GRIT, nos elabora la siguiente paleta:

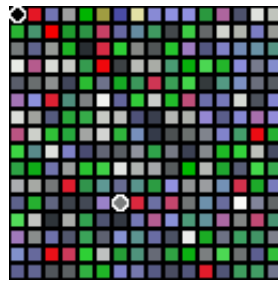


Figura 6.2: Paleta de la imagen generada por GRIT

Aquí hemos señalado los colores correspondientes al negro y al gris.

Si vemos la paleta generada por GRIT en el archivo ensamblador, es:

```
.section .rodata
.align 2
.global kanji_choose_options_sub_bgPal @ 512 unsigned chars
kanji_choose_options_sub_bgPal:
.hword 0x0000,0x105C,0x59CE,0x5293,0x02C0,0x2273,0x5129,0x539C
.hword 0x6A31,0x7251,0x7252,0x1E65,0x4D95,0x3549,0x6F9C,0x5F18
.hword 0x1AC5,0x3648,0x001E,0x1663,0x2245,0x3118,0x51AC,0x4E2C
.hword 0x3267,0x2528,0x2327,0x35AC,0x675A,0x56D6,0x61EF,0x4E73
.hword 0x3DEE,0x458C,0x4A72,0x12C3,0x18A5,0x20BA,0x1B25,0x4230
.hword 0x294A,0x1704,0x61F3,0x414A,0x6210,0x5A2E,0x524D,0x77BD
.hword 0x424A,0x4576,0x6B7B,0x6339,0x2666,0x043D,0x2107,0x5AF7
.hword 0x52B5,0x3A69,0x06C0,0x2B49,0x16C4,0x6E31,0x45EE,0x6630

.hword 0x356B,0x35AD,0x4A51,0x2D08,0x1AE5,0x45CD,0x12A2,0x2E47
.hword 0x28D9,0x2A67,0x2748,0x1CE6,0x426A,0x624F,0x59D4,0x7BDE
.hword 0x3DAD,0x316A,0x6E51,0x189B,0x3E29,0x420F,0x4D8C,0x1F26
.hword 0x6739,0x16E4,0x3D37,0x55AD,0x5DEE,0x6B7A,0x2D49,0x6A10
.hword 0x6F7B,0x4E94,0x3D6A,0x1E85,0x6759,0x26A6,0x6338,0x5EF7
.hword 0x3D49,0x24E7,0x5AD6,0x56D5,0x6A32,0x456B,0x6A30,0x55B4
.hword 0x6E32,0x1B06,0x4157,0x1E84,0x5294,0x5694,0x6B5A,0x1EC6
.hword 0x5E2E,0x24E6,0x2529,0x35CD,0x4610,0x6610,0x3668,0x5F17

.hword 0x083D,0x1AA4,0x2347,0x462B,0x620F,0x3929,0x398C,0x7BFF
.hword 0x316B,0x4E51,0x2685,0x41EE,0x2907,0x4630,0x12E3,0x1F46
.hword 0x3128,0x2285,0x0AA1,0x55CE,0x1B05,0x2348,0x5AD5,0x2D6A
.hword 0x02E0,0x1A84,0x396A,0x2768,0x2AA7,0x739C,0x5AB5,0x5AF6
.hword 0x52B4,0x3DCE,0x147B,0x2646,0x49AC,0x4A6B,0x464B,0x6613
.hword 0x2265,0x6E52,0x4A52,0x5A4E,0x5DF4,0x24DA,0x16A3,0x5DEF
.hword 0x458B,0x1AC4,0x418C,0x2507,0x2928,0x3DEF,0x1CBB,0x51CD
.hword 0x3518,0x6F9B,0x39CE,0x4DAD,0x564D,0x4E10,0x318C,0x2F6A

.hword 0x6318,0x1CC5,0x56B5,0x6317,0x2A86,0x3949,0x20E6,0x59EE
.hword 0x51B5,0x6A51,0x4E4C,0x3938,0x4210,0x3A49,0x3E69,0x4976
.hword 0x4631,0x55CD,0x354A,0x2E87,0x398B,0x4A4B,0x51AD,0x6A50
.hword 0x3149,0x12A3,0x41CD,0x3287,0x3248,0x460F,0x6650,0x4DAC
.hword 0x0C5D,0x2CF9,0x1B45,0x1A64,0x3E49,0x39AC,0x2286,0x0AC1
.hword 0x41EF,0x2B69,0x39CD,0x496B,0x1EE6,0x498B,0x498C,0x2D4A
.hword 0x318B,0x22C6,0x12C4,0x660F,0x55EE,0x416A,0x3129,0x3D4A
.hword 0x147C,0x394A,0x2E67,0x3268,0x396B,0x2949,0x45CE,0x12E4
```

Figura 6.3: Ejemplo de paleta generada por GRIT en hexadecimal

Aquí señalado los colores negro (0x0000) y gris (0x3DEF). Este es su valor combinando sus componentes para valores de rojo, verde y azul entre 0 y 31.

El negro (0x0000) será  $r=0$ ,  $g=0$ ,  $b=0$  y el gris (0x3DEF) será  $r=15$ ,  $g=15$ ,  $b=15$ .

Al igual que con estos colores, GRIT integrará los colores para el “tick” y el “cross” en la paleta, lo que nos hará posible dibujarlos en la pantalla fácilmente sólo recortando la zona y pintándola donde se desee.

## 6.2. Renderizado de texto

Para el renderizado de texto se usa la librería FreeType [2]. De manejarla se encarga la clase `TextBoxHandler` (ver 5.2.3) y `TextBox` (ver 5.2.4).

La clase `TextBoxHandler` se encarga de inicializar la librería y cargar nuevas fuentes que no se encuentren cargadas, así como mantener registradas las fuentes cargadas para liberarlas al salir.

Esta clase es necesaria porque si sólo usásemos la clase `TextBox` tendríamos la misma fuente cargada en memoria varias veces, algo muy ineficiente, además, dado que la cantidad de memoria es muy limitada, esto no sería siquiera posible.

De esta forma, cada vez que se quiera crear una nueva caja de texto, se llama a la clase `TextBoxHandler`, que se ocupará de cargar la fuente deseada si no lo estuviera y de crear la caja de texto, a la que devolverá un puntero.

La clase `TextBoxHandler` también guarda un puntero a los objetos `TextBox` creados para poder destruirlos cuando sea necesario y para imprimir las cajas ajustándose unas a otras.

La clase `TextBox` es la que se encarga de renderizar los símbolos llamando a la librería FreeType y de controlar que el texto se mantenga dentro de la caja, cambiando de línea si es necesario y posible.

Esta clase, se encarga de ir cogiendo los símbolos de la cadena de texto a imprimir a través de la librería UTF8-CPP [10], para coger cada símbolo aunque ocupe varios bytes en caso de que sea complejo, y llamar a la librería FreeType para que los renderice.

Esta librería renderiza cada símbolo en un bitmap que será un array de bytes, conteniendo cada uno un valor entre 0 y 255, y calcula las medidas de ese carácter, que nos dirán cuánto ocupa de ancho, lo que hay que avanzar el cursor, dónde empieza el carácter dentro del bitmap, etc.

Todos estos datos son los que usa la clase `TextBox` para calcular dónde tiene que imprimir el carácter. Para imprimirlo una vez tenga las coordenadas exactas sólo habrá que llamar a la función `PrintBitmap`. Que además convertirá los colores adecuadamente usando los colores negro y gris anteriormente buscados en la paleta como se describe en 6.1.

Cada caja de texto tiene una anchura y altura máximas, siendo esta última variable si se desea. En caso de que el texto no quepa porque sea demasiado largo, se continúa escribiendo en la siguiente línea, si la altura lo permite, y se actualiza ésta, en caso de que se haya podido.

Si la altura no es variable y no cabe más, no se imprimirá más.

## 6.3. Alineado de las cajas de texto

Es posible, y probable que el texto de una caja de texto ocupe más de una línea. Si ésto ocurre, y el texto invade la caja que se encuentre debajo de ella en la pantalla, ésta, y las que haya debajo suya, si se vuelven a invadir, habrá que moverlas hacia abajo.

Para controlar ésto, la clase `TextBoxHandler` (ver 5.2.3) implementa un método “Adjust” para mover las cajas si fuese necesario.

Éste método se llama desde la función “PrintTextBox” y “PrintAll”, que imprime todas las cajas de texto asociadas a una pantalla.

La función “PrintTextBox” es para imprimir una caja específica. Ésta mira hasta dónde llega la caja anterior y si invade la posición de la caja actual, mueve la caja hacia abajo hasta que no se superpongan y entonces imprime el texto de la caja.

La función “PrintAll” imprime secuencialmente las cajas de texto, ajustándolas todas entre sí. No se puede calcular inicialmente la altura que va a tener la caja, si ésta es variable, ya que calcularlo implicaría renderizar todos los símbolos, perdiendo un tiempo precioso, para luego tener que volverlos a renderizar (o guardarlos pero podría ocupar mucha memoria). Así que la altura se actualiza al imprimir cada caja de texto. Por ésto, las cajas sólo se pueden ajustar a las anteriormente impresas.

De esta forma, el funcionamiento de esta función es:

1. Se ajusta la caja de texto a la posición y dimensiones de la anterior.
2. Se imprime la caja de texto.
3. Se vuelve al paso 1

Las cajas de texto también es posible situarlas fuera de este flujo de cajas, simplemente diciendo que flotan. Con lo que ni ajustarán a otras cajas, ni se verán afectadas por la posición de otras cajas.

También es posible hacer a una caja de texto no visible, de forma que las cajas se ajustan a ella, porque el texto se renderiza, pero no se imprime en la pantalla.

Una última propiedad que tienen las cajas de texto es que pueden ser también independientes. Éstas cajas de texto serán ignoradas por las función “PrintAll” y deberán imprimirse manualmente con “PrintTextBox” o directamente llamando a “Print” para la caja deseada.

Éstos atributos y las funciones que permiten ajustarlos se encuentran documentados en 5.2.4.

Un ejemplo de esto puede verse aquí:

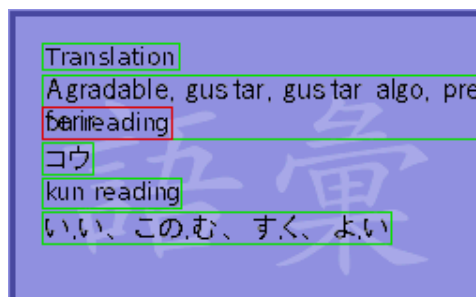


Figura 6.4: Ejemplo de cajas de texto no alineadas

En esta imagen se ve como la caja de la traducción del kanji es demasiado alta y se superpone con la caja del título de la lectura “on”.

Ajustando las cajas antes de imprimirlas con “PrintAll” imprimirá las cajas adecuadamente alineadas para que no se superpongan:

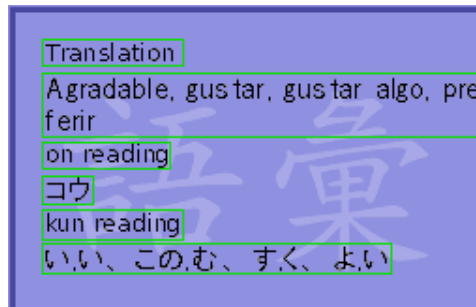


Figura 6.5: Ejemplo de cajas de texto alineadas

## 6.4. Namespace Graphics

En el espacio de nombres “Graphics” he implementado diversas funciones que detallo a continuación:

- namespace **ColorTransformation**

*Transformaciones de color.*

- namespace **SplashEffect**

*Efectos de splash disponibles.*

### 6.4.1. Funciones

- void **FactorizePalette** (const u16 \*src\_palette, u16 \*dst\_palette, int palette\_size, u8 factor, **ColorTransformation::mode** trans)

*Multiplica una paleta por un factor haciendo los colores más claros o más oscuros.*

- void **SplashImage** (const unsigned int \*bitmap, const unsigned short \*palette, **Types::Screen::selector** screen, int bgid, u8 effect)

*Hace un efecto “splash” con una imagen en una pantalla.*

- void **PrintBitmap** (int x, int y, int width, int height, const unsigned int \*bitmap, unsigned short key\_color, int bgid, **Types::Screen::selector** screen)

*Imprime un bitmap de GRIT en un background.*

- void **PrintBitmapRegion** (int dst\_x, int dst\_y, int region\_x, int region\_y, int region\_width, int region\_height, int bmp\_width, int bmp\_height, const unsigned int \*bitmap, unsigned short key\_color, int bgid, **Types::Screen::selector** screen, bool use\_key\_color=true)

*Imprime una región de un bitmap de GRIT en un background.*

- void **RedrawBgRegion** (int x, int y, int region\_width, int region\_height, **Types::MainScreenMode::mode** screen\_mode, **GameMode::mode** game\_mode, int bgid)

*Redibuja una región de un background en la pantalla superior.*

- void **RedrawBgRegion** (int x, int y, int region\_width, int region\_height, **Types::SubScreenMode::mode** screen\_mode, **GameMode::mode** game\_mode, int bgid)

*Redibuja una región de un background en la pantalla inferior.*

- void **PrintBitmap** (int x, int y, int width, int height, const unsigned char \*bitmap, int bgid, **Types::Screen::selector** screen)

*Imprime un bitmap de FreeType en un background en una pantalla.*

- void **Fill** (unsigned short color, **Types::Screen::selector** screen)

*Rellena la pantalla con un color.*

- unsigned short **FindColor** (unsigned short color, **Types::Screen::selector** screen, unsigned short offset=0)
- void **SetColors** ()

*Ajusta las variables de los colores negro y gris para renderizar el texto.*

- const unsigned int \* **GetBitmapPtr** (**Types::MainScreenMode::mode** screen\_mode)
- const unsigned short \* **GetPalPtr** (**Types::MainScreenMode::mode** screen\_mode)
- const unsigned int \* **GetBitmapPtr** (**Types::SubScreenMode::mode** screen\_mode, **GameMode::mode** game\_mode)
- const unsigned short \* **GetPalPtr** (**Types::SubScreenMode::mode** screen\_mode, **GameMode::mode** game\_mode)

## 6.4.2. Documentación de las funciones

**void Graphics::FactorizePalette** (const u16 \* src\_palette, u16 \* dst\_palette, int palette\_size, u8 factor, **ColorTransformation::mode** trans)

Multiplica una paleta por un factor haciendo los colores más claros o más oscuros.

Esta función está ideada para la función `SplashImage`. Lo que hace es multiplicar la paleta por un número, que dependiendo del tipo de transformación de color que se quiera aplicar, aclarará u oscurecerá los colores de la paleta en el factor dado.

Parámetros: *src\_palette* Paleta de colores de origen.

*dst\_palette* Paleta de colores de destino.

*palette\_size* Tamaño de la paleta en bytes.

*factor* Factor por el que multiplicar los valores. Deberá estar en el rango [0-256].  
Cuanto más bajo sea, más claro/oscura serán los colores.

*trans* Si *trans* == LIGHTER la paleta se aclarará.  
Si *trans* == DARKER la paleta se oscurecerá.

**unsigned short Graphics::FindColor (unsigned short *color*,  
Types::Screen::selector *screen*, unsigned short *offset* = 0)**

Busca un color en la paleta de la pantalla dada. Se le puede proporcionar un *offset* donde empezar a buscar.

**void Graphics::Fill (unsigned short *color*, Types::Screen::selector *screen*)**

Esta función rellena la paleta de colores con un color dado coloreando la pantalla de ese color. Se rellena la paleta en vez de cambiar los índices del color de los píxeles porque cambiar la paleta es más eficiente ya que su tamaño es mucho menor.

Para rellenar la paleta se usa DMA por lo que se hace muy rápido.

**void Graphics::SetColor ()**

Esta función busca en la paleta de colores (con la función `FindColor`) los colores negro y gris, necesarios para renderizar el texto.

Esta función deberá llamarse cada vez que se cambie de imagen de fondo para que el texto pueda ser renderizado con los colores correctos.

**const unsigned int \* Graphics::GetBitmapPtr (Types::SubScreenMode::mode  
*screen\_mode*, GameMode::mode *game\_mode*)**

Devuelve la dirección del bitmap de fondo para el modo de pantalla seleccionado para la pantalla inferior.

**const unsigned int \* Graphics::GetBitmapPtr (Types::MainScreenMode::mode screen\_mode)**

Devuelve la dirección del bitmap de fondo para el modo de pantalla seleccionado para la pantalla superior.

**const unsigned short \* Graphics::GetPalPtr (Types::SubScreenMode::mode screen\_mode, GameMode::mode game\_mode)**

Devuelve la dirección de la paleta de la imagen de fondo para el modo seleccionado para la pantalla inferior.

**const unsigned short \* Graphics::GetPalPtr (Types::MainScreenMode::mode screen\_mode)**

Devuelve la dirección de la paleta de la imagen de fondo para el modo seleccionado para la pantalla superior.

**void Graphics::PrintBitmap (int x, int y, int width, int height, const unsigned char \* bitmap, int bgid, Types::Screen::selector screen)**

Imprime un bitmap de FreeType en un background.

Esta función imprime un bitmap de Freetype dado por un vector de bytes indicando el nivel de gris de cada píxel en una escala [0-255], en una determinada posición de alguna capa de background de la pantalla seleccionada, transformando los colores a negro y gris.

Esta transformación se hace porque no podemos almacenar 256 niveles de gris en la paleta, (y tampoco haría falta), de forma que hemos establecido unos umbrales que quedan visualmente bien con los que se reducen los colores sólo a 2, negro y gris, a los que se transforman los colores antes de imprimirlos.

Los píxeles con color por debajo del umbral del gris no se imprimirán permaneciendo transparentes.

Parámetros: **x** Coordenada X de la esquina superior izquierda donde imprimir el bitmap.

**y** Coordenada Y de la esquina superior izquierda donde imprimir el bitmap.

**width** Anchura del bitmap.

**height** Altura del bitmap.

**bitmap** Puntero a un array de palabras de 4 bytes conteniendo los índices de los colores de los píxeles del bitmap.

**bgid** ID del background donde imprimir el bitmap.

**screen** Pantalla donde imprimir.

```
void Graphics::PrintBitmap (int x, int y, int width, int height, const unsigned int * bitmap, unsigned short key_color, int bgid, Types::Screen::selector screen)
```

Esta función imprime en un determinado background de una de las pantallas una imagen generada por la utilidad GRIT a partir de un png.

Esta función reordena las medias palabras de las estructuras generadas por la utilidad GRIT de las imágenes y permite aplicar un color clave para no imprimir los píxeles de ese color.

La reordenación obliga a tener imágenes cuyas dimensiones en píxeles sea múltiplo de 4.

Parámetros: *x* Coordenada X de la esquina superior izquierda donde imprimir el bitmap.

*y* Coordenada Y de la esquina superior izquierda donde imprimir el bitmap.

*width* Anchura del bitmap.

*height* Altura del bitmap.

*bitmap* Puntero a un array de palabras de 4 bytes conteniendo los índices de los colores de los píxeles del bitmap.

*key\_color* Color máscara.

*bgid* ID del background donde imprimir el bitmap.

*screen* Pantalla donde imprimir.

```
void Graphics::PrintBitmapRegion (int dst_x, int dst_y, int region_x, int region_y, int region_width, int region_height, int bmp_width, int bmp_height, const unsigned int * bitmap, unsigned short key_color, int bgid, Types::Screen::selector screen, bool use_key_color = true)
```

Esta función imprime en una determinada posición de un background de alguna pantalla, una determinada región de un bitmap.

Debido a que los bitmaps de entrada provienen de GRIT, se tiene que hacer una reordenación de las medias palabras y ésto obliga a que las regiones tengan dimensiones múltiplos de 4.

También admite que se aplique un color clave para no imprimir los píxeles de ese color.

Parámetros: *dst\_x* Coordenada X de la esquina superior izquierda donde imprimir el bitmap.

*dst\_y* Coordenada Y de la esquina superior izquierda donde imprimir el bitmap.

*region\_x* Coordenada X de la esquina superior izquierda donde empieza la región a imprimir en el bitmap.

*region\_y* Coordenada Y de la esquina superior izquierda donde empieza la región a imprimir en el bitmap.

*region\_width* Anchura de la región a imprimir.

*region\_height* Altura de la región a imprimir.

*bmp\_width* Anchura del bitmap.

*bmp\_height* Altura del bitmap.

*bitmap* Puntero a un array de palabras de 4 bytes conteniendo los índices de los colores de los píxeles del bitmap.

*key\_color* Color máscara.

*bgid* ID del background donde imprimir.

*screen* Pantalla donde imprimir

*use\_key\_color* Imprimir o no los píxeles del color de la máscara.

**void Graphics::SplashImage (const unsigned int \* *bitmap*, const unsigned short \* *palette*, Types::Screen::selector *screen*, int *bgid*, u8 *effect*)**

Esta función hace una animación de “splash” con la imagen que se le pase. Puede hacer dos efectos, o incluso uno después del otro, que son hacer a la imagen aparecer desde una pantalla en negro o hacerla oscurecerse hasta llegar al negro.

También es posible hacer los dos efectos haciendo un “OR” entre los efectos al pasarlos a la función como parámetros, de forma que primero aparecerá y luego se oscurecerá.

Parámetros: *bitmap* Bitmap de la imagen.

*palette* Paleta de la imagen.

*screen* Selector de pantalla

*bgid* ID del background donde imprimir.

*effect* Puede ser SplashEffect::APPEAR o SplashEffect::DISSOLVE.

Ambos valores pueden ponerse con un “OR” y el bitmap aparecerá y luego se oscurecerá.

# Capítulo 7

## Pruebas

La metodología de desarrollo de este juego ha sido basada en componentes por lo que los componentes se han ido desarrollando haciendo pequeñas pruebas sobre ellos a medida que se iban implementando nuevas funcionalidades. Por tanto, no existe la tradicional fase de pruebas propiamente dicha sino que éstas se han ido haciendo poco a poco a lo largo del desarrollo y no existen como una fase separada.

Durante el desarrollo de este juego no he podido probarlo sobre una nintendo DS física porque la librería que uso para acceder a los ficheros, EFS [11], tenía un bug que la hacía entrar en un bucle infinito al ejecutarse en una Nintendo DS. Así que he tenido que desarrollarlo probando las cosas en el emulador DeSmuMe.

Cuando por fin han corregido la librería EFS, he podido probar el juego en la videoconsola física y sólo he tenido que corregir dos bugs.

- Uno de ellos era que en el vector `faces_` de la clase `TextBoxHandler`, usaba el operador `[]` sin haber introducido previamente nada en el vector.
- El otro fallo era un glitch de la pantalla principal, que al cambiar de tarjeta, se imprimía por un momento el “background” donde está el texto desplazado. El fallo se producía al ajustar el desplazamiento del background en la pantalla sin haber realmente cambiado. He podido arreglarlo fácilmente añadiendo una nueva variable a la clase `MainScreenHandler` que almacena el scroll vertical actual, y sólo lo actualiza cuando se cambie de verdad.



# Capítulo 8

## Conclusiones

Este proyecto me ha gustado mucho hacerlo ya que he podido aprender sobre la Nintendo DS, que era una plataforma que desconocía por completo. Además, me ha resultado muy gratificante aprender a programar para una videoconsola, ya que es algo mucho menos general que un ordenador donde no se cuenta con un sistema operativo que te facilite las cosas.

En tiempo este proyecto he tenido que hacerlo muy rápido así que está realmente muy recortado para poder presentarlo ya.

Aun así creo que el juego ha quedado jugable por cualquier persona y con una interfaz sencilla e intuitiva que facilite la tarea del aprendizaje. Una vez presentado podré continuar añadiendo mejoras y muchas cosas que tengo pensadas y que detallo a continuación:

### 8.1. Mejoras futuras

En esta sección describiré las diversas mejoras que se pueden hacer en el sistema.

Las dividiré en tres categorías, las mejoras relativas a las funcionalidades disponibles en el juego, las mejoras relativas a la interfaz con el usuario y las mejoras relativas a las bases de datos que se usan.

#### 8.1.1. Funcionalidad

Relativas a nuevas funcionalidades que se pueden implementar en el sistema, o mejora de las que ya existen.

**Selección de bases de datos** Esta es una mejora prioritaria, que el usuario pueda, al entrar en el modo deseado, seleccionar una base de datos de las disponibles para ese modo.

Estos ficheros no deberían ir ya incluidos en el .nds sino que podrían estar en el mismo directorio del juego y luego se leería el contenido del directorio para

ofrecer al usuario la posibilidad de seleccionar uno de los ficheros adecuados para ese modo disponibles.

**Seguimiento del jugador** Podría implementarse una nueva funcionalidad que permitiese crear perfiles de jugadores en el juego, al estilo “BrainTraining”, con el que poder hacer un seguimiento de los kanjis y el vocabulario que el jugador ha aprendido, su porcentaje de acierto de un determinado tipo de kanjis o palabras, etc.

Ésto podría generar un gráfico donde se viesen los progresos del jugador, que es algo que siempre motiva.

### 8.1.2. Interfaz

Mejoras relativas a la interfaz entre el juego y el usuario.

**Internacionalización de la interfaz** Sería interesante que al iniciarse el juego por primera vez se preguntase por el idioma deseado, para que los mensajes que salgan en la pantalla saliesen en ese idioma.

**Mejora del Artwork** Para hacer el juego más bonito, sería bueno rediseñar las imágenes de fondo para que no fuesen tan planas y fuesen más atractivas para el jugador.

### 8.1.3. Bases de datos

Mejoras que se pueden hacer en cuanto a las bases de datos de palabras e ideogramas incluibles en el juego.

**Importador para diccionario EDICT** El diccionario EDICT <sup>1</sup> es un diccionario Japonés-Inglés muy grande. Se podría crear un pequeño traductor para convertir este diccionario en el formato adecuado para este juego y que pudiese usarse.

**Página web donde contribuir a la base de datos** Crear una base de datos es un trabajo realmente laborioso y largo, por lo que sería interesante crear una página web donde cualquier persona pudiese enviar una tarjeta para añadirla a la base de datos.

Las tarjetas permanecerían en un estado de moderación en una tabla de una base de datos en el servidor hasta que se confirmase su veracidad y entonces pasarían a otra tabla, que contuviese todas las entradas confirmadas.

A partir de ésta podría exportarse fácilmente a un fichero XML para el juego. De esta forma, se podría crear una base de datos realmente grande de forma colaborativa, aprovechando el tirón de páginas donde se ofrezcan cursos de japonés, como la mía <http://www.japonesporlibre.com>, se podría animar a los usuarios a que contribuyesen.

---

<sup>1</sup>Página del proyecto: <http://www.csse.monash.edu.au/~jwb/edict.html>

También podría hacerse otra página equivalente para traducir el diccionario EDICT al español. Y crear una base de datos.

## 8.2. Herramientas utilizadas

Para desarrollar este proyecto he usado el entorno de compilación devkitARM de devkitPro para la Nintendo DS. Así como las librerías libnds[8], EFS [11], UTF8-CPP[10], FreeType[2] y maxmod [9].

Para convertir las imágenes para incluirlas en el juego he usado la utilidad GRIT[6] y para los sonidos la proporcionada por la propia librería maxmod.

Para generar la documentación he usado doxygen [3]. Y como sistema de control de versiones he usado Git [1] en un repositorio de Gitorious<sup>2</sup>.

Esta memoria está escrita en  $\text{\LaTeX}$ [13][12] y para escribirla esta memoria he usado AU $\text{\TeX}$ [7].

Tanto el código fuente del juego como este documento se compilan con la utilidad “make”, una herramienta para controlar la recompilación[4].

Todo ello en Gentoo GNU/Linux.

---

<sup>2</sup>Accesible desde: <http://gitorious.org/dsmemorizer>



# Formato de los ficheros XML

DSMemorizer usa ficheros XML como bases de datos. Éstos ficheros tienen que estar codificados en UTF-8.

Tal como se veía en el diagrama de la capa de datos 5.1 los ficheros de las bases de datos se organizan en paquetes de tarjetas.

De esta forma, existen varios tipos de bases de datos. Éstos difieren en las variables que contienen las tarjetas.

Cada línea de las que se describen deberá estar exclusivamente en una línea, que no podrá ser dividida.

La primera línea en cada fichero siempre deberá ser:

```
<?xml version="1.0"?>
```

## .1. Descriptor de paquete

Cada paquete tiene varias propiedades, que deberán ir incluidas en la línea de información del paquete:

```
<package name="" records="" format="" license=""/>
```

## .2. Bases de datos para kanjis

Para las bases de datos de kanjis la variable formato del descriptor de paquete deberá ser

```
DSMemorizer-kanji-1.0.
```

Las líneas para las tarjetas de bases de datos de kanjis tienen el siguiente formato:

```
<card symbol="" reading="" reading2="" translation=""  
example_symbol="" example_reading="" example_translation=""  
grade="" strokes=""/>
```

```

<?xml version="1.0"?>
<package name="Kanjis" records="5" format="DSMemorizer-kanji-1.0"
  license="Creative Commons Reconocimiento-No comercial-Compartir bajo
  la misma licencia 3.0 Unported"/>
<card symbol="山" reading="サン、セン" reading2="やま"
  translation="Montaña" example_symbol="富士山" example_reading="ふじ
  さん" example_translation="Monte Fuji" grade="1" strokes="3"/>
<card symbol="日" reading="ジツ、ニチ" reading2="-か、 ひ、 -び "
  translation="Día, Japón, rayo de sol, sol, solar" example_symbol="
  あの日は月曜日でした" example_reading="あのひはげつようびでした"
  example_translation="Aquel día fue lunes" grade="1" strokes="4"/>
<card symbol="木" reading="ボク、モク" reading2="き、 こ-"
  translation="Árbol, madera" example_symbol="この木はとても大きいです"
  example_reading="このきはとてもおおきいです" example_translation="Este
  árbol es muy grande" grade="1" strokes="4"/>
<card symbol="人" reading="ジン、ニン" reading2="-と、 ひと、 -り"
  translation="Persona" example_symbol="この人は田中さんです"
  example_reading="このひとはたなかさんです" example_translation="Esta
  persona es el señor Tanaka" grade="1" strokes="2"/>
<card symbol="火" reading="カ" reading2="ひ、 -び、 ほ-"
  translation="Flama, fuego, llama" example_symbol="火は暑いです"
  example_reading="ひはあついです" example_translation="El fuego está
  caliente" grade="1" strokes="4"/>
</package>

```

### .3. Bases de datos para vocabulario

Para las bases de datos de kanjis la variable “formato” del descriptor de paquete deberá ser

DSMemorizer-vocabulary-1.0.

```
<card symbol="" reading="" translation="" />
```

```

<?xml version="1.0"?>
<package name="Vocabulario" records="5"
  format="DSMemorizer-vocabulary-1.0" license="Creative Commons
  Reconocimiento-No comercial-Compartir bajo la misma licencia 3.0
  Unported"/>
<card symbol="火山" reading="かざん" translation="Volcán" />
<card symbol="今日" reading="きょう" translation="Hoy" />
<card symbol="明日" reading="あした" translation="Mañana" />
<card symbol="日本人" reading="にほんじん" translation="Japonés
  (persona)" />
<card symbol="出口" reading="でぐち" translation="Salida" />
</package>

```

# Bibliografía

- [1] Comunidad Git. Git Community Book. <http://book.git-scm.com/>.
- [2] David Turner, Robert Wilhelm, and Werner Lemberg. Página oficial de The FreeType Project. <http://www.freetype.org>.
- [3] Dimitri Van Heesch. Pagina oficial de Doxygen. <http://www.doxygen.org>.
- [4] Gerardo Aburruzaga García. Make. Un programa para controlar la recompilación. <http://www.uca.es/softwarelibre/publicaciones/make.pdf>.
- [5] Albert Torres I Graell. *Kanji, La escritura japonesa*. Hiperión, 1995.
- [6] Jasper Vijn. GRIT: GBA Raster Image Transmogriber. <http://www.coranac.com/projects/grit/>.
- [7] Joaquín Ataz López. Creación de ficheros  $\text{\LaTeX}$  con GNU Emacs. <ftp://ftp.dante.de/tex-archive/info/spanish/guia-atx/guia-atx.pdf>.
- [8] Michael Noland (joat) y Jason Rogers (dovoto). Documentación de la librería libnds. <http://libnds.devkitpro.org>.
- [9] Mukunda Johnson. Maxmod Nintendo DS and Gameboy Advance Sound System. <http://www.maxmod.org>.
- [10] Nemanja Trifunovic. Página oficial de la librería UTF8-CPP. <http://utfcpp.sourceforge.net>.
- [11] Noda (initially based on Alekmaul's libNitro). Página oficial de la librería EFS. <http://nodadev.wordpress.com/nds-projects/efs-library/>.
- [12] Wikibooks. The Book of LaTeX. <http://en.wikibooks.org/wiki/LaTeX>.
- [13] Frank Mittelbach y Michel Goossens. *The LaTeX Companion*. Addison-Wesley, 2004.



# GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

## Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

## 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed

as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page”

means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The “**publisher**” means any person or entity that distributes copies of the Document to the public.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve the Title**” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you

may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## **5. COMBINING DOCUMENTS**

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

## **6. COLLECTIONS OF DOCUMENTS**

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## **7. AGGREGATION WITH INDEPENDENT WORKS**

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the

Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## **8. TRANSLATION**

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## **9. TERMINATION**

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

## 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

## 11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

## **ADDENDUM: How to use this License for your documents**

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with . . . Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.