ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

VIDEOJUEGO PARA LA ESTIMULACIÓN COGNITIVA Y CEREBRAL OPENBTRAINER

José Luis Falcón Ramírez

14 de septiembre de 2009

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO TÉCNICO EN INFORMÁTICA DE SISTEMAS

VIDEOJUEGO PARA LA ESTIMULACIÓN COGNITIVA Y CEREBRAL OPENBTRAINER

- Departamento: Lenguajes y sistemas informáticos
- Director del proyecto: Manuel Palomo Duarte, Alejandro Álvarez Ayllón
- Autor del proyecto: José Luis Falcón Ramírez

Cádiz, 14 de septiembre de 2009

Fdo: Jose Luis Falcón Ramírez

Agradecimientos

Me gustaría agradecer a Ana, a mi madre y a mi familia el apoyo que he recibido de ellos. Muchas gracias a todos los que me habéis ayudado para probar el juego y para conseguir los tiempos para hacer que este proyecto salga adelante y no se haya quedado en una simple idea.

Licencia

Convenciones, notación y formato

Para facilitar la lectura y comprensión del documento se citan las principales convenciones, notaciones y formatos utilizados durante la escritura de dicha memoria:

- Los comandos que han de usarse en la consola de cualquier sistema GNU/Linux tendrán el formato:

```
Esto es una prueba de comandos en una consola
```

- Los fragmentos de código o de ficheros de texto plano que puedan aparecer a lo largo de esta memoria cumplirá el formato:

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4    cout<<"Esto es una prueba de programa simple creado en C++"<<endl;
5    return 0;
6  }
```

- Cuando hablemos de las categorías o juegos que componen OpenBTrainer cumplirá: **Lógica**, **Luces y Sombras**.
- Cuando hablemos de programas y herramientas usadas durante el desarrollo del proyecto usaremos el formato: *GIMP*, *Planner*.

Índice general

Indice de figuras

Indice de tablas

Capítulo 1

Introducción

1.1. Objetivos

Se pretende crear un videojuego basado en el videojuego libre Gbrainy creado por Jordi Mas [1].
Se desea crear un videojuego intuitivo, fácil de usar y con una interfaz agradable para que resulte atractivo para el jugador. Un videojuego con pruebas simples que consigan hacer que el usuario note que va progresando a medida que se van realizando los ejercicios diariamente, para ello ayudará el sistema de control de perfiles que dará al usuario la posibilidad de ver claramente la progresión que va teniendo.

Se intenta así potenciar un mercado de juegos basados en software libre totalmente escalables, es decir, que cualquier persona con unos conocimientos informáticos y de programación, pueda aportar sus propios juegos para que sean incorporados a OpenBTRainer y así crear multitud de pruebas para que el usuario final actualice su producto y consiga así nuevas experiencias.

1.2. Composición del documento del Proyecto Fin de Carrera

El documento se divide en nueve apartados, el primero de ellos es en el que nos encontramos.

En el segundo punto nos encargaremos de comentar los conceptos básicos del proyecto, así como hablar del tema del que trata el videojuego y las tecnologías que hemos usado.

En el tercer apartado explicaremos las tareas realizadas y el tiempo que se ha dedicado a cada una de ellas, ya sean tareas gráficas como tareas de implementación de código. Todo ello se presentará mediante un Diagrama de Gannt[1] usando para ello la herramienta libre *Planner* incluida en los repositorios de cualquier distribución GNU/Linux.

En el cuarto apartado comentaremos la fase de análisis que hemos tenido que realizar para finalizar el producto, incluiremos algunos diagramas entidad-relación más conocidos como ERs.

En el quinto apartado hablaremos sobre la fase de diseño del juego, es decir, definir el sistema con total detalle.

En el sexto apartado comentaremos la implementación del juego, las técnicas y herramientas usadas, así como las diferentes estrategias que hemos seguido para realizar el proyecto.

[1] El Diagrama de Gannt es una popular herramienta cuyo objetivo es mostrar el tiempo de dedicacación previsto para diferentes tareas o actividades a lo largo de un tiempo determinado.

En el séptimo apartado comentaremos las diferentes pruebas a las que ha sido sometido el producto, comentaremos los resultados obtenidos por una serie de individuos en cada juego y lo veremos todo más claramente en forma de estadística y con diagramas.

En el octavo, hablaremos de las conclusiones a las que hemos llegado después de la finalización del proyecto, lo que se ha aprendido, posibles extensiones del proyecto...

En el último apartado incluiremos distintos apéndices tales como el manual de usuario, manual de instalación, manual del desarrollador, manual de ampliación de Juego Diferencias...
Al ser un producto de Software Libre incluimos la licencia GFDL 1.3 en la que se basa la documentación del proyecto.

Capítulo 2

Conceptos Básicos

2.1. Visión Global del Producto

El producto que presentamos responde a la demanda de crear un juego de desarrollo cognitivo bajo una licencia GPL y capaz de expandirse e ir insertándose nuevos juegos a medida de que avance el proyecto.

Como comentamos anteriormente existe un proyecto parecido y también distribuido bajo licencia GPL e integrado en el proyecto GNU, pero en mi opinión no cumple con el segundo punto que hemos comentado en el párrafo anterior, la escalabilidad del proyecto. OpenBTrainer está pensado para la creación de una comunidad de desarrolladores en la cuál se creen nuevos juegos y se vayan incorporando al producto, para así ir creando un videojuego atractivo y en constante expansión. Uno de los grandes problemas de los juegos privativos, como puede ser el caso de cualquier juego similar a Brain Training es que una vez que hayamos desbloqueado todos los juegos que nos incluye el producto, el jugador tiende a perder la atracción y el interés de seguir jugando. Posiblemente éste sea el mayor problema que tiene cualquier juego en la actualidad que posea el carácter privativo.

Imaginemos un juego libre y gratuito en constante expansión, en el que cada mes disponga de dos o tres pruebas distintas que hará que los jugadores vayan actualizando las versiones de su juego y sigan jugando sin parar a OpenBTrainer. Todo esto es una idea bastante interesante si llegásemos a encontrar un grupo de personas comprometidas con el proyecto y que aportaran nuevas pruebas a OpenBTrainer.

En la idea anteriormente descrita está la originalidad del proyecto OpenBTrainer, en crear juegos libres en constante expansión, sería una opción que conseguiría dar bastante protagonismo a los juegos bajo licencia libre en contra de los juegos privativos.

2.2. Antecedentes

La idea de este tipo de juegos no es nueva, a continuación vamos a repasar cómo apareció el tipo de juego de desarrollo cognitivo.

Brain Training del Dr. Kawashima ¿Cuántos Años Tiene tu Cerebro?, conocido como **Brain Training** y conocido en Estado Unidos como **Brain Age** fue creado por el Dr. Kawashima[1] y distribuido por Nintendo para Nintendo DS. Llegó al mercado el 19 de mayo de 2005 en Japón y tuvimos que esperar

[1] Ryuta Kawashima nacido el 23 de mayo de 1959 es un neurocientífico japonés. En 2003 escribió un libro conocido como *Train Your Brain: 60 Days to a Better Brain* que fue un gran éxito en Japón y luego fue distribuido por todo el mundo vendiendo más de 2.5 millones de copias. En 2005 volvió con Brain Training teniendo un gran éxito en todo el mundo.

hasta el 9 de junio de 2006 para verlo en Europa [2].

La aparición de este tipo de juego marcó un antes y un después en el mundo de los videojuegos. Existe en la actualidad una gran lucha por atraer al público joven y a los llamados gamers[2]. Este tipo de colectivo se caracteriza por estar comprendido en edades entre los 8 y los 25 años aproximadamente.

Nintendo supo ver todo el público que no entraba en este grupo y que con el juego desarrollado por el Dr. Kawashima existía la posibilidad de atraer a este colectivo. El juego fue todo un éxito de ventas con más de 30 millones de copias vendidas y actualmente es uno de los juegos que goza de mayor popularidad. Gracias a una gran campaña de publicidad y marketing. Personas que nunca habían sido amantes de las consolas se lanzaron a comprar este videojuego. La campaña se basaba en transmitirnos una idea, que a las personas nos preocupa mucho, como es la pérdida de la memoria y la pérdida de destreza mental.

Enfermedades modernas como el Alzheimer[3] produjeron continuos consejos por partes de médicos en diferentes medios de comunicación para que personas mayores ejerciten la memoria y la mente. Nintendo vio el momento perfecto para dedicarse a ello mediante un juego muy simple y que cualquier persona puede familiarizarse con la interfaz y con los distintos tipos de juego, gracias a unas explicaciones muy claras sobre lo que hay que hacer en los distintas situaciones existentes dentro de **Brain Training**.

Posteriormente han aparecido multitud de clónicos del juego del Dr. Kawashima, como puede ser **Big Brain Accademy**, **BrainStorm**, **Brain Boost**, **Mind Quiz**,...

El día 29 de Junio de 2007 aterrizó en España la segunda parte de Brain Training conocida con el nombre de **Más Brain Training**, en esta ocasión nos trae nuevas pruebas y retos, aunque se echó en falta algún cambio significativo ya que hay pocas diferencias con su famoso predecesor.

2.3. Alcance

Se intenta que el videojuego tenga una dificultad menor que Gbrainy y que consiga hacer que los usuarios jueguen diariamente para progresar y entrenarse ellos mismos poco a poco, con una interfaz agradable. La escalabilidad será mayor que Gbrainy ya que éste utiliza Mono[4] como lenguaje de programación y Cairo[5] como librería gráfica y que gozan de poca popularidad, esto hace que el código de **OpenBTrainer** realizado en C/C++ y libSDL sea más familiar para cualquier informático con conocimientos del lenguaje y de la librería anteriormente descrita.

[2]Un gamer es el término usado en el idioma español para definir al tipo de videojugadores que se caracterizan por jugar con gran dedicación e interés.

[3]El Alzheimer es una enfermedad neurodegenerativa que se manifiesta como deterioro cognitivo y trastornos conductales.

[4]Mono es el nombre de un proyecto de código abierto impulsado por Novell y con características como una máquina virtual de lenguaje y un compilador en tiempo de ejecución (JIT).

[5]Cairo es una librería gráfica basada en gráficos vectoriales y programación independiente de dispositivo. Está diseñada para trabajar con gráficos primitivos en dos dimensiones.

Capítulo 3

Planificación

3.1. Incrementos

En este proyecto debido a su naturaleza, hemos decidido que lo mejor será un **modelo incremental**. Esto es debido a que cada juego es independiente al otro, por tanto podremos conseguir una versión cada vez más completa hasta llegar a la versión final en la que se concentrarán todos los juegos planificados.

Como sabemos el primer incremento sólo se consideran los requisitos básicos del sistema, en nuestro caso la especificación inicial de requisitos y las imágenes, botones y código de los menús. Una vez realizado el primer incremento lo probaremos y a partir de él se planifican las modificaciones a realizar en la siguiente iteración y así progresivamente.

A continuación mostraremos los principales incrementos realizados en el proyecto **OpenBTrainer**.

3.1.1. Incremento 1: Requisitos básicos del sistema

En este incremento hemos planificado el tiempo que disponemos para la realización del proyecto y las distintas tareas que deberíamos realizar en cada momento. El tiempo inicial de trabajo será de 10 de la mañana a 3 de la tarde. Por tanto se intentará realizar cada día una tarea distinta, para hacer más liviano el trabajo de realizar un proyecto largo. Comenzamos con la especificación de lo que va a realizar globalmente el juego, es decir, lo que queremos conseguir con el juego sin pensar en los distintos juegos y puzzles de los que se compone el producto. Gracias al programa *GIMP*[1] realizamos el icono del juego, los fondos de los distintos menús y los botones.

Figura 3.1: Imagen del Logo de OpenBTrainer

[1] *GIMP* (GNU Image Manipulation Program) es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. Es un programa libre y gratuito. Está englobado en el proyecto GNU y disponible bajo la Licencia pública general de GNU (GPL)

Me encuentro en la circunstancia que no conocía LaTeXasí que gracias a que asistí al curso de LaTeXpara proyectos fin de carrera que impartió un compañero y organizado por la OSLUCA, disponía de las plantillas y con varios libros y tutoriales me lo tuve que preparar por mi cuenta. Así desarrollé la primera versión de la memoria del proyecto.

3.1.2. Incremento 2: Implementación de memorión

En este incremento, nos centramos a desarrollar un juego bastante famoso, el juego de las parejas, en el que el jugador dispone de varias cartas volteadas y deberá ir de dos en dos buscando las parejas. Primeramente hemos tenido que coseguir las imágenes de todas las cartas, para ello hemos usado un juego libre como es PokerTH [3], también hemos adquirido la imagen de la mesa donde se situarán las cartas.

Figura 3.2: Juego Memorión

En la fase de pruebas, después de arreglar algún bug, permitimos que algún usuario pruebe nuestro juego. Lo que más comentan al final del juego es la dificultad que encuentran en diferenciar entre picas y tréboles. Por tanto nuestro objetivo se ha cumplido, queremos conseguir que el jugador recuerde tanto el número de la carta como el palo de la misma.

Una vez que disponemos de todos los materiales y tenemos claro las normas del juego nos centramos en implementar el código.

3.1.3. Incremento 3: Implementación del juego de las diferencias

En esta ocasión queremos crear un juego de diferencias como el que encontramos en la mayoría de los períodicos y demás. Por tanto lo que necesitamos principalmente son las imágenes duplicadas con sus fallos. En esta ocasión encontramos el material en una página de educación en el que profesores incluían multitud de imágenes para que otros profesores lo usaran, por supuesto todo estaba bajo licencia Creative Commons [4].

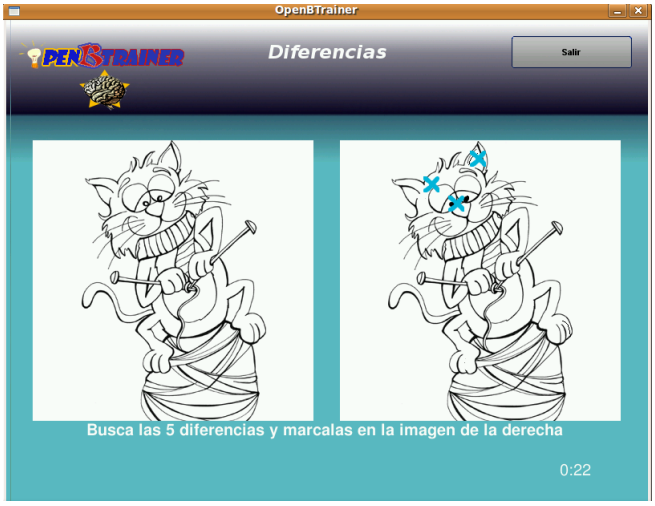

Figura 3.3: Juego Diferencias

En la fase de pruebas vamos probando los distintas imágenes que disponemos y nos encontramos la dificultad principal del juego, es difícil acertar con la región del fallo que hemos escrito en el fichero de texto, por tanto tenemos que ir una a una refinando las coordenadas e incrementando, en caso de ser necesario, su cuadrado de captación.

Una vez que tenemos todas las imágenes necesitamos encontrar las diferencias e introducir las coordenadas en un fichero de texto. Luego realizaremos el código del juego.

3.1.4. Incremento 4: Implementación del juego de Recordar Dígitos

Una vez realizados los dos minijuegos anteriores, nos adentramos en los juegos que definen realmente al juego OpenBTrainer, juegos de desarrollo cognitivo. En esta ocasión hemos elegido un juego de memoria a corto plazo. Se trata de un juego de fácil implementación y que sólo necesita la imagen del fondo que la hemos desarrollado en *GIMP*.

Figura 3.4: Juego Recuerda Dígitos

Luego realizamos las pruebas pertinentes con usuarios reales y vemos que el nivel de dificultad del juego va incrementando exponencialmente a medida de que aumenta el número de dígitos. Encontramos algunos bugs que son arreglados rápidamente.

También necesitamos en este incremento modificar los menús para incluir el juego en el apartado de juegos temáticos.

3.1.5. Incremento 5: Implementación de juego Observación

En esta ocasión nos dedicamos a realizar el juego de **Observación**, un juego que hemos decidido incluirlo en el apartado de rapidez visual, aunque también se podría considerar un juego de memoria, pero considero que lo importante es abrir bien los ojos para ver el mayor número de flechas en el menor tiempo posible.

En primer lugar realizamos los fondos y las flechas usando *GIMP*. Luego nos dedicamos a la implementación del juego, el cual no nos ha dedicado demasiado tiempo.

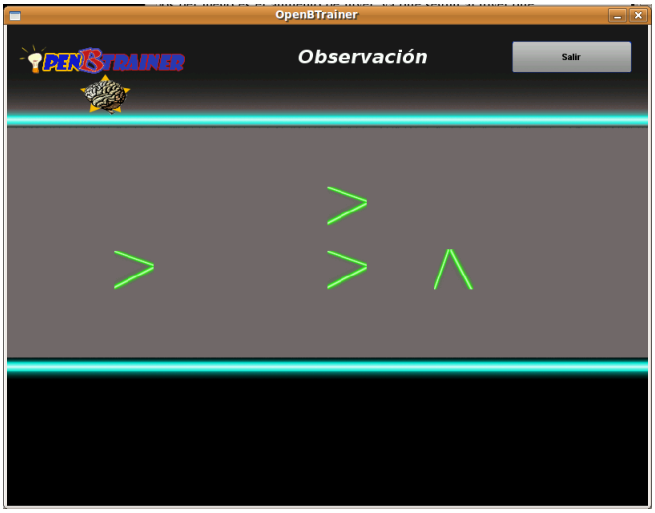

Figura 3.5: Juego Observación

Una vez que hemos terminado la fase de implementación del juego, realizamos las pruebas pertinentes con distintos usuarios.

3.1.6. Incremento 6: Implentación de juego Ojo de Cerradura y remodelación de menús

En primer lugar debemos de modificar los menús del apartado de juegos temáticos, ya que en el momento que haya bastantes juegos puede parecer algo difícil de navegar por los distintos juegos. Decidimos realizar la remodelación de los menús mediante botones que se despliegan al pinchar en las distintas categorías que componen el proyecto.

Una vez realizado los cambios en los menús nos adentramos en realizar uno de los juegos más originales del juego, lo he llamado **Ojo de Cerradura**. La realización de dicho juego supone sobre todo un gran trabajo de imágenes con *GIMP*, ya que necesitamos transparencias que nos la facilitan el uso de imágenes PNG[2].

[2]PNG (Portable Network Graphics) es un formato gráfico basado en un algoritmo de compresión sin pérdida para bitmaps no sujeto a patentes. Este formato fue desarrollado en buena parte para solventar las deficiencias del formato GIF y permite almacenar imágenes con una mayor profundidad de contraste y otros importantes datos.

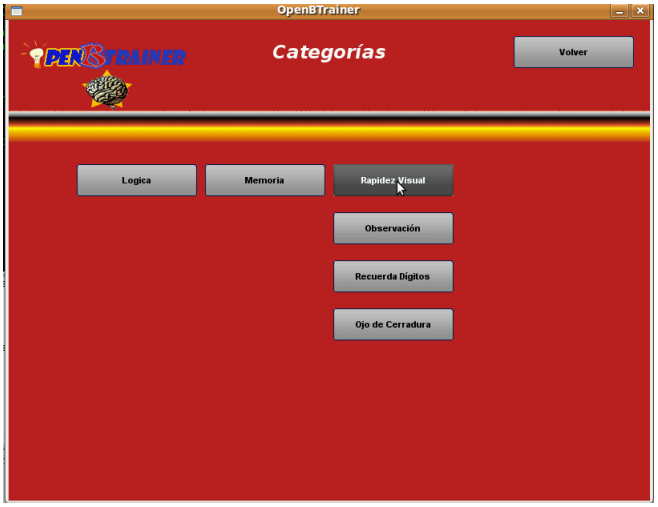

Figura 3.6: Remodelación de Menús

Uno de los grandes problemas que nos encontramos al realizar este juego es el tema de la eficiencia, hay que tener bastante cuidado al realizar el refresco de la imagen de fondo, ya que podemos hacer que el ordenador se sobrecargue si realizamos muchos limpiados de pantalla en un segundo. Una vez solventados estos problemas el juego toma forma y queda completado este incremento.

Figura 3.7: Juego Ojo de Cerradura

Por último realizamos las pruebas con usuarios que no han mantenido relación con el desarrollo del juego. Una de las cosas curiosas que se pueden observar es que resulta un poco complejo el juego y se necesita jugar varias veces hasta que el jugador avance varios niveles.

3.1.7. Incremento 7: Implementación de juego Acuerdate de mí

Para la realización de este juego debemos buscar imágenes en Internet, y nos decantamos por las imágenes de sprites de personajes de videojuegos, dichas imágenes son sacadas de Molotov [5], una página de recursos de sprites libres para que cualquiera que quiera pueda usar cualquiera de los personajes para su propio juego.

Una vez que hemos seleccionamos las imágenes que queremos las recortamos usando *GIMP*, las clasificamos en una carpeta que usaremos para cargarlas en el juego. Implementamos el código del juego y realizamos las pruebas.

Figura 3.8: Juego Acuerdate de Mí

Cabe destacar que para la realización de este juego hemos tenido que usar un reloj para controlar el tiempo que queda para que desaparezca el personaje que debemos recordar. Por tanto creamos una clase que controlará y dibujará un reloj - cronómetro en la posición que se le indique. Dicho reloj también es adaptado a los juegos que necesiten de tiempo para controlar las puntuaciones, como es el caso del juego del **Memorión** y el juego de las **Diferencias**.

3.1.8. Incremento 8: Implementación del control de perfiles

Este incremento es uno de los más importantes del juego, ya que nos da la posibilidad de controlar los mejores registros de los jugadores y llevar un control de los progresos de los usuarios.

Es una de las tareas más complejas. En primer lugar debemos realizar, en una clase, la lectura y escritura de los ficheros **.sav*, donde almacenaremos los datos de los jugadores. Más tarde debemos realizar funciones para almacenar los nuevos registros que van realizando los jugadores. Luego deberemos realizar una pantalla para seleccionar o crear un nuevo perfil. El mapeo del teclado para introducir el nombre del usuario. Por último realizamos una pantalla para indicar al usuario que ha batido el récord personal o no, lo que le da la posibilidad al usuario de comprobar como a medida de que juega va a ir progresando en sus resultados y le irá dando gran motivación para que continúe jugando a diario.

Figura 3.9: Pantalla de Actualización de Perfil

3.1.9. Incremento 9: Implementación de Luces y Sombras, Aritmética, Operadores y Arcade

Este incremento supone el último gran esfuerzo de código, ya que con ello conseguimos incluir todos los juegos necesarios para cerrar el proyecto.

La primera tarea es la de la implementación del primer juego de **Lógica**, se trata de **Luces y Sombras**. Necesitamos las imágenes de fondo y las luces encendidas y apagadas, todas ellas fueron realizadas con *GIMP*, luego realizamos el código del juego que supuso un par de días de trabajo.

Figura 3.10: Juego Luces y Sombras

La segunda tarea fue la de la implementación de los juegos de **Cálculo** llamados **Aritmética** y **Operadores**. Estos juegos son incluidos en una misma tarea debido a la similitud que existe con respecto a código de los dos juegos. Para ambos juegos realizamos un escaneado de un pequeño cuaderno y lo

adaptamos para que diera la impresión de que estuviésemos escribiendo en un cuaderno de anillas.

Figura 3.11: Juego Aritmética

La tercera tarea fue la de implementar el modo de juego **Arcade**, esta tarea fue tan simple como hacer la llamada a los cuatro juegos que componen dicho modo de juego y crear unos criterios de puntuación.

Las pruebas realizadas, como en incrementos anteriores, se realizaron con éxito sobre usuarios externos al desarrollo del proyecto.

3.2. Diagrama de Gantt

A continuación se muestra el diagrama de Gantt donde se comprueba la planificación de cada una de las iteraciones con sus respectivas tareas.

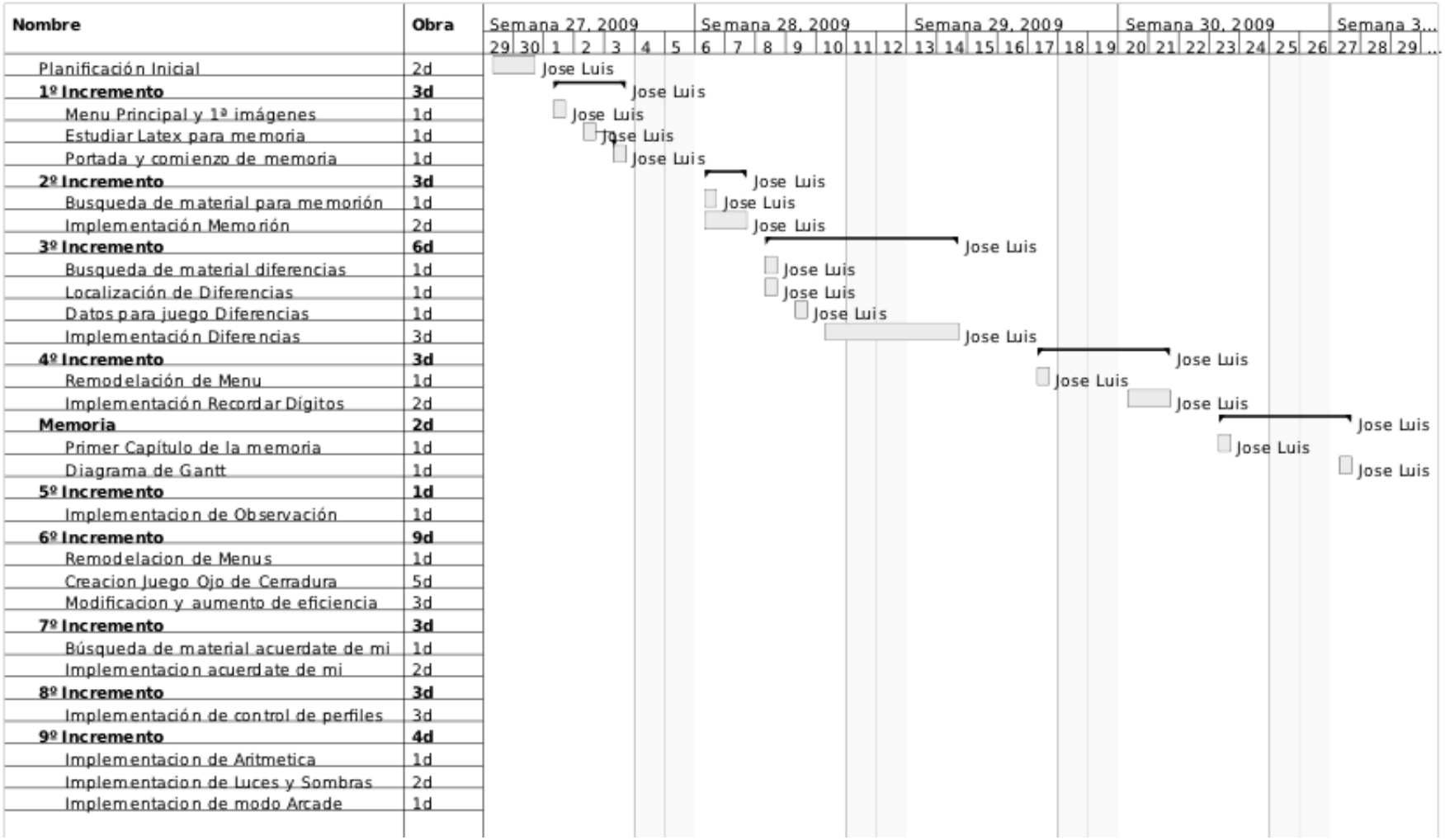

Figura 3.12: Primera parte del Diagrama de Gantt

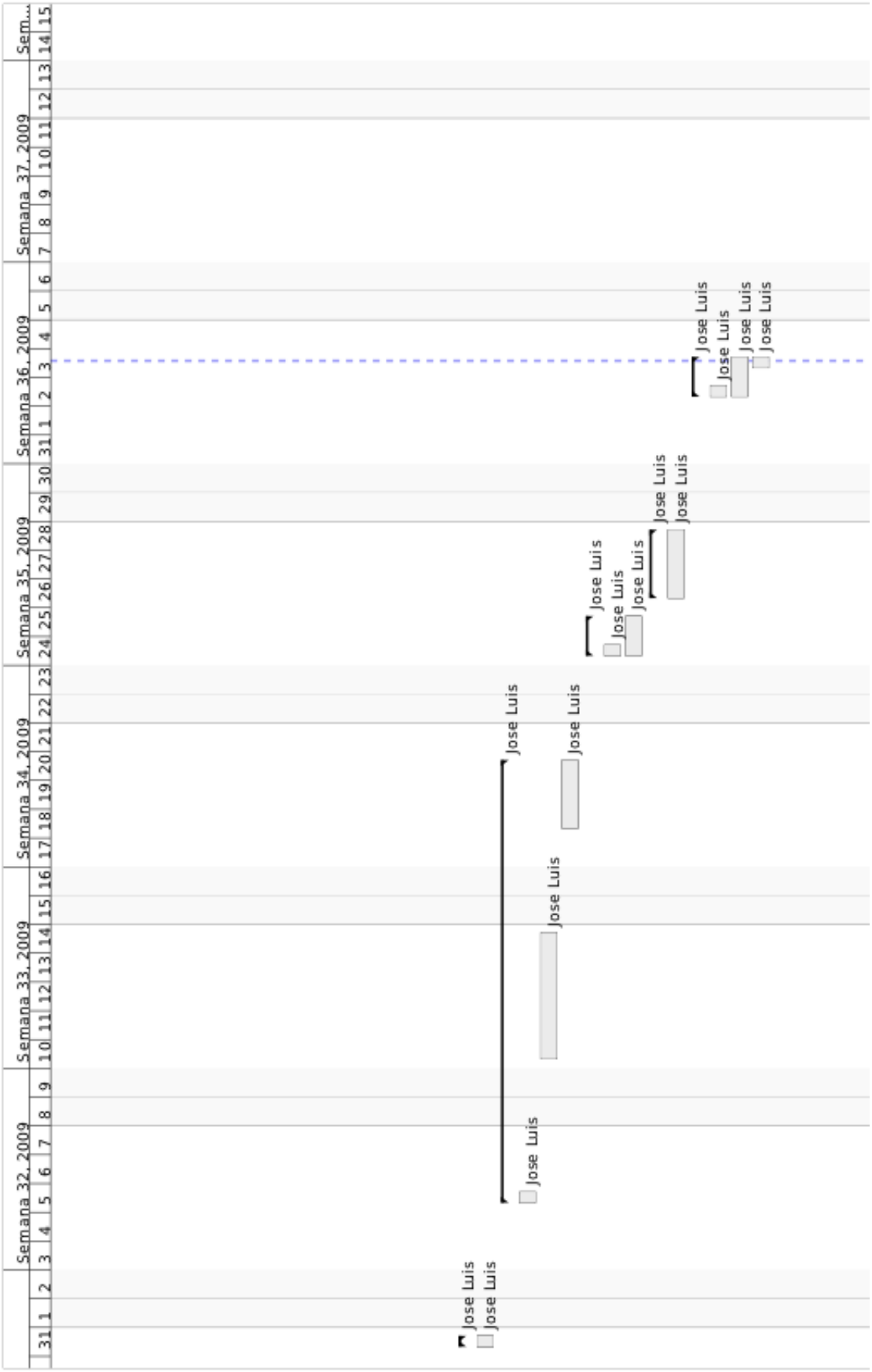

Figura 3.13: Segunda parte del Diagrama de Gantt

WBS	Nombre	Inicio	Fin	Obra	Duración	Asignado a
1	Planificación Inicial	jun 29	jun 30	2d	2d	Jose Luis
2	**1º Incremento**	**jul 1**	**jul 3**	**3d**	**3d**	Jose Luis
2.1	Menu Principal y 1ª imágenes	jul 1	jul 1	1d	1d	Jose Luis
2.2	Estudiar Latex para memoria	jul 2	jul 2	1d	1d	Jose Luis
2.3	Portada y comienzo de memoria	jul 3	jul 3	1d	1d	Jose Luis
3	**2º Incremento**	**jul 6**	**jul 7**	**3d**	**2d**	Jose Luis
3.1	Busqueda de material para memorión	jul 6	jul 6	1d	1d	Jose Luis
3.2	Implementación Memorión	jul 6	jul 7	2d	2d	Jose Luis
4	**3º Incremento**	**jul 8**	**jul 14**	**6d**	**5d**	Jose Luis
4.1	Busqueda de material diferencias	jul 8	jul 8	1d	1d	Jose Luis
4.2	Localización de Diferencias	jul 8	jul 8	1d	1d	Jose Luis
4.3	Datos para juego Diferencias	jul 9	jul 9	1d	1d	Jose Luis
4.4	Implementación Diferencias	jul 10	jul 14	3d	3d	Jose Luis
5	**4º Incremento**	**jul 17**	**jul 21**	**3d**	**3d**	Jose Luis
5.1	Remodelación de Menu	jul 17	jul 17	1d	1d	Jose Luis
5.2	Implementación Recordar Dígitos	jul 20	jul 21	2d	2d	Jose Luis
6	**Memoria**	**jul 23**	**jul 27**	**2d**	**3d**	Jose Luis
6.1	Primer Capítulo de la memoria	jul 23	jul 23	1d	1d	Jose Luis
6.2	Diagrama de Gantt	jul 27	jul 27	1d	1d	Jose Luis
7	**5º Incremento**	**jul 31**	**jul 31**	**1d**	**1d**	Jose Luis
7.1	Implementacion de Observación	jul 31	jul 31	1d	1d	Jose Luis
8	**6º Incremento**	**ago 5**	**ago 20**	**9d**	**12d**	Jose Luis
8.1	Remodelacion de Menus	ago 5	ago 5	1d	1d	Jose Luis
8.2	Creacion Juego Ojo de Cerradura	ago 10	ago 14	5d	5d	Jose Luis
8.3	Modificacion y aumento de eficiencia	ago 18	ago 20	3d	3d	Jose Luis
9	**7º Incremento**	**ago 24**	**ago 25**	**3d**	**2d**	Jose Luis
9.1	Búsqueda de material acuerdate de mi	ago 24	ago 24	1d	1d	Jose Luis
9.2	Implementacion acuerdate de mi	ago 24	ago 25	2d	2d	Jose Luis
10	**8º Incremento**	**ago 26**	**ago 28**	**3d**	**3d**	Jose Luis
10.1	Implementación de control de perfiles	ago 26	ago 28	3d	3d	Jose Luis
11	**9º Incremento**	**sep 2**	**sep 3**	**4d**	**2d**	Jose Luis
11.1	Implementacion de Aritmetica	sep 2	sep 2	1d	1d	Jose Luis
11.2	Implementacion de Luces y Sombras	sep 2	sep 3	2d	2d	Jose Luis
11.3	Implementacion de modo Arcade	sep 3	sep 3	1d	1d	Jose Luis

Tabla 3.1: Tabla de tareas incluidas en el diagrama de Gantt

3.3. Esfuerzo

Hay que destacar que la mayor parte del tiempo se ha dedicado a pensar, analizar e implementar nuevos y diferentes juegos que hagan que OpenBTrainer sea un juego interesante y serio. Otra de las tareas que me causó bastante esfuerzo fue la de implementar el control de perfiles, ya que hay mucho movimiento de ficheros y demás. Otra tarea que cabe destacar por su esfuerzo fue la de hacer los juegos bastante eficientes para que OpenBTrainer pudiese ejecutarse desde cualquier equipo que posea los requerimientos software que se piden y poco más.

Capítulo 4

Análisis

4.1. Metodología de Desarrollo

En este proyecto se ha seguido una metodología de desarrollo ágil de software, que se basa en procesos ágiles, es decir, se intenta evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados.

Esta metodología promueve iteraciones en el desarrollo a lo largo de la vida del proyecto, minimizamos los riesgos desarrollando software en cortos períodos de tiempo, estos períodos de tiempo se llaman iteraciones, la cual puede durar de una a cuatro semanas. Cada iteración debe de contener: planificación, análisis de requerimientos, diseño, codificación, revisión y documentación. Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del producto al mercado, pero la meta es tener una versión de prueba (demo) al final de cada iteración. Cada vez que se finaliza una iteración el proyecto debe pasar a manos de los revisores, que en mi caso puede ser cualquier persona ajena al desarrollo del proyecto y lo evalúa, comentan las cosas que más le ha llamado la atención y aconsejan cambios en cualquiera de las funcionalidades que se han implantado en esta nueva iteración.

Estos métodos ágiles enfatizan las comunicaciones cara a cara en vez de la documentación. Lo cual es más rápido y consigue un producto más cercano al público y en menor tiempo.

Como hemos ido viendo el proyecto comenzó con sólo el aspecto visual y a medida de que fueron avanzando las iteraciones o incrementos se fueron añadiendo nuevas funcionalidades y pruebas, hasta conseguir un producto final con varias pruebas y muchas funcionalidades. Siempre teniendo en cuenta las opiniones de las distintas personas que iban adquiriendo el papel de revisores, que al no disponer de un equipo de trabajo puede ser cualquier persona de la familia que conociera el proyecto y que no le importara perder tiempo revisando a fondo el trabajo que iba realizando semana tras semana.

4.2. Análisis del sistema

El objetivo del análisis de sistemas es realizar un **Documento de Especificación de Requisitos**: especificar qué tiene que hacer el sistema y no cómo desarrollarlo.

Para el análisis de nuestro sistema vamos a usar un método de análisis orientado a objetos.

Para desarrollar las ideas para nuestro proyecto hemos usado la técnica llamada **Brainstorming** (tormenta de ideas) en la cuál me reunía yo con algún miembro de mi familia o algún amigo y en el que le comentaba la categoría del juego, cada uno aportaba una idea sin juzgar su validez y generando ideas

en un ambiente libre de críticas y juicios en el que yo era el jefe de sesión y valoraba las ideas más interesantes. Una vez que tenía las ideas más interesantes, dedicaba un tiempo a valorar las ideas más originales y que se adaptaban más a la naturaleza del proyecto y a sus características.

4.3. Diagrama de Casos de Uso

A continuación mostraremos los dos casos de uso más importantes con sus descripciones.

4.3.1. Caso de Uso: Cargar Perfil

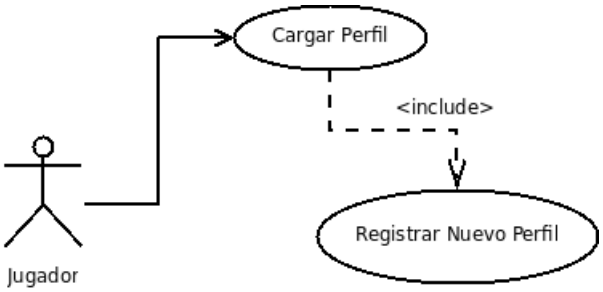

Figura 4.1: Diagrama de Caso de Uso Cargar Perfil

Descripción del Caso de Uso: *Cargar Perfil*

Descripción: Carga un perfil de un jugador.
Actor: Jugador.
Resumen: El jugador elige un perfil para cargar.

Escenario principal

1. El *Sistema* carga la pantalla de elección de perfiles con todos los perfiles existentes.
2. El *Jugador* elige una de las posiciones para cargar el perfil asociado a dicha posición.
3. El *Sistema* comprueba que la posición no está vacía y carga los datos del perfil elegido por el jugador y carga la pantalla de bienvenida.

Escenario alternativo

3a. El usuario elige una posición que está vacía.

3a1. Include *Registrar Nuevo Perfil*.

Descripción del Caso de Uso: *Registrar Nuevo Perfil*

Descripción: Registra los datos de un nuevo perfil.
Actor: Jugador.

Resumen: El sistema registra los datos de un jugador nuevo y le asigna una posición dentro de la pantalla de cargar perfil.

Escenario principal

1. El *Sistema* carga la pantalla de introducción de datos
2. El *Jugador* introduce los datos que le piden mediante el teclado.
3. El *Sistema* comprueba que los datos son correctos.

Escenario alternativo

3a. El usuario ha introducido un nombre de más de 20 caracteres.

3a1. El *Sistema* no permite que se siga escribiendo e indica el error.

3b. El usuario ha introducido un nombre vacío.

3b1. El *Sistema* indica el error y aborta el proceso de registro.

3c. El usuario ha introducido un espacio en blanco.

3c1. El *Sistema* no permite que se escriba el espacio en blanco y permite que el usuario siga escribiendo.

4.3.2. Caso de Uso: Jugar a un juego

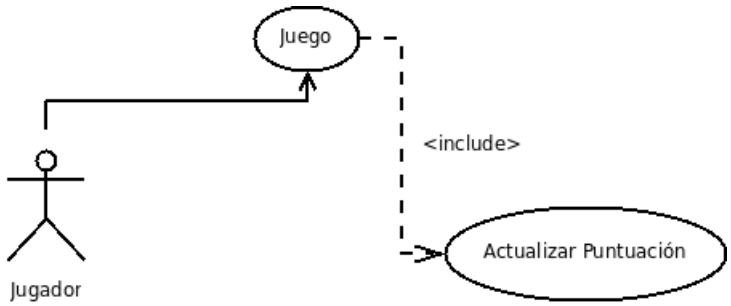

Figura 4.2: Diagrama de Caso de Uso Jugar a un juego

Descripción del Caso de Uso: *Jugar a un juego*

Descripción: El jugador elige un juego y juega.
Actor: Jugador.
Resumen: El jugador elige un juego en la pantalla de elección de juego, juega y después el sistema registra la puntuación obtenida.

Escenario principal

1. El *Sistema* carga la pantalla de elcción de juego.
2. El *Jugador* elige el juego que desee.
3. EL *Sistema* lo ejecuta.
4. El *Jugador* juega al juego elegido.
5. El *Sistema* comprueba cuando el jugador ha terminado.
6. Include *Actualizar Puntuación*.

Escenario alternativo

*a. En cualquier momento el jugador aborta el juego.

*a1. El *Sistema* cierra el juego y carga de nuevo la pantalla de elección de juego.

Descripción del Caso de Uso: *Actualizar Puntuación*

Descripción: El Sistema actualiza la puntuacion del perfil.
Actor: Jugador.
Resumen: El jugador obtiene una puntuación en un juego y el sistema comprueba si ha batido el récord y si es así actualiza la puntuación.

Escenario principal

1. El *Jugador* obtiene una puntuación en un determinado juego.
2. El *Sistema* comprueba que la puntuación obtenida sea mayor que la que tenía almacenada en su perfil.
3. El *Sistema* actualiza el perfil con la puntuación obtenida en el juego.
4. El *Sistema* muestra una pantalla con la puntuación que ha obtenido y dándole la enhorabuena.

Escenario alternativo

2a. La puntuación obtenida en menor que la que tenía almacenada en el perfil.

2a1. El *Sistema* muestra una pantalla al jugador informándole de que no ha batido su récord.

4.4. Diagrama de Clases

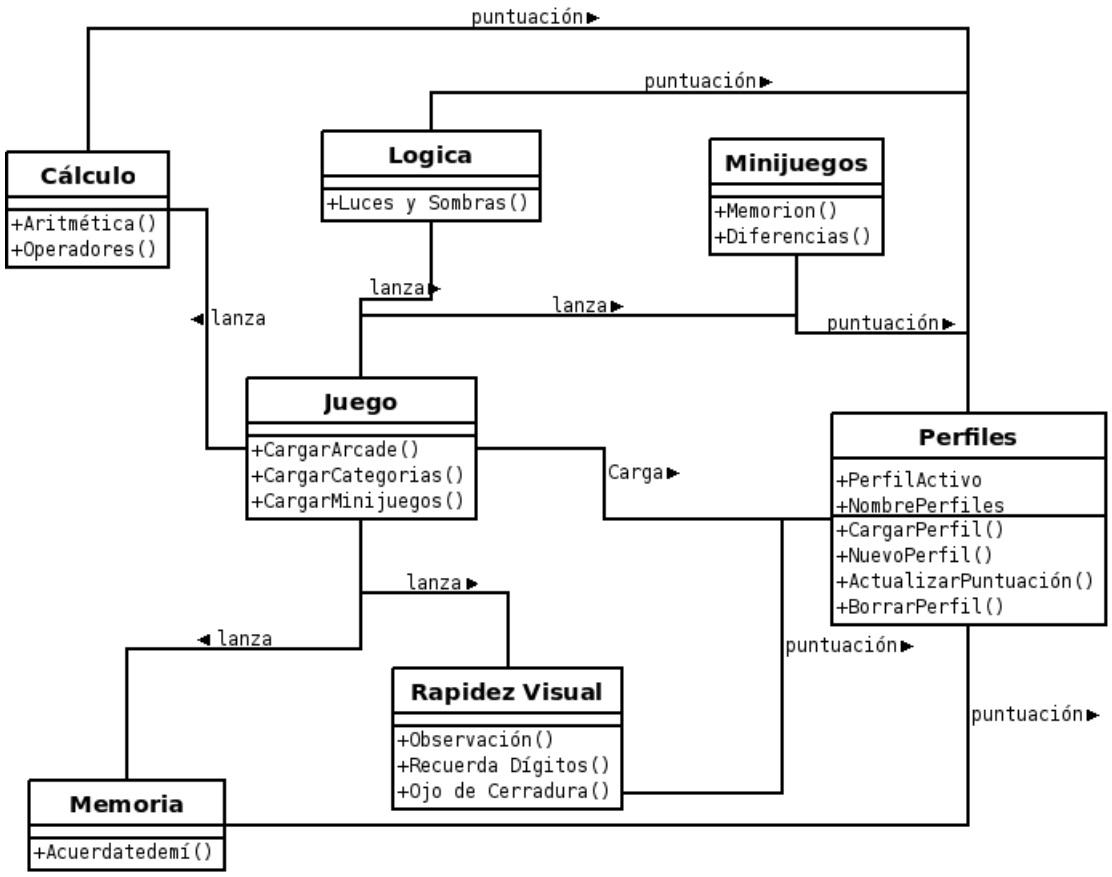

Figura 4.3: Diagrama de Clases

En el diagrama anterior vemos que la clase central es **Juego** ya que es la que controla todos los lanzamientos de juegos y carga todas las pantallas. Otro papel muy importante lo recibe la clase **Perfiles** ya que recibe las puntuaciones de los juegos una vez terminados y los almacena, según si ha obtenido una puntuación mayor, en los ficheros de datos.

Hemos decidido crear una clase para cada categoría de juego, para así tener agrupados a todos los juegos que comparten naturaleza.

En este proyecto no necesitamos demasiados datos, los datos importantes los maneja la clase **Perfiles**, por eso nos encontramos pocos atributos importantes en las clases.

En la figura se ha obviado la clase **Reloj** para simplificar el diagrama, sólo hay que decir que dicha clase será usada por los juegos que necesiten un control de tiempo como pueden ser: **Memorión, Diferencias, Acuerdate de Mí, Aritmética, Operadores y Luces y Sombras**.

4.5. Funciones del Producto

OpenBTrainer está formado por muchos juegos que están separados del núcleo del juego, por tanto primero veremos las funciones principales del núcleo del juego y luego iremos describiendo cada uno de los juegos que se incluyen en esta versión del proyecto.

4.5.1. Interfaz gráfica

Uno de los pilares de un videojuego, sea del tipo que sea, es la interfaz gráfica y el juego que nos ocupa no es ninguna excepción. OpenBTrainer dispone de una interfaz agradable y sencilla, siempre buscando que el usuario encuentre en menos de dos o tres clicks el juego o función que desea. Posiblemente podríamos haber realizado una interfaz más compleja pero me he preocupado en desarrollarla lo más sencilla posible pensando en el público final, buscamos que sea un público de todas las edades e incluso jugadores que no disponen de una soltura demasiado importante en el manejo de un dispositivo informático.

Posiblemente se podría mejorar bastante los gráficos del juego para hacerlo más atractivo, pero pienso que es un proyecto bastante largo y que hay que abarcar todos los campos con un tiempo reducido. Cabe destacar que cualquier juego o producto software de la actualidad está desarrollado usando un equipo de trabajo bien organizado y que cada miembro se ocupa de una parte del software, y que en mi caso mi persona está a cargo tanto del desarrollo de ideas para el juego como de la implementación, pasando por otros campos como el diseño gráfico.

4.5.2. Control de Perfiles

Una de las funciones más importantes del juego es la de controlar los perfiles de los distintos usuarios. El usuario final desea ver los progresos que va realizando.

OpenBTrainer nos ayuda mediante un control de perfiles basado en ficheros. En esta versión, nos permite almacenar el nombre y las puntuaciones de cuatro usuarios distintos, esta limitación viene dado por el apartado gráfico, ya que si optaramos por la opción de almacenar la información de un número casi ilimitado de perfiles, posiblemente no tendríamos la sencillez de la pantalla que mostramos a continuación:

Figura 4.4: Captura de la pantalla para seleccionar perfil

Como hemos visto en la captura anterior sólo tendremos que pinchar en cualquiera de las cuatro posiciones para cargar/crear el perfil que deseemos. Pongámonos ahora en la situación de que disponemos de un número casi ilimitado de perfiles, posiblemente tendríamos que cambiar la imagen anterior por un diálogo de abrir un fichero tipo *.sav como el siguiente:

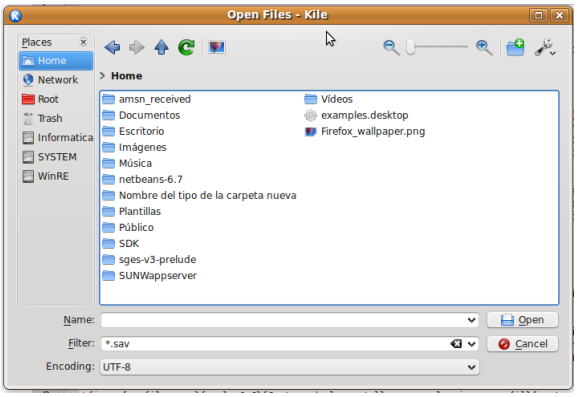

Figura 4.5: Captura de un hipotético diálogo para cargar perfiles

Al hacer así la carga de perfiles estamos complicando la interfaz y estamos reduciendo el público hacia el cuál está dirigido ya que debería saber donde se guardan los ficheros de datos y saber cuál es el fichero que corresponde a su perfil. Por tanto la opción que he elegido sea una de las mejores aunque seguramente exista otra que no me haya planteado.
Centrémonos en la funciones de la pantalla de inicio, tendremos dos opciones según el estado de la posición donde se haya pinchado:

- Si se ha pinchado en una posición Vacía o con etiqueta Nuevo, entraremos en una pantalla en la que nos hará la petición de que introduzcamos mediante el teclado de nuestro ordenador el nombre que vamos a usar en el perfil. El programa se encargará de introducir el nuevo nombre del perfil en el fichero de nombres y de crear el fichero personal con nuestras puntuaciones en los distintos juegos que empezarán a cero.

- Si se ha pinchado en una posición No Vacía, tendremos acceso a un perfil ya creado anteriormente y directamente nos dará acceso a la pantalla de selección de juegos. El programa cargará el fichero con las puntuaciones y dará el error en caso de que lo hayamos borrado por error.

Otra de las funciones principales del control de perfiles es la de almacenar la puntuación en el caso de que haya sido mejor que la tuviese almacenada en el fichero de perfil. Después de que el usuario haya completado un juego, se comprobará la puntuación que tenía y según la situación que describiremos ahora haremos una cosa u otra:

- En el caso de que nos encontremos en un juego en el que se almacenen los niveles máximos que hayamos obtenido, comprobaremos si el jugador ha obtenido anteriormente un nivel mayor, si lo ha obtenido no cambiamos nada, si por el contrario, ha obtenido un nivel superior al que tenía almacenado se cambiará el valor por el actual y se le indicará al usuario que ha batido su propio récord con una pantalla.
- En el caso de que nos encontremos en un juego en el que la puntuación se basa en el tiempo que haya obtenido al completar la prueba deberemos comprobar que el tiempo que haya obtenido es menor que el tiempo que hubiese tenido almacenado. En el caso de que haya batido su récord se le informara al usuario, en caso contrario, se le indicará al jugador que siga intentándolo con una pantalla.

Dentro del control de Perfiles también nos encontramos con las funciones de borrar perfil, que nos dará la posibilidad de eliminar el perfil activo y borrar todas las puntuaciones obtenidas y nos llevará a la pantalla inicial de selección de perfil. Otra de las posibilididades que nos aporta el control de perfiles es la función de cambiar de perfil, que nos llevará a la pantalla inicial de selección de perfil sin la incomodidez de tener que salir y entrar del juego.

4.5.3. Juego Memorión

Una vez que hemos hablado de las funciones que realiza el núcleo del juego, hablaremos de los distintos juegos que nos aporta OpenBTrainer.
El primer juego se incluye en el apartado de **Minijuegos** y se trata del **Memorión** un juego basado en el clásico juego de las parejas de cartas, en el que nos aparece un tablero lleno de cartas vueltas del revés y tendremos que ir pinchando en cada una de ellas de dos en dos hasta dar con las parejas que hay en la mesa. En este caso usamos una serie de cartas de la baraja francesa, concretamente los números del siete al diez de cada uno de los palos (picas, tréboles, corazones y diamantes), esta selección no se ha hecho al azar si no que se ha realizado pensando en los números y las cartas que supongan un esfuerzo adicional para el jugador, posiblemente tal y como está hecha la mente del ser humano, sea más sencillo almacenar una figura con sus colores que un número. Por tanto realizarlo así debe de suponer un reto para el usuario.

La originalidad que le hemos introducido a este juego es la introducción de niveles de dificultad, es decir, en un principio dispondremos de 12 cartas, una vez que hayamos resuelto todas las parejas, aumentaremos el nivel y nos iremos hasta las 18 cartas y por último en el nivel 3 dispondremos de 24 cartas. Una vez que hayamos completado los tres niveles comprobaremos el tiempo que hemos necesitado, pasándole el tiempo a la función que actualiza las puntuaciones de los perfiles.

4.5.4. Juego de las Diferencias

Nos centramos en otro de los juegos dentro de la categoría de **Minijuegos**. Se trata del juego de **Las Diferencias**, cualquier persona sabe en que se basa dicha prueba, tendremos dos imágenes y tendremos que señalar en la imagen de la derecha las diferencias que existen, una pista nos la dará el mensaje que

nos aparece más abajo donde nos indica la cantidad de errores que existen en las imágenes.

Posiblemente lo que llame la atención de este juego no son las imágenes de las diferencias, que son bastante sencillas, si no la forma de realizarse el juego. Hemos creado un motor de juego de diferencias que se podría aplicar a cualquier juego o software de cualquier tipo que querría disponer de algún juego para pasar el rato divirtiéndose. El juego parte de dos imágenes y de un fichero de datos donde almacena las coordenadas x e y donde se sitúan las diferencias y el tamaño del cuadrado de captación. Dicho cuadrado se entenderá mejor con una figura:

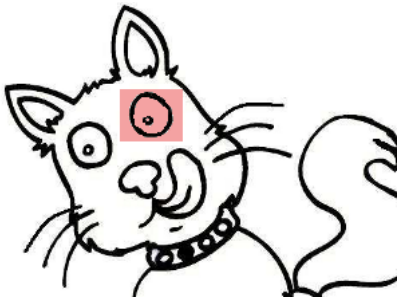

Figura 4.6: Representación de un cuadrado de captación

Como vemos en la imagen ese cuadrado representa al área donde se dará por válido que un usuario haya encontrado la diferencia, dicho cuadrado no debe de ser demasiado pequeño, ya que habría que afinar demasiado para pinchar en el área, ni tampoco muy grande ya que podría solaparse con otro cuadrado de captación que se encuentre alrededor suya.

Por tanto cualquier persona podría insertar imágenes nuevas con el mismo tamaño y modificando el fichero de datos, podría crearse su propio juego de diferencias usando el motor aquí creado.

En nuestro caso almacenaremos el tiempo que hemos tardado en realizar una imagen completa.

4.5.5. Juego Recuerda Dígitos

Ahora nos dedicamos a los juegos del apartado de **Rapidez Visual**, en este caso el juego de **Recuerda Dígitos**, este juego se basa en la apirición de unos números de unas cifras dadas por el nivel en el que nos encontremos y dicho nivel irá incrementándose hasta que llegue el momento en el que fallemos y acabará el juego.

Es un juego sencillo y que perfectamente podría haberse incluído en el apartado de memoria, pero consideramos que la esencia del juego, es que de un simple vistazo seamos capaces de almacenar todos los dígitos que nos aparece durante un corto período de tiempo, en concreto dos segundos. El nivel 1 vendrá dado por dos dígitos que rápidamente memorizaremos ya que en disponemos de un segundo por dígito, la razón de esta sencillez inicial es porque el cerebro es como un músculo, y como músculo deberemos de ir realizando un esfuerzo paulatino, es decir, poco a poco incrementando en cada nivel un dígito, hasta llegar a un nivel que no podremos almacenar todo el número completo.

El juego pasará el control al manejador de perfiles y éste almacenará el nivel máximo alcanzado.

4.5.6. Juego de Observación

Posiblemente el juego de **Observación** enclavado en el apartado de **rapidez Visual**, sea uno de los más complejos de completar. Se basa en la aparición de unas flechas que nos indican una dirección, durante un período corto de tiempo, un segundo, luego desaparecerá y el sistema nos hará una pregunta sobre la cantidad de flechas de un determinado tipo hemos visto y habrá que indicárselo mediante el teclado. La dificultad viene que a priori no conoces por el tipo de flecha que el sistema va a preguntarle y por tanto habrá que tratar de almacenar el número de flechas de todos los tipos.

El primer nivel comenzará con dos flechas de dos tipos diferentes y nos preguntará por una de ellas, no es necesario decir que el primer nivel es sumamente sencillo y la razón de ello es la que indicamos en la sección anterior.

El juego pasará el control al manejador de perfiles y éste almacenará el nivel máximo obtenido.

4.5.7. Juego Ojo de Cerradura

Se trata de otro juego de la categoría de **Rapidez Visual**. **Ojo de Cerradura** puede resultar un nombre tanto extraño pero es algo buscado ya que simplemente el nombre llamará la atención del jugador y posiblemente sea el juego que pruebe antes de la categoría. Es un juego original y se basa en la idea de que da la impresión de que estamos delante de una puerta asomándonos al ojo de la cerradura de dicha puerta y tras ella nos va a aparecer un código secreto y deberemos almacenarlo en nuestra memoria.

En el primer nivel nos aparece un número de dos dígitos que avanzará de derecha a izquierda por detrás de la cerradura, una vez que pase el sistema nos avisará para que introduzcamos el número que ha pasado. Uno de los puntos más complejos del juego es el aumento de nivel, ya que según al nivel que avancemos producirá que aumente el número de dígitos o que aumente la velocidad con la que pasa el número por detrás de la cerradura.

El juego, una vez terminado, dejará el control al manejador de perfiles que almacenará el nivel máximo obtenido.

4.5.8. Juego Acuerdate de Mí

Se trata de un juego de la categoría de **Memoria**. **Acuerdate de Mí** se basa en una serie de personajes, que nos aparecerán en forma de bloques durante unos segundos, que vendrá determinado por el nivel en el que nos encontremos y una vez acabado el tiempo nos desaparecerá un número de personajes, que también viene determinado por el nivel en el que nos encontremos. Luego una vez que hayan desaparecido los personajes se nos mostrarán todos los personajes y se nos pediremos que pinchemos en los personajes que han desaparecido. Una vez que los hayamos hecho el personaje volverá a su sitio y cuando completemos el nivel, éste aumentará y comenzará todo de nuevo pero con un nivel más, es decir, con un personaje que desaparece más y con un bonus de un segundo más para que nos centremos bien en los personajes que tenemos que memorizar.

Una vez que hayamos fallado más de dos veces el juego termina y se almacenará en el perfil el número de nivel máximo que hemos obtenido.

4.5.9. Juego Luces y Sombras

En esta ocasión nos encontramos con un juego de la categoría de **Lógica**. Se trata de un juego bastante entretenido y totalmente diferente a todos los juegos que se incluyen en el proyecto. Se basa en la idea

de tener que apagar todas las luces de una hipotética habitación.

El juego posee un tablero de 5 x 5 luces, el mismo comienza con un nivel en el que están encendidas las luces de las esquinas y deberemos apagarlas con la peculiaridad que caracteriza a este juego, al encender o apagar una posición, se encienden o apagan, según la circunstancia, dicha posición y todas las consecutivas, es decir, la situada arriba, derecha, abajo e izquierda. Deberemos usar la lógica para crearnos nuestras propias estrategias para conseguir, usando el menor tiempo posible, resolver todos los niveles que se han creado para el juego. Cabe destacar que puede resultar dificultoso las primeras veces que se juega, pero con la práctica puede convertirse en un juego muy entretenido y que nos hará pensar mucho para resolver los cuatro niveles.

Una vez que hayamos completado todos los niveles se almacenará, mediante el control de perfiles, el tiempo empleado en finalizar el juego.

4.5.10. Juego Aritmética

Ahora nos encontramos con un juego llamado **Aritmética** de la categoría **Cálculo**. Se trata del clásico juego de cálculos aritméticos, en el que aparecerá una determinada cuenta de manera aleatoria y deberemos resolverla. En esta ocasión nos encontramos con un juego en el que debemos averiguar el resultado del segundo operando, es decir, nos aparece una cuenta con la siguiente sintaxis *3 + ? = 6* y deberemos resolverla. Luego para escribir el resultado tenemos la posibilidad de usar el teclado alfanumérico o el teclado numérico que nos encontramos en la mayoría de los teclados convencionales.

El objetivo de la prueba es resolver, todas las cuentas posibles usando un minuto y medio de tiempo. Por tanto deberemos darnos prisa para superar todas las cuentas que podamos. Los fallos no serán penalizados. Una vez terminado el tiempo el juego devolverá al control de perfiles la cantidad de cuentas resueltas en el minuto y medio de tiempo.

4.5.11. Juego Operadores

El último juego que vamos a describir es el llamado **Operadores** de la categoría **Cálculo**. Se trata de un juego muy parecido al anterior, pero con la salvedad de que en el juego anterior debíamos averiguar el segundo operando de la cuenta y en esta ocasión deberemos resolver el operador de la cuenta. De tal forma que si nos aparece una cuenta de la siguiente manera *3 ? 3 = 6* nosotros deberemos indicarle al sistema que el operador que falta es el operador de suma. En esta ocasión los controles son distintos al juego anterior, tenemos dos posibilidades, o usamos el ratón y marcamos con el ratón en el botón del operador correspondiente o usamos el teclado numérico.

El objetivo del juego es similar al anterior, deberemos resolver todos los cálculos posibles antes de que acabe el minuto y medio de tiempo.

Capítulo 5

Diseño

El proceso de Diseño se puede definir como una actividad consistente en aplicar distintas técnicas y principios con la finalidad de definir un sistema con suficiente detalle para que se pueda implementar.

Después de un gran trabajo de análisis y tener todos el análisis de requisitos de todos los juegos bien planteados, la tarea de diseño es simple. Sólo tenemos que pensar en cómo lo hace la máquina, definir las operaciones más importante y una vez hecho esto pasar a la fase de implementación.

De manera breve comentaremos las distintas funciones y procedimientos que componen los distintos módulos, las distintas opciones que nos permiten y lo que nos aporta.

5.1. Juego

Módulo principal donde se manejan todas las llamadas a los distintos juegos.

- **Juego::Juego()**. Constructor de la clase juego en el que se inicializan todos los subsistemas necesarios de libSDL.
- **Juego::iniciar_pantalla()**. Crea y devuelve la superficie de la pantalla inicializada a un tamaño de 800x600.
- **Juego::Minijuegos(perfil, pantalla)**. Carga el menú de minijuegos y carga el juego que el usuario le indica mediante un click del ratón.
- **Juego::Categorías(perfil, pantalla)**. Carga el menú de elección de juego por categorías y carga el juego elegido.
- **Juego::DibujarEscenarioCategorias(pantalla)**. Dibuja en la pantalla, el escenario de categorías con los botones correspondientes.
- **Juego::DesplegarBotones(categoria, pantalla)**. Nos ayuda a desplegar la lista de botones de los juegos según la categoría elegida.

A continuación se muestra un diagrama con las distintas dependencias del módulo:

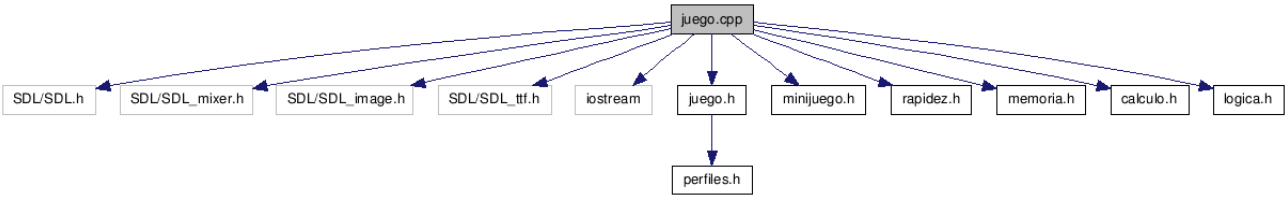

Figura 5.1: Diagrama de dependencia de Juego

5.2. Minijuegos

Módulo de los juegos que conforman la categoría de Minijuegos.

- **Minijuego::Minijuego()**. Constructor que usaremos para inicializar la semilla de números pseudoaleatorios que nos servirán para los juegos que componen la categoría.

- **Minijuego::Memorion(pantalla)**. Se lanza el juego del Memorión, una vez que finaliza devuelve la cantidad de segundos que ha tardado el usuario en completar los tres niveles.

- **Minijuego::Diferencias(pantalla)**. Se lanza el juego de las Diferencias, una vez que finaliza el juego devuelve la cantidad de segundos que ha empleado el usuario en completar la imagen.

Como en la sección anterior vemos el diagrama de dependencias del módulo:

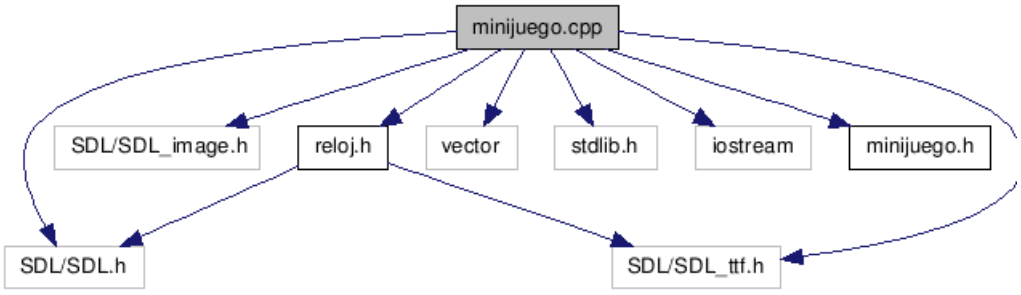

Figura 5.2: Diagrama de dependencia de Minijuegos

5.3. Memoria

Módulo de los juegos que componen la categoría de Memoria.

- **Memoria::Memoria()**. Como comentamos en la sección anterior, usaremos el constructor para generar una semilla pseudoaleatoria que nos sirve para los juegos que componen el módulo.

- **Memoria::acuerdatedemi(pantalla)**. Lanza el juego Acuerdate de mí y nos devuelve el nivel máximo que el jugador ha completado.

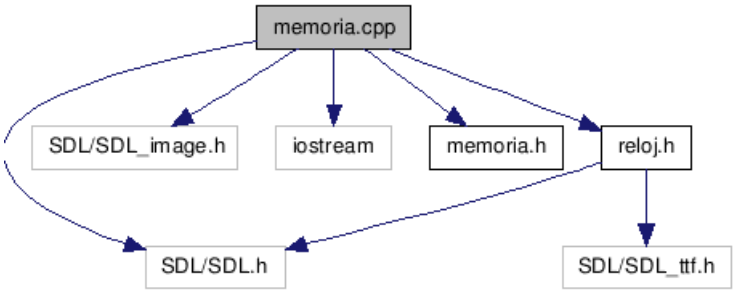

Figura 5.3: Diagrama de dependencia de Memoria

5.4. Rapidez

Módulo de los juegos que componen la categoría de Rapidez Visual.

- **Rapidez::Rapidez()**. Usaremos el constructor para generar una nueva semilla de números pseudoaleatorios.
- **Rapidez::recuerda(pantalla)**. Lanza el juego de Recuerda los Dígitos y una vez finalizado el mismo devuelve el nivel máximo completado.
- **Rapidez::observa(pantalla)**. Lanza el juego de Observación y una vez completado devuelve el nivel máximo alcanzado.
- **Rapidez::ojodecerradura(pantalla)**. Arranca el juego de Ojo de Cerradura, una vez que el usuario finalice devuelve el nivel máximo alcanzado.

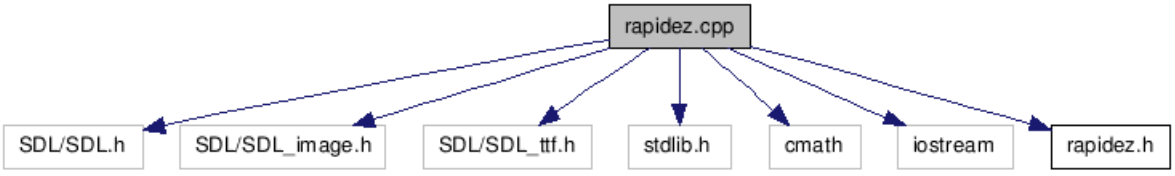

Figura 5.4: Diagrama de dependencia de Rapidez

5.5. Lógica

Módulo de los juegos que tienen la naturaleza de usar necesariamente la lógica.

- **Logica::LucesySombras(pantalla)**. Lanza el juego de Luces y Sombras, una vez que finaliza devuelve el tiempo que ha tardado el usuario en completar los distintos niveles.

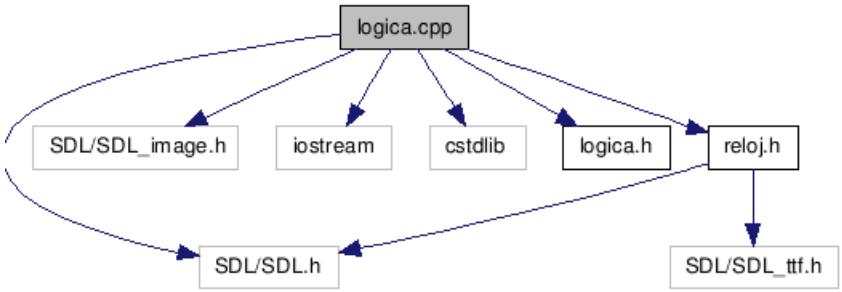

Figura 5.5: Diagrama de dependencia de Lógica

5.6. Cálculo

Módulo de la clásica categoría de cálculo aritmético.

- **Calculo::Calculo()**. Usaremos el constructor para generar una semilla aleatoria.
- **Calculo::Aritmetica()**. Lanza el juego de Aritmética, cuando finalice devolverá el número de cuentas que el usuario ha superado en los 90 segundos.
- **Calculo::Operadores()**. Lanza el juego de Operadores, cuando finalice devolverá, como en el juego anterior, el número de cuentas que el usuario ha superado en el tiempo del juego, 90 segundos.

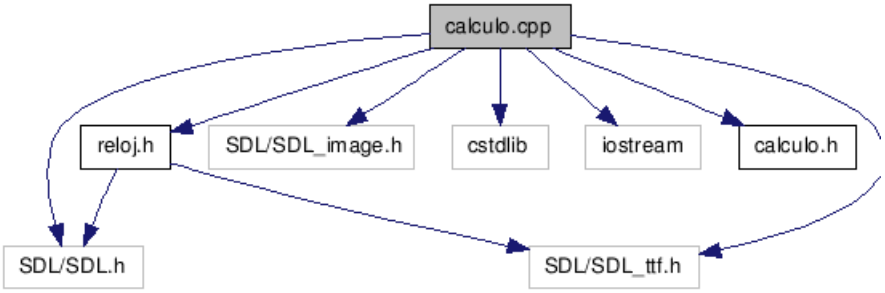

Figura 5.6: Diagrama de dependencia de Cálculo

5.7. Perfiles

Módulo con los manejadores de perfiles de usuarios.

- **perfiles::perfiles()**. Constructor de la clase que comprueba si existe el fichero de perfiles, si no existe lo crea con los cuatro perfiles vacíos, si no guarda los nombres de los perfiles existentes.
- **perfiles::cargar(pantalla)**. Carga la pantalla de selección de perfil y llama a nuevo perfil si se ha seleccionado un perfil vacío.
- **perfiles::nuevoperfil(pantalla)**. Carga la pantalla de introducción de datos del perfil que ha seleccionado anteriormente, mapea el teclado y almacena en el fichero de perfiles el nombre seleccionado al igual que crea un fichero con la información personal y con las puntuaciones a 0.

- **perfiles::actualizarpuntuacion(perfil, codigodejuego, puntuacion, tipopuntuacion, pantalla)**. Función que hace que los perfiles se vayan actualizando. Los parámetros que se le pasan son el perfil que actualmente está activo, el código del juego sobre el que se va a actualizar, la puntuación obtenida, el tipo de puntuación que será 1 si es de tipo nivel o similar, es decir, si la puntuación se debe actualizar si obtiene una puntuación mayor a la que tenía, y un número distinto de 1 si la puntuación es de tipo tiempo y el objetivo es conseguir el menor posible, además habrá que pasar como parámetro la variable de la pantalla ya que según si se ha obtenido una puntuación mayor o menor que la que se tenía, se mostrará una pantalla con los resultado obtenidos.
- **perfiles::perfilactual()**. Devuelve el perfil que está activo en ese momento, será un número del 0 al 3.
- **perfiles::nombreperfilactual(nombre)**. Nos escribe en la variable que le pasamos como parámetro el nombre del perfil activo, esto nos será de gran utilidad si queremos mostrarle algún tipo de mensaje personalizado al usuario.
- **perfiles::borrarperfil()**. Nos permite borrar el perfil activo. Borra el fichero con la información personal y las puntuaciones obtenidas y borra del de perfiles el nombre del registro.

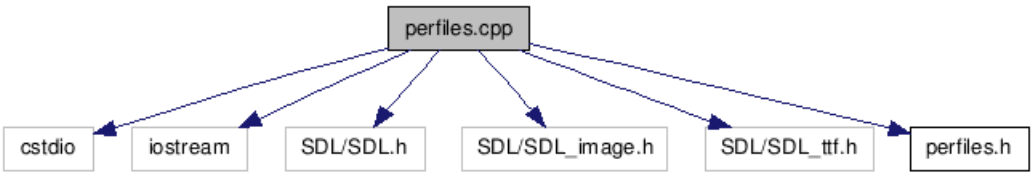

Figura 5.7: Diagrama de dependencia de Perfiles

5.8. Reloj

Módulo del reloj que usaremos para los juegos que lo necesiten.

- **reloj::reloj(posicionx, posiciony)**. Constructor de un reloj de tipo normal (ascendente) que se irá a dibujar en la posición x e y que se le pasa como parámetro.
- **reloj::reloj(tiempo, posicionx, posiciony)**. Constructor de un reloj de tipo cuenta atrás (descendente) que se irá a dibujar en la posición x e y que se le pasa como parámetro.
- **reloj::actualizartiempo(pantalla)**. Con este procedimiento consultamos si ha pasado un segundo y si es así se actualiza el tiempo en la pantalla.
- **reloj::consultartiempo(minutos, segundos)**. En cualquier momento podremos consultar los minutos y los segundos que han transcurridos desde que se hizo la llamada al constructor.
- **reloj::consultarseg()**. Nos ayuda devolviéndonos los segundos. Normalmente lo usaremos si el reloj es de tipo cuenta atrás para comprobar cuanto tiempo queda y si se ha llegado a completar.
- **reloj::dibujatiempo(pantalla)**. Dibuja el tiempo actual en la pantalla.
- **reloj::settime(minutos, segundos)**. Conseguimos con este procedimiento cambiar el valor de minutos y segundos en cualquier momento.

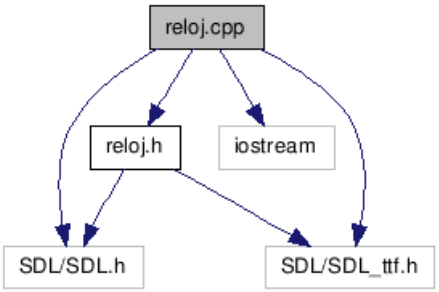

Figura 5.8: Diagrama de dependencia de Reloj

5.9. Características de los usuarios finales

OpenBTrainer se ha diseñado pensando en que los usuarios finales no necesitan grandes nociones de ningún campo del conocimiento en especial, simplemente esta diseñado para cualquier tipo de jugador con ganas de divertirse y de hacer funcionar la mente al cien por cien. Únicamente necesitará tener unas nociones mínimas de manejo de teclado y ratón. Las edades a las que va dirigido el juego puede ser de 10 a 80 años, aunque esta cifra es sólo orientativa ya que para adentrarse en el mundo de OpenBTrainer sólo se necesitan ganas.

Entrando en comparaciones, el juego Gbrainy de Jordi Mas, posiblemente tenga una variación con respecto a mi proyecto. Las características de los usuarios finales son distintas a las mías. La explicación a ello es que las pruebas que podemos encontrar resultan algo complejas para personas de corta edad y posiblemente no llamen la atención del público de menos de 15 años. Las pruebas de OpenBTrainer se han diseñado con la idea de llegar a cualquier usuario y con juegos fácilmente comprensibles y que no necesiten apenas instrucciones de juego.

Capítulo 6

Implementación

Una vez que hemos finalizado las fases de análisis y diseño, sólo nos queda codificar todas las instrucciones usando un lenguaje de programación y usando una herramienta gráfica. En nuestro caso hemos elegido el lenguaje de programación C++ y la librería gráfica libSDL.

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje multiparadigma. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales. C++ permite trabajar tanto a alto como a bajo nivel siendo muy óptimo.

El nombre C++ fue propuesto por Rick Mascitti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera de un laboratorio científico. Antes se había usado el nombre “C con clases”. En C++, la expresión “C++” significa “incremento de C” y se refiere a que C++ es una extensión de C [6].

El motivo de la elección de C++, es que el grado de familiarización con este tipo de lenguaje es muy alto. Ya que en la Universidad de Cádiz es el lenguaje por el cual se rigen la mayoría de las asignaturas. Es un lenguaje muy eficiente y que nos ayuda con multitud de herramientas que nos serán útiles a lo largo de la fase de implementación. La otra herramienta que usamos es libSDL, Simple DirectMedia Layer (SDL) que es un conjunto de bibliotecas desarrolladas con el lenguaje C que proporcionan funciones básicas para realizar operaciones de dibujado 2D, gestión de efectos de sonido y música, y carga y gestión de imágenes [7].La biblioteca se distribuye bajo la licencia LGPL, que es la que ha provocado el gran avance y evolución de las SDL. Se han desarrollado una serie de bibliotecas adicionales que necesitamos ya que extienden las funcionalidades de la librería:

- SDL Mixer: Extiende las capacidades de SDL para la gestión y uso de sonido y música en aplicaciones y juegos. Soporta formatos de sonido como Wave, MP3 y OGG, y formatos de música como MOD, S3M, IT, y XM.
- SDL Image: Extiende notablemente las capacidades para trabajar con diferentes formatos de imagen. Los formatos soportados son los siguientes: BMP, JPEG, TIFF, PNG, PNM, PCX, XPM, LBM, GIF, y TGA,
- SDL TTF: Permite usar fuentes TrueType en aplicaciones SDL.

Si tuviésemos que dar un motivo por el cual elegir esta herramienta sería que dicha herramienta es familiar para mí, ya que al haber cursado la asignatura de Diseño de Videojuegos, cuyo profesor es uno de los tutores de este proyecto, es la herramienta que usamos para desarrollar el curso. Por tanto usando los documentos y tutoriales que facilita la asignatura es bastante fácil aprender a usar dicha librería. Además es una librería simple y que da bastante buenos resultados, siempre que nos dediquemos a usarla en un juego en dos dimensiones, ya que si el proyecto fuese en tres dimensiones tendríamos que usar OpenGL[1] y una librería como Ogre[2].

6.1. Desarrollo de OpenBTrainer

En primer lugar para la codificación del programa, hemos usado un entorno integrado como es el caso de NetBeans[3] con el plugin de C/C++, dicho entorno nos ha facilitado la tarea de escribir el código ya que nos realiza un análisis sintáctico del código antes de pasarlo por el compilador y nos permite crear un código limpio y bien sangrado.

Para la compilación usamos el compilador de C++ llamado g++, que viene incluída en la distribución de cualquier sistema GNU/Linux.

El proyecto se divide en núcleo del juego y las pruebas que lo componen. El núcleo del juego lo componen la interfaz gráfica de los menús principales y la gestión de perfiles.

6.1.1. Núcleo del Juego

En primer lugar hablaremos de la interfaz gráfica. La interfaz gráfica se componen de imágenes desarrolladas en el programa GIMP y botones creados por mí. Todo ello mediante la librería libSDL, es usado y creado la interfaz fácilmente.

En segundo lugar hablaremos del desarrollo de la gestión de perfiles. El desarrollo de la gestión de perfiles se basa en la lectura/escritura de una serie de ficheros contenidos en la carpeta *perfiles*. En el fichero *perfiles.sav* mantenemos la información de los nombres de los cuatro perfiles que nos permite el juego, en los ficheros *0 - 4.sav* guardamos la información del nombre del perfil y todas las puntuaciones de las pruebas. Por tanto las operaciones que necesitará el módulo de perfiles será leer dichos ficheros y escribir las puntuaciones en el perfil adecuado.

[1] OpenGL (Open Graphics Library) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triángulos.

[2] OGRE 3D (acrónimo del inglés Object-Oriented Graphics Rendering Engine) es un motor de renderizado 3D orientado a escenas, escrito en el lenguaje de programación C++.

Sus librerías evitan la dificultad de la utilización de capas inferiores de librerías gráficas como OpenGL y Direct3D, y además, proveen una interfaz basada en objetos del mundo y otras clases de alto nivel.

[3] NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE) desarrollado usando la Plataforma NetBeans.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

6.1.2. Juegos de OpenBTrainer

En segundo lugar hablaremos de los distintos juegos que componen el proyecto. Cada juego se sitúa en la clase que pertenece por categoría. Cada juego tendrá su propio game loop, es decir el bucle principal del juego en el que se controlan todos los eventos de la prueba, y además tendrán sus funciones y procedimientos privados.

Memorión

Para el desarrollo hemos elegido como tipo de dato una matriz de enteros, ya que cada entero equivale a una carta, rellenaremos la matriz con enteros aleatorios, de modo que siempre habrá dos números iguales en posiciones distintas, para conseguir así que siempre haya parejas y no cartas sueltas.

Según el nivel en el que nos encontremos, evaluaremos los eventos de pulsado de botón del ratón y llamaremos a la función voltear carta, ésta comprobará si las cartas elegidas son iguales y si deben desaparecer de la mesa o volver a darse la vuelta. Una vez que todas las parejas hayan sido encontradas, el juego aumentará de nivel, y por tanto el número de cartas sobre la mesa, o por el contrario terminará el juego y se calcularán los segundos que se han empleado para completar la prueba.

Diferencias

Para la implementación del juego, cargamos el fichero que creamos con todas las soluciones de las diferencias. De modo pseudoaleatorio cargamos un número de imagen, una vez que hemos cargado las dos imágenes, la imagen de la izquierda y la derecha, la cual nos servirá para marcar las diferencias, metemos los datos del fichero en una estructura de datos formada por una variable formada por el número de diferencias y tres vectores, para las posiciones x, y y para el tamaño del cuadrado de captación.

Una vez que tenemos todos los datos cargados sólo nos falta evaluar los eventos de pulsado de botón izquierdo del ratón y si coinciden con la diferencia cargar la imagen de la “x” para marcarla.

Recuerda Dígitos

Para la implementación de este sencillo juego, simplemente generamos pseudoaleatoriamente un número de tantas cifras como niveles llevemos completados, necesitamos también una cadena donde iremos introduciendo los dígitos que nos indica el usuario y un contador para el índice de la cadena.

Una vez que tenemos todas las variables y tipos de datos, simplemente captamos las pulsaciones de teclas y las iremos introduciendo en la variable cadena hasta que coincida con el número que generamos. Cabe destacar que el juego está preparado para captar las teclas del teclado alfanumérico y las del teclado numérico ya que será más sencillos para todos aquellos usuarios que estén acostumbrados a usarlo. En nuevas versiones se ha incorporado la capacidad de borrar el número que anteriormente ha introducido el usuario para que así disponga de la posibilidad de arreglar fallos antes de evaluar si el número es correcto.

Observación

Para el desarrollo y codificación de este juego se ha necesitado una variable de tipo entero que equivaldría a las cuatro posiciones distintas de las flechas. Según en el nivel que estemos tendremos que generar más o menos números de flechas. Además de las flechas necesitamos una variable que controle las posiciones donde se van a situar las flechas en la pantalla. Dispondremos de 8 posiciones que asignaremos de

forma pseudoaleatoria. Una vez que tenemos todas las flechas correspondientes al nivel. El juego debe generar la pregunta de cuántas flechas de un tipo existen, esto también lo haremos de forma aleatoria y conseguiremos el número de flechas del tipo que hemos generado.

Una vez que hemos inicializado todas las flechas del nivel y generado la pregunta del nivel, debemos captar los eventos producidos por la pulsación de una tecla del teclado y evaluar que este número es correcto. Si es correcto aumentaremos de nivel o en el caso de llegar al nivel 8 concluirá el juego con la máxima puntuación posible.

Ojo de Cerradura

Para la codificación de este juego, lo principal es preparar un fondo en forma de puerta con una cerradura y que el ojo de la cerradura sea transparente. Por tanto el juego se basará en la generación de un número, cargar un fondo gris, imprimir el número y luego el fondo del escenario, para que éste quede por encima del número y de la sensación de que el número está detrás de la puerta. Para el movimiento simplemente crearemos una función que vaya varando la posición x del número según el nivel en el que nos encontremos.

Una vez que tenemos el juego inicializado deberemos esperar hasta que el número llegue a una determinada posición en la que no se vea y luego preguntar al usuario por dicho número. La manera de almacenar las cifras que nos indica el usuario, será igual que en el juego **Recuerda Dígitos**. Una vez que acierte el número que ha aparecido el juego aumentará el nivel por tanto, podrá incrementarse la velocidad con la que pasan las cifras (nivel impar) o aumentar la cantidad de dígitos que aparecen (nivel par).

Acuérdate de Mí

Para la implementación de este juego nos basamos nuevamente en una matriz de enteros, en los que cada uno de ellos representa a un personaje. Rellenaremos de manera pseudoaleatoria toda la matriz con dichos enteros y cargaremos en la pantalla los personajes asociados a dichos números. En este juego necesitamos usar la clase reloj para crear una cuenta atrás cuyo tiempo dependerá directamente del nivel en el que nos encontramos. Una vez que hayamos completado el tiempo, de manera aleatoria, desaparecerá una posición de la matriz. Luego mostraremos en la parte baja de la pantalla todos los personajes que existan para que el usuario elija el que crea que ha desaparecido.

Una vez llegado a este punto sólo tendremos que controlar los eventos de tipo pulsación de botón izquierdo del ratón y comprobar que el personaje que ha pulsado es el que realmente ha desaparecido, si no es así dispondremos de 3 oportunidades más antes de que el juego termine. En cada nivel que pasemos desaparecerá un personaje más y tendremos un bonus de un segundo más para que nos sea más fácil memorizar todos los personajes.

Aritmética

Para el desarrollo de este juego nos basamos en la generación pseudo aleatoria de tres números, para el primer operando, el segundo y para el operador respectivamente. Una vez que tengamos todos los operandos y el operador calcularemos la solución de la cuenta. Dibujaremos toda la cuenta en la pantalla menos el segundo operando que será el que permanecerá oculto mediante un signo de interrogación (“?”).

Una vez que tenemos todo el escenario cargado y generado toda la cuenta, simplemente tendremos que captar los eventos de teclado, tanto de teclado alfanumérico como de numérico y comprobar que el número pulsado sea correcto, si es incorrecto no haremos nada al respecto. Una vez que se acierte aumentaremos de nivel. Este aumento de nivel, en oposición a los demás juegos del proyecto no supondrá un aumento de la dificultad, sólo nos aparecerá otra cuenta similar a la anterior. El usuario deberá completar en un minuto y medio, que mostraremos mediante la clase reloj, todas las operaciones posibles.

Operadores

Para la implementación de este juego nos basaremos en el código del anterior juego que hemos descrito ya que presenta muchas similitudes con éste. Los cambios más significativos que nos encontramos que en vez del segundo operando será el operador el que desaparezca.

Para que los usuarios puedan elegir el operador y no sea demasiado tedioso, como por ejemplo usar teclas asignadas a cada operador, he decidido crear cuatro botones, cada uno asignado a cada operador y además la posibilidad de usar lo operadores del teclado numérico. Al igual que el juego anterior el aumento de nivel no supone un incremento en la dificultad. Dispondrá de un minuto y medio para conseguir el mayor número de operaciones resueltas.

6.1.3. Luces y Sombras

Para el desarrollo de este juego, nos basamos en una estructura de datos en forma de matriz de booleanos, si una posición está a *true* significará que la luz en esa posición estará encendida. Cargaremos el fichero de mapas del juego que se compone de unos y ceros y lo incorporaremos a la matriz de tablero. Una vez hecho esto cargaremos las imágenes de las luces apagadas y encendidas en la posición que corresponda.

Luego tendremos que ir captando los eventos del ratón y llamando a una función para que dibuje los cambios en la posición donde se pulsó el botón izquierdo del ratón, arriba, derecha, izquierda y abajo, siempre que sea posible. El juego dispondrá de un temporizador que usaremos de la clase reloj. Una vez que finalicemos un nivel, cargaremos el nivel siguiente siempre y cuando no hayamos llegado al último, en tal caso finalizaremos el juego y devolveremos los segundos que hemos empleado para completar el juego.

Capítulo 7

Pruebas y Estadísticas

En este apartado presentaremos la manera con la que hemos procedido ha realizar las pruebas a los distintos componentes del juego y además presentaremos unas estadísticas realizadas sobre usuarios reales sobre los resultados que obtuvieron en las distintas pruebas de OpenBTrainer.

7.1. Pruebas realizadas

Según las recomendaciones de G. J. Myers [8]:

1. Cada caso de prueba debe especificar el resultado esperado: Es decir el juego funciona correctamente y devuelve una puntuación válida.
2. El programador debe evitar probar sus propios programas: Los usuarios elegidos para probar el proyecto siempre son personas ajenas al juego.
3. Se debe analizar con detalle el resultado de cada prueba: Se debe comprobar con detenimiento que todas las funciones de cada prueba funcionan correctamente y escriben en el perfil la puntuación satisfactoriamente.
4. Se deben incluir entradas válidas e inválidas: El usario externo al proyecto prueba todas las teclas que el piensa y se comprueba que no producen reacciones extrañas.
5. Se debe:
   - Probar si el software no hace lo que debe. El juego debe captar los eventos necesarios y cargar todos los escenarios que hacen falta.
   - Probar si el software hace lo que no debe. El juego se vuelve inestable en algún caso en el que debería funcionar correctamente.
6. Los casos de prueba deben documentarse: Mostraremos las pruebas realizadas en cada iteración más adelante.
7. Hay que asumir que hay errores: En el momento en el que programamos códigos largos suele encontrarse muchos fallos.
8. Donde hay un defecto hay otros: En caso de que encontremos un fallo, ese fallo puede provocar otros más o menos graves que éste.
9. Las pruebas son una tarea creativa: No nos debemos quedar en las pruebas ideales, debemos probar el programa con todo tipo de casos y situaciones.

7.1.1. Incremento 1: Requisitos Básicos del sistema

¿El sistema carga correctamente los fondos y aparece la pantalla principal? ¿El sistema capta correctamente la pulsaciones de los distintos botones?

El sistema desde el primer momento carga los fondos, los botones después de unos ajustes son colocados en sus posiciones correspondientes.

El programa capta correctamente la pulsación de botones determinada por el tamaño del mismo, se ha probado pulsar en regiones muy cercanas a los botones e incluso entre dos botones y el sistema se comporta de manera satisfactoria para todos los casos.

7.1.2. Incremento 2: Implementación de Memorión

¿El juego carga correctamente las imágenes y el revés de las cartas? ¿Capta la pulsación con el ratón encima de las distintas cartas? ¿Existen siempre una carta homóloga a la carta pulsada? ¿En el momento de adivinar una pareja, el sistema la elimina de la mesa? ¿Es capaz el sistema de saber cuando una mesa está vacía?

El juego podemos comprobar que carga la imagen de fondo, así como el botón de salir del juego y su captación de evento correspondiente. Las cartas después de una serie de ajustes, logramos que se coloquen en las posiciones correctas y carguen correctamente con un tamaño correcto.

El sistema se comporta correctamente al pulsar en cada carta y nos informa de la correspondencia en valor de la posición de la matriz asociada a dicha carta.

Esta es una prueba importante, ya que a menudo nos encontramos en la situación de que se cargan correctamente las cartas pero hay algunas sin pareja. Solucionamos el problema imprimiendo el valor de la matriz inicial y haciendo varios ajustes.

La siguiente prueba la pasa correctamente el sistema, ya que reconoce correctamente cuando dos cartas son iguales. Uno de los problemas que nos econtramos usando a usuarios reales que cuando se pulsa dos veces sobre la misma carta, ésta desaparece, por tanto debemos indicarle al sistema que las dos cartas no deben de tener la misma posición.

La prueba siguiente nos aportó la necesidad de tener una variable controladora del número de cartas que hay en la mesa e ir decrementándola en dos cada vez que descubríamos una pareja.

7.1.3. Incremento 3: Implementación del juego de las Diferencias

¿El juego carga correctamente el fichero y lee los datos? ¿El juego introduce estos datos en la estructura de datos de la prueba? ¿El juego capta las diferencias y dibuja la cruz de marcado?

El juego desde un principio carga correctamente el fichero y lee bien los datos.

El juego una vez que tenemos todos los datos leídos, directamente son almacenados en memoria mediante la estructura de datos que creamos para el juego.

Las pruebas más duras a las que tuvo que hacer frente este juego fueron la captación de diferencias. Para solucionar los numerosos fallos con los que me encontraba diseñé el concepto de cuadrado de

captación. Una vez hecho esto, fuimos cargando cada imagen una por una y comprobando que cada diferencia funcionaba correctamente, en caso erróneo modificamos el fichero de soluciones. Gracias a la colaboración de usuarios opté por crear unos cuadrados de captación lo bastante amplios para que cualquier jugador localice y señale la diferencia en el primer intento.

7.1.4. Incremento 4: Implementacion del juego Recordar Dígitos

¿El juego genera un número válido para el nivel en el que nos encontramos? ¿El juego capta correctamente el número que el usuario introduce? ¿El juego comprueba la corrección del número satisfactoriamente? ¿El aumento de nivel se realiza de forma correcta?

El juego genera correctamente en todos los casos un número válido, sin que pueda aparecer el número cero y con las cifras que corresponde al nivel.

El juego genera algunos fallos al captar los número que el usuario introduce sobretodo al aumentar de nivel, ya que la variable genera basura y se imprimen trozos de memoria con caracteres extraños, para solucionarlo decidimos rellenar en cada nivel la variable cadena con aes, de modo que al captar los números, la función al encontrarse una a pare de captar dígitos. Otro aspecto importante es el borrado que después de unos problemas de fallos de segmentación fueron solventados y funciona correctamente.

El sistema mostraba errores al introducir un número que no era correcto, ya que permitía seguir introduciendo números indefinidamente. Para solventar el fallo tuvimos que incorporar un sistema que comprobara que si el número tenía las mismas cifras que el buscado y no era éste, lo interpretara como un error.

El sistema desde el primer momento se comporta bien al aumentar de nivel, se refresca la pantalla y aparece un nuevo número con las cifras correctas.

7.1.5. Incremento 5: Implementación del juego Observación

¿El juego carga correctamente el escenario y las distintas flechas? ¿El juego genera una pregunta correcta? ¿El juego evalúa correctamente la respuesta del usuario?

El juego carga correctamente las flechas, sólo se deben hacer ajustes en la posición donde se cargan las flechas, pero se comporta bien en cada nivel y con las flechas correctas.

El juego genera una pregunta correcta, ya que la flecha por la que pregunta, siempre se ha mostrado anteriormente.

El sistema evalúa correctamente la respuesta del usuario mediate el teclado alfanumérico y el numérico.

7.1.6. Incremento 6: Implementación del juego Ojo de Cerradura y Remodelación de Menús

¿El juego carga bien los escenarios con las transparencias? ¿El juego genera un número correcto según el nivel? ¿El movimiento del número es adecuado y natural? ¿Se evalúa correctamente la solución aportada por el jugador? ¿El despliegue de botones se realizaba de manera correcta?

El juego carga satisfactoriamente todos los escenarios comprobando que la cerradura posee la transparencia como característica.

El número es cargado correctamente de modo que en niveles pares aumenta el número de dígitos y en los impares aumenta la velocidad con la que se desplaza las cifras.

Esta prueba fue la más dificil de superar, ya que el juego en un principio mostraba movimientos bastante entrecortados y al poco tiempo el sistema operativo se mostraba inestable y probocábamos que no pudieramos salir del juego ni reiniciar el sistema operativo. Después de mucho esfuerzo conseguimos realizar el movimiento correctamente liberando espacio en memoria y eliminando el fondo de detrás de la cerradura, es decir sólo se rellena la pantalla de gris y se dibuja el escenario tranparente sobre él. Esta medida logró que el rendimiento aumentara de manera muy significativa.

Las soluciones aportadas por el jugador se evaluaban correctamente desde un principio, sólo tuvimos que variar que el usuario no pudiese escribir los dígitos mientra los estaba viendo, lo que hacía que resultara tremendamente fácil de completar todos los niveles. Obligamos al usuario a esperar a que desapareciera el número de la pantalla para que pudiese introducir su solución.

Debido a problemas de espacio decidimos incluir esta funcionalidad, que resultó muy satisfactoria y pasó la prueba sin ninguna dificultad.

7.1.7. Incremento 7: Implementación del juego Acuérdate de Mí

¿El juego carga correctamente todos los personajes en la pantalla? ¿Se completa la segunda fase del juego, en donde desaparece uno de ellos? ¿Se capta correctamente la solución aportada por el usuario?

El juego, después de unos ajustes propios de las posiciones donde se mostraban en la pantalla, pasó la prueba de manera satisfactoria.

El sistema después de completar el tiempo marcado por un objeto reloj, hace desaparecer a un número de personajes propio del nivel. Después de las pruebas realizadas a jugadores externos al proyecto y a petición de ellos, varié el tiempo de memorización ya que parecía ser muy escaso, a la vez que aporté un segundo de bonus cada vez que aumentamos de nivel.

El sistema muestra todos los personajes y capta correctamente los que son pulsados.

7.1.8. Incremento 8: Implementación del control de perfiles

¿El sistema crea correctamente el fichero de perfiles? ¿El juego carga correctamente la pantalla de carga de perfiles, así como los distintos nombres de los perfiles? ¿El sistema registra correctamente el nuevo perfil? ¿El sitema actualiza correctamente la puntuación de un determinado juego y escribe en el fichero la nueva puntuación? ¿El sistema borra un determinado perfil?

El sistema crea correctamente el fichero de perfiles, comprueba si no existe el fichero lo crea y si existe lo lee y almacena los datos en memoria. No se observan fallos al realizar esta prueba.

El juego carga correctamente la pantalla de carga de perfil, así como sus nombres, aunque observamos que si se incluyen espacios en blanco en el nombre del perfil el nombre se separa por el espacio y crea un falso registro con la otra otra parte del nombre, decidimos eliminar cualquier espacio, ya que observando

cualquier juego de las mismas características no suelen permitir espacios en blanco.

A simple vista el juego crea correctamente el perfil con cualquier tipo de nombre, pero al hacer las pruebas con usuarios reales vemos que al intentar pulsar sobre la tecla *"shift"* el sistema deja de captar teclas y se genera un perfil con un nombre extraño. Para solucionarlo averiguamos que al pulsar cualquiera de estas teclas se guardaba un "0" en la variable que almacenaba el nombre (dicha variable almacena los códigos unicode propios de cada carácter) y que ello provocaba un error, al evitar la introducción de este extraño 0 logramos que funcionen tanto las letras mayúscula como las minúsculas, así como la pulsación de la tecla *"bloq. Mayúsculas"* y otras de la misma naturaleza. Otro de los fallos que localizamos es que si pulsabamos *"Enter"* sin escribir nada, se creaba un perfil con caracteres extraños al igual que antes, para ello comprobamos antes de registrar el nuevo perfil que el nombre no haya quedado vacío.

El sistema guarda correctamente todas las puntuaciones en el fichero correspondiente al perfil activo, sólo nos encontramos con el caso de que si una puntuación correspondía a tiempo deberemos guardar el tiempo menor y no el mayor, por tanto deberemos tratar a este tipo de puntuación como un caso aparte. También nos encontramos con el error de que si en un momento, en juegos de tiempo, elegimos abortar el transcurso del juego, la puntuación que devolvía el juego era 0, por tanto lo almacenaba en el perfil como un tiempo menor al que poseíamos. Para solucionarlo evitamos que evalúe como puntuación válida al 0, ya que en ningún caso lograremos completar un juego en menos de un segundo.

El sistema logra borrar el pefil activo sin ningún tipo de problema.

7.1.9. Incremento 9: Implementación de Luces y Sombras, Aritmética, operadores y Arcade

¿El sistema lee los ficheros de mapas y carga el tablero de Luces y Sombras? ¿El sistema realiza los cambios de apagado/encendido de luces? ¿El sistema genera correctamente la cuenta de Aritmética y Operadores? ¿El sistema genera una puntuación correcta en Arcade?

El sistema no presenta problemas al cargar y dibujar el tablero en ninguno de los casos y niveles dispobibles.

El sistema tampoco presenta problemas graves al realizar las operaciones de apagado/encendido de luces, se insta a forzar al error, comprobando las luces de los bordes y no se presentan fallos de segmentación y funcionan correctamente.

El sistema genera bien las operaciones, sólo nos encontramos problemas al generar pseudoaleatoriamente el operador segundo como 0, ya que nos encontramos con errores de tipo número entre cero. Eliminado este caso, no encontramos más fallos significativos, ni en **Aritmética** ni en **Operadores**, debido a su similitud.

El modo de juego **Arcade** genera una puntuación totalmente correcta y afín a los criterios creados.

7.2. Estadísticas

En esta sección mostraremos los resultados obtenidos en los distintos juegos por diversos jugadores que se prestaron a realizar las pruebas de OpenBTrainer. El estudio se ha realizado sobre 15 individuos de edades comprendidas entre 20 y 50 años. Mostraremos unas gráficas con los resultados obtenidos por cada jugador y analizaremos estos resultados, ello nos servirá para averiguar el nivel de dificultad que

dispone cada juego.

Hay que reseñar que se brindó a los jugadores la oportunidad de realizar las pruebas como máximo dos veces, por si la primera vez no fueron capaces de entender la dinámica del juego y por tanto no hubiesen podido explotar todo su potencial.

7.2.1. Resultados Memorión

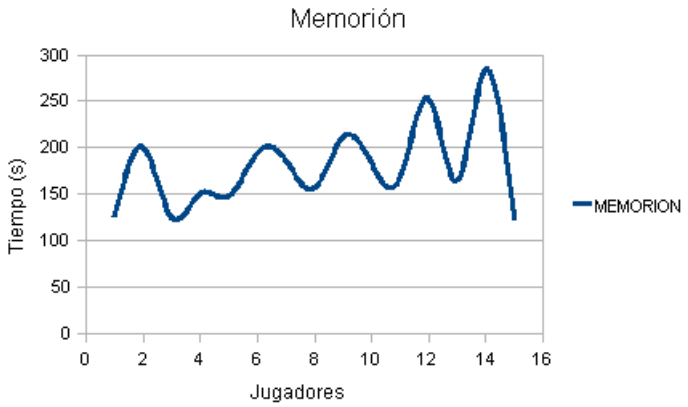

Figura 7.1: Gráfica de Resultados Memorión

En la anterior gráfica podemos ver la tendencia que suelen mantener los usuarios que juegan a **Memorión**. Estudiando la media podemos demostrar que los resultados giran entorno a los 177,87 segundos, algo menos de 3 minutos. Cerca del 80 % del tiempo empleado transcurre en el tercer nivel y dentro de ese 80 % podemos observar que más de la mitad del tiempo se tarda en localizar las dos o tres primeras parejas. Una vez que hemos pasado esa fase del juego el usuario completa el nivel en muy poco tiempo. Prácticamente a todos les agrada este juego, ya que les resulta familiar y sobre todo comentan la dificultad de diferenciar picas de tréboles, lo que supone un reto.

7.2.2. Resultados Diferencias

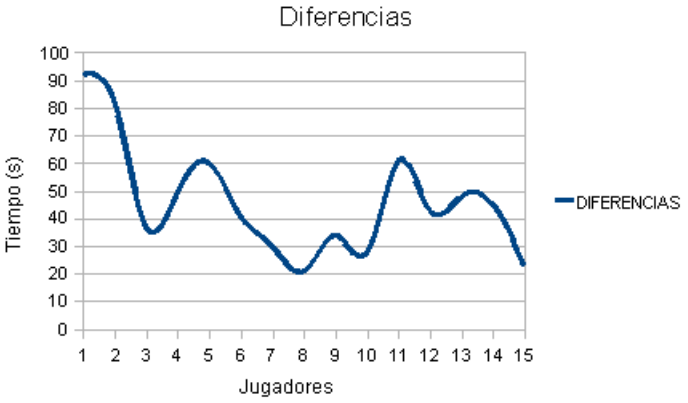

Figura 7.2: Gráfica de Resultados Diferencias

Primeramente cabe destacar que las pruebas se realizaron sobre la misma imagen para que los resultaros sólo dependieran del jugador y no de la imagen que haya aparecido. Después de este inciso podemos observar que la gráfica no es muy concluyente, ya que los resultados varían mucho según la habilidad del jugador y la rapidez visual de éste. Podríamos quedarnos con que la mayor parte de los jugadores completaron la imagen en unos 46,27 segundos, dato obtenido de realizar la media a todos los resultados. Por tanto no es un juego que suponga una dificultad demasiado grande ya que gran parte de los usuarios completan el nivel sin muchos problemas. Las opiniones giran entorno a las imágenes en las cuales algunas diferencias son complicadas de ubicar y otras demasiado obvias.

7.2.3. Resultados Recuerda Dígitos

Figura 7.3: Gráfica de Resultados Recuerda Dígitos

Los resultados obtenidos en este juego son los más claros de todos. La media se sitúa en 7 niveles completados, con valores que oscilan entre 5 y 9. Por tanto podemos ver que hay un salto enorme entre las siete, ocho y nueve cifras, es algo curioso ver que una cifra más resulta casi imposible de memorizar. Además de esta dificultad se suma la fatiga mental que producen el paso de niveles, seguramente si esta prueba la hicieramos a personas que hubiesen entrenado durante un mes aproximadamente comprobaríamos que se llegaría sin problema al nivel 9 o incluso alguno más dependiendo de las facultades mentales de cada persona. Con respecto a la opinión del juego, parece que tiene bastante éxito ya que los propios usuarios se ven capaces de memorizar los dígitos que componen un número de teléfono en algo más de un segundo. Por supuesto para este juego usamos la memoria a corto plazo y en pocos segundos esta cifra desaparecerá de nuestra mente.

7.2.4. Resultados Observación

Figura 7.4: Gráfica de Resultados Observación

La gráfica que obtenemos es también bastante clara, ya que la media se sitúa en 4 niveles y con resultados que oscilan entre los 3 y 6. Por ello podemos ver que el juego **Observación** parece que resulta algo complejo, ya que a los jugadores le produce gran esfuerzo pasar del cuarto nivel. Estos resultados son provocados por realizar las estadísticas con jugadores no entrenados, ya que si se hubiese jugado más se sabría que hay bastantes estrategias que pueden hacer que pases de nivel sin demasiado esfuerzo. Una de esas estrategias podría ser recordar las flechas repetidas y obviar las demás por tanto si preguntan por alguna flecha que no es una de las recordadas seguramente la respuesta será uno, aunque esta estrategia puede romperse si aparece una flecha que no haya aparecido. La opinión de los jugadores no es demasiado favorable debido a la dificultad, estos comentarios se explican a la necesidad por parte de los jugadores de ganar, normalmente al diseñar un videojuego se tiene bastante en cuenta esta cuestión, pero en juegos de tipo de desarrollo mental no se cumple demasiado.

7.2.5. Resultados Ojo de Cerradura

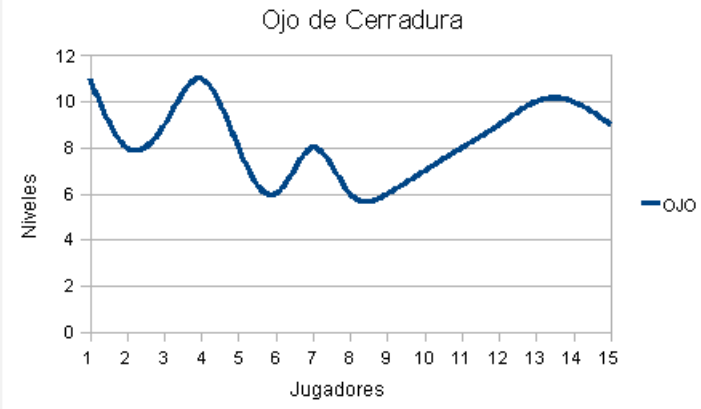

Figura 7.5: Gráfica de Resultados Ojo de Cerradura

Los resultados obtenidos se pueden observar claramente en una gráfica sin demasiados picos, esto significa que la mayoría de usuarios no han obtenido unos resultados demasiado distintos. La media se sitúa

en el nivel 8, esto se debe que apartir de dicho nivel, el juego se complica bastante, sólo algunos de ellos han sobrepasado esta barrera. Nos encontramos en este juego con dos factores, la dificultad que supone la prueba y las limitaciones hardware que nos aporta las pantallas TFT y LCD de varios años, una vez que el número adquiere una velocidad los números se van distorsionando y provocan un factor de dificultad costoso de superar. Las opiniones de los usuarios son bastante positiva ya que resulta un juego muy original y que llama rápidamente la atención del jugador.

7.2.6. Resultados Acuérdate de Mí

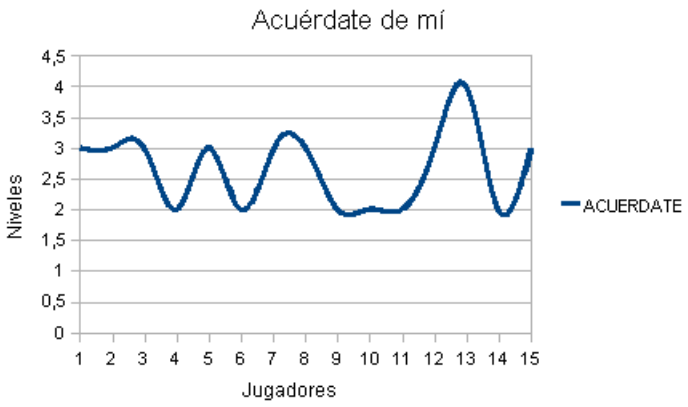

Figura 7.6: Gráfica de Resultados Acuérdate de Mí

La gráfica obtenida nos aporta buena información ya que la mayor parte de los usuarios no pasan del nivel 3, posición en la que se sitúa la media. Esto es debido a que este juego ejercita la memoria a medio-largo plazo, lo que supone una dificultad ya que posiblemente este tipo de memoria se encuentra bastante poco usada. Pues prácticamente las cosas que necesitamos memorizar, en la vida cotidiana, como números de teléfono o listas de la compra, se apuntan en un papel o en la memoria de algún dispositivo electrónico. Con respecto a las opiniones de los usuarios, resulta un gran éxito entre los usuarios consultados, ya que es un juego muy simpático y que está bastante bien para ejercitar la mente con un juego bastante distraído, gran parte del éxito del juego se basa en las imágenes elegidas, ya que rompe la monotonía de memorizar dígitos y otros objetos.

7.2.7. Resultados Aritmética

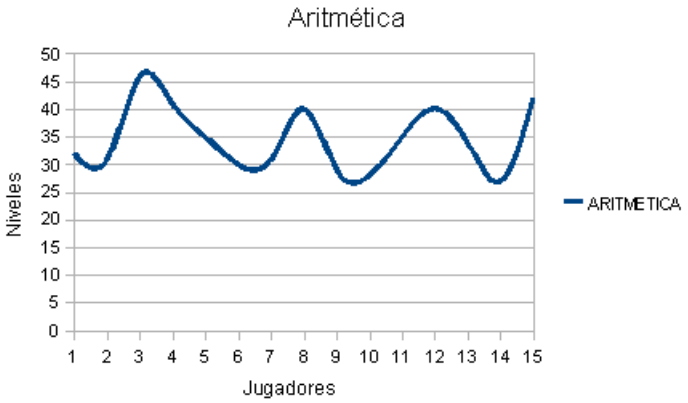

Figura 7.7: Gráfica de Resultados Aritmética

En la anterior gráfica podemos ver los resultados que hemos obtenido al realizar el estudio, podemos ver que hay mucha variación entre los 28 niveles del peor resultado y los 46 del mejor. La media la situamos en los 34 niveles completados, lo que nos lleva a pensar que no resulta un juego demasiado difícil. La gran variación de resultados obtenidos se debe a la costumbre de realizar operaciones mentales y no mediante dispositivos electrónicos, ya sean móviles, ordenadores o calculadoras. Por tanto en una persona que acostumbra a realizar los cálculos sencillos mentalmente, obtendremos unos resultados bastante buenos y en otras personas que no realizan cálculos le costará realizar las distintas operaciones. Las opiniones de los usuarios tampoco tienen mayor importancia, ya que resulta un juego bastante común y los jugadores no comprueban un cambio demasiado grande.

7.2.8. Resultados Operadores

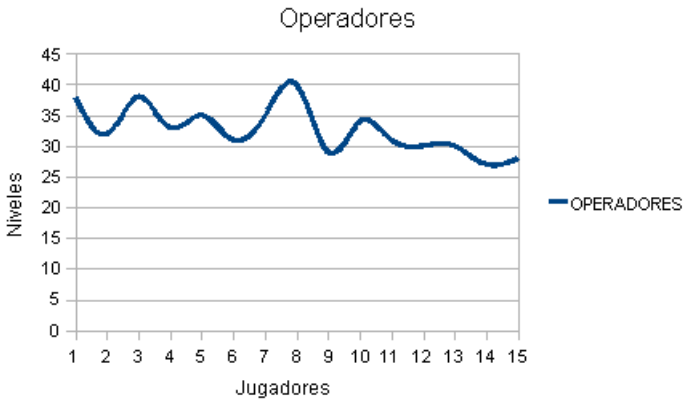

Figura 7.8: Gráfica de Resultados Operadores

En la gráfica que hemos obtenido vemos que, hemos obtenido unos resultados bastante igualados y la media la situamos en torno a los 32 niveles superados. Cualquiera se extrañaría que al ver un juego bastante similar al anterior, los resultados no sigan el mismo patrón, esto se explica por que el juego se ha probado en un ordenador portátil y sin teclado numérico, lo que nos obliga a usar el ratón, lo que

puede provocar una penalización de tiempo mientras localizas el botón en la pantalla y dirigir el puntero a la posición del botón. En el anterior caso sólo debíamos pulsar una tecla del teclado. La opinión del juego es buena y se basa en la que obtuvimos en el anterior juego.

7.2.9. Luces y Sombras

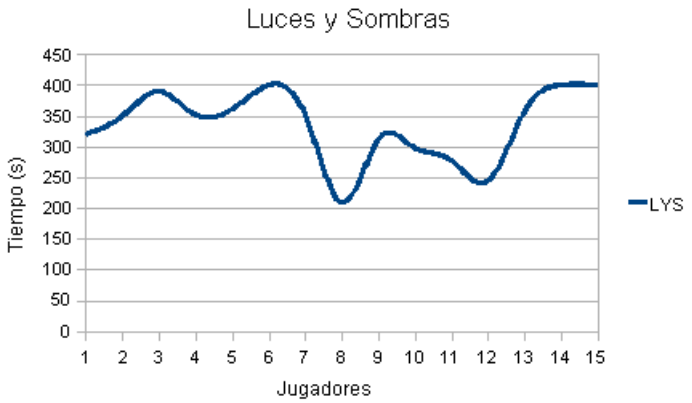

Figura 7.9: Gráfica de Resultados Luces y Sombras

La gráfica anterior puede ser la que más tienda a errores, ya que el juego luces y sombras, al contrario de los demás tiene un proceso de aprendizaje lo que provoca que el usuario le cueste trabajo adaptarse y perder mucho tiempo probando las estrategias que se deben utilizar para completar el nivel. Hay que reseñar que el 80 % de los jugadores necesitaron ayuda para completar el último nivel y por tanto los resultados no son demasiado válidos. Lo que nos indica que el juego necesita de jugadores expertos y demuestra la variedad de juego de OpenBTrainer que ofrecen experiencias distintas a cada jugador. La opinión a pesar de ser un juego bastante difícil es bastante buena debido a que el usuario ve que poco a poco va entendiendo mejor el juego y usa mejor la lógica para completar cada nivel.

7.2.10. Arcade

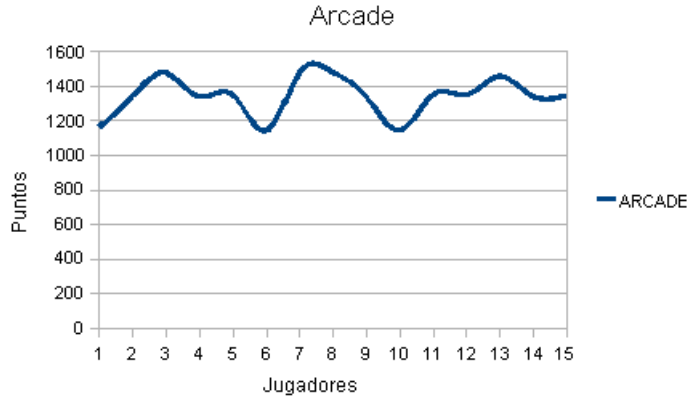

Figura 7.10: Gráfica de Resultados Arcade

Los resultados del modo de juego **Arcade** se basa en las distintas habilidades y puntos fuertes de los usuarios, normalmente personas con un buen nivel de rapidez visual y de aritmética consigue muy buenos resultados en la prueba. La media obtenida de los resultados registrados se sitúa entorno a los 1339 puntos. La opinión de los usuarios de esta prueba es buena, ya que provoca que los jugadores se esfuercen por practicar las pruebas que le provocan mayor esfuerzo y que se le dan peor.

Capítulo 8

Conclusiones

8.1. Posibles Ampliaciones

En futuras versiones el proyecto podría tener una expansión bastante importante, una de las numerosas ideas que se podía implantar serían:

1. Reconocimiento por voz. Podría implantarse un sistema de reconocimiento de voz para, con ayuda de un micrófono, se pudiera responder a las distintas pruebas mediante la voz.

2. Creación de exámenes personalizados. Podrían crearse pruebas exclusivamente en forma de exámenes que los usuarios debieran afrontar para que pudiesen avanzar en el juego.

3. Creación de una comunidad web. Como hablamos en otros capítulos se podría crear una comunidad web entorno a OpenBTrainer con la idea de que muchos desarrolladores trabajarán juntos para hacer crecer el proyecto y que se fueran incrementando el número de juegos y pruebas que existen actualmente en el proyecto.

4. Control de perfiles más avanzado. Se podría usar una pequeña base de datos en el que los usuarios debieran introducir varios datos personales para que el juego se comportarse de una forma distinta según la edad del jugador por ejemplo.

5. Inclusión de distintas categorías. Como es el caso de categorías musicales, esta categoría se está poniendo de moda con la inclusión de nuevos juegos sobre todo para nintendo DS, como es el caso de Rhythm Paradise, en el que se entrena la capacidad musical y el ritmo.

6. Inclusión de la posibilidad de crear retos online con otros jugadores para ver quien consigue más puntuación en un determinado juego.

8.2. Opinión final del proyecto realizado

Llegado a este punto debemos realizar un pequeño resumen sobre las conclusiones que sacamos después de haber realizado tanto esfuerzo para realizar el proyecto OpenBTrainer.

Habría que destacar que el proyecto OpenBTrainer necesita aún varios juegos para hacer que el juego pueda extenderse y llegar al público. Esto lo dejamos en el aire por si algún compañero quiere retomar el proyecto y expandirlo, desde este punto animo a cualquier compañero a seguir con el juego, es un proyecto bastante gratificante y que fácilmente ves resultados y con trabajo y esfuerzo poco a poco el proyecto cada vez queda más preparado. Por supuesto me presto a ayudar a todo aquel que necesite

ayuda para seguir con el proyecto y del mismo modo adjunto en los apéndices un manual para el desarrollador que nos realizara un pequeño paseo por el código realizado y los primeros pasos que deben tomar para que cualquier persona cree sus propios juegos.

Con respecto a la calidad del proyecto creo que el resultado es bastante bueno, un proyecto sólido, en el que encontraremos pocos fallos, en gran parte gracias a todas las pruebas que hemos hecho a usuarios de todo tipo, que es el momento en el que observas los bugs más importantes.

Resulta muy gratificante haber utilizado *GIMP* para todo el apartado gráfico, ya que es una herramienta muy completa y una vez que aprendes a usarlo es muy útil para cualquier aspecto gráfico, ya sea recortar, editar o crear imágenes.

Ha sido muy interesante usar un entorno de programación como es el caso de **NetBeans**. Mi opinión se basa en la posibilidad de tener a mano todos los ficheros de código en un simple click, pero sobre todo me he decantado por este tipo de entornos por su sistema de análisis de sintaxis y de autocompletado que poseen. De esta manera conseguimos solucionar el 90 % de los fallos de compilado, son los fallos de falta de *;*, falta de alguna llave (*}*), etc...

Otra de las herramientas usadas son las distintas librerías que forman **libSDL**. Estoy muy contento de haber conseguido un nivel bastante alto en el uso de este tipo de librerías, ya que facilitan enormemente la tarea de realizar juegos sencillos. Por tanto el uso de estas librerías las recomiendo a cualquiera ya que con un poco de nivel en C/C++ pueden llegar a conformar pequeños juegos con una serie de códigos que podemos encontrar por ejemplo en el gran trabajo realizado por mi compañero *Antonio García Alba*, el **Tutorial Wiki libSDL** [9].

Con respecto a lo que he aprendido. Pienso que un proyecto fin de carrera, sobre todo un proyecto de tipo software, te ayuda a aprender a organizar de forma muy eficiente el tiempo del que dispones. Gracias a herramientas como *Planner* con sus diagramas de Gantt es más fácil e intuitivo ver las tareas que tienes pendiente y el tiempo estimado que dispones para realizarlas. Se puede aprender también que cualquier imprevisto que pueda ocasionar no es una pérdida de tiempo, si no una oportunidad para ver que las cosas pueden hacerse mejor, y que desarrollar con prisas te puede ocasionar que aparezcan numerosos bugs que al final te harán perder más tiempo todavía.

El uso de las metodologías de desarrollo que nos aporta la Ingeniería del Software es muy buena idea, ya que nos ha aportado la oportunidad de tener un proyecto cada vez más completo y siguiendo un orden. Para ello también resulta muy útil el uso de los diagramas de Gantt, ya que te ayudan a hacerte una idea del tiempo que se dispone, aproximadamente, para la realización de una tarea. Una vez finalizado el proyecto, podemos decir que hemos cumplido con todas las especificaciones que citamos en capítulos anteriores e incluso hemos creado alguna característica que no habíamos pensado en un principio.

Otra de las cosas que se aprende enormemente es a usar la ayuda que puede prestarte cualquier persona de tu entorno, escuchar sus opiniones, a saber tratarlas y acatar sus consejos en la medida de lo posible, este puede ser uno de los puntos más fuertes en el aprendizaje que he obtenido en estos meses de trabajo. Los desarrolladores pueden estar tan involucrados en el proyecto que no ven realmente los deseos y necesidades del público que realmente es hacia el que va a ser dirigido el juego.

Apéndice A

Manual de Usuario

A continuación detallamos las instrucciones básicas que un usuario normal debe saber sobre el proyecto OpenBTrainer.

A.1. Ejecución

Es posible la ejecución del programa en cualquier entorno GNU/Linux. Una vez finalizado la instalación del producto simplemente tendremos que escribir en una consola:

```
./openbtrainer
```

Una vez ejecutado el comando anterior veremos la pantalla inicial de selección de perfiles.

A.2. Primeros Pasos

Si es la primera vez que ejecuta OpenBTrainer desde su instalación, el sistema creará los ficheros de perfiles vacíos, por tanto verá una pantalla como la siguiente:

Figura A.1: Captura de la pantalla Inicial de OpenBTrainer

A continuación deberá pulsar en cualquiera de las cuatro posiciones que dispone. Una vez que ha elegido una de las posiciones donde desea que el perfil se guarde, el sistema le pedirá que introduzca el nombre del perfil. No introduzca espacios en blanco, ya que el sistema no los permite y dispone de 20 caracteres.

Una vez completado la tarea de completar el perfil usted verá una pantalla como la siguiente, suponemos un registro con el nombre de Prueba.

Figura A.2: Captura de la pantalla Principal

Como puede verse en la imagen anterior tenemos varios botones que nos darán la posibilidad de realizar varias acciones que ahora citaremos. Comenzamos con las funcionalidades que le ofrece el sistema de perfiles.

A.2.1. Borrar Perfil

En caso de que desee eliminar un perfil de forma permanente, pulse en el botón. Debe tener cuidado ya que en caso de error sus puntuaciones pueden eliminarse.

A.2.2. Cambiar Perfil

En caso de que desee volver a la pantalla de selección de perfil use el botón. En el momento de que salga a la pantalla de selección de perfiles todos los datos quedarán correctamente guardados, es decir, no tiene que guardar los datos de manera manual.

A.2.3. Arcade

Este modo de juego consiste en que el sistema le ofrece un juego de cada categoría de forma consecutiva. Deberá intentar conseguir la mayor puntuación posible en cada juego para batir sus propios récords, cuando finalice todos los juegos el sistema le mostrará la puntuación obtenida y si consiguió batir su propia marca.

A.2.4. Categorías

En caso de que desee jugar algún juego específico, esta es su elección idónea. Se le mostrará en una pantalla todas las categorías existentes en OpenBTrainer. Cuando haya elegido la que desee pulsa en el botón correspondiente y elija uno de los juegos que se ha desplegado al pinchar en la categoría.

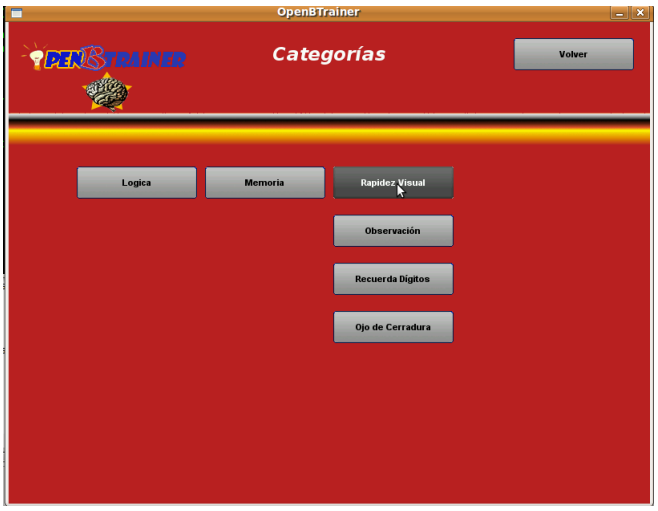

Figura A.3: Captura de la pantalla Categorías

Elija el juego y pinche sobre él, el juego arrancará y podrá disfrutar de lo que le ofrece, una vez finalizado el mismo, el sistema le mostrará la puntuación obtenida, siempre teniendo en cuenta la puntuación almacenada en el perfil.

A.2.5. Minijuegos

En caso de que lo que quiera es distraerse y sin demasiado esfuerzo mental puede optar entre los dos juegos que existen en la categoría Minijuegos, Memorión y Diferencias. De la misma manera que lo hemos hecho antes elija el juego que desee y pinche sobre el botón.

Figura A.4: Captura de la pantalla Minijuegos

A.3. Juegos de OpenBTrainer

A continuación pasaremos a explicar la forma de jugar a cada una de las pruebas existente dentro del proyecto OpenBTrainer.

A.3.1. Recuerda Dígitos

Nos encontramos en la categoría de **Rapidez Visual**, y como su propio nombre indica lo importante es tener el sentido de la visión muy atento y la memoria muy fina.

En el momento que lanzamos el juego aparecerá un número en la parte central de la pantalla, deberemos recordarlo ya que a los dos segundos después de su aparición, se eliminará y ahora el sistema nos pedirá que tecleemos el número que hemos visto. Si el resultado es correcto avanzaremos de nivel y a continuación nos enfrentaremos a un número con una cifra más, y así respectivamente. En el momento que fallemos en algún dígito el juego terminará y mostrará el resultado del nivel máximo alcanzado.

Ver figura 3.4.

A.3.2. Observación

El siguiente juego que nos ocupa es el juego **Observación**. Un juego en el que aunque al principio cueste realizarlo, con un poco de práctica se pueden llegar a unos resultados sorprendentes.

Al principio de arrancar el juego, nos aparecerán unas flechas de dirección que apuntarán hacia una posición determinada durante un segundo, el objetivo es memorizar en el segundo todas las direcciones de las flechas. Una vez que finaliza el tiempo, las flechas desaparecerán y el sistema nos preguntará por la cantidad de flechas de un tipo determinado, posiblemente lo más complejo es memorizar las flechas sin saber por cuál te va a preguntar el juego.

El juego tiene siete niveles, en el que cada vez aparecerán más flechas y habrá que hacer un esfuerzo mayor para conseguir, de un vistazo, memorizar todos los elementos.

Ver figura 3.5

A.3.3. Ojo de Cerradura

Sin salirnos de la categoría de **Rapidez Visual** nos encontramos con un nuevo juego. **Ojo de Cerradura**, que nos hará creer que estamos delante de una puerta misteriosa y debemos ver por el ojo de la cerradura y recordar los códigos númericos que nos van a ir apareciendo.

El juego comienza con un número de dos cifras que se irá deslizando por el escenario, dejándose ver solamente por la zona de la cerradura. Dicho número tendrá una velocidad constante. En el momento que el número desaparezca deberemos esperar a que el sistema nos avise de que es el momento adecuado para escribir el número que acabamos de ver.

A medida de que lo vayamos contestando correctamente el nivel irá aumentando y la dificultad será mayor. En un nivel aumentará la velocidad con la que pasa el número por la pantalla y en el siguiente lo que aumentará será el número de dígitos que aparecerán. En el momento de que se falle un dígito, el juego finalizará y el sistema mostrará el nivel máximo alcanzado.

Ver figura 3.7

A.3.4. Acuérdate de Mí

Dentro de la categoría de **Memoria**, nos encontramos con el juego **Acuérdate de Mí**, en el que tendremos la difícil tarea de memorizar los personajes que nos aparecerán en un tiempo determinado.

En el momento de lanzar el juego nos aparecerán una serie de personajes apilados, abajo a la derecha nos encontramos con una cuenta atrás. Deberá memorizar todos los personajes de la pantalla en los segundos que restan, ya que cuando finalice el tiempo uno de los personajes se perderá. Una vez que el personaje haya desaparecido, se dibujarán en la pantalla todos y cada uno de ellos, deberemos señalar cuál de ellos es el que falta.

Un aspecto importante es la etiqueta que tenemos a nuestra izquierda que nos indican cuantos fallos se nos permiten hasta que el juego finalice. Inicialmente son 2, es decir, a la tercera vez que nos equivoquemos el juego acabará. Por tanto deberemos guardar bien de no equivocarnos en exceso, ya que los fallos pueden sernos útiles en niveles superiores, en donde el número de personajes que desaparezcan será mayor, al igual que el tiempo que tenemos para memorizar, que se aumentará cada nivel un segundo, para que prestemos más atención.

Ver figura 3.8

A.3.5. Memorión

En el apartado de **Minijuegos** nos encontramos con el juego del **Memorión**, basado en el clásico juego de las parejas, en el que iremos volteando cartas de dos en dos hasta encontrar todas las parejas existentes en la mesa.

En el juego nos encontraremos tres niveles, cada uno con más cantidad de cartas que el anterior. Deberemos encontrar todas las parejas escondidas. Nos encontramos con una baraja francesa con números del 7 al 10 de los cuatro palos de la baraja. Nuestro objetivo es completar todas las parejas que existen en la mesa.

El juego una vez que hemos encontrado todas las parejas del tercer nivel, acabará y devolverá los segundos que hemos tardado en completar los niveles.

A.3.6. Diferencias

El último de los juegos de la categoría de **Minijuegos** es Diferencias, se trata de un juego en el que debes de encontrar todas las diferencias que existen entre dos imágenes semejantes.

El juego de manera aleatoria carga dos imágenes, deberemos observarlas bien y en el momento que encontremos una diferencia, marcarla en la imagen de la derecha. Deberemos estar atentos ya que puede que haya diferencias muy pequeñas y que nos resulte un poco difícil de encontrar. Debemos fijarnos en el rótulo que nos aparece abajo, donde nos indica el número de diferencias totales que deberemos de buscar.

En el momento de que hayamos acabado todas las diferencias el juego terminará y nos apuntará en nuestro perfil el tiempo que ha tardado en completar la figura.

A.3.7. Aritmética

Nos encontramos con un juego de la categoría de **Cálculo**, llamado **Aritmética**, en el que tendremos que completar una serie de cálculos aritméticos.

Una vez que lanzamos el juego, nos aparecerá la primera cuenta, y rápidamente podemos observar que nos falta el segundo operando, nosotros mediante el teclado, deberemos indicarle al sistema el número que falta. En caso de que hayamos acertado nos aparecerá otra y así sucesivamente.

Por tanto el objetivo del juego será completar el mayor número de cuentas en un minuto y medio de tiempo, luego el sistema nos mostrará si hemos batido nuestro propio récord.

Ver figura 3.11

A.3.8. Operadores

Seguimos en la categoría de **Cálculo** y nos encontramos con el juego **Operadores**, un juego similar al anterior pero con la salvedad de que deberemos indicar el operador que falta en la operación.

De manera similar al anterior juego, nos aparecerán cuentas y deberemos solucionarlas indicando el operador correspondiente. Para seleccionar el operador existen dos alternativas, usando el ratón y pulsando sobre los botones que aparecen en negro, o usando los operadores que tenemos en el teclado numérico.

Deberemos completar todas las cuentas posibles en el minuto y medio de tiempo. El sistema nos indicará si hemos batido el récord o deberemos intentarlo de nuevo.

Ver figura 3.11

A.3.9. Luces y Sombras

Cambiamos de categoría y nos vamos a la categoría de **Lógica** y nos encontramos con el juego **Luces y Sombras** en el que nos aparecerán unas luces encendidas y deberemos apagarlas.

El juego comienza en el primier nivel y nos aparecen unas luces, nuestro objetivo es apagarlas, pero teniendo en cuenta un determinado criterio. Al seleccionar usando el ratón una de las luces, se encenderán o apagarán todas las consecutivas, es decir, la de arriba, abajo, derecha e izquierda. Por tanto, necesitaremos pensar bien nuestros movimientos para que no nos quede ninguna encendida. Si llegamos a un punto en el que deseamos volver a empezar el nivel, podremos usar el botón *Resetear* situado arriba a la derecha.

El juego posee cuatro niveles que deberemos superar lo más rápido posible, ya que es la única forma de batir nuestro récord.

Ver figura 3.10

A.3.10. Arcade

Realmente **Arcade** no es un juego en sí, sino un modo de juego formado por cuatro juego, uno por cada categoría existente en OpenBTrainer. Puede considerarse como un pequeño examen, en el que deberemos sacar la máxima puntuación posible en base a los siguientes criterios:

1. Juego **Recuerda Dígitos**, por cada nivel obtenido, tendremos 100 puntos.
2. Juego **Acuérdate de mí**, por cada nivel obtenido, tendremos otros 100 puntos.
3. Juego **Aritmética**, por cada cuenta resuelta obtendremos 10 puntos.
4. Juego **Luces y Sombras**, por cada segundo que tardemos en completar el nivel obtendremos una penalización de 3 puntos.

Apéndice B

Manual de Instalación

B.1. Obtener OpenBTrainer

En primer lugar deberemos obtener una versión de OpenBTrainer. Lo podemos encontrar en la página del proyecto en la forja de rediris [10] o haciendo un chekout mediante subversion de la siguiente manera:

```
svn checkout https://forja.rediris.es/svn/openbtraining
```

B.2. Instalación

Si hemos optado por bajarnos el paquete comprimido de la página del proyecto, primero deberemos descomprimirlo, podremos hacerlo fácilmente haciendo:

```
tar -xvzf openbtrainer-v1.0c.tar.gz
```

Una vez que hemos descomprimido el juego o hemos optado por usar subversion, tenemos que compilar el proyecto. En tal caso deberemos entrar en la carpeta src.

```
cd openbtrainer/src/
```

Luego para compilar simplemente haremos “make”.

```
make
```

Si la compilación ha tenido éxito nos generará un fichero ejecutable llamado openbtrainer, por tanto sólo tendremos que ejecutarlo.

```
./openbtrainer
```

Apéndice C

Manual del Desarrollador

En primer lugar indicar que este manual está indicado para que cualquier desarrollador con unas nociones de programación en C/C++ y libSDL puedan desarrollar sus propios juegos y añadirlos a OpenBTrainer.

En segundo lugar para publicar cualquier cambio, realiza una petición de colaboración en la plataforma de la forja de Rediris y tendrá libre acceso para subir y modificar el proyecto OpenBTrainer mediante subversion.

C.1. Estructuración Básica del Juego

El juego posee una jerarquía bastante bien definida, a continuación iremos explicando el papel de cada módulo y fichero dentro del juego.

C.1.1. Main.cpp

El fichero **main.cpp** nos servirá para crear el objeto perfil, llamará a la pantalla de carga de perfil y una vez que le devuelva el control, cargará la pantalla principal, ver figura A.2. Una vez que entre en el main loop, main controlará los eventos del ratón, y comprobará la pulsación de los distintos botones. En caso de que se pulse los botones arcade, categorías o minijuegos le pasará el control al objeto juego, que realizará las operaciones pertinentes.

C.1.2. Juego

El módulo **Juego** es el más importante, ya que controlará todo el flujo del programa. Dentro de su constructor llamaremos e inicializaremos los distintos subsistemas que necesitemos y en la función correspondiente crearemos la variable pantalla, que será la que controle lo que imprima la pantalla en cualquier momento.

El módulo **Juego** dispone de varias funciones importantes:

- Categorías. Esta función será la que llame main si el botón categorías ha sido pulsado. Cargará la pantalla de elección de juego según la categoría, controla el despliego de botones que comentaremos más adelante, y además controla el lanzamiento de los distintos juegos.
- Minijuegos. Del mismo modo que el anterior si el botón Minijuegos se ha pulsado, se llamará a esta función. Cargará la pantalla de elección de minijuegos y lanzará el juego que el usuario elija mediante la pulsación de los botones.

- Arcade. Este modo de juego llamará a los distintos juegos que componen Arcade y realizará unos cálculos para averiguar la puntuación total obtenida.

C.1.3. Perfiles

Es el módulo de control de perfiles desde donde se controlará la carga de perfiles, el registro de nuevos perfiles, la actualización de puntuaciones, el borrado de perfiles y varias opciones interesantes que comentaremos en su momento.

C.1.4. Módulos de todas las categorías

En los distintos módulos de categorías se incluyen los juegos de acuerdo a la categoría en la que concuerdan. Así que si desea introducir un juego deberá introducir su código en alguna de las categorías existentes o crear una nueva.

C.1.5. Reloj

Este módulo es bastante útil para cualquier juego que necesite un control de tiempo ya que nos simplifica la tarea de tener que dibujar el tiempo en la pantalla y posee bastantes opciones, tanto para crear un reloj cronómetro o incluso para crear una cuenta atrás introduciendo un tiempo determinado.

C.2. Estructuración de Carpetas

Una vez que obtenemos OpenBTrainer disponemos de cinco carpetas distintas, comentaremos a continuación el contenido de las mismas.

doc Es la carpeta en la que se incluye esta memoria y los fuentes latex con la que se ha creado dicha memoria.

fuentes Es la carpeta en la que se incluyen las fuentes de texto de tipo TrueType[1], para que se impriman los rótulos en OpenBTrainer. Si se desea incluir nuevas fuentes o modificar las que posee simplemente incluya aquí su fichero de fuentes.

imagenes En esta carpeta nos encontramos con todas las imágenes que se incluyen en OpenBTrainer. Dentro de esta carpeta se incluyen otras seis.

- **botones** En esta carpeta se incluyen todos los botones del proyecto, son de tipo **.png* y están sacados de la página Buttonator [11].
- **cartas** Aquí se incluyen las imágenes de las cartas que usaremos en el juego de **Memorión**.
- **diferencias** Dentro se incluyen las imágenes que usa el juego de **Diferencias**, además se incluye el fichero *soluciones.dat* en el que encontramos todas las soluciones de las imágenes.
- **fondos** En esta carpeta incluimos todos los fondos que necesitamos para cualquier pantalla de OpenBTrainer.
- **sprites** En dicha carpeta incluimos las imágenes de los personajes usados en el juego **Acuérdate de Mí**.

[1] TrueType es un formato estándar de fuentes tipográficas escalables desarrollado inicialmente por Apple Computer a fines de la década de los ochenta oara competir comercialmente con el formato "Tipo 1" de Adobe. Una de las principales fortalezas de TrueType era que ofrecía a los diseñadores de fuentes un mayor grado de control sobre la forma en que los caracteres se desplegaban en pantalla o impresos a tamaños menores, con lo cual se lograba una mejor legibilidad

otros En esta carpeta incluimos todas aquellas imágenes que no podemos incluir en ninguna de las anteriores.

perfiles En esta carpeta nos encontramos con los ficheros de datos de los perfiles, la primera vez que obtengamos el juego, dicha carpeta estará vacía.

src Esta carpeta incluye todos los ficheros fuente del proyecto.

C.3. Control de Perfiles

Pasemos ahora a explicar más detenidamente el funcionamiento del control de perfiles. Una vez que consigamos entender bien cómo funciona el módulo de perfiles, lograremos usarlo con facilidad e incluso conseguir mejorarlo y adaptarlo a una base de datos o similar.

En la carpeta se crea la primera vez que se ejecuta el juego un fichero llamado *perfiles.sav*. La sintaxis del fichero es la siguiente:

```
Pepe
Nuevo
Nuevo
Nuevo
```

En este ejemplo de fichero podemos ver que cada línea equivale a un nombre de perfil, el primero de ellos, equivale al perfil *pepe* y los demás equivalen a perfiles vacíos.

Una vez que seleccionamos y creamos nuestro perfil, el programa generará un fichero con el número de posición del perfil y una terminación *.sav*. Un ejemplo de fichero de datos de un perfil puede ser el siguiente:

```
Pepe
<--->
0
0
4
5
5
0
0
0
200
31
```

En este fichero se puede comprobar que la primera línea es el nombre del perfil, la segunda línea es de separación y las restantes equivalen a la puntuación en distintos juegos, en base a la tabla siguiente:

Número de Línea	Juego
3	Memorión
4	Diferencias
5	Recuerda
6	Observación
7	Ojo de Cerradura
8	Acuérdate de Mí
9	Aritmética
10	Luces y Sombras
11	Arcade
12	Operadores

Tabla C.1: Concordancia de Líneas con Juegos en fichero de perfil

Por tanto debemos tener muy en cuenta este fichero, para comprobar que al introducir un juego nuevo, la puntuación se escriba correctamente.

Una vez que hemos visto cómo funciona los ficheros de control de perfiles, veamos ahora cómo funciona la actualización de puntuaciones. Esta función tiene una determinada sintaxis, que mostraremos a continuación:

```
Uint32 actualizarpuntuacion(Uint32 perfil, Uint32 numjuego, Uint32
    puntuacion, Uint32 tipopuntuacion, SDL_Surface *pantalla);
```

Como primer parámetro le pasamos el perfil actual, que podemos conseguir mediante una función existente, como segundo parámetro hay que pasarle el número de juego que equivaldría a la tabla C.3 menos dos, es decir, le restamos dos debido a que debemos restarle las dos posiciones equivalentes al nombre y a la línea separadora. En tercer lugar se pasa como parámetro la puntuación obtenida en el juego, como consejo y para ahorrarse una variable, se podría introducir el código para lanzar el juego, ya que éste siempre devuelve la puntuación obtenida. En cuarto lugar nos encontramos con un parámetro importante, en OpenBTrainer existen tres tipos de puntuación:

1. Puntuación tipo **Niveles**, es el caso de juegos en los que deberemos llegar al nivel máximo.
2. Puntuación tipo **Tiempo**, es el caso de juegos en los que deberemos completar los niveles en el menor tiempo.
3. Puntuación tipo **Puntos**, es el caso del juego Arcade, deberemos obtener la puntuación máxima.

El motivo de que existan tres tipos de nivel, es debido a que la función se basa en leer del fichero la puntuación en base a la posición que ocupa ésta, la compara y evalúa si, en el caso de encontrarnos en los puntos 1 o 3, hemos obtenido una puntuación mayor a la que teníamos registrada. En caso de que nos encontremos en el caso 2, debemos evaluar si el número de segundos, que es lo que se almacena en el perfil, es mayor del que hemos obtenido. Tiene mucha importancia destacar que en el perfil se guarda el número de segundos que hemos tardado en completar un juego, si tardamos un minuto y medio se almacenará en el perfil 90.

Por último se le pasa la variable pantalla para que aparezca la pantalla en la que te indica si has obtenido una puntuación mayor o menor de la que teníamos registrado.

Cabe destacar, además de estas funciones, toma gran importancia una constante que se define en el fichero *perfiles.h*:

```
#define NUMJUEGOS 10
```

Esta constante controla el número de juegos cuyas puntuaciones se introducirán en los ficheros de perfiles. Esto quiere decir que si estamos interesados en introducir un nuevo juego en OpenBTrainer, tendremos que cambiar esta constante.

C.4. Introducción de un Nuevo Juego en OpenBTrainer

La tarea de introducir un juego en OpenBTrainer es muy simple y trataremos de comentar los detalles más importantes para que cualquier desarrollador se preocupe más de su propio juego que el modo de adaptar el juego a OpenBTrainer. Trataremos la introducción del juego en varios pasos, que deberán ser cumplidos por cualquier programador.

C.4.1. Paso 1: Codificación del Nuevo Juego

El primer paso consiste en elegir una categoría para el nuevo juego, una vez elegida deberemos introducir una función cuya sintaxis debe cumplir en la medida de lo posible el siguiente criterio:

```
Uint32 NombreJuego(SDL_Surface * pantalla);
```

En nuevo juego debe devolver la puntuación obtenida y recibir como mínimo la variable pantalla para que así pueda dibujar los distintos objetos que conformará el juego.

Las funciones que puedan ayudar al código como funciones de dibujado de fondo, deberán incluirse en la parte privada de la clase, para evitar que puedan ser accesibles desde otro módulo del proyecto OpenBTrainer.

En la medida de lo posible se deberá hacer comentarios en los aspectos más complicados del código, así como documentar el juego usando para ello la herramienta *Doxygen*. Se deberá tener en cuenta el rendimiento del juego, que antes de comenzar, el usuario tenga claro de qué va el juego y cómo afrontar la prueba. Además todas los recursos que el juego necesite deberán cumplir los requisitos de licencia e incluirlos al proyecto en lugares y carpetas que anteriormente describimos.

C.4.2. Paso 2: Adaptar OpenBTrainer

Una vez que hemos terminado la codificación y deseamos incluirlo en el proyecto tendremos dos posibilidades, según la categoría utilizada:

Minijuegos

En caso de que hayamos desarrollado un minijuego, deberemos irnos a la función siguiente:

```
void juego::minijuegos(perfiles *perfil, SDL_Surface* pantalla){
```

Veamos este trozo de código:

```
      SDL_Surface* boton2 = IMG_Load("../imagenes/botones/Diferencias.png");
     if(!boton2)
     {
         cerr<<"Imposible cargar bitmap: "<< SDL_GetError()<<endl;
         exit(1);
     }
     rectbtn.x=500;
     rectbtn.y=240;
     SDL_BlitSurface(boton2, 0, pantalla, &rectbtn);
```

Debajo de este código crearemos un trozo similar a éste quedando de esta forma:

```
      SDL_Surface* boton3 = IMG_Load("../imagenes/botones/NuevoBoton.png");
     if(!boton3)
     {
         cerr<<"Imposible cargar bitmap: "<< SDL_GetError()<<endl;
         exit(1);
     }
     rectbtn.x=500;
     rectbtn.y=300;
     SDL_BlitSurface(boton3, 0, pantalla, &rectbtn);
```

Este trozo nos servirá para dibujar en la pantalla de selección de **Minijuego** el nuevo botón del juego a introducir. Como posición de la coordenada y deberemos darle la posición del anterior botón incrementándole 60 píxels, para seguir el mismo formato.

A continuación buscaremos el siguiente código:

```
if(event.type==SDL_MOUSEBUTTONDOWN)
{
  Uint32 mousex, mousey;
  mousex=event.button.x;
  mousey=event.button.y;
  if(mousex>500&&mousey>180&&mousex<(500+150)&&mousey<(180+40)){
    cout<<"Pulsacion de boton: Memorion"<<endl;
    perfil->actualizarpuntuacion(perfil->perfilactual(),1,minigame.memorion
        (pantalla),2,pantalla);
    goto carga;
  }
  if(mousex>500&&mousey>240&&mousex<(500+150)&&mousey<(240+40)){
    cout<<"Pulsacion de boton: Diferencias"<<endl;
    perfil->actualizarpuntuacion(perfil->perfilactual(),2,minigame.
       diferencias(pantalla),2,pantalla);
    goto carga;
  }
  if(mousex>630&&mousey>20&&mousex<(630+150)&&mousey<(20+40)){
    cout<<"Pulsacion de boton: Volver"<<endl;
    done=true;
  }
}
```

Este trozo de código cumple el papel de evaluar el evento mousebuttondown, es decir, si se ha pulsado un botón del ratón. Se almacenará la posición x e y donde se hizo click y luego se evaluará mediante los

ifs. Por tanto deberemos hacer que si se pulse en el nuevo botón que hemos introducido realice la tarea de lanzar el juego que acabamos de crear. Para ello deberemos introducir el siguiente código.

```
  if(mousex>500&&mousey>300&&mousex<(500+150)&&mousey<(300+40)){
    cout<<"Pulsacion de boton: NuevoJuego"<<endl;
    perfil->actualizarpuntuacion(perfil->perfilactual(),codigoNuevoJuego,
       minigame.NuevoJuego(pantalla),TipoPuntuacion,pantalla);
    goto carga;
  }
```

Un aspecto muy importante es crear el botón con un tamaño de 150 x 40 px, ya que si no se cumple esto el código anteriormente descrito no serviría. Más importante aún es realizar correctamente la llamada a la función actualizarpuntuacion del módulo de perfiles. El códigoNuevoJuego será uno más que el último juego que se ha creado, si no ha habido cambios desde que se creo el proyecto sería el 11. El tipoPuntuación dependerá de la naturaleza del juego y de cómo evalúa éste la puntuación, como comentamos anteriormente.

Categorías

En el caso de que nuestro juego pertenezca a cualquiera de las categorías que existen en el proyecto excepto **Minijuegos**, deberemos cumplir lo siguiente.

En primer lugar vemos que la pantalla de categorías dispone de un sistema de desplegado de botones cada vez que se pulsa en una de los botones de las categorías, esto hace que se complique un poco, pero no demasiado. Para ello deberemos desplazarnos dentro de la función:

```
void juego::DesplegarBotones(Uint32 categoria, SDL_Surface* pantalla)
```

Aquí encontraremos un código similar al siguiente, cabe destacar que pondremos un ejemplo suponiendo que la categoría del juego sea **Lógica**:

```
switch(categoria){
        case 1:{
            //Carga de boton efecto pulsado
            SDL_Surface* boton1 = IMG_Load("../imagenes/botones/Logica_p.
               png");
            if(!boton1)
            {
                cerr<<"Imposible cargar bitmap: "<< SDL_GetError()<<endl;
                exit(1);
            }
            rectbtn.x=85;
            rectbtn.y=180;
            SDL_BlitSurface(boton1, 0, pantalla, &rectbtn);
            //Despliego de botones
             SDL_Surface* boton3 = IMG_Load("../imagenes/botones/
                Luces_y_Sombras.png");
            if(!boton3)
            {
                cerr<<"Imposible cargar bitmap: "<< SDL_GetError()<<endl;
                exit(1);
            }
```

```
            rectbtn.x=85;
            rectbtn.y=240;
            SDL_BlitSurface(boton3, 0, pantalla, &rectbtn);
        }
        break;
```

Esta función cargará los botones de los juegos en el caso de que hayamos elegido la categoría **Lógica**. Por tanto deberemos introducir después de la carga del último botón el siguiente código:

```
SDL_Surface* boton4 = IMG_Load("../imagenes/botones/BotonoNuevoJuego.png");
if(!boton4)
{
  cerr<<"Imposible cargar bitmap: "<< SDL_GetError()<<endl;
  exit(1);
}
rectbtn.x=85;
rectbtn.y=300;
SDL_BlitSurface(boton4, 0, pantalla, &rectbtn);
```

Vemos que cargamos el nuevo botón 60 píxels más abajo que el anterior botón, por lo demás este código no es muy complejo.

A continuación nos desplazaremos de nuevo hasta la función:

```
void juego::Categorias(perfiles *perfil, SDL_Surface* pantalla)
```

De manera similar a lo que hemos estado haciendo, lo único que hay que hacer ahora es informar al sistema de que hay un nuevo botón y de la acción que debe de realizar cuando se pulse sobre la posición que ocupa. Para ello deberemos buscar el siguiente trozo de código:

```
//Si hay algun menu desplegado...
if(desplegadoCategoria[0]==true){
  if(mousex>85&&mousey>240&&mousex<(85+150)&&mousey<(240+40)){
    cout<<"Pulsacion de boton: Luces y Sombras"<<endl;
    perfil->actualizarpuntuacion(perfil->perfilactual(),8,log.lucesysombras
        (1,false,pantalla),2,pantalla);
    desplegadoCategoria[1]=false;
    DibujarEscenarioCategorias(pantalla);
  }
}
```

Debemos comentar que el vector *desplegadoCategoria* almacena booleanos que nos indica si hemos pulsado anteriormente en alguna de las categorías del proyecto. Por tanto si hemos pulsado en la categoría de **Lógica** la posición 0 del vector se pondrá a **true** y el sistema será capaz de evaluar si se pulsa en los botones que se han desplegado. Nosotros simplemente deberemos introducir la posición del botón que hemos introducido anteriormente de la siguiente manera:

```
//Si hay algun menu desplegado...
if(desplegadoCategoria[0]==true){
  if(mousex>85&&mousey>300&&mousex<(85+150)&&mousey<(300+40)){
    cout<<"Pulsacion de boton: Nuevo Juego"<<endl;
    perfil->actualizarpuntuacion(perfil->perfilactual(),codigoNuevoJuego,
        minigame.NuevoJuego(pantalla),TipoPuntuacion,pantalla);
```

```
6       desplegadoCategoria[0]=false;
7       DibujarEscenarioCategorias(pantalla);
8     }
9   }
```

Del mismo que explicamos en el subapartado de **Minijuegos** lo más importante de este código es la llamada al control de perfiles para que actualice la puntuación que tenemos registrado. El códigoNuevoJuego será uno más que el último juego que se ha creado, si no ha habido cambios desde que se creo el proyecto sería el 11. El tipoPuntuación dependerá de la naturaleza del juego y de cómo evalúa éste la puntuación, como comentamos anteriormente.

C.4.3. Paso 3: Actualizar el Módulo de Control de Perfiles

En el tercer y último paso simplemente deberemos actualizar el módulo de control de perfiles para indicarle que hay un juego más y que debe escribir en el perfil de usuario una posición más para almacenar la puntuación que obtengamos.

En primer lugar deberemos desplazarnos hasta el fichero *perfiles.h* y localizar la constante siguiente:

```
1   #define NUMJUEGOS 10
```

Una vez que veamos la constante NUMJUEGOS deberemos incrementar en uno su valor. A continuación deberemos borrar todos los perfiles para evitar fallos de segmentación y puntuaciones erróneas, así que deberemos irnos con una consola a la carpeta perfiles y escribir en el terminal:

```
rm *.sav
```

Otro aspecto a tener en cuenta que si salimos del juego de forma abrupta o sin terminar el nivel, simplemente devolviendo 0, el control de perfiles no almacenará ninguna puntuación.

Como hemos podido observar OpenBTrainer está pensado para que se puedan introducir multitud de juegos, ya que simplifica mucho la tarea de ampliación. Unas pocas líneas y ya tenemos nuestro nuevo juego listo para que funcione perfectamente.

Apéndice D

Manual de Apliación de Juego de Diferencias

Como comentamos en capítulos anteriores, el juego de Diferencias estaba pensado para que cualquiera, sin tener conocimientos de programación, pudiera cambiar o incluir nuevas imágenes con sus diferencias, siempre que cumpla con unos criterios que comentaremos en este manual.

D.1. Imágenes

Las imágenes que queramos introducir deben cumplir con un tamaño de 350 x 350 pixeles y el nombre debe cumplir con la siguiente criterio:

Si la última imagen es el número 25, tendremos que nombrarla *26.bmp* la primera imagen y la segunda, donde se marcarán los fallos deberá tener el nombre de *26-2.bmp*

En el caso de que no se cumpla con el criterio de tamaño, la imagen se verá distorsionada y si en el caso de que no se cumpla con el criterio de nombres, la imagen no será cargada y dará un error grave, que provocará el cierre inmediato del juego.

D.2. Fichero de datos de Diferencias

Este fichero nos serivirá para almacenar la cantidad de imágenes que poseemos, y para almacenar la posición x e y donde se encuentra la diferencia.

Dicho fichero lo podemos encontrar en *imagenes/diferencias/soluciones.dat*. Mostratemos un fragmento para explicar su sintaxis y cómo podemos modificarlo a nuestro antojo.

```
25
----
6
46 145 30
89 46 50
160 91 50
178 215 20
289 246 30
250 95 40
```

```
----
6
65 131 20
64 155 40
226 91 20
170 278 30
263 250 40
310 144 50
----
```

Comenzaremos explicando lo que significa cada línea del fichero. La primera de ella informa al juego de cuántas imágenes de diferencias poseemos y por tanto si queremos introducir nuevas imágenes deberemos modificar este valor. La segunda de ella supone una separación que nos ayuda a separar la primera línea del siguiente trozo. Dicho trozo que va desde la línea 3 hasta la 9 nos indica los datos de la imagen *1.bmp* y *1-2.bmp*.

El trozo de la imagen deberemos estudiarlo con detenimiento. La primera línea de cada trozo indicará al sistema el número de diferencias que posee la imagen. Las siguientes líneas nos indican la posición x e y de la diferencia. El siguiente valor equivale al tamaño del cuadrado de captación del que hablamos anteriormente y podemos ver un ejemplo consultando la imagen 4.6. No debemos crear un cuadrado muy pequeño ya que hará que los jugadores tengan que ser muy precisos para lograr acertar con la posición, lo que hará que el jugador desespere en sus intentos y abandone por completo la prueba. Deberemos por tanto, crear un cuadrado de dimensiones compensadas por un lado para que un jugador no le cueste acertar con la posición de la diferencia pero teniendo en cuenta de que no debemos solapar el cuadrado con otro de otra diferencia distinta. Lo que provocaría que la segunda diferencia nunca se encontrase. Cabe destacar que la posición que definimos en el fichero es el de la esquina superior izquierda del cuadrado de captación lo que puede provocar que tengamos que ir jugando con las posiciones hasta dar con la posición idónea.

Con esto terminamos el manual para ampliar el juego de diferencias, hay que comentar que el sistema carga aleatoriamente las imágenes, por tanto se puede dar el caso de que no logremos cargar nuestra nueva imagen y no podamos probarla para ver si las posiciones que hemos introducido son correctas para ello podemos abrir el fichero *minijuego.cpp* y comentar (mediante "//") la línea 1125 que equivale a:

```
numfig=1+rand()\%maxfig;
```

Una vez que modifiquemos esta línea obtendríamos la siguiente sentencia, suponiendo que la imagen fuera la número 26:

```
//numfig=1+rand()\%maxfig;
numfig=26;
```

Una vez que finalicemos de probar nuestra imagen modifica el fichero para dejarlo como estaba, ya que si no siempre cargará la misma imagen.

Podemos comprobar lo sencillo que es añadir nuevas imágenes al juego **Diferencias** y cómo este juego se puede ampliar hasta hacer de él un juego que se podría distribuir a parte del proyecto OpenBTrainer, aunque no es lo que deseamos por el momento ya que este juego supone una forma de que la mente descanse un poco antes de ponernos a realizar las distintos juegos que componen OpenBTrainer.

En este manual se ha explicado y demostrado que aunque el juego **Diferencias** pueda parecer el más flojo del proyecto, está ideado para ser un juego completamente escalable y esto es lo que se debe ver y no la calidad de sus imágenes ya que es difícil encontrar imágenes de diferencias bajo licencia GPL o similar.

Apéndice E

GNU Free Documentation License

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 noncommercially. Secondarily, 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 noncommercially, 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.

Bibliografía

[1] Jordi Mas. Gbrainy - gnome live! http://live.gnome.org/gbrainy, Consultado: Junio 2009.

[2] Brain training - wikipedia. http://es.wikipedia.org/wiki/Brain_Training_del_Dr._Kawashima_%C2%BFCu%C3%A1ntos_A%C3%B1os_Tiene_tu_Cerebro%3F, Consultado: Julio 2009.

[3] Florian Thauer and Felix Hammer. Pokerth - community portal. http://www.pokerth.net/, Consultado: Julio 2009.

[4] Orientación Andújar. Fichas de atención; encuentra las diferencias, departamento de orientación. http://orientacionandujar.wordpress.com/2009/01/17/fichas-atencion-encuentra-las-diferencias/, Consultado: Julio 2009.

[5] Molotov.nu - game graphics. http://www.molotov.nu/?page=graphics, Consultado: Agosto 2009.

[6] C++ - wikipedia. http://es.wikipedia.org/wiki/C%2B%2B, Consultado: Agosto 2009.

[7] Simple directmedia layer - wikipedia. http://es.wikipedia.org/wiki/LibSDL, Consultado: Agosto 2009.

[8] Glenford J Myers. *The art of software testing, ISBN: 0-471-46912-2*. Hoboken: John Wiley & Sons, 2º edition, 2004.

[9] Antonio García Alba. Tutorial wiki de libsdl para la programación de videojuegos. http://softwarelibre.uca.es/wikijuegos, 2008.

[10] Jose Luis Falcón Ramírez. forja: Openbtrainer: Información del proyecto. https://forja.rediris.es/projects/openbtraining/.

[11] Buttonator - drag and drop web buttons. http://www.buttonator.com, Consultado: Julio 2009.

[12] Bernardo Cascales Salinas. *El libro de LaTeX , ISBN: 84-205-3779-9*. Pearson Educación, 2003.

[13] Salvador Pozo, Steven R. Davidson, and Julián Hidalgo. Curso online de programación c y c++. http://www.conclase.com/.

[14] Emilio José Rodríguez Posada. Detector y corrector automático de ediciones malicionsas en wikipedia. http://emijrp.googlepages.com/mpfc.pdf, Enero 2009.