Universidad Complutense de Madrid
Facultad de Informática

Ingeniero en Informática
Curso académico 2006-2007

Proyecto de Sistemas Informáticos

Videojuego Educativo para el Aprendizaje de SQL

Autores:
Sonia Martín Moreno
Francisco Javier Remesal Escalero
Laura Rivera Rodríguez

Profesor Director:
Fernando Rubio Diez

Índice general

Índice de figuras

1. Resumen

1.1 Resumen en Castellano

El mundo de los videojuegos hoy en día es un campo en constante desarrollo. La posibilidad de aplicar los videojuegos al aprendizaje para poder aprender de una forma amena es uno de los motivos que nos ha llevado a realizar este proyecto.

El desarrollo de este proyecto ha consistido en la investigación de las herramientas utilizadas (SDL, Dev-C++ y C++) y la aplicación de éstas para la elaboración de un videojuego de aprendizaje de uno de los lenguajes más importantes hoy en día, SQL.

El videojuego consiste en la realización de una serie de pruebas que muestran al jugador las estructuras básicas para consultar y modificar las bases de datos de una forma amena.

1.2 Resumen en Inglés

The world of the videogames nowadays is a field in constant development. The possibility of applying videogames to learn in a pleasant form is one of the reasons that has led us to make this project.

The development of this project has consisted of the research of the used tools (SDL, Dev-C ++ and C ++) and the application of these tools to elaborate a videogame to learn one of the most important languages nowadays, SQL.

The videogame consists of the accomplishment of a series of trials that show to the player the basic structures to consult and to modify the databases in a pleasant form.

1.3 Palabras clave

Videojuego, Juego, Educativo, SQL, SDL, C++, MySQL.

2. Introducción

El contexto en el que se sitúa nuestro proyecto son los videojuegos educativos. Para ello, haremos una pequeña introducción a los videojuegos contando un poco de su historia: cómo surgieron, los avances, etc. Como segundo apartado de esta introducción profundizaremos un poco más en el tipo de videojuegos que nos interesan para este proyecto, los videojuegos educativos.

2.1 Reseña histórica de los videojuegos

Los primeros pasos de los actuales videojuegos se detectan en los años 40, cuando los técnicos americanos desarrollaron el primer simulador de vuelo, destinado al entrenamiento de pilotos. En 1962 apareció la tercera generación de ordenadores, reduciendo su tamaño y coste de manera drástica y a partir de ahí el proceso ha sido continuado.

En 1969 nació el microprocesador, que en un reducido espacio producía mayor potencial de información que los grandes ordenadores de los años 50. Es lo que constituye el corazón de nuestros ordenadores, videojuegos y calculadoras.

En 1972 se desarrolla el primer juego, llamado PONG, que consistía en una rudimentaria partida de tenis o ping-pong. En 1977, la firma Atari™ lanzó al mercado el primer sistema de videojuegos en cartucho, que alcanzó un gran éxito en Estados Unidos y provocó, al mismo tiempo, una primera preocupación sobre los posibles efectos de los videojuegos en la conducta de los niños.

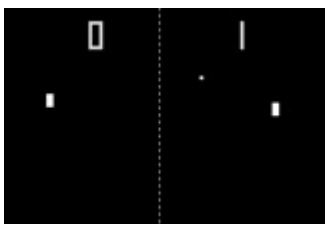

Figura 2.1: PONG de Atari™

Tras una rápida evolución, en la que el constante aumento de la potencia de los microprocesadores y de la memoria permitieron nuevas mejoras, en 1986, la casa Nintendo™ lanzó su primer sistema de videojuegos que permitió la presentación de unos juegos impensables nueve años atrás.

Figura 2.2: Famicom, primer sistema de videojuegos de Nintendo™

La extensión masiva de los videojuegos en los años 90 ha provocado una segunda oleada de investigaciones, desde la medicina, la sociología, la psicología hasta la educación.

2.2 Videojuegos educativos

Los videojuegos representan en la actualidad una de las entradas más directas de los niños a la cultura informática y a la cultura de la simulación.

Muy utilizados por niños y adolescentes, los videojuegos son, en muchos casos, muy criticados por sus contenidos y muy poco utilizados por los educadores que desaprovechan una potente herramienta educativa.

Una de las cosas que hacen que esto sea posible es la "motivación".El funcionamiento de la motivación y su influencia en el aprendizaje puede ser condensado en las siguientes reglas:

1. Lo fundamental es que la tarea tenga en sí misma el suficiente atractivo o motivación para promover el aprendizaje.
2. En caso contrario, existen otras fuentes de motivación, entre las que destacan los refuerzos, que pueden ejercer un papel importante. Estos refuerzos pueden tener características de tipo material (premios, recompensas, dinero, etc.), psicológico (alabanzas), intelectual (conocer las tareas y los resultados), y social (reconocimiento social, amplificadores sociales).
3. El sistema de refuerzos tiene una mayor influencia cuando se cumplen determinados requisitos:

- ✓ Que tengan carácter positivo, recompensador, en lugar de castigo.

- ✓ Que sea un programa definido, no arbitrario, de refuerzo.
- ✓ Que suponga una dificultad progresiva.
- ✓ Que el refuerzo o la recompensa sea inmediata.
- ✓ Que esté adaptado a las características y ritmo del individuo (niveles).
- ✓ Que se conozcan los resultados rápidamente.
- ✓ Que tengan un reconocimiento social.

Si comparamos la realidad de los videojuegos podemos comprobar que éstos reúnen muchas de las reglas que anteriormente se describen.

- Los videojuegos: algo más que un entretenimiento.

El juego fue introducido en el colegio como algo más que un entretenimiento o una diversión, los educadores intuyeron algo que muchos años después ha sido corroborado por numerosas investigaciones: los videojuegos tienen un potencial educativo importante. Pero el valor de los videojuegos no es sólo su factor motivacional sino que a través del juego se puede aprender, se pueden desarrollar destrezas, habilidades, estrategias. En definitiva, ya nadie discute que se puede aprender jugando.

Al hablar de los videojuegos tenemos que tener presente que el medio en sí mismo posee de unas características propias diferentes a otros productos informáticos.

Los videojuegos presentan unos atributos propios que podemos sintetizar en los siguientes aspectos:

- ❖ Integran diversas notaciones simbólicas.

En la mayoría de los juegos actuales podemos encontrar informaciones textuales, sonido, música, animación, vídeo, fotografías, imágenes en tres dimensiones. Diversas notaciones se encuentran presentes en una sola pantalla.

Las investigaciones sobre los medios todavía no han llegado a ninguna conclusión relevante sobre la eficacia real de la combinación de las diferentes notaciones simbólicas en un sólo medio.

El hecho en sí supone un avance técnico indudable. Sin embargo, los cambios en el diseño del software no se producen a la misma velocidad y la armonización de las diferentes notaciones no siempre se consigue. Muchos programas combinan tantas notaciones simbólicas diferentes en una única pantalla que se hace difícil pensar que el usuario tenga la capacidad suficiente como para lograr decodificar dicha

información. Aunque, por otra parte, se observan diferencias significativas en la decodificación de los mensajes entre los jugadores expertos y los jugadores noveles.

❖ Los videojuegos son dinámicos

El medio informático permite mostrar en pantalla fenómenos de procesos cambiantes. Las imágenes producidas por el ordenador pueden crear modelos de cualquier fenómeno real, posible o imaginario. En este sentido, la creación de simulaciones y entornos virtuales se va haciendo cada vez más sofisticada y el usuario tiene una sensación cada vez mayor de implicación en las historias ofrecidas a través de la pantalla. De hecho, uno de los campos de aplicación prioritarios en el diseño de la realidad virtual lo constituyen los juegos de ordenador.

❖ Los videojuegos son altamente interactivos

La mayoría de los juegos de ordenador son altamente interactivos. El grado de interactividad de un medio puede medirse a través de muchas variables diferentes, destacaremos las siguientes:

- Las posibilidades de apropiación y de personalización del mensaje recibido, sean cual sea su naturaleza.
- La reciprocidad de la comunicación.
- La virtualidad.
- La implicación de la imagen de los participantes en los mensajes.

❖ Adicción

Todos los juegos crean una cierta adicción, es una de las claves del éxito de un juego, incluyendo a los ya tradicionales. El hecho de jugar conlleva a que sea trascendente mientras se juega pero debe ser intranscendente una vez terminado.

Una vez superado o alcanzado un nivel de ejecución suficiente como para dominar el jugador al programa, la atracción disminuye y entra en los cauces de la normalidad. El hecho sigue siendo comparable a cualquier otra actividad de ocio.

- <u>Los videojuegos en la enseñanza</u>

Si bien las investigaciones no son definitivas, la mayoría de ellas indican que muchos videojuegos favorecen el desarrollo de determinadas habilidades, de atención, concentración espacial, resolución de problemas, creatividad, etc. por lo que se concluye que en su conjunto, desde el punto de vista cognitivo, los videojuegos suponen algún tipo de ayuda en el desarrollo intelectual. Se sugiere que

quienes juegan a los videojuegos adquieren mejores estrategias de conocimiento, modos de resolver problemas, se benefician en sus habilidades espaciales y aumenta su precisión y capacidad de reacción. No hay evidencia de los efectos contrarios.

Desde el punto de vista intelectual, la complejidad de la mayor parte de los juegos de ordenador actuales permiten desarrollar no sólo aspectos motrices sino, sobre todo, procedimientos tales como las habilidades para la resolución de problemas, la toma de decisiones, la búsqueda de información, la organización, etc. Desde el punto de vista afectivo, los juegos ejercen una importante motivación y pueden utilizarse para el trabajo de aspectos relativos a la autoestima.

Los juegos educativos se diseñan para aprender. No se trata de "perder" el tiempo jugando, hay que aprender algo durante el juego. Esta diferencia en cuanto a intención encierra unas consecuencias importantes. Los juegos educativos mayoritariamente están pensados para ser utilizados en los colegios.

Hay que tener en cuenta que los videojuegos cumplen muchos de los requisitos que una eficaz enseñanza debe contemplar, y en muchos casos lo hacen mejor incluso que nuestros actuales sistemas educativos.

Existen siete características que hacen de los videojuegos un medio de aprendizaje más atractivo y efectivo:

1. Permiten el ejercicio de la fantasía, sin limitaciones espaciales, temporales o de gravedad.
2. Facilitan el acceso a "otros mundos" y el intercambio de unos a otros a través de los gráficos, contrastando de manera evidente con las aulas convencionales y estáticas.
3. Favorecen la repetición instantánea y el intentarlo otra vez, en un ambiente sin peligro.
4. Permiten el dominio de habilidades. Aunque sea difícil, los jugadores pueden repetir las acciones, hasta llegar a dominarlas, adquiriendo sensación de control.
5. Facilitan la interacción con otras personas.
6. Hay una claridad de objetivos. El jugador cuando juega al videojuego sabe que hay una tarea clara y concreta: abrir una puerta, rescatar a alguien, hallar un tesoro, etc. lo cual proporciona un alto nivel de motivación.
7. Favorece un aumento de la atención y del autocontrol favoreciendo el éxito individual.

Hoy en día, los videojuegos forman parte de la vida lúdica de una gran mayoría de personas. Siendo éste un elemento más dentro de la tecnología, consideramos oportuno y conveniente analizar las potencialidades que los videojuegos podrían tener dentro del proceso educativo, de tal manera que su utilización contribuya a mejorar y elevar la calidad de dicho proceso.

3. Objetivos

El objetivo principal de este proyecto era realizar una herramienta de aprendizaje que permitiera adquirir una serie de conocimientos sobre el lenguaje de base de datos SQL.

Una vez comprobada la inexistencia de herramientas educativas sobre este tema, pensamos que podría ser un proyecto original y, sobre todo, útil.

Uno de los objetivos más importantes a la hora de llevar a cabo esta herramienta era que su uso resultara sencillo al jugador además de permitirle adquirir una serie de conocimientos relacionados con lenguaje SQL de forma amena y divertida.

Para llevar esto a cabo, el desarrollo del proyecto se centró en el uso de herramientas gratuitas, con licencias de libre distribución para que el resultado fuese un programa de código abierto.

En el desarrollo de la herramienta se pensó en posibles ampliaciones futuras, por lo que su diseño y ejecución se realizó en base a esta premisa.

Otro de los objetivos era permitir que el resultado final no necesitara excesivos recursos para su uso, pudiéndose ejecutar en prácticamente cualquier máquina con unos requerimientos mínimos.

Para resumir, hemos tratado de realizar un juego que pueda utilizar cualquier persona con independencia de sus conocimientos informáticos y de los recursos de que disponga su ordenador personal, y que su uso suponga el aprendizaje de uno de los lenguajes más importantes y utilizados en la actualidad.

4. Análisis y Diseño

En este apartado explicaremos brevemente las herramientas utilizadas para el desarrollo del proyecto y el porqué de su elección. Además, profundizaremos en el diseño interno de nuestro programa.

4.1 Herramientas

4.1.1 Definición de las herramientas

- SDL

SDL (Simple DirectMedia Layer) es una API gráfica desarrollada con el lenguaje C para realizar operaciones de dibujado en 2D, gestionar efectos de sonido y música, y cargar imágenes.

Pese a estar programado en C, es aplicable a otros lenguajes de programación como C++, Ada, C#, Basic, Lua, Java, Pitón, etc. También proporciona herramientas para el desarrollo de videojuegos y aplicaciones multimedia.

Una de sus grandes virtudes es el tratarse de una librería multiplataforma, soportando oficialmente los sistemas Windows, Linux, MacOS y QNX, además de otras arquitecturas/sistemas como Dreamcast, GP32, GP2X, etc.

Fueron desarrolladas inicialmente por Sam Lantinga, desarrollador de videojuegos para la plataforma Linux.

La librería se distribuye bajo la licencia LGPL, que es la que ha provocado el gran avance y evolución de las librerías SDL.

Se han desarrollado una serie de librerías adicionales que complementan las funcionalidades y capacidades de la librería base. Entre ellas destacan:

- SDL_image: Carga de imágenes en diferentes formatos: png, jpg, etc.
- SDL_mixer: Carga de formatos de sonido como wav, mp3, ogg, etc.
- SDL_net: Comunicación de datos en redes.
- SDL_ttf: Uso de fuentes TrueType.

- DEV-C++

Dev-C++ es un compilador y entorno de desarrollo para C/C++. Consiste en un editor de múltiples ventanas integrado con un compilador que nos permitirá una compilación, un enlace y una ejecución de aplicaciones rápida y de fácil uso.

El editor nos ofrece la opción de realzamiento de sintaxis. Incluye también un debugger y un creador de instaladores, además de 150 ficheros y 100 librerías.

Algunas de las posibilidades que nos ofrece son: recopiladores de la ayuda GCC-based, corrección de errores integrado (Usa GDB), ayuda para los idiomas múltiples (localización), explorador de clases, depurador de código, poder eliminar errores en el explorador de variables, asistente de Proyecto, editor más óptimo, crea rápidamente en la consola de Windows, bibliotecas estáticas y DLL, ayuda de plantillas para crear sus propios tipos del proyecto, creación del makefile, instalación fácil de bibliotecas adicionales, ayuda de CVS, etc.

- Lenguaje de programación C++

El C++ es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Las principales características del C++ son el soporte para programación orientada a objetos y el soporte de plantillas o programación genérica. Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos trae (obliga a hacerlo casi todo manualmente al igual que C) lo que dificulta mucho su aprendizaje.

El nombre C++ fue propuesto por Rick Masciatti 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++, "C++" significa "incremento de C" y se refiere a que C++ es una extensión de C.

4.1.2 Elección de las herramientas

- ¿Por qué SDL?

La primera decisión que se nos planteó a la hora de comenzar a realizar nuestro proyecto fue qué librería de gráficos usar. Buscando información sobre programación de videojuegos nos dimos cuenta que las opciones que podíamos considerar eran Allegro y SDL, ya que ambas son librerías de código abierto y que pueden utilizarse en diversos sistemas operativos.

Investigando sobre ambas, fue complicado decidirnos porque las dos son muy similares en cuanto a lo que ofrecen y lo que necesitan para ser desarrolladas, pero decidimos elegir SDL porque encontramos una documentación bastante completa y en los comienzos cuanto más amplia sea la base en la que nos podemos basar, mucho mejor.

- Elección del lenguaje de programación

Tras elegir la librería de gráficos, el segundo paso fue el lenguaje de programación en el que realizaríamos nuestro proyecto. No hubo dudas puesto que SDL está programado en C. Así que decidimos elegir C/C++ aunque nuestra experiencia previa no fuese muy amplia.

- Elección del entorno de desarrollo

Elegimos Dev-C++ porque es un programa gratuito y en su última versión viene integrado con el compilador Mingw. De esta forma teníamos ambas necesidades cubiertas. Además, parte de la documentación que encontramos sobre SDL estaba basada en este entorno de desarrollo.

4.2 Estructura del programa

Un videojuego es un programa diferente a los programas convencionales. Debe funcionar en tiempo real, en todo momento mientras se está ejecutando, el juego debe estar realizando alguna tarea. También debe estar esperando que ocurra algún evento, ver si se ha pulsado una tecla determinada, si se ha movido el ratón o presiona algún botón de éste y luego actuar en consecuencia. Todo esto, y más, debe ocurrir en un ciclo.

La estructura principal de nuestro programa podría resumirse en el siguiente gráfico. Donde:

- Cargar Opciones: En nuestro videojuego se ofrece la posibilidad de cambiar las opciones principales. Estas son: audio, sonido y modo de video. Por lo tanto, al comenzar el juego, se carga la última configuración guardada.
- Iniciar: Inicializa la biblioteca y los recursos básicos.
- Correr: Es el bucle principal del videojuego.
- Eliminar: Al terminar el programa, libera los recursos utilizados.

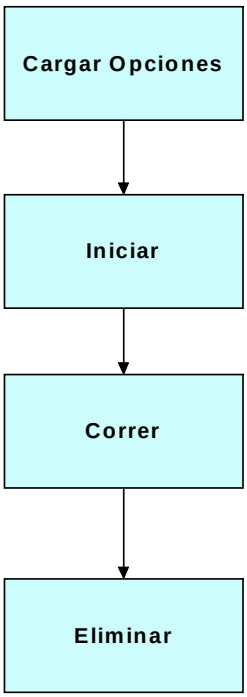

Figura 4.1: Estructura principal del programa

Estos métodos son llamados desde la clase main, pero están contenidos dentro de la clase *juego*.

Si estudiamos detalladamente los elementos principales:

- Iniciar:
  Este método se encarga de preparar el comienzo del videojuego. Se encarga de: inicializar la librería SDL, asignar las opciones cargadas previamente, iniciar el Audio y cargar la primera escena que podrá observarse. Resumido:

Figura 4.2: Método Iniciar

- Correr:
  En este método se encuentra el bucle principal del programa. En él se comprueba continuamente si se ha producido algún evento que deba ser considerado.

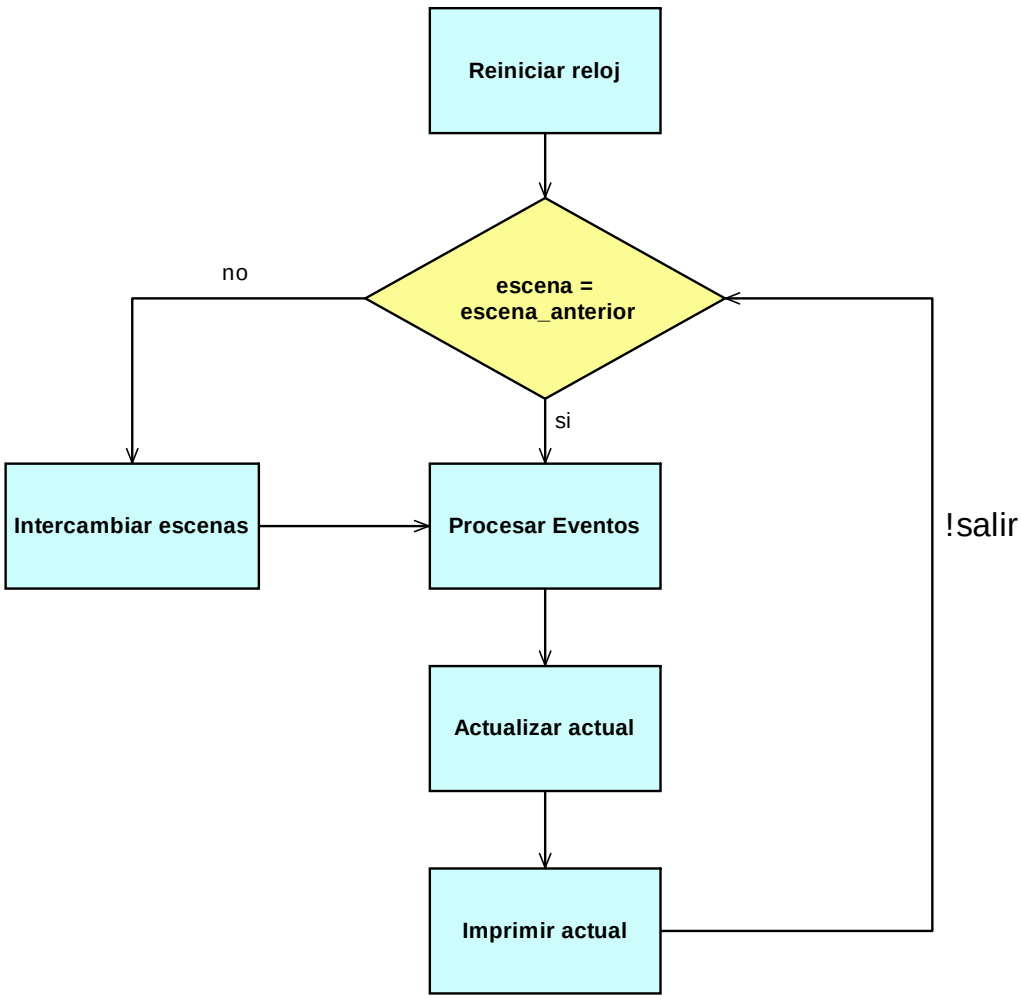

Figura 4.3: Método correr

Los métodos referenciados por correr son los siguientes:

- Reiniciar reloj:
  Debido a que no en todos los equipos la velocidad es la misma, debemos evitar que en un equipo lento, el juego funcione a baja velocidad y en cambio si se ejecuta en un equipo muy rápido sea imposible jugar dada su velocidad. Para esto, en nuestro bucle principal se gestiona la velocidad del juego de manera independiente a la arquitectura del equipo o dispositivo. Para conseguir esa independencia, controlamos nuestro bucle mediante rutinas de tiempo.
  Este método reinicia los contadores de tiempo y durante el bucle se gestionan los valores obtenidos.

- Procesar eventos:
  Es el método específico para observar los eventos que se producen durante la ejecución. Determina el valor de la variable salir.

- Intercambiar escenas:
  Elimina la escena actual e inicia la siguiente escena correspondiente. Si no se puede iniciar, el juego termina. Desde aquí también se llama al método de reiniciar reloj para tener en cuenta los tiempos de ejecución.

- Actualizar:
  Es un método específico de cada pantalla. Después de ser creada la siguiente escena, creándose una instancia de la clase correspondiente, se llama a este método, que contiene el bucle de cada pantalla.

- Imprimir:
  Al igual que el método anterior es diferente para cada pantalla. Se encarga de introducir todos los elementos gráficos sobre la pantalla actual.

4.3 Estructura de clases

A continuación explicaremos el contenido de las clases que forman nuestro programa y mostraremos con diagramas las relaciones entre ellas.

- juego.cpp/juego.h:

Es la clase principal del juego. Ya hemos explicado los métodos que influyen en el comportamiento general del programa en el apartado anterior. Su contenido completo es el siguiente:

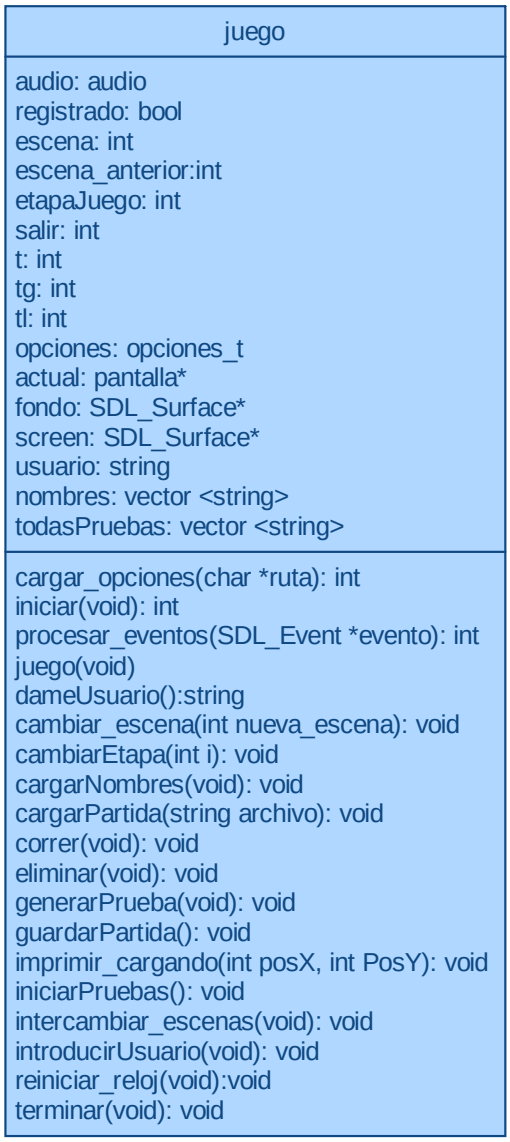

Figura 4.4: Diagrama de la clase juego

Además, de los elementos que ya hemos comentado, podemos destacar lo siguiente:

- Nuestro juego tiene la opción de guardar el estado del juego para siguientes ejecuciones del juego. Los elementos utilizados para que esto sea posible son: usuario y todasPruebas como atributos y dameUsuario, guardarPartida y cargarPartida como métodos. La información de cada usuario se guarda en un archivo de texto con el nombre de usuario donde se hace referencia a las pruebas que ya han sido superadas.
- Se tiene conocimiento de todos los usuarios que han guardado partida en el juego, esto se almacena en el atributo nombres mediante el método cargarNombres.

- pantalla.h:

Es una clase abstracta. Todas las pantallas de nuestro juego la heredan definiendo los métodos iniciar, actualizar e imprimir. Su contenido:

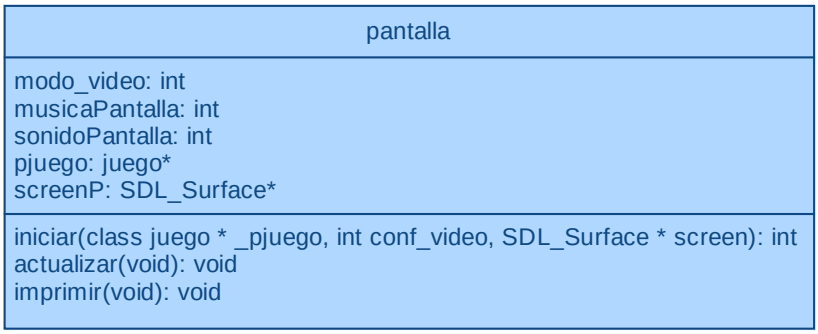

Figura 4.5: Diagrama de la clase pantalla

- audio.cpp/audio.h:

En esta clase están los métodos para controlar las opciones de audio de nuestro juego. Dependiendo de lo que seleccionemos en el menú de opciones, se activará la música o los sonidos.

El audio funciona de manera general. Si se activa, estará activado para todas las pantallas, si se desactiva, también será para todas las pantallas.

El diagrama siguiente muestra su contenido:

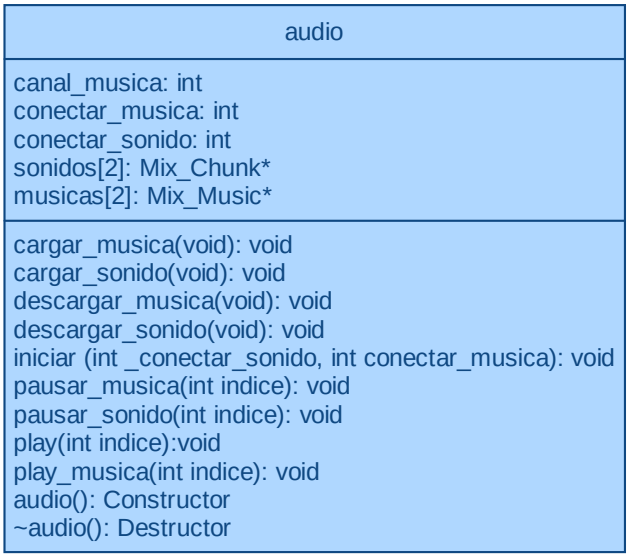

Figura 4.6: Diagrama de la clase audio

Las cuatro clases siguientes recogen métodos de utilidad que se usan en prácticamente el resto de clases.

- util_dibujo.cpp/util_dibujo.h:

Debido a que en todas las pantallas se muestran imágenes, tenemos una clase con los métodos para cargar las imágenes sobre nuestra pantalla de juego. DrawIMG carga la imagen y DrawBG carga el fondo de la pantalla:

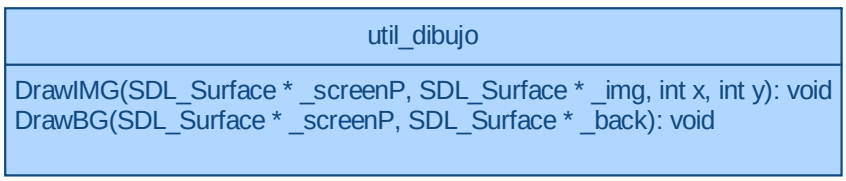

Figura 4.7: Diagrama de la clase util_dibujo

- util_fuente.cpp/util_fuente.h:

Para utilizar archivos de caracteres creados como imágenes, debemos separar cada letra como una imagen para poder mostrarlas en pantalla. Esto lo hemos utilizado en bastantes pantallas, por lo tanto hemos recogido en esta clase los métodos necesarios:

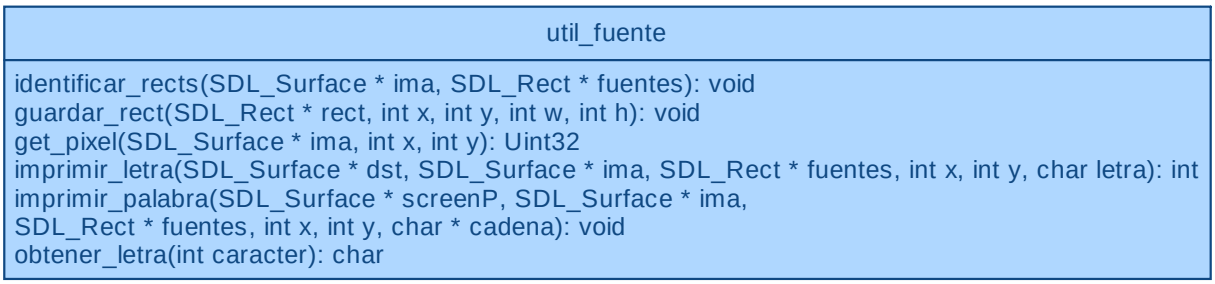

Figura 4.8: Diagrama de la clase util_fuente

- util_pantalla.cpp/util_pantalla.h:

Usamos los métodos de esta clase para cambiar el tamaño de la pantalla. Los tamaños pueden ser 640x480 o pantalla completa:

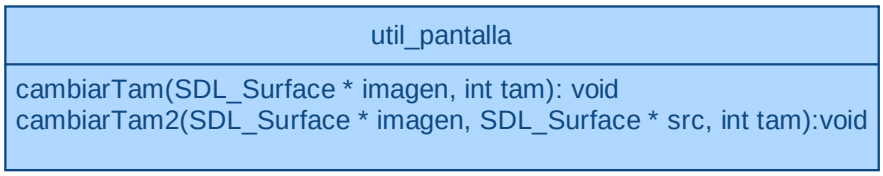

Figura 4.9: Diagrama de la clase util_pantalla

- boton.cpp/boton.h:

Junto con util_dibujo, esta es una de las clases más utilizadas dentro de nuestro programa. Podemos crear botones de diferentes tamaños y detectar si el ratón realiza una pulsación sobre él conociendo sus dimensiones. Tenemos dos tipos de botones, los normales y los de tipo check que actuan sobre la variable seleccionado.

Su contenido es el siguiente:

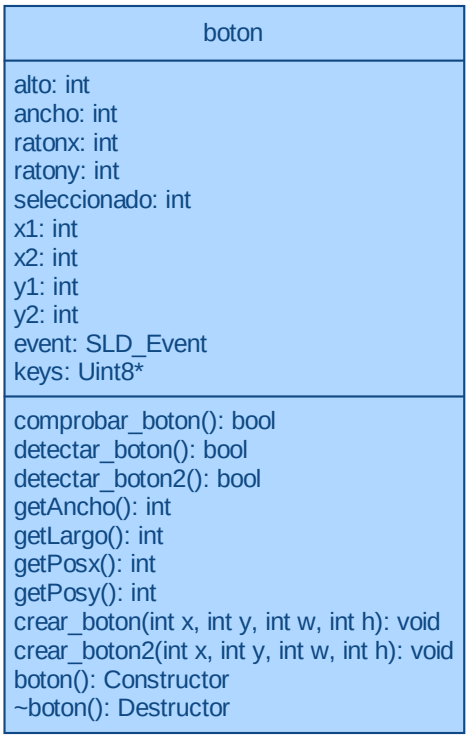

Figura 4.10: Diagrama de la clase boton

A continuación haremos un listado de las clases que heredan de la clase pantalla. La estructura principal de todas ellas es similar. Todas tienen los métodos iniciar, actualizar e imprimir, por lo tanto no mostraremos sus diagramas. El resto de métodos de cada una de ellas es específico para realizar su función.

- historia.cpp/historia.h: Es la introducción del videojuego
- menu_principal.cpp/menu_principal.h: Menú principal del videojuego.
- menu_opciones.cpp/menu_opciones.h: Menú para seleccionar las opciones de audio y video.
- menu_controles.cpp/menu_controles.h: Muestra la función de cada tecla.
- pantalla_nombre.cpp/pantalla_nombre.h: Pantalla donde se introduce el nombre del usuario y se muestran todos los usuarios que han salvado su partida.
- historiaSQL.cpp/historiaSQL.h: Pantalla que hace de cabecera al resto de pantallas. En su constructor se pasa como variable el archivo de texto del cual extrae las palabras a mostrar en cada uno de los casos.

- guardar.cpp/guardar.h: Pantalla que se muestra al intentar salir del juego para guardar el estado de las pruebas o no.

- pasoEtapa.cpp/pasoEtapa.h: Pantalla intermedia entre una etapa y otra. En su constructor se introduce como variable el número de la etapa siguiente.

Las siguientes pantallas recogen las diferentes pruebas de cada una de las etapas:

- etapa1.cpp/etapa1.h
- etapa2.cpp/etapa2.h
- etapa3.cpp/etapa3.h
- etapa4.cpp/etapa4.h
- etapa5.cpp/etapa5.h

Las siguientes clases también heredan de pantalla.h su característica es que son las que representan a cada una de las pruebas:

- ahorcado.cpp/ahorcado.h
- sopa.cpp/sopa.h: utiliza las clases informacion.cpp/informacion.h y Letra.h
- camareroSelect.cpp/camareroSelect.h
- camareroUpdate.cpp/camareroUpdate.h
- camareroDelete.cpp/camareroDelete.h
- inicial.cpp/inicial.h
- secuencia.cpp/secuencia.h
- puzzle.cpp/puzzle.h
- testSelect.cpp/testSelect.h
- testUpdate.cpp/testUpdate.h
- testDelete.cpp/testDelete.h
- testInsert.cpp/testInsert.h
- trivialSelect.cpp/trivialSelect.h
- trivialUpdate.cpp/trivialUpdate.h
- trivialDelete.cpp/trivialDelete.h
- trivialInsert.cpp/trivialInsert.h
- simon.cpp/simon.h
- acertijos.cpp/acertijos.h
- enigmas.cpp/enigmas.h
- adivinanzas.cpp/adivinanzas.h
- varios.cpp/varios.h
- chistes.cpp/chistes.h

- pelotaUp.cpp/pelotaUp.h
- pelotaDel.cpp/pelotaDel.h
- pasoEtapa.cpp/pasoEtapa.h
- pantalla_ultima.cpp/pantalla_ultima.h

En resumen, diremos que en la clase main tenemos una instancia de juego, que a su vez tiene una instancia de pantalla. En el siguiente gráfico se puede ver más claramente la relación de las clases principales:

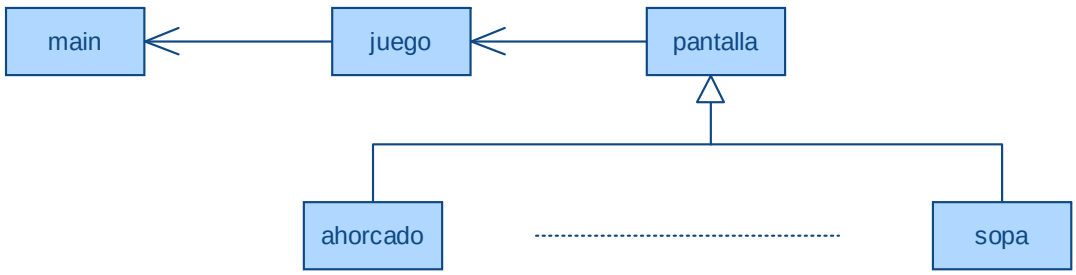

Figura 4.11: Diagrama de clases general

5. Desarrollo

Cuando alguien se plantea el reto de desarrollar su primer videojuego, las ganas por ver resultados inmediatos pueden llevar en algunos casos a realizar un trabajo inicial que debido, por ejemplo, a las herramientas elegidas no es posible finalizar como estaba planeado.

Por esto, es muy importante que antes de lanzarse a escribir cientos de líneas de código se realice un estudio detallado de todos aquellos elementos que estarán presentes en el proyecto.

A continuación detallaremos las fases principales por las que ha pasado el desarrollo de nuestro videojuego.

5.1 Idea principal

Al plantearnos el hecho de desarrollar un videojuego educativo relacionado con la informática, estuvimos barajando varias ideas entre las partes de hardware/software que habíamos aprendido durante la carrera.

Cuando surgió la idea de realizarlo sobre SQL nos decantamos por ella porque creemos que las bases de datos están presentes en el día a día y sería interesante explicárselo a personas sin conocimientos informáticos avanzados.

Además, creemos que SQL es uno de los lenguajes que a la vez de útil, no es muy complicado de aprender mediante este método.

5.2 Investigación

Gran parte del esfuerzo para llevar a cabo este proyecto ha estado dedicado a la investigación del uso de las herramientas elegidas.

Al comenzar el proyecto no sabíamos, por una parte, con qué lenguaje realizar el juego y, por otra, no conocíamos ninguna librería aplicable para gráficos. Entonces comenzamos a barajar distintas posibilidades y a estudiar las compatibilidades entre unas herramientas y otras.

La idea era decantarnos por aplicaciones y librerías que pudieran usarse en distintos sistemas operativos y que, por supuesto, fuesen eficaces para el tipo de juego que elegimos, que consiguiésemos todo lo que nos proponíamos sin que el PC aumentara mucho su uso de CPU. Esto último también nos ha llevado parte de estudio puesto que lo que queríamos también era que se pudiese ejecutar en distintos ordenadores con unos requerimientos mínimos.

Tras elegir las herramientas, estudiamos qué librerías auxiliares de SDL íbamos a necesitar para comenzar a utilizarlas. El primer paso fue configurar el entorno de desarrollo para poder utilizar la librería SDL.

Para comprobar que todo estaba configurado correctamente, el primer paso fue intentar mostrar una imagen por pantalla. Después de esto llegaron otros logros como es el mover una imagen mediante teclado, introducir sonido o el reconocimiento del movimiento y el pulsado del ratón.

Uno de los últimos pasos fue introducir características de la sublibrería SDL_ttf.

Otra de las tareas de investigación que realizamos fue en relación con MySQL. Queríamos que el usuario no solo jugara y aprendiera, sino que construyera tablas, modificara datos, etc. para que con la práctica se fijen los conocimientos adquiridos con la teoría. Por este motivo, decidimos primero realizar el juego y una vez que el usuario se viese en condiciones de aplicar lo aprendido a la práctica, le guiaríamos para que crease su propia base de datos. Para todo esto necesitaríamos hacer una conexión de la base de datos con nuestras herramientas.

Este proceso nos llevó aproximadamente 3 meses de trabajo, en los cuales obtuvimos algún primer resultado para nuestro programa.

5.3 Fase creativa

Después de conocer las capacidades de la librería gráfica, comenzamos por desarrollar la cabecera y los menús del juego.

Antes de comenzar a desarrollar las pruebas estudiamos el orden en el que enseñaríamos las diferentes características del lenguaje SQL. Así, decidimos que fuera el siguiente: Select, Where, Insert/Update/Delete, Create.

La primera prueba que pensamos fue “Tu Primera Query”, en la que se debían ordenar cuatro palabras para formar una consulta correcta.

Tras esto, surgieron ideas como el ahorcado o la sopa de letras. Pensamos que unos sencillos juegos para comenzar a aprender las palabras clave del lenguaje sería una buena manera de empezar.

Necesitábamos explicar las características del lenguaje, así que creamos una pantalla que fuese contando todo esto en forma de lecciones. De esta forma

aprovecharíamos estas lecciones para mostrar cosas que no pudiésemos abarcar con las pruebas. Hemos dividido las lecciones de la siguiente forma:

- Fase 1:
  - Construcción correcta de la consulta SELECT con uno y varios campos.
  - Operadores matemáticos: <, >, =, etc.
  - Funciones AVG, COUNT, SUM, MAX y MIN.
  - Uso de DISTINCT.

- Fase 2
  - Utilización de WHERE.
  - Operadores lógicos: AND y OR.
  - ORDER BY + ASC/DESC.
  - GROUP BY.
  - LIKE y el uso de %.

- Fase 3
  - Construcción correcta de una expresión de tipo UPDATE con uno y varios campos.
  - Uso de IN y NOT IN.
  - BETWEEN.

- Fase 4
  - Construcción correcta de una expresión de tipo DELETE.
  - COMMIT y ROLLBACK.

- Fase 5
  - Construcción correcta de una expresión de tipo INSERT.
  - Introducción a CREATE.

- Fase Final
  - Guía para la utilización de herramientas de MySQL.

Después de organizar las fases, comenzamos a desarrollar pruebas para aplicar estas lecciones. Decidimos que juegos tipo test o trivial serían los más útiles para poder valorar los conocimientos, así que decidimos hacer uno para cada tipo de comando.

Por último, surgió la idea de crear pantallas que no tuvieran nada que ver con SQL, que fuesen un entretenimiento durante el juego, así que se crearon pantallas de chistes, adivinanzas, etc. para cada una de las fases.

5.4 Cambios

Durante el desarrollo del videojuego, surgieron varios cambios, tanto por mejoras como por problemas encontrados.

Uno de los principales problemas fue el rendimiento. Cuando ya teníamos desarrollado gran parte del programa, nos dimos cuenta que en pantallas como las de los menús, el rendimiento era muy malo. La carga de transacciones podía alcanzar hasta los 2 Gb si dejábamos mucho tiempo la pantalla del menú activa.

La solución no fue sencilla porque los menús son pantallas con una complejidad bastante alta, pero finalmente logramos encontrar la solución. En estas pantallas teníamos un método que procesaba los eventos continuamente. Lo eliminamos e hicimos que esto se comprobara en el bucle de cada pantalla y el problema se solucionó.

Otro de los problemas ha sido la unión de las pantallas. En muchas ocasiones al intentar pasar de una pantalla a otra, el programa se cerraba. Entre otras cosas era causado por la repetición del nombre de alguna variable que estuviese en uso dentro de nuestra clase juego.

Las mejoras que hemos llevado a cabo han sido sobre todo para una mejor jugabilidad:

- Poder hacer uso del ratón y de las teclas para seleccionar los botones de Aceptar/Cancelar.
- La opción de guardar en cada fase.
- Poder ir hacia la siguiente o la anterior fase pulsando una tecla.
- Poder salir en todo momento a la fase actual tanto en las pruebas como en las pantallas de las lecciones.
- Mostrar el nombre de los usuarios que han guardado partida para poder recordar con qué nombre se ha jugado.

6. Documentación

6.1 Manual de SDL

Para utilizar SDL deberemos importar las cabeceras de las clases que necesitamos:

```
#include "SDL/SDL.h"
#include "SDL/SDL_image.h"
#include "SDL/SDL_mixer.h"
…
```

Para utilizar cada sublibrería de SDL necesita incluir su cabecera para poder utilizar sus métodos.

Para inicializar SDL y crear una pantalla de 640x480:

```
SDL_Surface *screen;

if (SDL_Init (0))
      {
            printf ("Error: %s\n", SDL_GetError ());
            exit (1);
      }

screen = SDL_SetVideoMode (640, 480, 16, SDL_HWSURFACE |
         SDL_DOUBLEBUF);
```

Insertar un título a nuestra pantalla y mostrar un icono en ella:

```
SDL_WM_SetCaption ("Prueba Inicial", NULL); //Título de la
ventana

SDL_WM_SetIcon(SDL_LoadBMP("imagenes/icono.bmp"),NULL);
//Colocar icono en ventana
```

Detección de eventos:

```
SDL_Event eventos;

// Detiene el programa hasta que recibe un evento
            if (SDL_WaitEvent (& eventos))
            …
```

Finalización de la librería y liberación de recursos:

```
SDL_Quit ();
```

- Insertar una imagen:

Una imagen es de tipo SDL_Surface, SDL soporta varios tipos de archivos de imagen. Pondremos un ejemplo de cómo crear una imagen y asignarle un archivo de tipo *.png:

```
SDL_Surface * imagen;
imagen = IMG_Load ("imagen.png");
```

Para que se muestre en nuestra pantalla, debemos asignarle una forma. Las imágenes suelen ser rectángulos, le asignamos las características de nuestra imagen: w → width (anchura) y h → height (altura) Así, a través del rectángulo podremos asignarle las coordenadas x e y que queremos para nuestra imagen y controlar su posición consultando estas coordenadas. También será útil que la imagen sea transparente, para ello podemos crear una imagen que tenga de fondo un color que asignaremos como color transparente. En nuestro ejemplo, hemos elegido el color rojo:

```
SDL_Rect rectangulo;
rectangulo.x=x;
rectangulo.y=y;
rectangulo.w = imagen->w;
rectangulo.h = imagen->h;

// Asignamos el color transparente al color rojo.
SDL_SetColorKey(imagen,SDL_SRCCOLORKEY|SDL_RLEACCEL,
SDL_MapRGB(imagen->format,255,0,0));

imagen=SDL_DisplayFormat(imagen);

//Relacionamos el rectangulo con nuestra imagen y nuestra
pantalla
SDL_BlitSurface(imagen, NULL, screen, &rectangulo);
```

También podemos determinar como píxel transparente uno en concreto de nuestra imagen, por ejemplo, el primer píxel (0,0):

```
Uint32 pixel_transparente = get_pixel (ima, 0, 0);
int fila;

/* busca un pixel opaco */
for (fila = 0; fila < ima->h; fila ++)
{
      if (pixel_transparente != get_pixel (ima, columna,
fila))
            return 0;
}
```

Eliminación de la imagen y liberación de recursos:

```
SDL_FreeSurface (imagen);
```

- Insertar sonido:

Para empezar y finalizar con el sonido:

```
if
(Mix_OpenAudio(MIX_DEFAULT_FREQUENCY,MIX_DEFAULT_FORMAT,1,40
96) < 0) {
            printf("No se pudo iniciar SDL_Mixer: %s\n",
            Mix_GetError());
            return 1;
     }

   atexit(Mix_CloseAudio);
```

Podemos poner una música de fondo y generar sonidos por sucesos que ocurran en nuestro juego. SDL soporta numerosos tipos de archivos:

```
Mix_Music *musica;

Mix_Chunk *explosion;

musica = Mix_LoadMUS("sounds/musica2.mid");
explosion = Mix_LoadWAV("sounds/explosion.wav");
```

Para que suene una u otra, tenemos diferentes canales:

```
Mix_PlayMusic(musica,-1);
Mix_PlayChannel(-1,explosion,0);
```

- Insertar texto desde un archivo de fuentes TrueType:

En primer lugar debemos crear una instancia de tipo fuente. Después tendremos que inicializar la librería, asignarle a la instancia un archivo de fuentes TrueType y definir el estilo que queremos para nuestra fuente:

```
TTF_Font * font;

TTF_Init();

font = TTF_OpenFont("comic.TTF", 18);

TTF_SetFontStyle (font, TTF_STYLE_NORMAL);
```

Los posibles estilos son los siguientes: TTF_STYLE_NORMAL para estilo normal, TTF_STYLE_BOLD para estilo negrita, TTF_STYLE_ITALIC, para estilo itálica, y TTF_STYLE_UNDERLINE para estilo subrayado.

El texto para que se muestre por pantalla debe tratarse como una imagen, para ello, utilizamos la función TTF_RenderText_Solid:

```
SDL_Surface * texto;
texto = TTF_RenderText_Solid(font,temp.c_str(), color);
```

donde temp es de tipo string y color es de tipo SDL_Color:

```
SDL_Color color;

color.r=23;
color.g=80;
color.b=250;
```

Para terminar con la librería, tendremos que escribir:

```
TTF_CloseFont(font);

TTF_Quit();
```

- Reconocimiento de los eventos de ratón:

Primero debemos habilitar el ratón:

```
SDL_ShowCursor(SDL_ENABLE);
```

Para saber en qué posición se encuentra el cursor tendremos dos variables de tipo entero que pasaremos como punteros a la función SDL_GetMouseState. Será de la siguiente forma:

```
int ratonx,ratony;
SDL_GetMouseState(&ratonx, &ratony);
```

Tras esto, para detectar que el botón ha sido pulsado se comprueba que lo siguiente es verdadero:

```
SDL_BUTTON (1)
```

- Ejemplos completos:

  - Hola mundo:

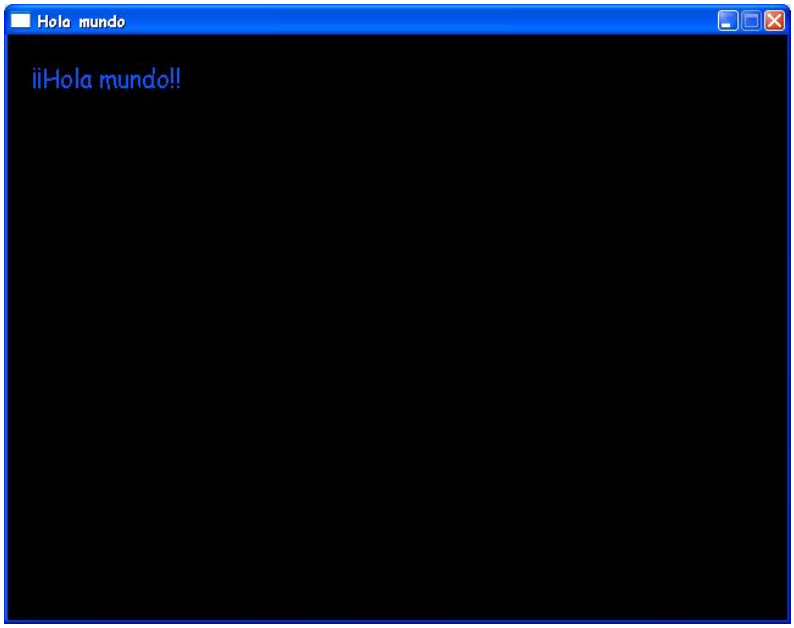

Figura 6.1: Pantalla "Hola Mundo"

El código que deberíamos escribir para obtener la pantalla anterior es el siguiente:

```
#include "SDL/SDL.h"
#include "SDL/SDL_image.h"
#include "SDL/SDL_ttf.h"
#include <string>


using std::string;

int main (int argc, char * argv []) {

     SDL_Surface * screen;
     TTF_Font * font;
     SDL_Color color;

     SDL_Surface * texto;

     string temp;

     if (SDL_Init (0))
        {
             printf ("Error: %s\n", SDL_GetError ());
             exit (1);
        }

        screen = SDL_SetVideoMode (640, 480, 16,
        SDL_HWSURFACE|SDL_DOUBLEBUF);

        SDL_WM_SetCaption ("Hola mundo", NULL);
```

```
        TTF_Init();

        font = TTF_OpenFont("comic.ttf", 20);

        color.r=23;
        color.g=80;
        color.b=250;

        TTF_SetFontStyle (font, TTF_STYLE_NORMAL);

        SDL_Rect dest;
        dest.x = 20;
        dest.y = 20;

        SDL_Event event;


        temp="¡¡Hola mundo!!";
        texto = TTF_RenderText_Solid(font,temp.c_str(),
        color);
        SDL_BlitSurface(texto, NULL, screen, &dest);

        while (SDL_WaitEvent(&event)){
           if (event.type == SDL_QUIT)
              break;
        }
        SDL_Quit();
        return 0;
}
```

- Imagen que puede moverse mediante los cursores:

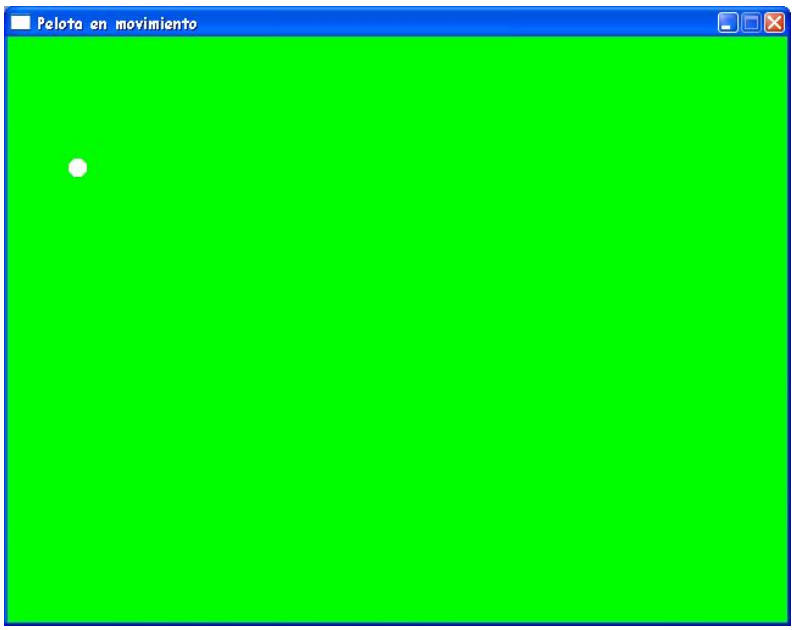

Figura 6.2: Pantalla “Pelota en movimiento”

Tendremos que escribir lo siguiente:

```
#include "SDL/SDL.h"
#include "SDL/SDL_image.h"

int main (int argc, char * argv [])
{

     SDL_Surface * screen;

     int baj;
     int der;
     int der2;
     int der3;

     int xpos;
     int ypos;

     xpos = 50;
     ypos = 100;

     int xpos2;
     int ypos2;

     int xpos3;
     int ypos3;

     SDL_Surface *pelota;
     SDL_Surface *back;

     SDL_Event event;

     int done = 0;


     if (SDL_Init (0))
      {
             printf ("Error: %s\n", SDL_GetError ());
             exit (1);
      }

      screen = SDL_SetVideoMode (640, 480, 16,
      SDL_HWSURFACE|SDL_DOUBLEBUF);

      SDL_EnableKeyRepeat(SDL_DEFAULT_REPEAT_DELAY,
      SDL_DEFAULT_REPEAT_INTERVAL);

      SDL_WM_SetCaption ("Pelota en movimiento", NULL);


      SDL_Rect dest;


      pelota = SDL_LoadBMP("pelota.bmp");
      back = IMG_Load("bg.png");

      SDL_BlitSurface(back, NULL, screen, &dest);

      SDL_Flip(screen);
      SDL_PollEvent(NULL);
```

```
      while(done == 0 )
      {
         if (SDL_WaitEvent(&event))
         {
            if ( event.key.keysym.sym == SDLK_ESCAPE )  {
               done = 1;
           }
            if (baj ==1){
                 if (ypos < 460){
                      ypos += 2;
                 }else{ baj = 0; }
            }

            if (der ==1){
                 if (xpos < 620){
                      xpos += 2;
                 }else{ der = 0; }
            }

            if (der == 0){
                 if (xpos > 0){
                      xpos -= 2;
                 }else{ der = 1; }
            }
            if (baj == 0){
                 if (ypos > 100){  //Limite superior
                      ypos -= 2;
                 }else {baj = 1;}
            }

            dest.x = 0;
            dest.y = 0;
            SDL_BlitSurface(back, NULL, screen, &dest);

            dest.x = xpos;
            dest.y = ypos;

            SDL_BlitSurface(pelota, NULL, screen, &dest);

            SDL_Flip(screen);

            if ( event.type == SDL_KEYDOWN )
            {
            if ( event.key.keysym.sym==SDLK_UP ) baj = 0;

            if ( event.key.keysym.sym==SDLK_DOWN ) baj = 1;

            if ( event.key.keysym.sym==SDLK_RIGHT) der = 1;

            if ( event.key.keysym.sym==SDLK_LEFT ) der = 0;
            }
         }
       }
       SDL_Quit();
       return 0;

}
```

6.2 Manual de Usuario

A continuación, mostraremos una guía para cada una de las pantallas de nuestro juego.

La siguiente figura muestra la secuencia normal a seguir hasta llegar a la primera Etapa. Tras terminar todas las etapas necesarias, podremos ver una etapa final que a modo de presentación explicará cómo podríamos conectarnos a MySQL para crear nuestra propia base de datos.

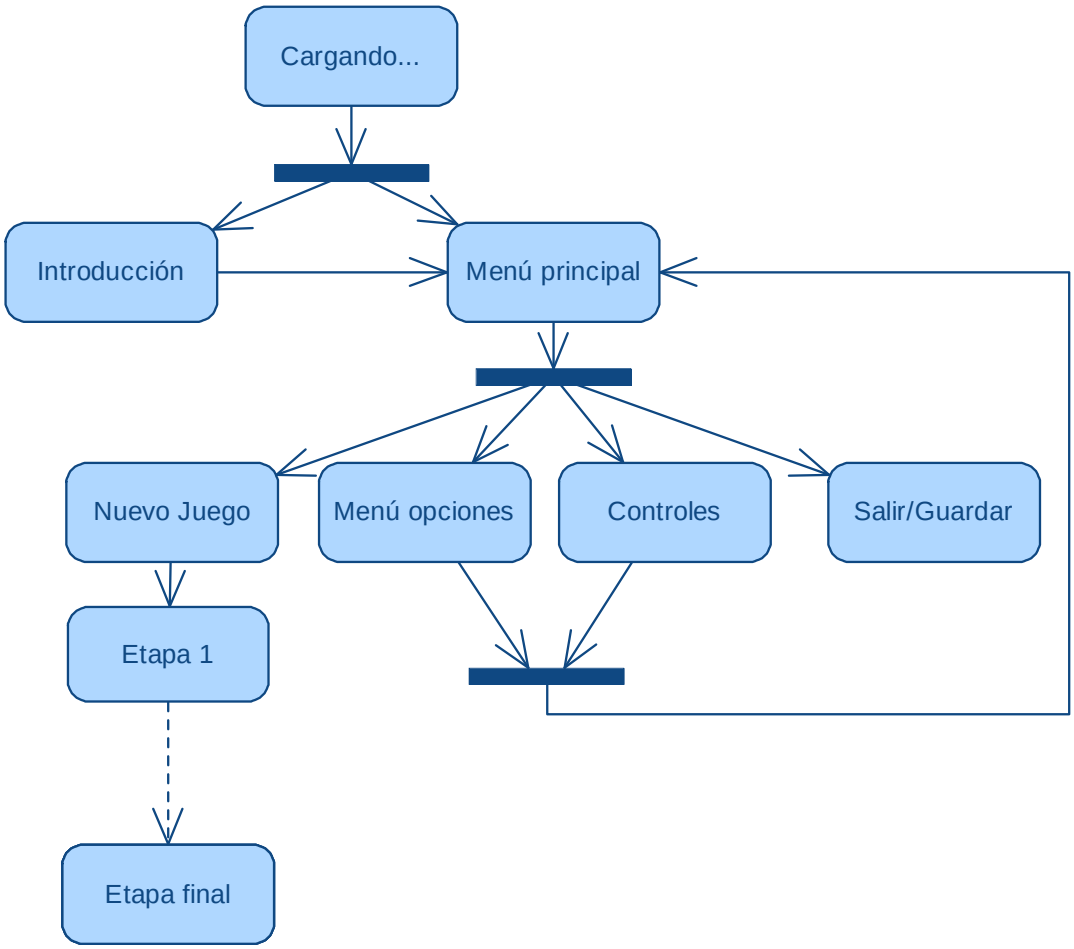

Figura 6.3: Secuencia de ejecución I

Al comenzar el juego se podrá ver una pantalla que indica que el juego está Cargando. A continuación, podrá verse una pequeña introducción al juego:

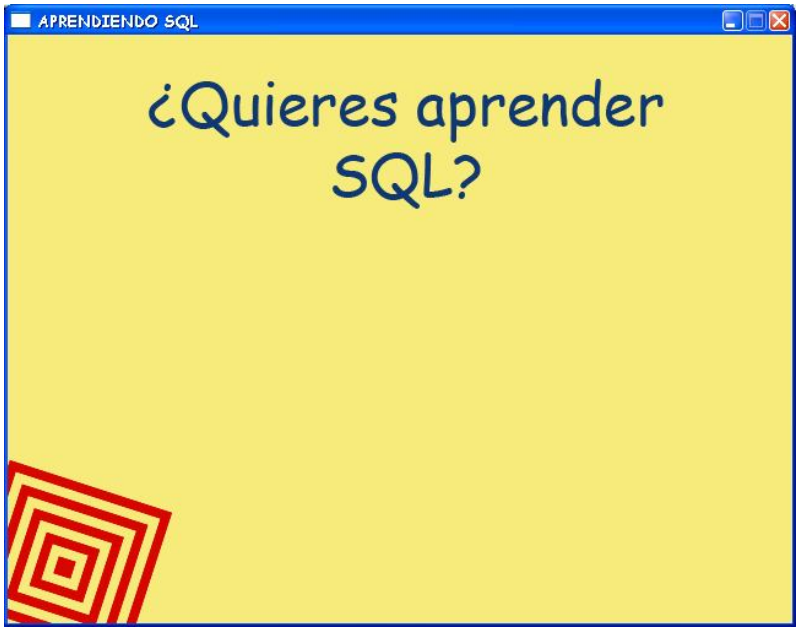

Figura 6.4: Pantalla de Introducción

La introducción puede interrumpirse pulsando Escape, Intro o Espacio. Tanto si esperamos como si decidimos terminarla antes de que acabe, se accede al menú principal:

Figura 6.5: Menú principal

Para acceder a las diferentes opciones se debe mover la bolita roja con los cursores de arriba y abajo. Cuando estemos situados sobre la opción que queremos, accederemos a ella pulsando Intro o Espacio.

Al seleccionar "OPCIONES", entraremos en la pantalla de Menú de opciones:

Figura 6.6: Menú de Opciones

El funcionamiento de este menú es igual que el anterior. En él podremos seleccionar el tipo de pantalla que queremos, podremos escoger entre 640x480 pixeles y pantalla completa. También se podrá escoger si queremos escuchar tanto la música como los efectos de sonido. Si pulsamos Escape o seleccionamos "Salir y guardar" volveremos al Menú principal. En un caso saldremos sin guardar los cambios y en el otro se guardarán.

Si en el Menú Principal pulsamos sobre "CONTROLES", accederemos a la página donde nos explican las teclas que usamos durante todo el juego. Para activar y desactivar la música de fondo del juego se utiliza F2 y para los efectos de sonido es F3. Esta pantalla podrá observarse en cada prueba pulsando F1:

Figura 6.7: Controles

Al pulsar Escape, volveremos al Menú principal o a la prueba donde estuviésemos.

Si en el Menú principal pulsamos sobre "Nuevo Juego", accederemos a la siguiente pantalla:

Figura 6.8: Pantalla para introducir el usuario

En ella se muestran aquellos usuarios que han grabado su partida en el juego. Podremos introducir un nombre ya existente o uno nuevo. Si se pulsa Cancelar o

Escape, volveremos al Menú Principal. Al pulsar Aceptar o Intro, se podrá comenzar a jugar. Si hemos introducido un usuario nuevo se accederá a la pantalla de presentación del lenguaje SQL:

Figura 6.9: Introducción al lenguaje SQL

Para salir de ella podremos pulsar Intro y comenzar la primera etapa de pruebas o Escape para volver al Menú principal.

Si el usuario que ingresamos ya estaba en la lista directamente accederemos a la pantalla de la primera fase de pruebas:

Figura 6.10: Primera Etapa de Pruebas

La secuencia normal de las etapas es la siguiente. Como indicamos en la siguiente figura:

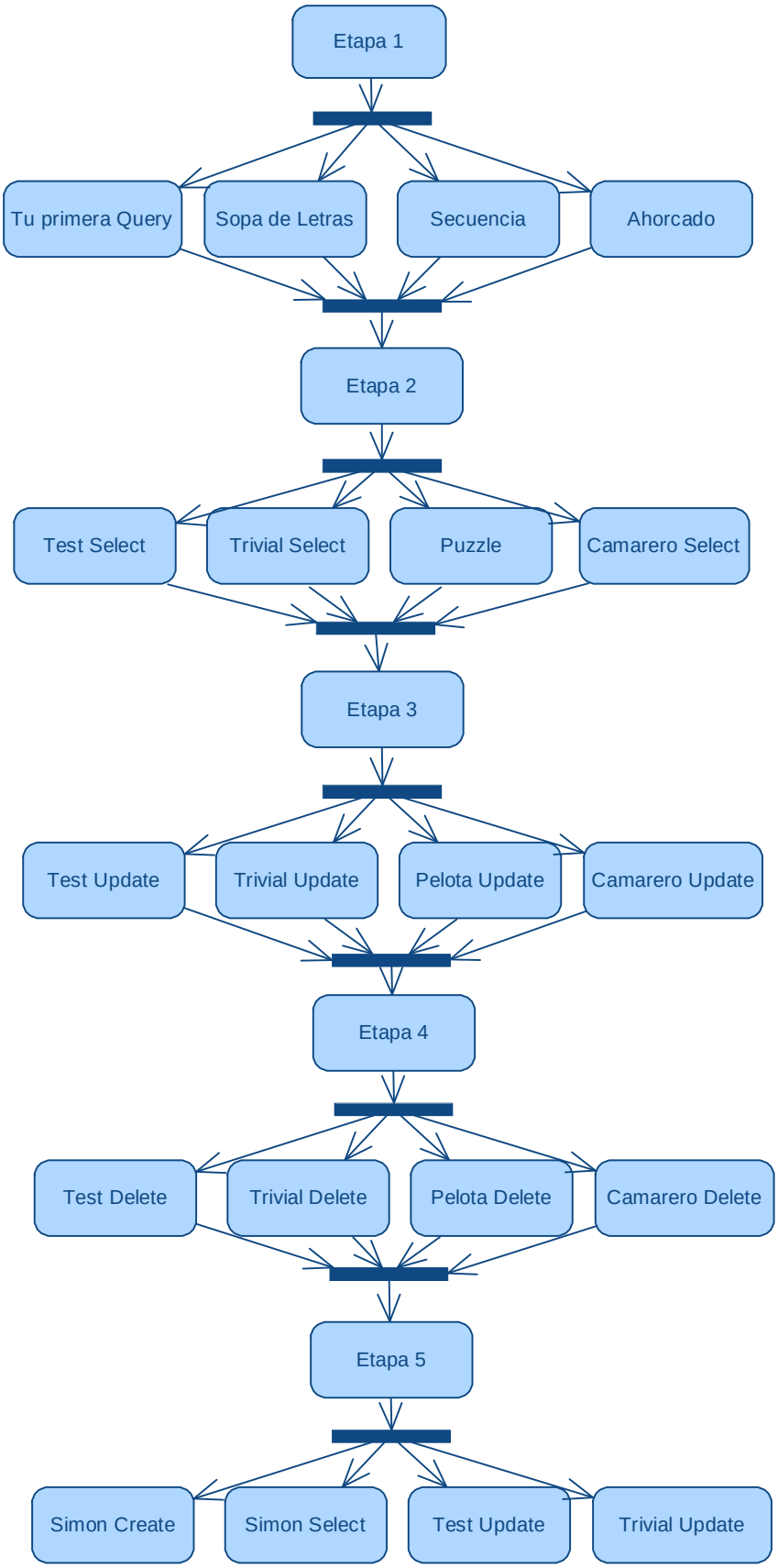

Figura 6.11: Secuencia de ejecución II

Al acceder a cada una de las pantallas del gráfico anterior, primero aparecerá la ventana que aparece en la figura 6.8 con una explicación diferente para cada una de ellas.

Dentro de cada etapa, el funcionamiento es el mismo. Pulsando sobre cada botón de las pruebas se accede a ella. Al pulsar sobre el emoticono central se accede a las pantallas de entretenimiento.

En cada etapa, si pulsamos sobre  , guardaremos la partida, es decir, las pruebas que han sido superadas hasta ese momento.

Cuando alguna prueba ha sido superada, aparece rodeada con un círculo rojo y ya no podrá jugarse de nuevo:

Figura 6.12: Ejemplo de prueba superada

Para ir a la siguiente etapa, pulsaremos “X”, para ir a la etapa anterior, habrá que pulsar “Z”.

Ahora, explicaremos el funcionamiento de cada una de las pruebas.

La primera prueba de la primera etapa consiste en ordenar una secuencia. Se pulsa sobre cada palabra en el orden que elijamos y pulsamos Aceptar para comprobar si es correcto:

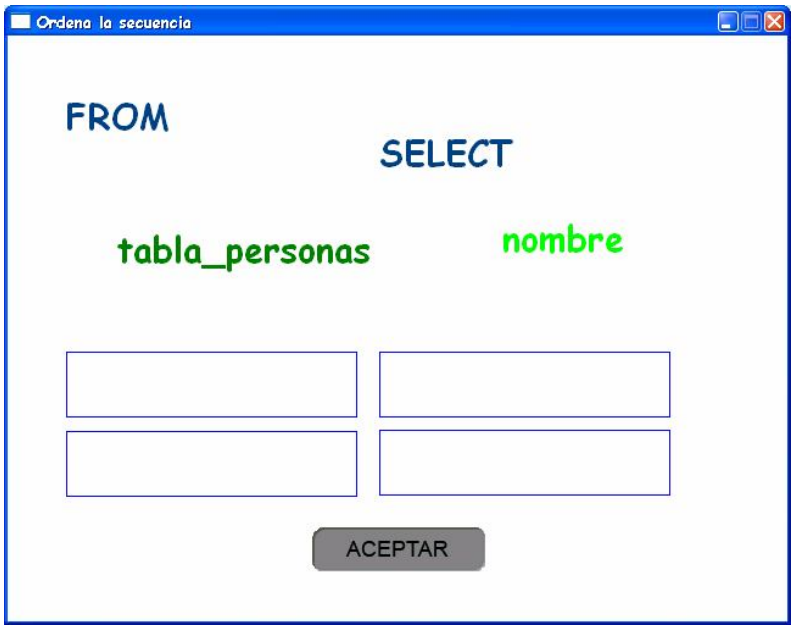

Figura 6.13: Tu primera Query

En la sopa de letras, tendremos que encontrar las palabras que aparecen en la parte derecha de la pantalla. Se seleccionarán pulsando con el ratón sobre las letras:

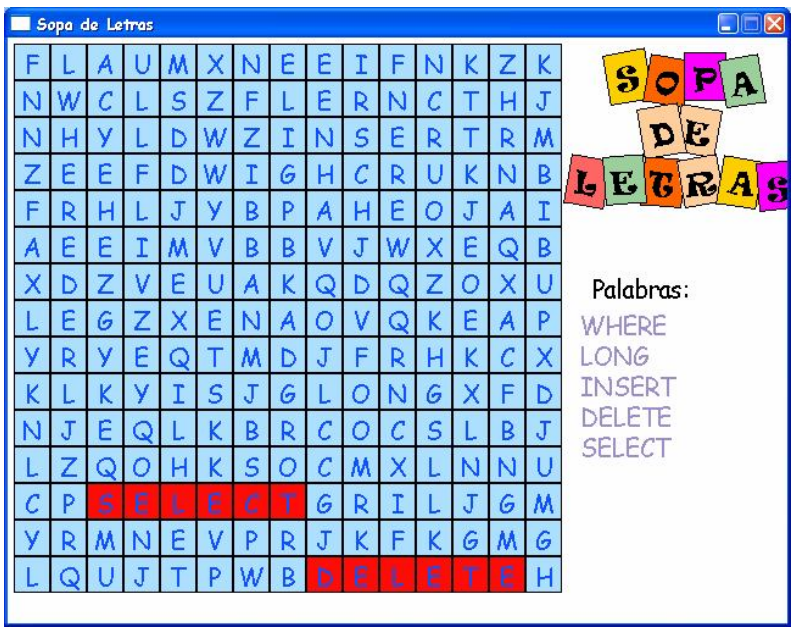

Figura 6.14: Sopa de letras

Para la prueba de la Secuencia, tendremos que levantar las cartas en el orden correcto para formar una secuencia. Para levantar una carta, habrá que pulsar sobre ella con el ratón:

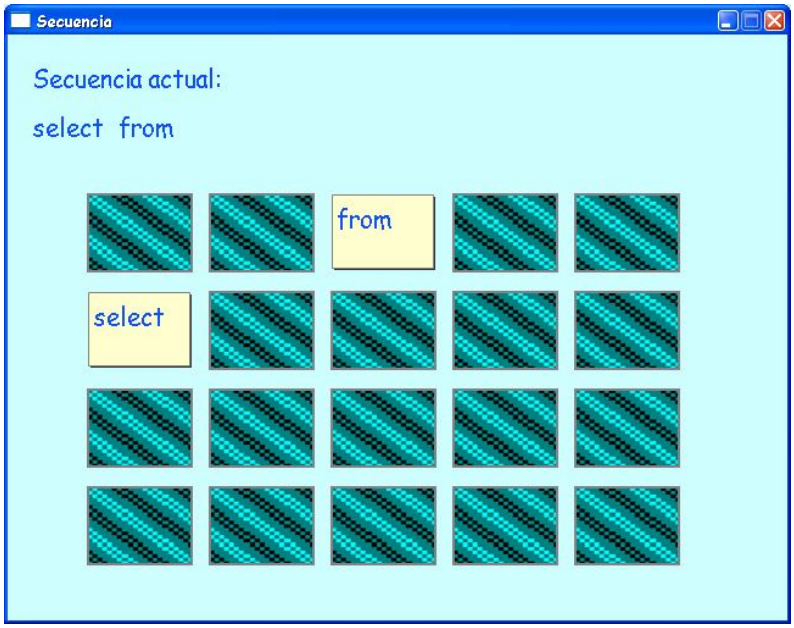

Figura 6.15: Secuencia

En el ahorcado, tendremos que adivinar la palabra antes de que el muñeco se complete. Para ello, introduciremos una letra que aparecerá en el recuadro superior y pulsaremos sobre Aceptar para comprobar si está en la palabra a descubrir.

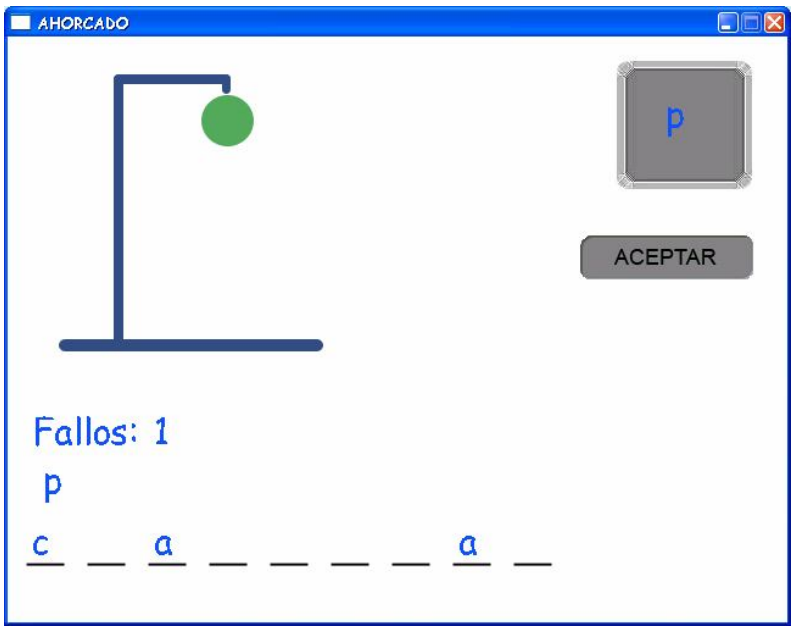

Figura 6.16: Ahorcado

El entretenimiento de la primera fase de pruebas es una pantalla de acertijos. Para saber la solución hay que pulsar sobre el botón con la interrogación. Como ejemplo de prueba de entretenimiento mostraremos la de la quinta fase:

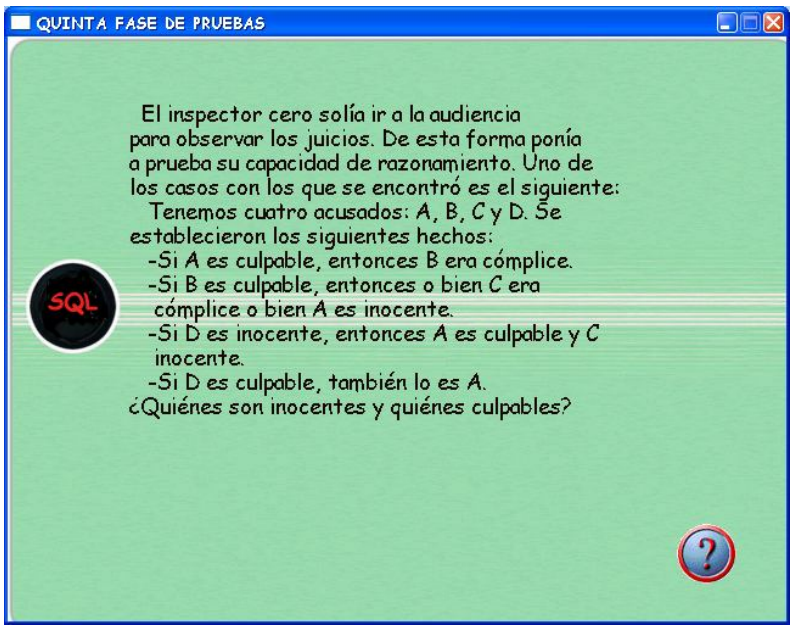

Figura 6.17: Enigmas

Las pantallas de entretenimiento no son obligatorias para poder ver la etapa final del juego.

Al pasar de una etapa a otra, se muestra por unos segundos una pantalla de Paso de etapa:

Figura 6.18: Paso de etapa

La segunda fase de pruebas es como sigue:

Figura 6.19: Segunda Etapa de Pruebas

En el Test, tendremos que marcar aquellas que creamos que son correctas y comprobarlo pulsando sobre Aceptar. Se mostrarán aquellas que están bien o mal.

Figura 6.20: Test

El juego de Simon consiste en seguir el orden en el que se van encendiendo las palabras:

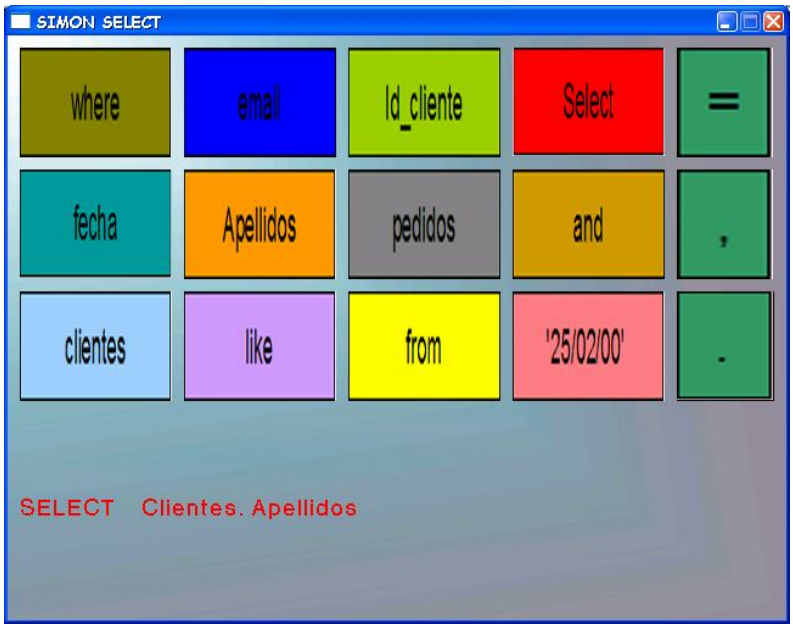

Figura 6.21: Simon

En el Camarero, tendremos que recoger las palabras en el orden correcto para formar una secuencia:

Figura 6.22: Camarero

Para jugar al Trivial, debemos pulsar sobre las palabras en el orden correcto para formar la secuencia que nos pide. Si en algún momento nos equivocamos, al pulsar 'R' podremos comenzar la secuencia que estábamos formando:

Figura 6.23: Trivial

La tercera Fase de Pruebas es de la siguiente forma:

Figura 6.24: Tercera Etapa de Pruebas

El funcionamiento del Test, el Trivial y el Camarero es el mismo que en la etapa anterior.

Para jugar a la Pelota Loca tendremos que esquivar las líneas negras para alcanzar las palabras en el orden correcto. Disponemos de 10 vidas para conseguirlo:

Figura 6.25: Pelota Loca

La cuarta etapa tiene las siguientes pruebas:

Figura 6.26: Cuarta Etapa de Pruebas

El funcionamiento de las pruebas en esta etapa es igual que en las etapas anteriores.

La quinta Fase tiene la siguiente forma:

Figura 6.27: Quinta Etapa de Pruebas

Para resolver el Puzzle debemos mover la casilla que queramos hacia el hueco que aparece. En la parte inferior aparece la secuencia que tendrá que aparecer al ordenar todas las piezas. Si la prueba se supera, se podrá leer un chiste. No es obligatorio superar esta prueba para poder ver la etapa final del juego:

Figura 6.28: Puzzle

En todo momento del juego si se pulsa Escape dentro de las pantallas de las etapas, aparece una pantalla preguntando si se desea salir del juego:

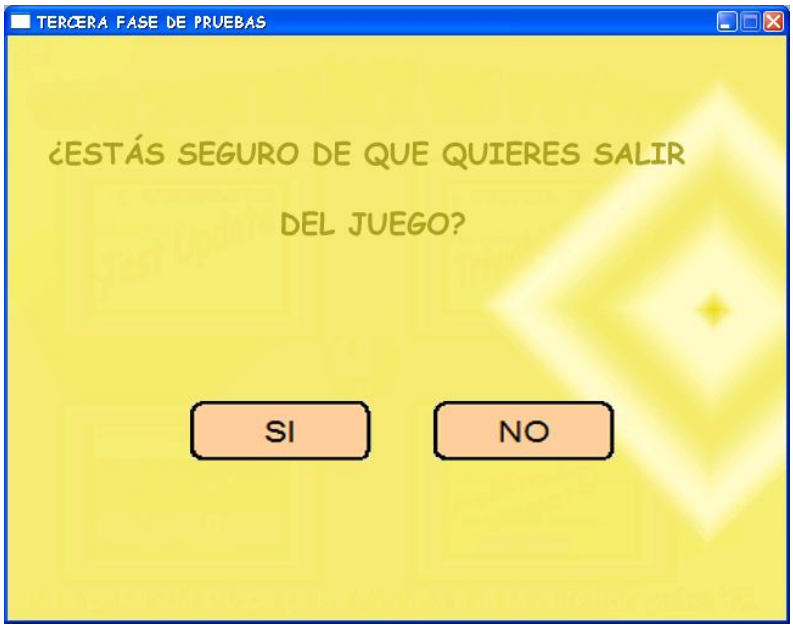

Figura 6.29: Salir del juego

Si se pulsa sobre No, vuelve a la primera Fase de Pruebas. Si, por el contrario, se pulsa Intro o sobre Sí, muestra la pantalla siguente pantalla:

Figura 6.30: Guardar la partida

Con ella guardamos la partida si elegimos Aceptar. También podemos acceder a ella desde el Menú Principal seleccionando la opción de “Salir”.

Como etapa final, hemos querido mostrar cómo podría crearse una tabla y realizar consultas y modificaciones sobre ella mediante MySQL.

Tras superar todas las pruebas necesarias, si en la etapa 5 pulsamos ‘X’, accederemos a la siguiente ventana:

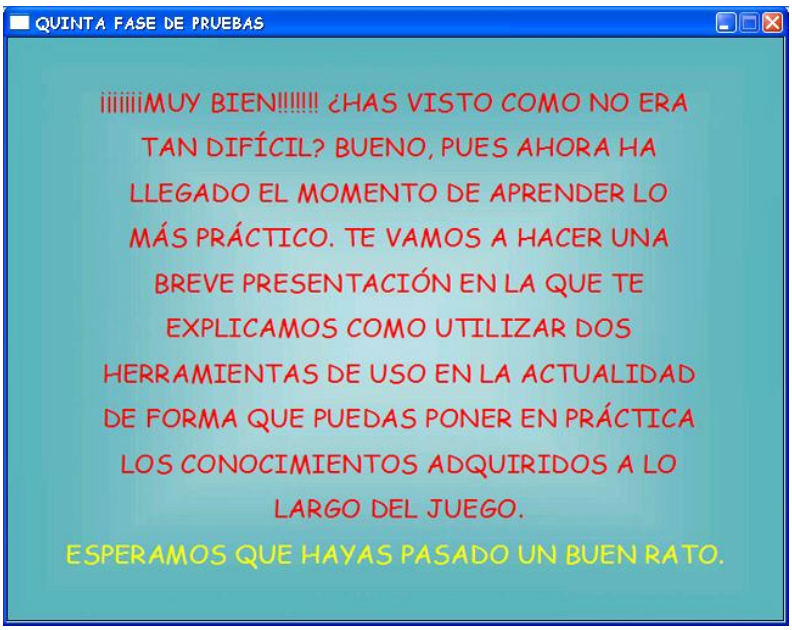

Figura 6.31: Paso a etapa final

Al pulsar Intro, podremos comenzar la etapa final, si queremos regresar a la etapa 5, pulsaremos Escape.

Esta última etapa tiene un carácter meramente ilustrativo. Se trata con ello de dar al usuario del juego una visión diferente de SQL. La ventana a la que se accede es la siguiente:

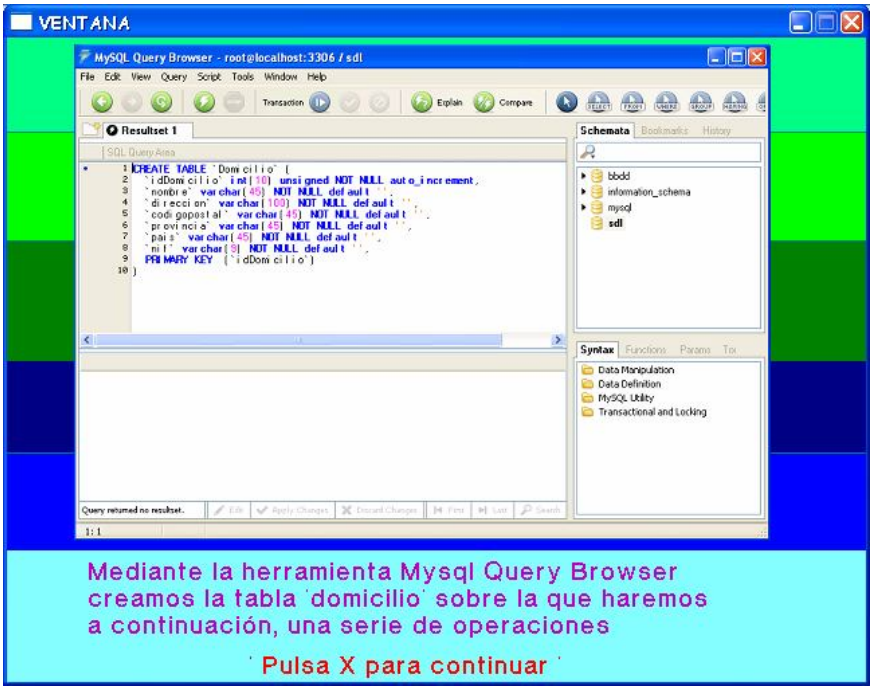

Figura 6.32: Etapa final

En primer lugar mostramos una imagen de la herramienta MySQL Query Browser donde se puede ver una consulta mediante la cual se realiza la creación de una tabla.

En la siguiente imagen se muestra la herramienta MySQL Administrator, donde aparece la estructura de la tabla creada anteriormente.

En pantallas posteriores se realizan diferentes operaciones sobre la tabla, todas ellas usando MySQL Query Browser .Estas operaciones son, una consulta Insert para introducir datos en la tabla, una consulta Update y una consulta Delete todas ellas seguidas de una consulta Select para ir mostrando los cambios que sufren los datos de la tabla.

Al terminar esta guía, el juego habrá acabado, mostraremos una pantalla como despedida en la cual si pulsamos Escape volveremos a la etapa 5 y si pulsamos Intro irá a la pantalla para guardar.

Figura 6.33: Pantalla Fin de Juego

7. Conclusiones

En este capítulo se detallarán las conclusiones obtenidas tras el desarrollo del presente proyecto.

Partiendo de los requisitos iniciales, se puede verificar que el resultado final cumple con los objetivos propuestos.

En primer lugar podemos decir que el resultado obtenido es una herramienta que permite el aprendizaje del lenguaje SQL con una relativa facilidad y con una variedad de pruebas que hacen que el usuario adquiera una serie de conocimientos de forma estructurada, ampliando la dificultad de las pruebas según se avanza el juego.

Otro de los objetivos del proyecto era el uso de herramientas gratuitas, con licencias de libre distribución. Para esto se realizó una etapa previa de investigación para buscar cuales podrían ser las herramientas idóneas que se ajustaran a nuestras necesidades para el desarrollo del proyecto. Una vez superada esta etapa nos decidimos por el uso de C++ y la librería para el desarrollo de videojuegos SDL.

La ejecución de la herramienta ha sido diseñada y desarrollada de tal forma que la realización de ampliaciones se pueda ejecutar sin necesidad de realizar ningún cambio a nivel estructural.

También podemos afirmar que los recursos necesarios para la ejecución de nuestro juego educativo en un PC son mínimos cumpliendo con lo esperado en los objetivos.

A nivel personal, se han adquirido conocimientos del lenguaje C++ que previamente no se tenían, así como conocimientos relativos a la librería SDL. Además se han afianzado y ampliado conocimientos sobre bases de datos y el lenguaje SQL.

Durante la etapa final del desarrollo del proyecto empezamos a pensar posibles ampliaciones o mejoras para una segunda parte del videojuego. Podríamos enumerar algunas de ellas:

- Ampliar el número de pruebas relativas a la enseñanza de SQL para poder crear una base de datos completa.
- Realizar las modificaciones pertinentes para además de poder ejecutar el juego en un entorno Windows pueda ejecutarse bajo Linux.

- Trasladar la idea principal del juego a una aventura gráfica en la que un personaje fuese encontrando pistas del lenguaje para aplicar a retos que se plantean en su camino.

8. Bibliografía

Información general, reseñas históricas sobre videojuegos:

www.wikipedia.es

http://tecnologiaedu.us.es/edutec/paginas/19.html

http://www.usal.es/~teoriaeducacion/rev_numero_02/n2_art_etxeberria.htm

http://ares.cnice.mec.es/informes/02/documentos/indice.htm

Sobre SDL y videojuegos:

www.libsdl.org

http://gpwiki.org/

http://www.losersjuegos.com.ar/principal/principal.php

http://www.wired-weasel.com/users/serhid/blog/?page_id=4

http://x-ezine.todo-linux.com/x1/1x003-sdl.html

Sobre C++:

http://arco.infcr.uclm.es/~david.villa/pensar_en_C++/products/vol1/index.html

http://www.codersource.net/codersource_cppprogramming.html

http://www.cppgameprogramming.com/cgi/nav.cgi?page=index

Sobre SQL:

http://www.asptutor.com/zip/sql.pdf

9. Anexos

9.1 Configuración del entorno de desarrollo

Mostraremos una pequeña guía de Instalación de Dev-C++ con la librería gráfica utilizada para su utilización en un entorno Windows.

1. Instalación de Dev-C++

Instalamos Dev-C++ en el directorio c:/dev-cpp, para ello podemos descargarlo de su página:

http://www.bloodshed.net/dev/devcpp.html

Nosotros hemos usado la versión "*Dev-C++ 5.0 beta 9.2 (4.9.9.2)*" con Mingw/GCC 3.4.2 que incluye el compilador Mingw con GCC y GDB (depurador)

El enlace directo es el siguiente:

http://prdownloads.sourceforge.net/dev-cpp/devcpp-4.9.9.2_setup.exe

2. Inclusión de las bibliotecas SDL:

SDL tiene varias bibliotecas, todas ellas pueden encontrarse en la página:

www.libsdl.org

Hemos necesitado las siguientes bibliotecas:

- SDL (básica)
- SDL_mixer
- SDL_image
- SDL_ttf

Necesitaremos 2 versiones de cada biblioteca:

- La versión de ejecución (.dll en Windows)
- La versión de desarrollo

- SDL (básica)

- Descarga:

La página principal desde donde podemos descargarla para Windows y para otros sistemas operativos es:

http://www.libsdl.org/download-1.2.php

La versión de ejecución se descarga directamente de:
http://www.libsdl.org/release/SDL-1.2.11-win32.zip

La versión de desarrollo:
http://www.libsdl.org/release/SDL-devel-1.2.11-mingw32.tar.gz

- Instalación:

Descomprimimos ambos archivos.

En el primero, la versión de ejecución, al descomprimir obtendremos el archivo SDL.dll, se debe copiar a la carpeta System dentro de la carpeta principal de Windows, generalmente la ruta completa es "c:\windows\system\"

En la versión de desarrollo, al descomprimir tendremos varios archivos y carpetas:
- Debemos copiar el contenido de la carpeta \lib en "c:\dev-cpp\lib\"
- Entraremos en la carpeta include y copiaremos todos los archivos .h dentro de "c:\dev-cpp\include\SDL"

- SDL_mixer

- Descarga:

La página principal desde donde podemos descargarla para Windows y para otros sistemas operativos es:

http://www.libsdl.org/projects/SDL_mixer/

La versión de ejecución se descarga directamente de:
http://www.libsdl.org/projects/SDL_mixer/release/SDL_mixer-1.2.7-win32.zip

La versión de desarrollo:
http://www.libsdl.org/projects/SDL_mixer/release/SDL_mixer-devel-1.2.7-VC6.zip

- Instalación:

Descomprimimos ambos archivos.

En el primero, la versión de ejecución, al descomprimir obtendremos los siguientes archivos .dll: SDL_mixer.dll, ogg.dll, smpeg.dll, vorbis.dll y vorbisfile.dll, se deben copiar a la carpeta System dentro de la carpeta principal de Windows.

En la versión de desarrollo, al descomprimir tendremos varios archivos:
- Debemos copiar SDL_mixer.h en c:\dev-cpp\include\SDL
- El archivo SDL_mixer.lib dentro de c:\dev-cpp\lib

- SDL_image

- Descarga:

La página principal desde donde podemos descargarla para Windows y para otros sistemas operativos es:

http://www.libsdl.org/projects/SDL_image/

La versión de ejecución se descarga directamente de:
http://www.libsdl.org/projects/SDL_image/release/SDL_image-1.2.5-win32.zip

La versión de desarrollo:
http://www.libsdl.org/projects/SDL_image/release/SDL_image-devel-1.2.5-VC6.zip

- Instalación:

Descomprimimos ambos archivos.

En el primero, la versión de ejecución, al descomprimir obtendremos los siguientes archivos .dll: SDL_image.dll, jpeg.dll, libpng12.dll, libtiff.dll y zlib1.dll, se deben copiar a la carpeta System dentro de la carpeta principal de Windows.

En la versión de desarrollo, al descomprimir tendremos varios archivos:
- Debemos copiar SDL_image.h en c:\dev-cpp\include\SDL
- El archivo SDL_image.lib dentro de c:\dev-cpp\lib

- SDL_ttf

- Descarga:

La página principal desde donde podemos descargarla para Windows y para otros sistemas operativos es:

http://www.libsdl.org/projects/SDL_ttf/

La versión de ejecución se descarga directamente de:
http://www.libsdl.org/projects/SDL_ttf/release/SDL_ttf-2.0.8-win32.zip

La versión de desarrollo:
http://www.libsdl.org/projects/SDL_ttf/release/SDL_ttf-devel-2.0.8-VC6.zip

- Instalación:

Descomprimimos ambos archivos.

En el primero, la versión de ejecución, al descomprimir obtendremos el siguiente archivo .dll: SDL_ttf.dll, se debe copiar a la carpeta System dentro de la carpeta principal de Windows.

En la versión de desarrollo, al descomprimir tendremos varios archivos:
- Debemos copiar SDL_ttf.h en c:\dev-cpp\include\SDL
- El archivo SDL_ttf.lib dentro de c:\dev-cpp\lib

3. Configuración de la compilación de Dev-C++

Cuando creamos un proyecto, para que su ejecución con las librerías sea correcta, tendremos que definir los parámetros de compilación para vincular las bibliotecas al programa.

Abrimos el menú "Herramientas -> Opciones del Compilador" ("Tools" -> "Compiler Options") y seleccionamos la opción:

"Añadir estos comandos a la línea de comandos del Linker" (Add these commands to the linker...)

Luego se debe completar el campo siguiente con:

→ Para SDL básica:

```
-lmingw32 -lSDLmain –lSDL
```

→ Para el resto de bibliotecas:

```
-lSDL_image -lSDL_mixer –lSDL_ttf
```

Quedaría de la siguiente forma:

Figura 9.1: Configuración de Dev-C++ para SDL

9.2 Conexión con MySQL

En primer lugar se debe obtener el paquete de instalación desde el servidor en Internet: http://www.mysql.com/, y después instalarlo en nuestro ordenador.

Se puede descargar el servidor MySQL y los clientes estándar directamente desde este enlace: http://www.mysql.com/downloads/index.html. Hay que elegir la versión que se quiere descargar, preferiblemente la recomendada, ya que suele ser la más actualizada y estable.

Una vez seleccionada la versión, hay que elegir la distribución adecuada al sistema operativo que usemos: Windows, Linux, Solaris, etc. El fichero descargado se podrá instalar directamente.

Terminar de instalar el servidor depende en gran medida del sistema operativo que vayamos a utilizar.

Explicaremos la forma de instalarlo para WINDOWS.

En el momento de escribir estas líneas, la versión recomendada es la 5.0.27. El fichero que hay que descargar se llama "mysql-5.0.27-win32.zip".

Después de descargarlo se descomprime y se ejecuta el fichero "setup.exe".

El proceso de instalación está muy mejorado con respecto a versiones anteriores, y bastará con seguir las indicaciones en cada pantalla.

- Instalación de librerías para Dev-C++

A continuación se detalla cual es el proceso a seguir para la instalación de las librerías necesarias para realizar la conexión entre la base de datos MySQL y Dev-C++.

1. Usar la opción Ayuda->Acerca de Dev-C++..., se mostrará este cuadro de diálogo:

Figura 9.2: Acerca de Dev-C++

2. Pulsar el botón "Buscar actualizaciones", se mostrará el cuadro de diálogo de actualizaciones:

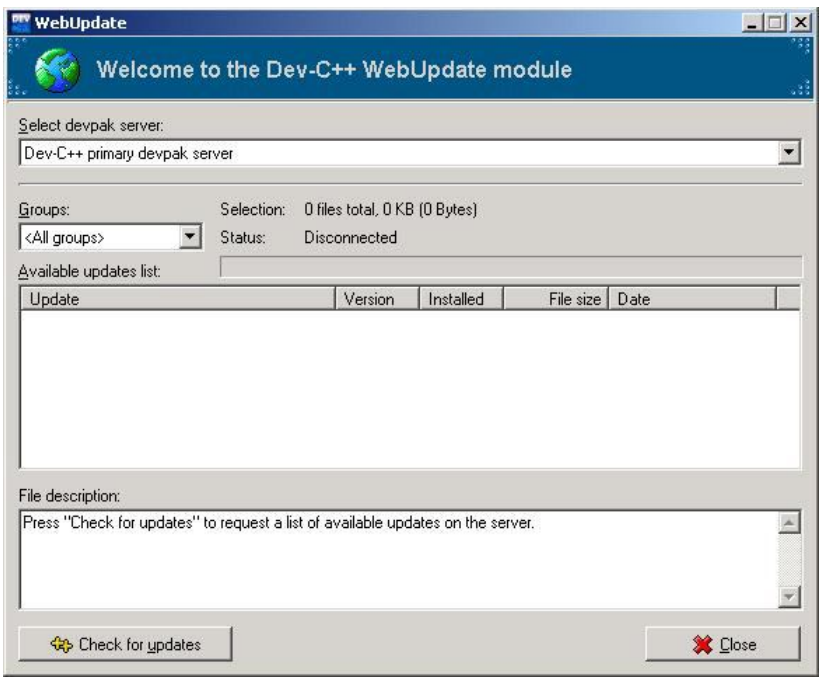

Figura 9.3: WebUpdate de Dev-C++

3. Pulsar el botón "Check for updates". Se leerá una lista de los paquetes disponibles.

4. Seleccionar el paquete MySQL y pulsar el botón "Download selected":

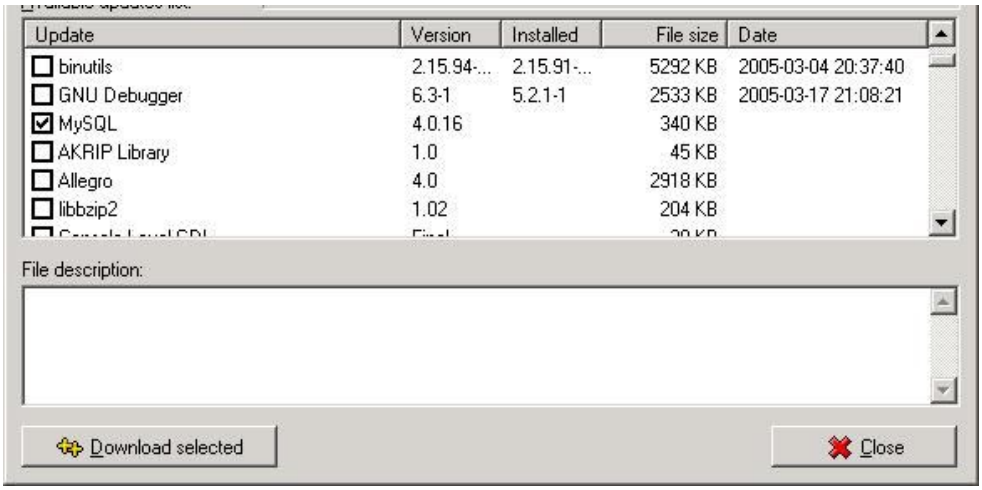

Figura 9.4: Descargar en Dev-C++

5. Se descargará el paquete desde Internet, y aparecerá este mensaje: "The updates you selected have been downloaded. Now they will be installed.", es decir, que se han descargado las actualizaciones y se procederá a instalarlas. Pulsamos "Ok".

6. Se abrirá otra ventana, con el "Installation Wizard", pulsamos "Install>":

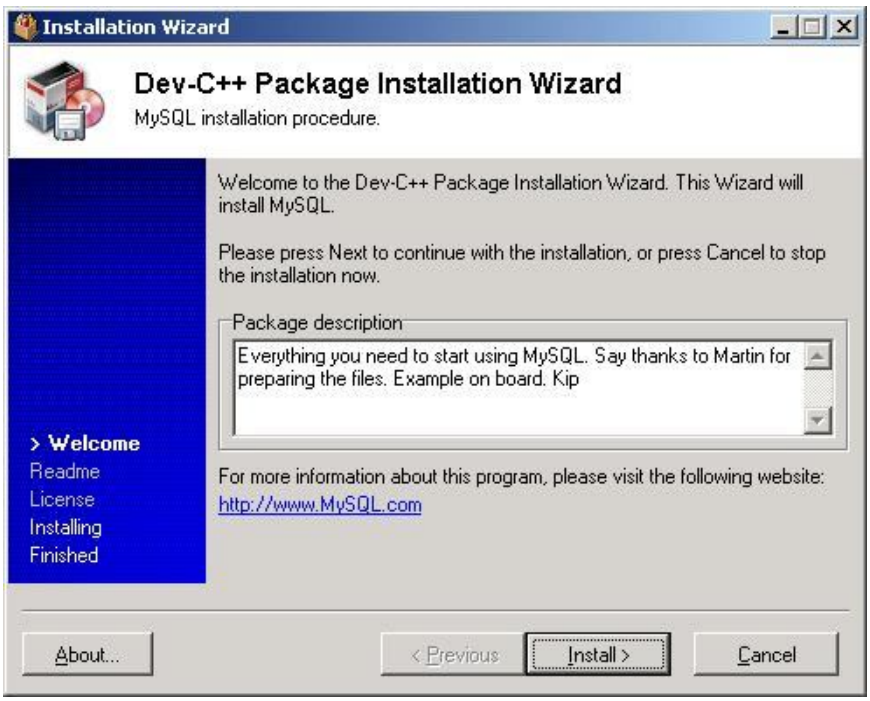

Figura 9.5: Instalación de MySQL

7. Cuando termine la instalación pulsamos "Finish", y se cerrará el "Installation Wizard".

Ahora tendremos un subdirectorio MySQL bajo el directorio “include” de Dev-C++, que contendrá los ficheros de cabecera ".h". En el directorio “lib” de Dev-C++ se habrán copiado los ficheros "libmySQL.a" y "libmySQL.def". Y en el directorio de “examples” de Dev-C++ habrá una carpeta con un proyecto de ejemplo MySQLClientTest.

- Primeros Auxilios (Manejo C++ y Mysql)

En primer lugar deberemos incluir la opción "-lmysql" en las opciones del linker dentro de las opciones de proyecto.

Figura 9.6: Configuración de Dev-C++ para usar MySQL y SDL

Para iniciar un objeto MYSQL dinámico, usaremos lo siguiente:

```
MYSQL *myData;

// Intentar iniciar MySQL:
if(!(myData = mysql_init(0))) {
   // Imposible crear el objeto myData
   return 1;
}
```

Una vez hemos inicializado el objeto MYSQL, intentaremos establecer una conexión con el servidor. Para ello usaremos la función mysql real connect()

Esta función necesita una serie de parámetros para su funcionamiento:

- Un puntero a un objeto MYSQL.
- El nombre del ordenador donde se está ejecutando el servidor. Puede ser también una dirección IP. Si el servidor está ejecutándose en la misma máquina que la aplicación, este nombre puede ser "localhost", o simplemente, un puntero nulo.
- El nombre del usuario. En sistemas Unix puede ser un puntero nulo, para indicar el usuario actual. En Windows ODBC debe especificarse.
- La contraseña del usuario seleccionado.
- Base de datos por defecto. Puede ser NULL si no queremos usar una base de datos determinada.
- Número de puerto. Generalmente usaremos la constante MYSQL_PORT.
- Socket Unix. Usaremos NULL para conexiones locales.

```
if(!mysql_real_connect(myData, NULL, "curso", "clave",
"prueba", MYSQL_PORT, NULL, 0)) {
      // No se puede conectar con el servidor en el puerto
      especificado.
      cout << "Imposible conectar con servidor mysql en el
            puerto "
           << MYSQL_PORT << endl;
      mysql_close(myData);
      return 1;

   }
```

Cuando hayamos terminado de trabajar con las bases de datos, cerramos la conexión con el motor de bases de datos, para eso usamos la función mysql_close().

Cerrar la conexión tiene un doble propósito. Por una parte, se cierra la conexión abierta con el servidor y por otra parte se libera la memoria correspondiente al objeto MYSQL.

```
// Cerrar la conexión
mysql_close(myData);
```

Para realizar una consulta o para realizar algún tipo de modificación sobre nuestra base de datos usaremos mysql_query():

```
mysql_query(myData, "SELECT * FROM TABLA");
```

Necesitaremos almacenar la información obtenida. Para ello es necesario la función mysql_store_result() y para liberar el resultado usaremos mysql_free_result():

```
// Almacenar el resultado de la consulta, lo necesitaremos
después:
if((res = mysql_store_result(myData))) {
   // Procesar resultados
   ...
   // Liberar el resultado de la consulta:
   mysql_free_result(res);
}
```

Cuando realicemos alguna modificación en los datos de alguna de las tablas de nuestra base de datos tenemos dos opciones:

Si la operación se realizó satisfactoriamente para confirmar esto se utiliza la función

```
mysql_commit(MYSQL *mysql)
```

Si por el contrario queremos anular los efectos producidos por la operación usaremos:

```
mysql_rollback(MYSQL *mysql)
```

Ambas funciones devuelven 0 en caso de éxito y distinto de 0 si ocurre algún error.

10. Autorización

Los autores de este proyecto, Sonia Martín Moreno, Francisco Javier Remesal Escalero y Laura Rivera Rodríguez, autorizamos a la Universidad Complutense de Madrid a difundir y utilizar, con fines académicos y no comerciales, la memoria, el código y el prototipo desarrollado sobre este Videojuego Educativo para el Aprendizaje de SQL.

Firma de los autores:

Sonia Martín Moreno

Francisco Javier Remesal Escalero

Laura Rivera Rodríguez