oFlute: blablablá título largo

Alumno: José Tomás Tocino García
Tutores: Manuel Palomo Duarte, Antonio García Domínguez

Agosto de 2010

Resumen

oFlute se modela como una herramienta lúdico-educativa para alumnos que comiencen a aprender a usar la flauta dulce, proporcionando un entorno atractivo y ameno para el estudiante. Éstos tendrán la posibilidad de comprobar sus conocimientos sobre el uso de la flauta de forma totalmente práctica, gracias a un motor de análisis del sonido capaz de detectar las notas que emite el jugador con la flauta, capturadas por un micrófono, mediante el que la aplicación valorará la pericia del estudiante con la flauta.

Además, los jugadores podrán recorrer una serie de pequeñas lecciones sobre música en general, y el uso de la flauta dulce en particular. Estas lecciones son totalmente ampliables, dando al usuario la posibilidad de crear las suyas propias.

Índice

1. Introducción **2**
1.1. Contexto y motivación . . . 2
1.2. Objetivos . . . 2

2. Planificación **3**
2.1. Primera iteración: conocimientos preliminares . . . 3
2.2. Segunda iteración: analizador básico . . . 3
2.3. Tercera iteración: interfaz gráfica de usuario . . . 3
2.4. Cuarta iteración: motor de lecciones . . . 3
2.5. Quinta iteración: motor de canciones . . . 3
2.6. Diagrama de Gantt . . . 3

3. Descripción general **5**
3.1. Secciones de la aplicación . . . 5
3.1.1. Análisis de notas . . . 5
3.1.2. Motor de canciones . . . 5
3.1.3. Motor de lecciones . . . 5
3.1.4. Calibración del micrófono . . . 5

4. Implementación **5**
4.1. Implementación del analizador básico . . . 6
4.2. Carga y uso de fuentes TrueType en Gosu . . . 6
4.3. Animaciones dinámicas . . . 6
4.4. Gestión de estados . . . 6
4.5. Internacionalización . . . 7

5. Conclusiones y difusión **7**
5.1. Objetivos 7
5.2. Posibles mejoras 7
5.3. Conclusiones personales 8
5.3.1. Conocimiento adquirido 8
5.4. Difusión 8
5.4.1. Conocimiento generado 8
5.4.2. Freegemas 9
5.4.3. IV Concurso Universitario de Software Libre 9

1. Introducción

1.1. Contexto y motivación

Las nuevas tecnologías van filtrándose gradualmente en los centros educativos, y las técnicas de enseñanza se están adaptando a las opciones que ofrecen. El reparto de ordenadores portátiles a los alumnos andaluces de 5º y 6º de primaria, dentro del marco de la Escuela TIC 2.0, es buena muestra de ello.

Por otro lado, las nuevas generaciones están en plena simbiosis con las tecnologías de la información, cada vez más acostumbradas al empleo de dispositivos electrónicos interactivos, y su uso ya les es prácticamente instintivo. Por tanto, es beneficioso buscar nuevos métodos educativos que hagan uso de las nuevas tecnologías.

En la búsqueda de materias educativas en las que aplicar el uso de las nuevas tecnologías, la música, parte fundamental del programa curricular en la educación primaria, ofrece una gran variedad de aspectos que podrían desarrollarse utilizando tecnologías de la información. Es ahí donde este proyecto hace su aportación, en la flauta dulce, un instrumento económico y fácil de aprender que se usa tradicionalmente en la educación musical obligatoria en España.

1.2. Objetivos

Los principales objetivos a alcanzar con **oFlute** son los siguientes:

- Crear un **módulo de análisis del sonido** en el dominio de la frecuencia para poder identificar las notas emitidas por una flauta dulce y capturadas mediante un micrófono en tiempo real.
- Crear una **aplicación** de usuario que identifique y muestre en pantalla las notas que toca el usuario con la flauta dulce en cada momento.
- Reutilizar el módulo de análisis en un juego en el que el usuario debe **interpretar una canción** tocando correctamente las notas que aparecen en pantalla sobre un pentagrama.
- Incluir un **sistema de lecciones** multimedia individuales que sirvan al alumno de referencia y fuente de aprendizaje.
- Potenciar el uso de **interfaces de usuario amigables**, con un sistema avanzado de animaciones que proporcione un aspecto fluido y evite saltos bruscos entre secciones.
- Obtener una **base teórica** sobre cómo se representa y caracteriza digitalmente el sonido.
- Conocer las **bases del DSP**, y su uso en aplicaciones de reconocimiento básico de sonidos, tales como sintonizadores y afinadores de instrumentos.

- Adquirir soltura en la **programación de audio** bajo sistemas GNU/Linux.
- Utilizar un enfoque de análisis, diseño y codificación **orientado a objetos**, de una forma lo más clara y modular posible, para permitir ampliaciones y modificaciones sobre la aplicación por terceras personas.
- Hacer uso de herramientas básicas en el desarrollo de software, como son los Sistemas de Control de Versiones para llevar un control realista del desarrollo del software, así como hacer de las veces de sistema de copias de seguridad.

2. Planificación

El proyecto se ha desarrollado siguiendo un calendario basado en fases, utilizando un modelo de desarrollo iterativo incremental.

2.1. Primera iteración: conocimientos preliminares

En esta etapa se adquirieron los fundamentos teóricos para poder afrontar el desarrollo con todas las garantías. Se llevaron a cabo labores de documentación y aprendizaje autodidacta con las que se asentaron los conocimientos necesarios.

2.2. Segunda iteración: analizador básico

La segunda iteración se basó en el diseño de un analizador de notas básico, que sería el corazón del programa. Del éxito del desarrollo temprano del módulo que se encargaría del análisis de sonidos dependería la viabilidad completa del proyecto.

2.3. Tercera iteración: interfaz gráfica de usuario

En esta tercera iteración se propusieron numerosos diseños para la interfaz gráfica de usuario y se comenzó el desarrollo de los elementos de la interfaz, haciendo énfasis en conseguir un aspecto dinámico y jovial.

2.4. Cuarta iteración: motor de lecciones

En esta iteración se llevó a cabo el motor de lecciones, que presenta una serie de unidades didácticas en formato multimedia, compuestas de imágenes y textos. El sistema resultante es muy sencillo de ampliar y utilizar.

2.5. Quinta iteración: motor de canciones

Durante la quinta iteración se elaboró el sistema de canciones, encargado de listar y cargar las diferentes canciones, y puntuar al usuario según cómo interprete, mediante la flauta, las canciones que aparece en pantalla.. Es la parte de la aplicación con mayor interactividad.

2.6. Diagrama de Gantt

Se ha diseñado un diagrama de Gantt para reflejar la distribución de las tareas a lo largo del tiempo (figura 1 en la página 4).

Nombre	Trab...	2010				2011		
		Ctr 1	Ctr 2	Ctr 3	Ctr 4	Ctr 1	Ctr 2	Ctr 3
Planificación	5d							
Investigación preeliminar	**83d**							
Análisis estado del arte	13d							
Bases teóricas	45d							
Blibiotecas desarrollo	25d							
2a iter: analizador básico	**35d**							
Análisis y diseño	7d							
Implementación	25d							
Pruebas	3d							
3a iter: interfaz gráfica	**46d**							
Diseño logotipo	7d							
Diseño menú principal	5d							
Diseño genérico secciones	9d							
Impl. animaciones din.	20d							
Pruebas	5d							
4a iter: motor lecciones	**110d**							
Análisis	25d							
Diseño	22d							
Implementación	53d							
Pruebas	10d							
5a iter: motor canciones	**112d**							
Análisis	25d							
Diseño	18d							
Implementación	60d							
Pruebas	9d							
Memoria	105d ...							

Figura 1: Diagrama Gantt de iteraciones

3. Descripción general

oFlute se modela como una herramienta lúdico-educativa para alumnos que comiencen a aprender a usar la flauta dulce, proporcionando un entorno atractivo y ameno para el estudiante. Éstos tendrán la posibilidad de comprobar sus conocimientos sobre el uso de la flauta de forma totalmente práctica, gracias a un motor de análisis del sonido capaz de detectar las notas que emite el jugador con la flauta, capturadas por un micrófono, mediante el que la aplicación valorará la pericia del estudiante con la flauta.

Además, los jugadores podrán recorrer una serie de pequeñas lecciones sobre música en general, y el uso de la flauta dulce en particular. Estas lecciones son totalmente ampliables, dando al usuario la posibilidad de crear las suyas propias.

3.1. Secciones de la aplicación

3.1.1. Análisis de notas

Permite a los usuarios comprobar, de manera individual y pausada, que la interpretación de cada una de las notas en la flauta es correcta. Para ello, se presenta un pequeño analizador que responderá al sonido emitido por el usuario con la flauta mostrando la nota tocada en pantalla.

3.1.2. Motor de canciones

Mediante este sistema, el usuario tendrá la oportunidad de interpretar canciones completas a la vez que el computador analiza la eficacia del jugador, otorgándole una puntuación en tiempo real. Además, el motor de canciones es fácilmente expansible mediante ficheros de definición de canciones.

3.1.3. Motor de lecciones

Este sistema ofrece al usuario una serie de lecciones multimedia, con las que podrá aprender sobre diferentes aspectos de la música en general y la flauta en particular. Las lecciones cuentan con imágenes, texto y animaciones, que harán que el aprendizaje sea entretenido y ameno. También es posible añadir nuevas lecciones al sistema de forma sencilla.

3.1.4. Calibración del micrófono

oFlute ofrece la posibilidad de calibrar el micrófono, de forma que el sistema se adapte al ruido ambiental y el análisis del sonido capturado sea lo más exacto posible

4. Implementación

Durante el desarrollo del proyecto surgieron numerosas cuestiones y decisiones de implementación. A continuación se presenta una selección de las que más interés han generado. En la memoria del proyecto se desarrollan en mayor extensión.

4.1. Implementación del analizador básico

La creación del analizador básico de notas era crucial para el buen transcurso del resto del proyecto, por lo que fue uno de los objetivos que antes se abordaron. El desarrollo se dividió en dos partes. Por un lado, había que iniciar la captura de audio, lanzando el subsistema de sonido y empezando a capturar datos. Por otro lado, se tenía que hacer el análisis de los datos leídos para determinar qué nota se estaba tocando.

La gestión del subsistema de audio se hizo mediante la *Simple API* de PulseAudio, la biblioteca de audio que se empleó en oFlute. Se utilizó para la configuración y creación de un flujo de audio de entrada que permitiera captar, muestrear y samplear el sonido del micrófono, ofreciéndolo en forma de datos en punto flotante en un búffer.

El siguiente paso fue el análisis del audio capturado. Mediante el uso de la Transformada Rápida de Fourier y la biblioteca KissFFT, de forma iterativa se iba analizando el búffer de sonido cada vez que éste se llenaba, dando lugar a una aproximación de la frecuencia fundamental capturada y, por ende, la nota que se estaba interpretando con la flauta.

Una vez implementado y probado el sistema, se dió por bueno el analizador y se siguió construyendo el resto de la aplicación

4.2. Carga y uso de fuentes TrueType en Gosu

La biblioteca principal utilizada en la implementación del proyecto es Gosu, una librería de desarrollo de videojuegos 2D, multi-plataforma y con numerosas características. Desafortunadamente, no era posible utilizar fuentes TrueType cuando se utilizaba la biblioteca en GNU/Linux.

Utilizando los conocimientos de otra popular biblioteca, SDL, se consiguió implementar un sistema para cargar y usar fuentes TrueType en Gosu de forma eficiente. Esta implementación finalmente acabó formando parte oficial de la biblioteca Gosu.

4.3. Animaciones dinámicas

Una de las decisiones iniciales de diseño fue la de hacer la interfaz gráfica de usuario lo más atractiva posible, intentando utilizar gráficos amigables y, en la medida de lo posible, animaciones y efectos dinámicos.

Con esto, se tornaba necesario crear un sistema de animaciones lo más versátil posible, de forma que dotar a los elementos de la interfaz de movimiento fuera un proceso sencillo.

Para satisfacer este objetivo se ideó una serie de clases de animación que permiten animar las propiedades de los objetos de manera muy sencilla. Mediante el uso de las ecuaciones de animación de Robert Penner, el sistema creado cuenta con un gran número de opciones y diferentes formas de movimiento, según el tipo de función utilizada para calcular los valores: cuadrático, cúbico, etcétera.

Gracias a este sistema, la interfaz de usuario de oFlute cuenta con un gran dinamismo y vistosidad.

4.4. Gestión de estados

La gestión de estados es uno de los retos principales a la hora de desarrollar un videojuego. Es necesario contar un sistema de gestión de estados robusto, que permita

pasar de un estado a otro de forma sencilla, sin errores, y manteniendo información entre transiciones si fuera necesario.

En el caso de oFlute, se creó un gestor de estados que permitió modularizar fácilmente el programa, haciendo una clara división entre secciones y facilitando la transición entre ellas. Además, mediante pruebas exhaustivas y el uso de herramientas de depuración, el gestor está implementado de forma que la carga y descarga de estados sea limpia y sin errores, evitando fugas de memoria.

4.5. Internacionalización

Durante el desarrolllo del proyecto surgió la necesidad de preparar el proyecto para su traducción. Inicialmente se optó por utilizar un sistema propio de traducción, basado en un script en Python y en una pequeña clase que generaba un diccionario según el idioma elegido. Sin embargo, esta opción resultó ineficiente y, sobre todo, alejada del resto de soluciones estándar a la hora de traducción de proyectos.

Por ello, se decidió investigar sobre las tecnologías de traducción más utilizadas en el panorama del software libre, y finalmente se optó por **GNU Gettext** [4]. Para afianzar los conocimientos y facilitar el aprendizaje a otros desarrolladores, se editó un conciso manual sobre internacionalización de proyectos, presente como apéndice en la memoria del proyecto fin de

5. Conclusiones y difusión

Durante el transcurso del desarrollo de oFlute, y sobre todo al término del mismo, se han obtenido unas conclusiones y unos resultados, tanto de forma personal como para con la comunidad, que se resumen en esta sección y se tratan más en profundidad en el capítulo dedicado a tal efecto en la memoria principal.

5.1. Objetivos

Al término del desarrollo del proyecto, el proyecto ha completado todos los objetivos a cumplir, detallados en el planteamiento inicial. En particular:

- Se llevó a cabo la creación del módulo de análisis de sonido, consiguiendo una efectividad en la detección de las notas muy alta.
- Se hizo uso del módulo en una sección de análisis de notas y en el propio juego de interpretación de canciones, con resultados muy satisfactorios y funcionamiento correcto.
- Además, el sistema de lecciones planteado se llevó a cabo correctamente, consiguiendo un motor dinámico bastante potente y fácilmente ampliable, con opción a añadir nuevas lecciones al sistema.
- Finalmente, todos los objetivos se llevaron a cabo respetando la premisa de mantener una interfaz de usuario amigable y fluida, agradable a la vista y sencilla de usar.

5.2. Posibles mejoras

Hay un gran número de posibles mejoras y extensiones de la funcionalidad de oFlute, en unos casos descubiertas de forma personal y en otros casos aportadas por terceras personas. Algunas de las más interesantes son las siguientes:

- Extensión del sistema de lecciones para permitir lecciones de varios pasos, así como integrar las lecciones con el analizador de audio para comprobar los conocimientos in-situ.
- Mejora de la jugabilidad del sistema de canciones, añadiendo bonus y puntos extra.
- Intentar extender el sistema a otros instrumentos o a la voz humana.
- Portar el videojuego a la plataforma Windows.

5.3. Conclusiones personales

oFlute ha sido el proyecto más longevo al que me he enfrentado hasta ahora. A pesar de tener conciencia de la envergadura del mismo desde el principio, el tiempo para completarlo ha superado todas mis expectativas, sobre todo en lo que a documentación se refiere.

5.3.1. Conocimiento adquirido

Gracias a oFlute he aprendido las técnicas básicas de la programación de audio, sobre todo en la parte técnica más que teórica. Es esta parte teórica, sobre todo la de análisis, la que me gustaría reforzar en un futuro, ahora que cuento con las bases para conseguirlo. Los videojuegos relacionados con el audio son un nicho aún poco explorado y que puede dar muchas satisfacciones, sobre todo cuando el producto se orienta a un público joven.

Por otro lado, oFlute también me ha ayudado a aprender a usar bastantes tecnologías auxiliares, en algunos casos meras herramientas que han facilitado el trabajo y, en otros casos, elementos que resultaron ser de inestimable ayuda e imprescindible uso al término del proyecto.

Una de estas tecnologías es **Boost** [2], un conjunto de bibliotecas para C++ que amplían en gran medida la biblioteca estándar del lenguaje. Un amplio número de componentes de Boost formarán parte del nuevo estándar C++0x, por lo que me ha servido para ponerme al día en las novedades que están por llegar.

Al tratarse de la principal biblioteca utilizada durante el desarrollo, oFlute me ha provisto de un profundo conocimiento de **Gosu** [17], permitiéndome implementar videojuegos con mucha más fluidez y labrándome un pequeño *framework* personal que utilizar de base en próximos proyectos.

Otra de las tecnologías que he aprendido ha sido **GNU Gettext**, que ha servido para internacionalizar el proyecto. Aunque inicialmente se pensó en utilizar una solución propia para la internacionalización el proyecto, posteriormente se decidió usar esta tecnología, ampliamente conocida y utilizada.

5.4. Difusión

5.4.1. Conocimiento generado

Además de servir como recurso para el correcto transcurso del proyecto, los conocimientos adquiridos me permitieron, en numerosos casos, generar documentación adicional e impartir talleres relacionados las tecnologías previamente mencionadas.

En cuanto a Boost, se llevó a cabo un taller durante los *Cursos de Verano de la OSLUCA* [3], en el que se explicaron las partes más importantes de esta colección de bibliotecas, con numerosos ejemplos de muchas de ellas. Toda la documentación es libre [14].

Por otro lado, impartí un taller [16] sobre la biblioteca Gosu en colaboración con la ADVUCA [1], cuya afluencia superó las 50 personas. Los materiales pueden descargarse libremente [15].

Por último, tras la aplicación de GNU Gettext en oFlute, escribí una guía concisa sobre traducción de proyectos con esta herramienta, que se puede encontrar en uno de los apéndices de la memoria, así como de forma online en la url http://hdl.handle.net/10498/10772.

5.4.2. Freegemas

Uno de los proyectos *"hijos"* de oFlute ha sido **Freegemas** [13], un clon libre del popular juego tipo puzzle *Bejeweled*. Freegemas es multiplataforma, funciona tanto en GNU/Linux como en Windows, y además forma parte oficial de Guadalinex [5], por lo que es posible encontrarlo en los repositorios oficiales.

Además, Freegemas sirvió como base para una serie de tres artículos que publiqué, junto al director del presente PFC, en la revista Linux Magazine [7] sobre desarrollo de videojuegos en C++. Es posible encontrar estos artículos en el archivo de la revista [10] [11] [12] bajo una licencia Creative Commons.

5.4.3. IV Concurso Universitario de Software Libre

Por otro lado, gracias a oFlute tuve la oportunidad de participar en el IV Concurso Universitario de Software Libre [6]. En el transcurso del concurso formé parte de una comunidad muy unida, en la que reinó el apoyo y la ayuda entre los concursantes. La final del concurso se celebró en la Escuela Superior de Ingeniería de Cádiz, en la que oFlute obtuvo una mención especial [9].

Previa a la final nacional tuvo lugar la fase local del concurso, en el que el proyecto también recibió un accésit al mejor proyecto de innovación [8].

Referencias

[1] *Asociación de Diseño de Videojuegos de la Universidad de Cádiz*. http://www.advuca.com.

La Asociación de Diseño de Videojuegos de la UCA promueve el uso y desarrollo de videojuegos dentro de la Universidad, organizando talleres y conferencias.

[2] *Boost C++ Libraries*. http://www.boost.org.

Boost es un conjunto de bibliotecas para C++ que ofrecen herramientas para una gran diversidad de situaciones. Entre sus desarrolladores se encuentran muchos de los mejores programadores de C++. Dada la calidad de Boost, una gran número de sus componentes parte del nuevo estándar C++0x.

[3] *Cursos de Verano de la OSLUCA*. http://osl.uca.es/node/1132.

Del 28 de junio al 2 de julio de 2010 se celebraron, dentro del marco de la final del IV CUSL, unis Cursos de Verano organizados por la Oficina de Software Libre y Conocimiento Abierto de la UCA.

[4] *GNU Gettext*. http://www.gnu.org/s/gettext/.

GNU Gettext es un conjunto de herramientas libres de internacionalización de proyectos.

[5] *Guadalinex*. http://www.guadalinex.org.

Guadalinex es una distribución Linux promovida por la Junta de Andalucía para fomentar el uso del software libre en su comunidad autónoma.

[6] *IV Concurso Universitario de Software Libre*. http://concursosoftwarelibre.org/0910/.

Cuarta edición del Concurso Universitario de Software Libre

[7] *Linux Magazine, edición en español*. http://linux-magazine.es.

Edición en español de la popular revista Linux Magazine.

[8] *Premios de la fase local del IV CUSL*. http://softwarelibre.uca.es/node/1120.

Noticia de la fase local del IV CUSL en el que se detallan los premios otorgados, entre los que oFlute recibió un accésit al mejor proyecto de innovación.

[9] *Premios del IV Concurso Universitario de Software Libre*. http://concursosoftwarelibre.org/0910/finalistas-iv-cusl, abril 2010.

Noticia del IV CUSL en que se da un listado de los premios finales del CUSL, en los que oFlute tiene el honor de aparecer como Mención Especial.

[10] Manuel Palomo Duarte y José Tomás Tocino García. *Gosu I - Creando un videojuego en C++*. *Linux Magazine, edición en Español*, (66), Diciembre 2010.

[11] Manuel Palomo Duarte y José Tomás Tocino García. *Gosu II - Creando un videojuego en C++*. *Linux Magazine, edición en Español*, (67), Enero 2011.

[12] Manuel Palomo Duarte y José Tomás Tocino García. *Gosu III - Creando un videojuego en C++*. *Linux Magazine, edición en Español*, (68), Febrero 2011.

[13] José Tomás Tocino García. *Freegemas*. http://freegemas.googlecode.com.

Videojuego libre, un clon multiplataforma del clásico juego tipo puzzle *Bejeweled*. Está disponible para sistemas GNU/Linux y Windows, y también se encuentra disponible en los repositorios de Guadalinex.

[14] José Tomás Tocino García. *Materiales del curso de Boost*. http://josetomastocino.com/varios/taller_boost.tar.gz.

Materiales libres del curso sobre Boost que impartí durante los Cursos de Verano de la OSLUCA en junio de 2010.

[15] José Tomás Tocino García. *Materiales Taller Gosu, marzo 2011*. http://advuca.com/blog/actividades.

Materiales del taller sobre desarrollo de videojuegos con Gosu que impartí en marzo de 2011.

[16] José Tomás Tocino García. *Taller: aprende a programar videojuegos en C++ con Gosu*. http://advuca.com/blog/talleres/talleres-blender-y-gosu-en-marzo/.

Anuncio del taller sobre desarrollo de videojuegos con Gosu que impartí en marzo de 2011.

[17] Julian Raschke y otros. *Gosu*. http://libgosu.org.

Gosu es una biblioteca de desarrollo de videojuegos 2D para Ruby y C++, con aceleración gráfica por OpenGL y orientación a objetos.