UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Extendiendo OpenAL con SDL. Caso de estudio MP3

Apellidos, nombre	**Agustí i Melchor, Manuel** (magusti@disca.upv.es)
Departamento	**Departamento de Informática de Sistemas y Computadores** (DISCA)
Centro	Escola Tècnica Superior d'Enginyeria Informàtica Universitat Politècnica de València

1 Resumen de las ideas clave

OpenAL es un motor de audio capaz de *renderizar* el sonido que escucha un oyente inmerso en una escena sonora tridimensional, de modo que el usuario se vea envuelto entre la fuentes de sonido dispuestas a su alrededor. Para ampliar el conjunto de ficheros de los que OpenAL puede **importar el audio** se implementó la *Audio Library Utility Toolkit* (ALUT) y se propuso un mecanismo de ampliación (las "extensiones") para que otros pudieran aportar nuevas funcionalidades al motor básico.

Aún así, la aparición de nuevos formatos de ficheros y la creciente necesidad de gestionar esquemas de compresión en el audio contenido en los ficheros, ha hecho que la inclusión de nuevos formatos de audio venga de la mano de librerías específicas. Una posible solución para este problema es utilizar la biblioteca de funciones *Simple DirectMedia Layer* (SDL), tanto por sus capacidades multimedia, como por su carácter multiplaforma, su naturaleza de código abierto y por su asentado desarrollo. En ese trabajo, veremos cómo incluir las funciones del *Application Programming Interface* (API) de SDL para ampliar el conjunto de formatos soportados en aplicaciones propias que utilicen OpenAL para implementar audio posicional.

2 Objetivos

La intención de este documento es proporcionar una solución al problema de la limitación de formatos que puede importar OpenAL (véase [1] y [2]). Abordaré el uso de SDL [3] como solución al problema de acceso al mayor número de formatos de ficheros de audio posible. Se quiere así conseguir la mayor flexibilidad y disponibilidad posible, en el acceso a ficheros de audio, para una aplicación que tiene este media como principal elemento.

Los ejemplos de uso e implementación están hechos sobre plataforma GNU/Linux pero, dada la capacidad de portabilidad de las bibliotecas utilizadas, el desarrollo puede ser llevado con facilidad a otros sistemas operativos. También cabe destacar que se utilizará el formato MP3·como ejemplo de uso por su popularidad, pero se podrá aplicar a otros formatos que se detallarán.

Una vez que el lector haya explorado el documento y los ejemplos que en él se explican, será capaz de:

- Ampliar el abanico de formatos de fichero de audio que soporten sus propias aplicaciones sobre OpenAL, mediante SDL.
- Reutilizar, de los ejemplos proporcionados, la secuencia de instrucciones necesaria de SDL para incorporar en sus desarrollos estas capacidades.
- Instalar las dependencias y compilar una aplicación que pueda hacer uso del API de SDL.

No es objetivo de este documento dar una exhaustiva relación de operaciones disponibles en la solución propuesta, para lo que se aconseja la revisión de la documentación de SDL.

Nota: En el texto, cuando me refiera a una de las figuras que se incluyen en él, utilizaré la abreviatura "fig." listada en la RAE[1] para este menester.

Para empezar voy a revisar el planteamiento de diseño de OpenAL. Hablaré de las posibles variantes de soluciones al problema de acceso a los ficheros y cómo

1 Lista de las abreviaturas convencionales más usuales en español. Disponible en <http://www.rae.es/diccionario-panhispanico-de-dudas/apendices/abreviaturas>.

SDL puede servir para homogeneizar y simplificar estas tareas relativas al audio. Y, finalmente, la propuesta de pequeños ejemplos que puedan ser reutilizados como plantillas en otros desarrollos con SDL y OpenAL, permitirá experimentar de forma práctica con estas librerías.

3 Introducción

OpenAL [1] **no se encargará de cargar audio desde un fichero** en disco, sino de enlazarlo a una fuente de sonido. cuando ya está cargado en memoria y sin compresión. Esta ha sido una decisión de diseño [4] y [2] y, aunque se han creado extensiones, el resultado no es completamente portable a todas las plataformas en las que OpenAL está disponible. Por tanto, no es una solución sólida. Por ejemplo, cabe citar [1] dos casos:

- La extensión AL_EXT_MP3, que pertenece al grupo de ampliaciones multiplataforma, no está soportada en Microsoft/Windows[2] y, tampoco, por las nuevas versiones de Linux.
- La extensión EXT_vorbis, que permite el uso del formato Ogg Vorbis, solo está disponible en plataformas basadas en Linux.

El poco éxito de estas ampliaciones y la creciente necesidad de descomprimir el audio contenido en los ficheros, ha hecho que la inclusión de nuevos formatos de audio venga de la mano de librerías específicas. Esto trae como consecuencia que se puede ampliar la operativa de una aplicación, al tiempo que se limita el peso de este componente en una aplicación, a solo los formatos que se sepa se van a utilizar. Por contrapartida, el desarrollador se ve inmerso en la necesidad de entender y adaptar más de una de estas librerías, con sus propias estructuras de datos y algoritmos de funcionamiento.

Otra solución es que el conjunto de ficheros a utilizar sea estático y los puedas tener todos en un formato de tu elección y que habrá de soportar la aplicación que desarrollas[3]. La descarto por buscar una solución mas flexible: un sistema que pudiera lidiar con cualquier formato o, al menos, tener una cierta holgura para ello.

La librería SDL puede servir de solución casi global, por su gran número de formatos aceptados y porque[4] la carga del soporte a los formatos FLAC, MOD, Ogg Vorbis y MP3 se hace en tiempo de ejecución. Así que, si no se utiliza alguno de ellos, no se desaprovechará memoria con código que no es utilizado.

3.1 Un breve comentario sobre MP3

Quizás te preguntes por qué se ha utilizado MP3 como “caso de estudio”. Quizás te has quedado un poco distraído mirando que he mencionado Vorbis y hayas recalado en que hay otros formatos (AAC, H.264, Opus, ...) y que incluso, dicen[5], son más eficientes y tienen características más avanzadas que MP3.

---

2 “Openal Driver Issue on windows xp”. Disponible en <http://openal.996291.n3.nabble.com/Openal-Driver-Issue-on-windows-xp-td5065.html>.

3 Te ayudará en este aspecto el trabajo de Lippens, S. (2009). “Audio format conversion cheat sheet (aka how to)”. Disponible en <http://stefaanlippens.net/audio_conversion_cheat_sheet/>.

4 Como dice en su sito web: <https://www.libsdl.org/projects/SDL_mixer/ >.

5 Y así lo dicen los propios creadores de MP3 <https://www.iis.fraunhofer.de/en/ff/amm/prod/audiocodec/audiocodecs/mp3.html>.

Lo que pasa es que las patentes han expirado y ni el Instituto *Fraunhofer*, ni *Technicolor* (las dos empresas que las poseían), van a seguir vendiendo productos con este formato, dando soporte a los usuarios de los anteriores desarrollos o desarrollando nuevas aplicaciones sobre él.

MP3 revolucionó la industria musical[6], tanto en su intercambio entre particulares como en la venta en Internet. Ahora que es libre, no tiene porque desaparecer, ya que es un formato tremendamente extendido y que cuenta en muchas plataformas con la aceleración del hardware que se encarga del audio. Así que veamos cómo incorporarlo a OpenAL.

3.2 Introducción a SDL

SDL y OpenAL son viejos conocidos. Ambos proyectos nacen dentro de *Loki Software [4]*, que fue una empresa especializada en portar juegos a plataforma Linux. Se debe a Sam Lantinga[7] la creación de SDL. Esta biblioteca de funciones constituye, a día de hoy, una referencia importante en el campo de desarrollo multimedia por: su carácter abierto, su capacidad de ser portada a un gran número de plataformas (oficialmente, Mac OS X, Linux, iOS, Android y Windows) y su amplio y asentado conjunto de operaciones. SDL está escrito en lenguaje C (se puede utilizar también con C++, C# y Python) y proporciona acceso a bajo nivel para audio, periféricos y gráficos en pantalla.

El conjunto de funciones del API de SDL de "Audio Device Management, Playing and Recording" da soporte al acceso al hardware y a ficheros en formato WAVE. Para las plataformas de escritorio, existe un módulo para SDL, llamado ***SDL_sound***[8] que es un complemento especializado en la lectura y decodificación de formatos de audio usuales como *raw* (formato descomprimido que no especifica sus propiedades), *WAVE, MP3, MIDI, Ogg Vorbis, Speex, FLAC* y otros treinta más, según su sitio web. Con *SDL_sound* será posible determinar las propiedades del audio contenido en un fichero, descomprimir los datos que este contiene y llevar a memoria el audio en un formato que sea compatible con las estructuras de OpenAL

También relacionada con el audio, existe una extensión a SDL que es ***SDL_mixer***, cuyo objetivo la reproducción, en un numero (sin limite) de canales separados para efectos (audio de corta duración, de 16 bits y estéreo) y uno dedicado a la reproducción de la música (una pista de larga duración). No lo abordaremos en este trabajo por quedar fuera de los objetivos fijados.

Cuidado, porque ahora puedes encontrar ejemplos e instalaciones de SDL en su versión 1.2 y en la nueva 2.0. En mi máquina coexisten ambas versiones de SDL, actualmente. Puedes ejecutar las siguientes órdenes para comprobar la situación en tu computador:

```
$ sdl-config --version
1.2.15
$ sdl2-config --version
2.0.2
```

Para instalar los paquetes que necesitamos lo puedes hacer desde un terminal con la orden:

```
$ sudo apt install libsdl2-dev libsdl-sound1.2-dev
```

---

6 Stephen Witt en "How Music Got Free" dice que lo fue porque ¡lo eligieron como formato para el pirateo de música!

7 Véase <https://en.wikipedia.org/wiki/Sam_Lantinga>.

8 Lo encontrarás en <http://www.icculus.org/SDL_sound/>.

3.2.1 Formatos soportados por SDL_sound

¿Estás intrigado por cómo se puede utilizar SDL/SDL_sound en tus propios desarrollos? Te voy a contar algunas cosas, ten a mano la documentación, porque ahí hay más. A partir de la documentación del API de SDL_sound[9], se puede comprobar, en tiempo de ejecución, los formatos que se soportan en una máquina en concreto. Ya que esto depende de si existen las librerías externas correspondientes con las que SDL puede trabajar:

```
#include <SDL2/SDL.h>
#include <SDL/SDL_sound.h>
int main(int argc, char* argv[]){
  if (SDL_Init(SDL_INIT_AUDIO) < 0)   // Initialize SDL.
    return 1;
  int i, count = SDL_GetNumAudioDevices(0);
  printf("Detectats %d AudioDevice/s\n", count);
  for (i = 0; i < count; ++i) {
    SDL_Log("Audio device %d: %s",
                i, SDL_GetAudioDeviceName(i, 0));
  }
  if(Sound_Init() == 0)     {     // Inicializace SDL_sound
      fprintf(stderr, "Unable to initialize SDL_sound: %s\n",
         Sound_GetError());      return 0;
    }
 // Obtener la lista de formatos soportados por SDL_sound
  const Sound_DecoderInfo **dec_info = Sound_AvailableDecoders();
  printf("Supported sound formats:\n");
  for(int i = 0; dec_info[i] != NULL; i++)     {
      printf("%s\t- %s\n",
        *dec_info[i]->extensions, dec_info[i]->description);
    }
  printf("\n");
  SDL_Quit();   // End SDL
}
```

Listado 1: Obtención de los formatos soportados por SDL_sound.

Lo vemos con un ejemplo, el que se muestra en el listado 1, donde he resaltado las funciones de SDL y SDL_sound. He llamado al fichero "sdl_formatsEntrada.c", que se compila y ejecuta, desde un terminal, con:

```
$ gcc sdl_formatsEntrada.c -o sdl_formatsEntrada `sdl2-config --cflags --libs`
-lSDL_sound

$ sdl_formatsEntrada
Detectats 1 AudioDevice/s
INFO: Audio device 0: Audio Interno Estéreo analógico
Supported sound formats:
```

9 Y de los comentarios en <http://i.iinfo.cz/urs-att/sdl_18_b-111951903649740.html>.

```
MP3     - MP3 decoding via internal mpglib
669     - Play modules through MikMod
WAV     - Microsoft WAVE audio format
AIFF    - Audio Interchange File Format
AU      - Sun/NeXT audio file format
OGG     - Ogg Vorbis audio through VorbisFile
VOC     - Creative Labs Voice format
RAW     - Raw audio
SHN     - Shorten-compressed audio data
FLAC    - Free Lossless Audio Codec
spx     - SPEEX speech compression format
```

Vale y, suponiendo que el fichero de audio que quieres utilizar está en ese grupo, ¿qué se necesita incluir de **SDL_sound** para nuestra propia aplicación en OpenAL? La fig. 1 muestra, de forma gráfica, la relación y la secuencia de pasos para llevar la información de audio descomprimida (en PCM) a OpenAL. Para ello, a través de la extensión **SDL_sound**, SDL proporciona las funciones:

- *Sound_NewSampleFromFile* nos permite abrir el fichero y obtener de su cabecera las propiedades del fichero. Con ello será posible inicializar y configurar las estructuras de OpenAL.
  ¿A qué me refiero? A parámetros como la frecuencia de muestreo, el número de canales o el tamaño de muestra. También será posible pedir recursos al sistema para albergar la información descomprimida. Este dato es conocido puesto que al generar el fichero se ha guardado esa información.
- *Sound_DecodeAll se encargará de la lectura del fichero y su descompresión. Con ella obtendremos el audio en PCM.*
- En ese punto ya es posible trabajar con las funciones de OpenAL. Así que es posible liberar los recursos asociados al tratamiento del fichero que ha facilitado SDL. Como *los* datos de audio ya están guardados en un *buffer* (con *alBufferData*), ya los podemos asociar a una fuente. La fuente es el objeto de OpenAL que tiene la capacidad de reproducir el audio (*alSourcePlay*), así que ya se podrá escuchar.

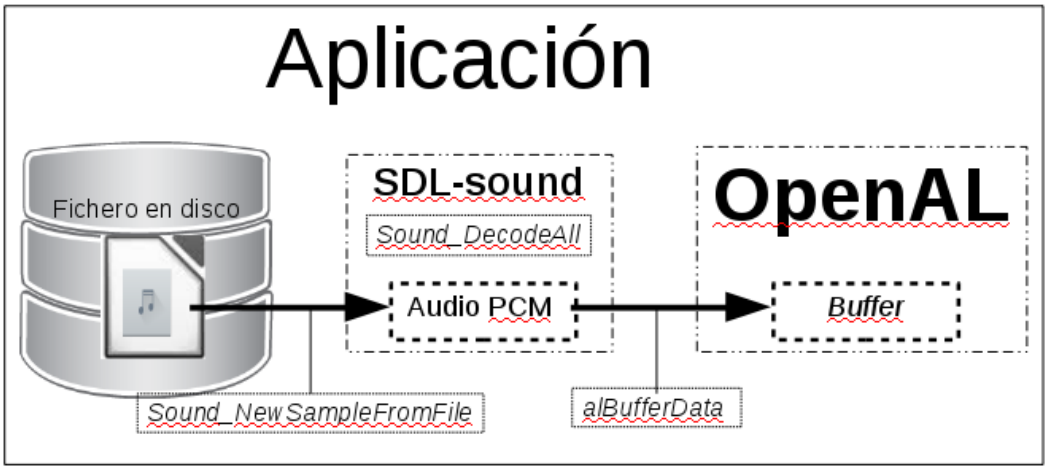

Figura 1: Relación entre el SDK de SDL (SDL_sound) y OpenAL, para la carga de ficheros de audio.

Dicho todo esto, en el siguiente apartado hablaré de cómo encajar todas las piezas en un ejemplo combinado de SDL y OpenAL.

4 Ejemplo conjunto de SDL y OpenAL

Voy a comentar ahora el ejemplo “alplay.c “ [5] que permite cargar el contenido de un fichero de audio, de los formatos soportados por *SDL_sound*, en un *buffer* de OpenAL y hacer que este lo haga sonar.

```
#include <stdio.h>
#include <assert.h>
#include <SDL_sound.h>
#include "AL/al.h"
#include "AL/alc.h"
#include "common/alhelpers.h" // InitAl, CllooseAL
```

Listado 2: Cabeceras del ejemplo "alplay.c" [5].

El listado 2, muestra las dependencias de este proyecto. De entre las que cabe destacar **SDL_sound**, como venimos avanzando para la gestión de los formatos de ficheros de audio. El fichero *alhelpers* contiene las funciones *InitAl* y *CllooseAL* que solo tienen como misión facilitar la inicialización y el cierre ordenado de OpenAL, mediante las primitivas que este ofrece, agrupadas de la forma que el autor del ejemplo suele utilizar.

El resto del ejemplo lo he divido en dos partes y, en ellas, voy resaltando el uso de las funciones de **SDL_sound**:

- Por un lado, el listado 3, contiene la función que hace uso de **SDL_sound** para abrir el fichero de audio, cargar su contenido en memoria (lo que puede implicar la descompresión del mismo) y asignarlo a un *buffer* de OpenAL. Estas tareas están repartidas entre varias líneas porque hay que comprobar algunas situaciones que pueden dar errores y que he eliminado por brevedad en la exposición. ¡Pero, tranquilo, en la referencia original están todas!
  Esta función implementa el esquema esbozado en la fig. 1, así que no voy a insistir más en los detalles. He remarcado en negrita las funciones propias de **SDL_sound** (empiezan con el prefijo **Sound_**) y las de OpenAL (que comienza con **al**) para ayudarte en la lectura del código. Compáralo con el esquema de la fig. 1 y, siguiendo los comentarios originales del ejemplo que he dejado en el mismo, “verás” como el sonido, en el formato que esté guardado en el fichero, es cargado en una estructura de datos en memoria (*sample*). Y, ahora que ya está el sonido descomprimido en PCM, de ahí es asignado a una estructura de OpenAL (buffer).
- Por otro lado, el listado 4, contiene la función *main*. Esta es la encargada de inicializar OpenAL, **SDL_sound** y reproducir el sonido. He remarcado las funciones de cada uno y he dejado los comentarios originales para facilitar la lectura del código .
  OpenAL se inicializa y se cierra con las funciones **InitAL** y **CloseAL**. Estas son funciones que recogen la secuencia de instrucciones de OpenAL que hace posible estas dos etapas. Dada su repetición en otros ejemplos, el autor de los mismos las ha recogido en estas funciones. Como no hay salida gráfica, ni otros componentes de SDL que necesiten almacenar su estado, SDL no se inicializa como tal; pero **SDL_sound** si necesita su propia configuración y liberación de recursos (que se realizan con *Sound_Init* y *Sound_Quit,* respectivamente).

```
// Loads the file into an OpenAL buffer & returns the new buffer ID.
static ALuint LoadSound(const char *filename) {
    Sound_Sample *sample;
    ALenum err, format;
    ALuint buffer;
    Uint32 slen;

    sample = Sound_NewSampleFromFile(filename, NULL, 65536);    // Open
    if(!sample)    {                                            // the audio file
        fprintf(stderr, "Could not open audio in %s\n", filename); return 0;
    }
    if(sample->actual.channels == 1) {          // Get the sound format, and figure
        if(sample->actual.format == AUDIO_U8)             // out the OpenAL format
            format = AL_FORMAT_MONO8;
     // ... [Aquí va el resto de instrucciones para identificar los  posibles formatos ]
    else    {
    fprintf(stderr, "Unsupported channel count: %d\n", sample->actual.channels);
        Sound_FreeSample(sample); return 0;
    }
    slen = Sound_DecodeAll(sample);          // Decode the whole audio stream
    if(!sample->buffer || slen == 0)    {
        fprintf(stderr, "Failed to read audio from %s\n", filename);
        Sound_FreeSample(sample); return 0;
    }
    buffer = 0;                    // Buffer the audio data into a new buffer object,
    alGenBuffers(1, &buffer);
    alBufferData(buffer, format, sample->buffer, slen, sample->actual.rate);
    Sound_FreeSample(sample);                  // free the data and close the file
    err = alGetError();               // Check if an error occured, and clean up if so.
    if(err != AL_NO_ERROR)    {
        fprintf(stderr, "OpenAL Error: %s\n", alGetString(err));
        if(buffer && alIsBuffer(buffer))
          alDeleteBuffers(1, &buffer);
        return 0;
    }
    return buffer;
}
```

Listado 3: Función "LoadSound" para la precarga de archivos de audio con SDL_sound.

La reproducción del sonido, una *vez ya* está en memoria, es la parte más visible de esta función *main* y se realiza con instrucciones de OpenAL. Para ello, se crea una fuente (*alGenSources*) se asigna el sonido en PCM (*alSourcei*) y se hace sonar en un bucle (con *alSourcePlay* y *alGetSourcei*). Al terminar, se liberan los recursos (con *alDeleteSources y alDeleteBuffers*).

```
int main(int argc, char **argv) {
   ALuint source, buffer;
   ALfloat offset;
   ALenum state;
   if(argc < 2)    {              // Print out usage if no arguments were specified
      fprintf(stderr, "Usage: %s [-device <name>] <filename>\n", argv[0]);
      return 1;
   }
   argv++; argc--;                                          // Initialize OpenAL.
   if(InitAL(&argv, &argc) != 0)       return 1;
   Sound_Init();                                          // Initialize SDL_sound
   buffer = LoadSound(argv[0]);                  // Load the sound into a buffer.
   if(!buffer)    {
      Sound_Quit();       CloseAL();       return 1;
   }
   source = 0;                          // Create the source to play the sound with.
   alGenSources(1, &source);
   alSourcei(source, AL_BUFFER, buffer);
   assert(alGetError()==AL_NO_ERROR && "Failed to setup sound source");
   alSourcePlay(source);                          // Play the sound until it finishes.
   do {
      al_nssleep(10000000);
      alGetSourcei(source, AL_SOURCE_STATE, &state);
      alGetSourcef(source, AL_SEC_OFFSET, &offset);    // Get the source offset.
      printf("\rOffset: %f  ", offset);       fflush(stdout);
   } while(alGetError() == AL_NO_ERROR && state == AL_PLAYING);
   printf("\n");
   alDeleteSources(1, &source);                    // All done. Delete resources,
   alDeleteBuffers(1, &buffer);       //and close down SDL_sound and OpenAL.
   Sound_Quit();
   CloseAL();
   return 0;
}
```

Listado 4: Función "main" de alplay.c

Para compilarlo debes bajarte los archivos del GitHub de *OpenAL Soft*[10] y, si te haces una copia propia de *examples/alplay.c* para poder trastear con él, lo puedes compilar con la orden[11]:

```
$ gcc alplay.c  openal-soft-master/examples/common/alhelpers.c
 openal-soft-m
aster/common/threads.c openal-soft-master/build/config.h -o alplay
-I. -Iopenal-soft-master/common/
-Iopenal-soft-master/examples/ `sdl2-config --cflags --libs` `pkg-
config openal --cflags --libs` `sdl-config --cflags --libs`
-lSDL_sound -lpthread
```

Con la orden *ldd* puedo ver que ha enlazado con las diferentes librerías que dan soporte a los mencionados formatos de ficheros de audio:

```
$ ldd alplay
…
       libopenal.so.1 => /usr/lib/x86_64-linux-gnu/libopenal.so.1
(0x00007fca1d0c3000)
       libSDL_sound-1.0.so.1 => /usr/lib/x86_64-linux-
gnu/libSDL_sound-1.0.so.1 (0x00007fca1ce7b000
…
       libSDL-1.2.so.0 => /usr/lib/x86_64-linux-gnu/libSDL-1.2.so.0
(0x00007fca1bee6000)
       libmikmod.so.3 => /usr/lib/x86_64-linux-gnu/libmikmod.so.3
(0x00007fca1bca2000)
       libvorbisfile.so.3 => /usr/lib/x86_64-linux-
gnu/libvorbisfile.so.3 (0x00007fca1ba99000)
       libFLAC.so.8 => /usr/lib/x86_64-linux-gnu/libFLAC.so.8
(0x00007fca1b824000)
       libogg.so.0 => /usr/lib/x86_64-linux-gnu/libogg.so.0
(0x00007fca1b61b000)
       libspeex.so.1 => /usr/lib/x86_64-linux-gnu/libspeex.so.1
(0x00007fca1b402000)
...
```

Ahora ya puedes comprobar si esto funciona o no. Yo he cogido un archivo y lo he convertido a diferentes formatos : WAVE, MP3, OGG, etc y ahora a probarlo. Verás que sí funciona, se escucha el audio mientras, en pantalla, se ven dos sencillos mensajes:

$ alplay drHouse.mp3

Opened "Audio Interno Estéreo analógico"

Offset: 6.932948[12]

---

10 Bájate el archivo “`openal-soft-master.zip`” y descomprímelo en un directorio local. Entras al directorio *build* y ejecutas la orden “cmake ..”. También puedes clonarlo con la orden *git*

11 Recuerda que es una sola línea, solo que está aquí recortada por el ancho de la página.

12 Este “Offset” es un mensaje de la aplicación que dice en qué instante de la reproducción del fichero está.

5 Cierre

El objetivo central de este documento ha sido el de proporcionar una solución al problema de la limitación de formatos que puede importar OpenAL. Se ha propuesto el uso de SDL como solución. Y se han construido ejemplos de uso de este SDK para ilustrarlo. Los detalles de instalación y compilación permiten la experimentación práctica de lo que se ha expuesto en un equipo real.

Ahora, para comprobar la utilidad de este trabajo, sería buena cosa que localizaras ejemplos de ficheros de audio y probaras a reproducirlos con este proyecto que se ha explorado. Sería interesante ver el consumo de recursos, porque se carga el fichero completo en memoria, así que puede pasar un tiempo hasta que empiece a sonar ... y puede que no deje mucha memoria libre para otros menesteres. ¡Es el momento de que empieces a experimentar por tu cuenta!

6 Bibliografía

[1] OpenAL. Disponible en <http://www.openal.org>.

[2] - . (2005).*OpenAL 1.1 Specification*. Disponible en <http://www.openal.org/documentation/openal-1.1-specification.pdf>.

[3] *Simple DirectMedia Layer – Homepage*. Disponible en <https://www.libsdl.org/ >.

[4] Loki Software. (2000). *OpenAL 1.0. Specification.* Disponible en <https://pdfs.semanticscholar.org/831a/72e74a6f63dafb1ff74dfa5e311f416bc238.pdf>.

[5] Robinson, C. (2017). *OpenAL Soft GitHub.* <https://github.com/kcat/openal-soft/blob/master/examples/alplay.c>.