UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Reproducción de ficheros FLAC con OpenAL y dr_flac

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 [1] un motor de audio 3D capaz de renderizar el sonido que llega hasta el oyente, de modo que el usuario se escuche envuelto entre las fuentes de sonido. OpenAL no se encargará de cargar audio de 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 [2] y [3] y para proporcionar una capa de nivel superior que se encargue de cargar ficheros de audio se creó ALUT [4] que ofrece la posibilidad de cargar ficheros WAVE desde disco.

Actualmente hay muchos ficheros de audio que compiten por el "mercado" digital de este formato, puede ver una pequeña introducción y caracterización de algunos de ellos en [6].

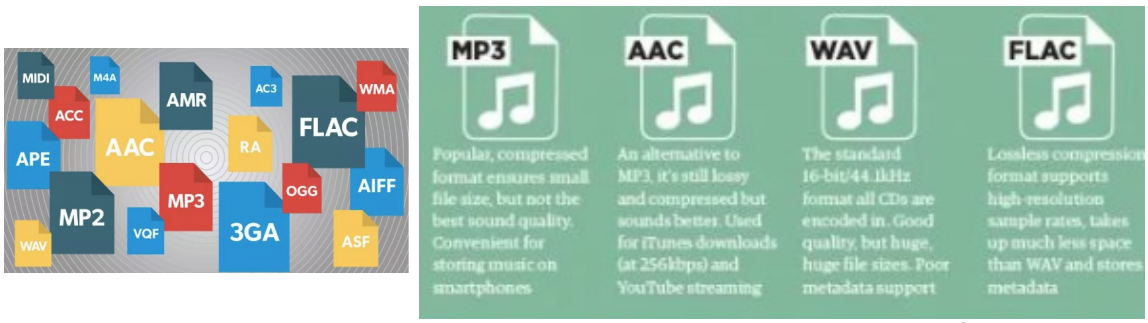


Figura 1: Algunas opciones de formatos de audio existentes. Extraído de [6].

Con la aparición de nuevos formatos de audio, se puede recurrir a las librerías propias de cada uno de ellos para ampliar el conjunto de ficheros de los que OpenAL puede importar audio. De esta manera, se mantiene bajo el consumo de recursos que necesita una aplicación: ajustándose solo los formatos que se sepa que se van a utilizar. En este artículo se verá el uso del formato de audio *Free Lossless Audio Codec* (FLAC) [5] y cómo utilizar la librería *libFLAC* a través del interfaz simplificado que aporta *dr_flac*, sobre la dinámica de uso típica de OpenAL.

2 Objetivos

A partir del estudio del ejemplos que se aborda en este documento, el lector será capaz de:

- Incorporar ficheros de formato FLAC a un desarrollo sobre OpenAL.
- Instalar y compilar una aplicación que hace uso del formato FLAC sobre OpenAL, con las funciones de la biblioteca *libFLAC*.
- Identificar una posible estrategia para la carga completa en memoria de ficheros FLAC.

El presente documento está encaminado a ofrecer una perspectiva inicial de cómo ampliar el conjunto de formatos de ficheros que puede utilizar el motor de audio 3D OpenAL. No es el objetivo de este documento dar una solución global para todos los formatos, ni adentrarse en el detalle de cómo es FLAC, sino mostrar cómo incorporar este formato al repertorio de archivos que puede utilizar una aplicación basada en OpenAL. Así que buscamos una opción sencilla, en cuanto a número de líneas de código, que nos permita precargar los ficheros FLAC para usarlos como efectos de sonido puntuales en nuestras aplicaciones que, con OpenAL, construyan una escena sonora 3D alrededor del usuario.

3 Introducción

FLAC es un codificador y decodificar (codec) de código abierto para audio sin pérdidas desarrollado por Josh Coalson y que ha sido incorporado a la fundación Xiph.org, donde es mantenido por Erik de Castro. El objetivo de este organismo es propiciar protocolos y software abiertos mantener la independencia de los formatos relacionados con la multimedia en Internet.

Los archivos de audio en formato FLAC pueden ser gestionados con la librería *libFLAC* (que es el desarrollo original y está escrita en C). Esta ofrece acceso tanto a la codificación y decodificación, como al formato de fichero (que puede diferente extensión en función de si es el "nativo" de FLAC o el Ogg FLAC), como a los metadatos para obtener las características del fichero. También existe la *libFLAC++*, que es un interfaz de la anterior para ser utilizada desde C++.

Además, a parte de un plugin para XMMS2[1], podemos encontrar en el mismo proyecto dos aplicaciones:

- *flac*, Un conversor de línea de órdenes, especializado en FLAC desde y hacia otros formatos[2].
- *metaflac*, Otra aplicación de línea de órdenes para obtener las propiedades de un fichero FLAC y/o modificar los metadatos del mismo[3].

El formato FLAC es complejo, puede ver una comparativa en [7]. Destacaremos que puede utilizar muestras de 16 y 24 bits y frecuencias de muestreo de entre los 44,1 hasta los 192 kHz. Además, su esquema de compresión sin pérdidas se estima que reduce en hasta un 60% el tamaño de un WAVE y que es superior a ALAC en términos de eficiencia y calidad de codificación de audio [8]. En este trabajo nos centraremos en los valores de 16 bits y 44.1 kHz que son los referentes de alta calidad de audio en CD-A [9]. Ánimo, es momento de tener algún fichero a mano para hacer pruebas o bajarse uno como. p. ej., el primer movimiento de la Sinfonía n.º 6 "Pastoral" de Beethoven[4] o los que se mencionan en [9]. ¿Lo tenemos ya? Entonces seguimos adelante.

Si utiliza las aplicaciones mencionadas verá la complejidad de su contenido. En este artículo, como hemos dicho, nos interesa ver cómo el desarrollador puede cargar en memoria el contenido completo de un archivo en este formato. Por ello haremos uso del interfaz dr_flac [11].

3.1 Reproducción y formatos de fichero de audio en OpenAL

OpenAL es capaz de reproducir audio, en PCM, desde memoria, pudiendo trabajar en modo de precarga o de reproducción en continuo (*streaming*). La precarga supone llevar a memoria, por completo, el contenido del fichero de audio, para después reproducirlo. La fig. 2 muestra de forma gráfica como la librería *Audio Library Utility Toolkit* (ALUT) [4] complementa a OpenAL, en tanto que es capaz de leer archivos WAVE de disco (con la llamada al sistema *read*) y

1 Puede encontrarlos en <https://github.com/xmms2/>. i

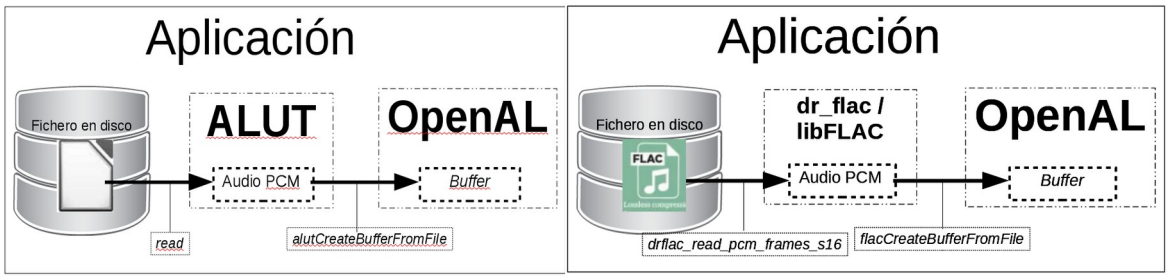
2 Puede ver ejemplos de uso de *flac* en <https://xiph.org/flac/documentation_tools_flac.html >.

3 Puede ver ejemplos de uso de *metaflac* en <https://xiph.org/flac/documentation_tools_metaflac.html >.

4 Puede verla en vídeo en < https://www.youtube.com/watch?v=hvbRwb8twiA> y descargarla de <https://filesamples.com/formats/flac >.

extraer de ese contenido el audio en PCM para asociarlo (con *alutCreateBufferFromFile*) a un *buffer* de OpenAL.

Cuando el formato del fichero no es reconocido por ALUT, como p. ej. el caso de FLAC, es necesario decodificar el formato del fichero para ser capaz de extraer el audio. Para ello es necesario conocer a fondo FLAC. Aquí queremos ofrecer una solución que permita al desarrollador seguir centrándose en su aplicación de audio y no en el estudio de un formato de fichero. La Figura 2 resume, de forma gráfica, los cambios que proponemos: mantener el flujo de trabajo que proporciona ALUT, creando una nueva función que haga transparente el uso del formato FLAC. En este caso, el uso de la librería *libFLAC* permite obtener el audio en PCM para asignárselo a un *buffer* de OpenAL y, cómo queremos una solución que centralice la operativa de cargarlo completamente en memoria, recurrimos a *drl_flac*.



a) *b)*

Figura 2: Esquema básico de gestión de formatos de ficheros en OpenAL: (a) con ALUT y (b) la variación propuesta en este trabajo con libFLAC/dr_flac..

4 Desarrollo

Para llevar a cabo el ejemplo que se muestra en este artículo, se necesita instalar los paquetes de desarrollo de *libFLAC*, lo que puede hacer con la orden:

```
$ sudo apt-get install libflac-dev
```

Puede comprobar que la instalación está operativa ejecutando las aplicaciones *flac* o *metaflac* comentadas en el punto de Introducción.

Ya hemos mencionado que vamos a utilizar la capa de abstracción que nos proporciona el interfaz de dr_flac a libFLAC, así que es necesario también descargar el fichero de [11].

Vamos ya a ver un ejemplo de uso de FLAC con OpenAL que está repartido entre el Listado 1 y el Listado 2. El Listado 1 muestra la parte de declaraciones del código: dependencias y el uso de OpenAL para reproducir el sonido. En él hemos destacado que:

- Las líneas 5 y 6 constituyen la declaración de uso de *dr_flac* y, con ella, la importación del interfaz que esta ofrece sobre *libFLAC*. Implícitamente, se hará la declaración de uso de *libFLAC* transparente para nosotros, puesto que *dr_flac* opera sobre ella.
- Las líneas 17 y 33 son la manera habitual de inicializar el motor de OpenAL con el interfaz de ALUT, seleccionando el hardware a utilizar y liberando los recursos al terminar.
- Entre medias de esas líneas y, siguiendo el modelo de trabajo de ALUT, hemos creado una función (línea 18) *flacCreateBufferFromFile* que nos

permite reunir el código de acceso al fichero de formato FLAC en un solo lugar.

- Para animar el resultado en pantalla, las líneas 42 y 43 muestran un mensaje sobre las dos propiedades que necesitamos conocer para establecer la ligazón con las estructuras de datos de OpenAL: ya hemos dicho que trabajaremos con muestras de 16 bits y frecuencia de muestreo de 44100 Hz, así que es necesario averiguar la duración de sonido (en número de muestras[5]) y el número de canales 25 a la 30 muestran el punto de reproducción en que se encuentra el sonido en el instante actual.
  Así, para los dos siguientes ejemplos de ejecución de la orden con dos ficheros diferentes, vemos el nombre del archivo junto con estos valores que se muestran durante su reproducción. Y, en la línea siguiente, el valor de milisegundos que llevamos de la reproducción y un carácter que va cambiado para dar dinamismo a la visualización del avance.

```
$ flac_OpenAL Symphony\ No.6\ \(1st\ movement\).flac

Symphony No.6 (1st movement).flac: totalPCMFrameCount 32108544 channels 2

    34356 \

$ flac_OpenAL BIS1447-002-flac_16.flac

BIS1447-002-flac_16.flac: totalPCMFrameCount 18451440 channels 2

    978459 -
```

Por su parte, en el Listado 2 se agrupan las instrucciones de acceso sobre FLAC: es la parte encargada de extraer la información de audio digital codificada en el fichero y llevarla a una zona de la memoria central del computador. Hemos resaltado:

- La línea 40 abrirá el fichero con drflac_open_file, lo que permitirá acceder a los metadatos del mismo.
- La línea 41 detectará si no se puede acceder al fichero y terminaría la aplicación en ese punto.
- Si lo es, la línea 45 preparará la estructura de datos que contendrá la información de audio en PCM, esto es, el audio digital decodificado.
- La línea 46 rellenará la estructura mencionada, recodificando las muestras a 16 bits por canal, si es necesario, para poder asignarlo directamente a un buffer de OpenAL.
- Las líneas 53 y 54 hacen la asignación, con lo que tenemos una copia dentro del motor de OpenAL y podemos liberar los recursos asociados al acceso del fichero FLAC (líneas 56 y 57).

Ya casi lo tenemos, solo es necesario que compilemos este ejemplos para poderlo utilizar. Para ello, desde Linux (Ubuntu 20.04), una vez instaladas las dependencias señaladas anteriormente se habrá de ejecutar una órden del estilo de:

```
$ gcc flac_OpenAL.c -o flac_OpenAL  -lFLAC  -lalut  `pkg-config openal --cflags --
libs`
```

o su equivalente:

```
$ gcc flac_OpenAL.c -o flac_OpenAL -lFLAC  -lalut  -I/usr/include/AL
-lopenal
```

5 Si se requiere, sabiendo la frecuencia de muestreo y el número de muestras totales se puede obtener la duración, en tiempo, de la reproducción del sonido.

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. ##include <unistd.h> // usleep
4.
5. #define DR_FLAC_IMPLEMENTATION
6. #include "dr_flac.h"
7. #include <AL/alut.h>
8.
9. ALuint flacCreateBufferFromFile( char *argv );
10.
11.int main(int argc, char **argv) {
12.     ALuint source, bufferOpenAL;
13.     ALint state;
14.     static char spinner[] = "|/-\\";
15.     static int spinpoint = 0;
16.
17.     alutInit (&argc, argv); // Inicializa OpenAL
18.     bufferOpenAL = flacCreateBufferFromFile( argv[1] );
19.     if( !bufferOpenAL ) { alutExit ();  return( 1 );   }
20.     alGenSources(1, &source);
21.     alSourcei(source, AL_BUFFER, bufferOpenAL);
22.     alSourcePlay(source);
23.     state = AL_PLAYING;
24.     alSourcef(source, AL_GAIN, 0.5);
25.     while ( state == AL_PLAYING ) {
26.        printf("%13d %c\r",
27.               spinpoint, spinner[spinpoint++ % 4]); fflush(stdout);
28.        usleep(1000);
29.        alGetSourcei(source,AL_SOURCE_STATE,&state);
30.     }
31.     alDeleteSources(1, &source);
32.     alDeleteBuffers(1, &bufferOpenAL);
33.     alutExit ();
34.     return EXIT_SUCCESS;
35.} // main
...
```

Listado 1: Listado de flac_Openal (parte 1)

Cuento con el lector está copiando el código en un fichero propio, en la siguiente hoja está el resto del código. ¿A que es cortito? Venga, que aquí no lo puedo hacer sonar. ¿Lo tenemos ya? Entonces seguimos adelante.

```
...
36.ALuint flacCreateBufferFromFile( char *fitxer ) {
37.     ALuint bufferOpenAL;
38.     ALenum err, format;
39.
40.     drflac* pFlac = drflac_open_file(fitxer, NULL);
41.     if (pFlac == NULL) { return -1; }
42.     printf("%s: totalPCMFrameCount %llu channels %d \n",
43.            fitxer, pFlac->totalPCMFrameCount, pFlac->channels );
44.
45.     int16_t* pSampleData = (int16_t*)malloc((size_t)pFlac-
>totalPCMFrameCount * pFlac->channels * sizeof(int16_t));
46.     drflac_read_pcm_frames_s16(pFlac, pFlac->totalPCMFrameCount,
pSampleData);
47.     pFlac->channels == 2?
48.       format = AL_FORMAT_STEREO16 : format = AL_FORMAT_MONO16 );
49.
50.     alGenBuffers(1, &bufferOpenAL);
51.     err = alGetError();
52.     if (err) { printf("%s\n", alutGetErrorString(err)); return -1; }
53.     alBufferData(bufferOpenAL, format, pSampleData,
54.                  pFlac->totalPCMFrameCount, 44100 );
55.
56.     drflac_free(pSampleData, NULL);
57.     drflac_close(pFlac);
58.    return bufferOpenAL;
59.} // flacCreateBufferFromFile
```

Listado 2: Listado de flac_Openal (parte 2)

Esperamos que el lector haya tenido cuidado respetando la sintaxis de cualquiera de las dos posibles líneas de compilación: comillas, guiones, espacios en blanco, mayúsculas o minúsculas (si su sistema lo permite, algunos más simples no lo hacen). ¿Lo tenemos ya compilado sin errores?

Y a qué espera, escoja un fichero en formato FLAC y compruebe que se reproduce. ¿Lo hace? Bien. Entonces es hora de ir recapitulando.

5 Conclusiones y cierre

El motor de audio 3D, OpenAL, parte de una especificación reducida para conseguir un impacto mínimo en el consumo de recursos de una aplicación. Sus capacidades de importación de ficheros se pueden ampliar utilizando una librería genérica como ALUT.

La existencia de un número tan alto de formatos de audio que podemos encontrar actualmente hace interesante que el desarrollador pueda seleccionar

el uso de bibliotecas especializadas en el conjunto de formatos que vaya a utilizar para poder reducir la sobrecarga de código que puede no utilizarse en una aplicación. En ese sentido, se ha revisado el camino que sigue la información de audio desde ficheros FLAC para su inclusión en forma de trabajo habitual de OpenAL y, así, proponer un ejemplo de uso de audio basado en la librería *libFLAC*.

Al lector interesado en los detalles de bajo nivel de uso de FLAC puede consultar la documentación en [10] y el ejemplo de código de FlacDecoder[6]. El lector puede ahora experimentar con esta base para incorporarla a su aplicación. ¿Por qué no lo comprueba?

6 Bibliografía

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

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

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

[4] The OpenAL Utility Toolkit. Disponible en <http://distro.ibiblio.org/rootlinux/rootlinux-ports/more/freealut/freealut-1.1.0/doc/alut.html >.

[5] Free Lossless Audio Codec. Sitio web. Disponible en <https://xiph.org/flac/>.

[6] B. Scarrott, (2021). MP3, AAC, WAV, FLAC: all the audio file formats explained. Disponible en <https://www.whathifi.com/advice/mp3-aac-wav-flac-all-the-audio-file-formats-explained >,

[7] -. (2013). Size Comparison Chart of Various Formats DSD-WAV-FLAC-mp3. Dispoinible en <https://dsd-guide.com/size-comparison-chart-various-formats-dsd-wav-flac-mp3#.YLorj5ozaV4>.

[8] R. Boffard. (2018). WAV VS. FLAC VS. MP3: AUDIO FILE FORMATS EXPLAINED. Disponible en <https://www.themasterswitch.com/wav-vs-flac-vs-mp3-audio-file-formats-explained>.

[9] eClassical.com. (-) About audio formats FLAC and MP3. Disponible en <https://www.eclassical.com/pages/24-bit-faq.html>.

[10] Documentación FLAC: API 1.3.1 Disponible en <https://xiph.org/flac/api/> .

[11] D. Reid. dr_flac. Disponible en <https://mackron.github.io/dr_flac>.

6 Puede encontrarlo en < ddiakopoulos / libnyquist

https://github.com/ddiakopoulos/libnyquist/blob/master/src/FlacDecoder.cpp >