UNIVERSIDAD POLITECNICA DE VALENCIA

# Uso del micrófono para captura de audio en OpenAL


| | |
|---|---|
| **Apellidos, nombre** | Agustí i Melchor, Manuel (magusti@disca.upv.es) |
| **Departamento** | Departamento de Informática de Sistemas y Computadores |
| **Centro** | Universidad Politécnica de Valencia |

UNIVERSIDAD POLITECNICA DE VALENCIA

# 1 Resumen de las ideas clave

En este artículo vamos a presentar una introducción a las funciones de captura de audio mediante el micrófono dentro del API de OpenAL. Se puede encontrar un desarrollo más detallado de esta librería de procesado de audio espacial en [1] y [3]. En este caso de estudio la atención se centra en el aspecto que da título a este artículo.

Se va a presentar un enfoque que permita trabajar en cualquier sistema operativo desarrollando aplicaciones que hagan uso de operaciones sobre audio. Los ejemplos se han probado sobre una instalación de x86_64 *GNU/IUbuntu* 10.04 (*lucid*) y *MS/Windows* XP 32 bits.

# 2 Objetivos

Una vez que el alumno lea con detenimiento este documento y experimente con los ejemplos propuestos, será capaz de:

- Generar un ejecutable que utilice las bibliotecas de programación de OpenAL y *libsndfile*, sin instalar ningún entorno pesado, desde la línea de órdenes.
- Presentar un ejemplo que integra OpenAL con *libsndfile.* Ambos son ejemplos de bibliotecas de funciones de carácter abierto y multiplataforma que permitarán al lector proseguir su autoaprendizaje de forma independiente.
- Averiguar la disponibilidad del hardware e inicializar el mismo para el uso del micrófono.
- Generar ficheros de audio sin formato para disponer de una operativa que no precisa grandes recursos. Así como también generar ficheros de audio con formato un estándar (WAVE) para poder compartir resultados con diferentes aplicaciones.
- Utilizar las operaciones propias de una librería externa a OpenAL, como lo es *libsndfile,* para manipular ficheros en formato WAV y, así, complementar lo que no es parte del API de OpenAL

# 3 Introducción

El presente desarrollo está encaminado a exponer las posibilidades de captura de audio a través del hardware disponible utilizando las funciones del API de OpenAL que permiten añadir al sistema la funcionalidad de audio en tres dimensiones.

De forma breve, OpenAL se estructura ([2] y [4]) en base a cinco objetos: el oyente (*listener*), las fuentes (source) de sonido, el sonido sin procesar en alguna zona de memoria (*buffer*), el contexto (*context*) o conjunto de factores que definen la escena y el dispositivo (device) o manejador con el que comunicarse con la tarjeta de sonido. La figura 1 muestra los objetos básicos de OpenAL y sus relaciones.

El acceso a la información de audio proveniente del exterior es una faceta de esta librería que suele verse poco explicada en la documentación. El motivo,

generalmente, es por que en la mayor parte de ocasiones esta información se guarda en un fichero binario sin formato y no es un tema prioritario en sus líneas de desarrollo actuales. Alguno de estos comentarios se puede ver en foros como el de OpenAL ("RFC: OpenAL audio recording extension"[1] y "[Openal] Recording and Playing Simultaneously"[2]) o en StackOverflow "Recording Audio with OpenAL"[3]

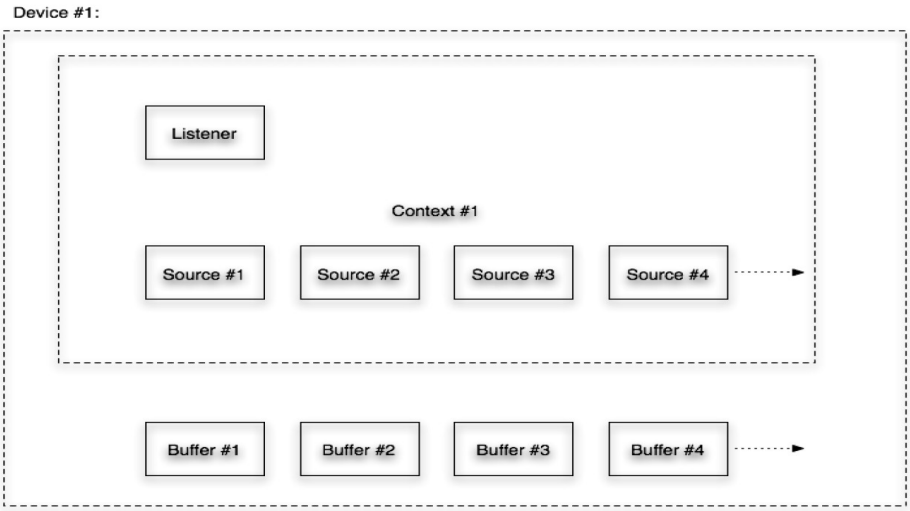


*Figura 1: Un ejemplo de jerarquía entre objetos de OpenAL [2]*

Si se quiere facilitar a otros el uso de los ficheros que se generen o si no es para una aplicación propia es interesante utilizar algún formato estándar. OpenAL no ofrece esta operatividad, por definición. No es su cometido.

Es necesario recurrir a otras librerías para guardar en disco, con un determinado formato, la información de audio adquirida. Aquí haremos uso de libsndfile[5] para esta tarea, creando un fichero en formato WAVE. Se puede consultar más información al respecto del formato creado por Microsoft e IBM en 1991 en [6]. Al mismo tiempo, generaremos un fichero sin cabecera para comparar prestaciones y requerimientos del entorno de trabajo.

## 3.1 La plataforma de trabajo

En este apartado propondremos un diagrama de bloques del sistema a desarrollar, el conjunto de funciones que vamos a emplear de OpenAL y la librería *libnsdfile* que vamos a utilizar para llevar a cabo lo que no es misión de OpenAL. Sea la que sea que escoja el usuario hay que comprobar que se dispone de las herramientas necesarias y que se han instalado correctamente para poder proceder a la compilación del código fuente.

---

[1]<http://opensource.creative.com/pipermail/openal-devel/2004-January/002337.html>.

[2]<http://opensource.creative.com/pipermail/openal/2007-July/010573.html>.

[3]< http://stackoverflow.com/questions/3056113/recording-audio-with-openal>.

### 3.1.1 Esquema de bloques de la aplicación

El diagrama de bloques de la aplicación desarrollada se muestra en la fig. 2 y consta de tres grandes pasos que se describen con detalle en los subapartados del punto de “Desarrollo”.

Básicamente es OpenAL quien lleva la voz cantante: comprobará la disponibilidad del hardware e inicializará el mismo; mientras no se dé la condición de salida irá repetidamente obteniendo datos del micrófono y los guardará en disco; al terminar debe liberar los recursos que ha obtenido para realizar una salida elegante.

Para guardar los ficheros sin formato (también llamado “en crudo” o “RAW”) no es necesaria ninguna funcionalidad aparte de la que proporciona el lenguaje de programación, respecto al acceso a los ficheros en formato binario. Pero para guardar la información en formato WAVE, ya hemos dicho que OpenAL no dispone de funciones debido a su propia naturaleza, por lo que utilizaremos *libsndfile* para guardar los datos con ese formato.

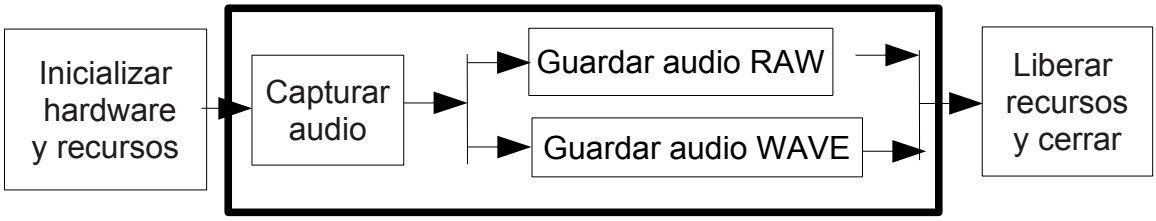


*Figura 2: Diagrama de bloques de la aplicación expuesta.*

### 3.1.2 Comprobación e instalación de software

Suponemos que se dispone de una versión de OpenAL operativa. De no ser así, en GNU/Linux se recomienda utilizar el instalador gráfico que se acostumbre o las órdenes siguientes. Se puede comprobar que se tiene una versión de OpenAL instalada con

```
$ sudo apt-cache search openal
```

En caso contrario, se recomienda instalar OpenAL y ALUT con:

```
$sudo apt-get install libopenal1 libopenal-dev
```

```
$ sudo apt-get install libalut0 libalut-dev
```

Pruebe a compilar los ejemplos que ha instalado (en la plataforma indicada están en /usr/share/doc/libopenal1/examples). Necesitará una orden del estilo de

```
$ gcc -o nombreEjecutable -lalut -lopenal -lm -lsndfile"
ficheroCodigoFuente.c
```

o bien, si dispone de la utilidad *pkg-config* instalada

```
$ gcc -o nombreEjecutable `pkg-config freealut --cflags --libs`
ficheroCodigoFuente.c
```

que hará lo propio adaptándose a las particularidades de la plataforma en que trabaja.

El API de OpenAL dispone de las funciones básicas [2] para adquisición de información desde micrófono agrupadas en la extensión “ALC_EXT_CAPTURE”:

- ALCdevice* alcCaptureOpenDevice(const ALCchar *deviceName, ALCuint freq, ALCenum fmt, ALCsizei bufsize);
- ALCboolean alcCaptureCloseDevice(ALCdevice *device);
- void alcCaptureStart(ALCdevice *device);
- void alcCaptureStop(ALCdevice *device);
- void alcCaptureSamples(ALCdevice *device, ALCvoid *buf, ALCsizei samps);

Otras instrucciones propias de OpenAL nos permiten averiguar ciertas propiedades que aquí no entramos a analizar, por ejemplo para listar los dispositivos en entrada disponibles:

- alcGetString(Null, ALC_CAPTURE_DEVICE_SPECIFIER)
- alcGetString(Null, ALC_CAPTURE_DEFAULT_DEVICE_SPECIFIER)

O para obtener el número de muestras obtenidas, con

- void alcGetIntegerv(ALCdevice * deviceHandle, ALCenum token, ALCsizei size, ALCint *dest);

También se definen estas constantes:

- ALC_CAPTURE_DEFAULT_DEVICE_SPECIFIER
- ALC_CAPTURE_DEVICE_SPECIFIER
- ALC_CAPTURE_SAMPLES

Por su parte, Libsndfile [5] es una biblioteca de funciones escrita en C para lectura y escritura de ficheros de audio en los formatos *MS/Windows WAVE* y *Apple/SGI AIFF*. Se distribuye bajo licencia *Gnu Lesser General Public* y se puede utilizar en GNU/Linux así como en cualquier UNIX (incluyendo Mac OS/X) y en distribuciones de MS/Windows de 32 y 64 bits. Para la instalación de *libsndfile* haremos:

```
$ sudo apt-get libsndfile1

$ $ apt-get install libsndfile-dev libsndfile
```

Y para la compilación del código que la utilice, añadiremos la librería en la línea de órdenes a utilizar:

```
$ gcc -o nombreEjecutable -lalut -lopenal -lsndfile -lm
ficheroCodigoFuente.c
```

# 4 Desarrollo

En este apartado se detallan las etapas expuestas en el diagrama de bloques de la fig. 2. El código encargado de estas tareas se muestra en los siguientes listados de código. Empezando por el listado 1 que muestra el esquema básico donde encajan las piezas. Todo gira sobre la variable “Buffer” que se rellenará

con datos extraídos del micrófono, se guardará en disco y así hasta que el usuario lo pare. Si se hace lo bastante rápido no se “notará”.

```
#include <stdio.h>
#include <math.h>
#include <malloc.h>
#include <sndfile.h>
#include <AL/al.h>
#include <AL/alc.h>
#include <AL/alut.h>
#include <termios.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

int kbhit();

#define  OUTPUT_WAVE_FILE                    "Capture.wav"
#define  OUTPUT_WAVE_FILE_RAW                "CaptureCrudo"
#define BUFFERSIZE                           4410

const int SRATE = 22050;
ALint nMostres;

int main(int argc, char *argv[])
{
  double freq;
  const ALCchar  *szDefaultCaptureDevice;
  FILE *pFile;
  ALchar* Buffer[BUFFERSIZE];
  int i,  error;
  ALuint buffer, fuente;
  ALint sourceState;

 double freq = SRATE;

  // Inicialización (véase listado 2)
  // Captura del audio (véase listado 3)
  // Cerrar la aplicación (véase listado 4)
}


// Función para detectar que se ha pulsado cualquier tecla en terminales UNIX.
int kbhit()
{
  struct termios oldt, newt;
  int ch;
  int oldf;

  tcgetattr(STDIN_FILENO, &oldt);
  newt = oldt;
  newt.c_lflag &= ~(ICANON | ECHO);
  tcsetattr(STDIN_FILENO, TCSANOW, &newt);
  oldf = fcntl(STDIN_FILENO, F_GETFL, 0);
  fcntl(STDIN_FILENO, F_SETFL, oldf | O_NONBLOCK);
```

```
  ch = getchar();
  tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
  fcntl(STDIN_FILENO, F_SETFL, oldf);

  if(ch != EOF)
    {
     ungetc(ch, stdin);
     return 1;
    }
  return 0;
}
```

*Listado 1. Inicializando la aplicación.*

Estas instrucciones en negritas que aparecen como comentarios en el programa principal son a las que les hemos dedicado un subapartado para desglosar allí sus detalles. El primero es la inicialización que atañe a las cuestiones relativas a la detección del hardware presente y su configuración, así como las tareas previas necesarias al uso de los ficheros de audio. En segundo lugar el bucle del programa principal que se ocupará de la adquisición propiamente dicha del audio y, al tiempo, guardará en fichero con el formato escogido la información obtenida. El último bloque es el encargado de cerrar la aplicación, teniendo cuenta de liberar los recursos hardware utilizados y cerrar oportunamente los ficheros de audio.

Aquí hemos dejado ya a la vista el código correspondiente a la función de detección de que se ha pulsado una tecla, por ser un tema tangencial al del audio. Utilizaremos una función "kbhit"[4] para que portar este código a otras plataformas sea lo más claro posible.

## 4.1 Inicialización

Este apartado es el que atañe a las cuestiones relativas a la detección del hardware presente y su configuración, así como las tareas previas necesarias al uso de los ficheros de audio. El listado 2 muestra el contenido que debe sustituir a la línea comentada correspondiente en el programa principal.

Cabe distinguir dos bloques de inicializaciones. El primero es el relativo a la inicialización de hardware. Mediante OpenAL se averigua el contexto y si la versión de OpenAL dispone de la extensión de grabación de audio; en cuyo caso se puede preguntar por el nombre que el sistema le asigna para que lo podamos identificar de cara al usuario. Finalmente se establecen las propiedades de grabación a unos parámetros de calidad media-alta para la grabación de voz: 22100 Hz de frecuencia de muestreo y 16 bits de cuantización. De ahí que se haya establecido el tamaño del vector que contendrá las muestras a BUFFERSIZE elementos-.

El segundo de los bloques es el encargado de la inicialización de los ficheros de resultados que, como hemos expuesto serán dos. Un primer fichero de datos en crudo (sin cabecera) o RAW, que se realiza como un fichero regular especificando que lo que se escriba en él es tal cual lo que se guarda, en binario. El segundo es el fichero con formato, para lo que *libsndfile* nos facilita el abrirlo y establecer las propiedades correspondientes para el archivo WAVE.

---

[4]El código de esta función está tomado de <http://cboard.cprogramming.com/c-programming/63166-kbhit-linux.html>

```
...
 // Hardware
  ALCcontext *pContext = alcGetCurrentContext();
  ALCdevice *pDevice = alcGetContextsDevice(pContext);
  if (alcIsExtensionPresent(pDevice, "ALC_EXT_CAPTURE") == AL_FALSE)
    {
     printf("Fallo al detectar extensión de captura.\n");  return 0;
    }
  szDefaultCaptureDevice = alcGetString(NULL,
                                        ALC_CAPTURE_DEFAULT_DEVICE_SPECIFIER);
  printf("\nDispositivo de captura por defecto es '%s'\n\n", szDefaultCaptureDevice);

  ALCdevice *device;
 device = alcCaptureOpenDevice(szDefaultCaptureDevice, freq, AL_FORMAT_MONO16,
                                    BUFFERSIZE);
   printf("Dipositivo de captura: '%s' esta abierto\n\n", alcGetString(device,
                                                    ALC_CAPTURE_DEVICE_SPECIFIER));

 // Ficheros: RAW
    pFileRAW = fopen(OUTPUT_WAVE_FILE_RAW, "wb");
  // y WAVE mediante libsndfile.
  SF_INFO info;
  info.format = SF_FORMAT_WAV | SF_FORMAT_PCM_16;
  info.channels = 1;
  info.samplerate = SRATE;
  SNDFILE *pFileWAVE = sf_open(OUTPUT_WAVE_FILE, SFM_WRITE, &info);
...
```

*Listado 2. Inicializando la aplicación.*

Asumimos, para simplificar el código, que no hay errores al abrir los ficheros, se recomienda al lector que no deje esto en manos del azar en sus realizaciones.

## 4.2 Captura del audio

El programa principal esperará a que el usuario pulse una tecla para decir que ya no quiere grabar más. La tarea de este bloque es obtener los datos de la adquisición del audio y depositarlas en una estructura de datos de donde se puedan manipular. El listado 3 corresponde al comentario del programa principal con el texto que da nombre a este apartado.

Obsérvese cómo se guarda la información de audio. En el caso del fichero en crudo, simplemente se escribe las muestras, en binario, en disco. Para el caso del fichero WAVE se recurre a la instrucción *sf_writef* para que se realice la estructura con las convenciones propias de este formato.

```
...
  alcCaptureStart(device);

  float bloq = info.channels * 16/8; // bytes por bloque
  while (!kbhit()) {
    alcGetIntegerv(device, ALC_CAPTURE_SAMPLES, BUFFERSIZE, &nMostres);
    printf("nMostress: %d\n", nMostres);

    if (nMostres > (BUFFERSIZE / bloq))
      {
```

```
            alcCaptureSamples(device, Buffer, BUFFERSIZE/bloq);
            fwrite(Buffer,BUFFERSIZE , 1, pFileRAW);
            sf_writef_short(pFileWAVE, (short*)Buffer, BUFFERSIZE/bloq);
   }
    else{
            alcCaptureSamples(device, (ALCvoid *)Buffer, nMostres);
            fwrite(Buffer, nMostres, 1, pFileRAW);
            sf_writef_short(pFileWAVE, (short*)Buffer, nMostres/bloq);
   }
  }
...
```

*Listado 3. Guardar la información de audio en fichero.*

## 4.3 Cerrar la aplicación

Terminada la adquisición de datos la aplicación concluye de manera concienzuda: por un lado, teniendo cuenta de liberar los recursos hardware utilizados y, por otro, cierra oportunamente los ficheros de audio utilizados. El listado 4 muestra la secuencia de instrucciones encargadas de esta parte.

El hardware se libera con las instrucciones de OpenAL, mientras que para dejar los ficheros de datos listos para su uso por otras aplicaciones y/o usuarios es necesario concluir correctamente estos: el fichero en crudo es, nuevamente, el más sencillo; mientras que el WAVE necesita algo más de trabajo que *libsndfile* nos facilita.

```
...
  //cerramos todo: Hard.  ...
  alcCaptureStop(device);
  alcCaptureCloseDevice(device);

  // y ficheros: RAW
  fclose( pFileRAW );
  // …            y WAVE
  sf_write_sync( pFileWAVE );
  sf_close( pFileWAVE );
  printf("\nSaved captured audio data to '%s'\n", OUTPUT_WAVE_FILE);

  alcMakeContextCurrent(NULL);
  alcDestroyContext(pContext);
  alcCloseDevice(pDevice);
 return EXIT_SUCCESS;
}
```

*Listado 4. Cerrando la aplicación.*

Se podría haber reproducido el fichero de audio, con la ayuda de ALUT mediante OpenAL, para que el usuario pueda comprobar cómo ha ido la grabación. Por simplicidad lo hemos obviado.

# 5 Conclusiones y trabajos futuros

A lo largo de este objeto de aprendizaje hemos visto cómo es posible adquirir audio en OpenAL y guardarlo para su uso posterior. Se han utilizado dos formatos: en crudo y WAVE para ofrecer dos soluciones. Su elaboración y flexibilidad deben ser tenidas en cuenta a la hora de elegir una de las dos, por que cada una tiene una vertiente atractiva y debe ser Ud. quién decida la rentabilidad de una y otra en un contexto dado. Recordemos que es necesaria una librería externa a OpenAL (en nuestro caso *libsndfile*, aunque hay otras opciones) para manipular los fichero de audio WAVE.

Observe que la calidad, si los reproduce, es la misma. Para esta tarea le recomendamos utilizar *Audacity*. Eso si, tendrá que ayudarle con el cómo leer del fichero formato. Esto no será necesario con el fichero WAVE, pero verá que esto tiene un coste en su ocupación en bits en disco.

Ahora toca experimentar. Aquí se han fijado algunos elementos por simplificar el código mostrado, se deja de manos del lector interesado profundizar en estos detalles. Un apartado que puede ser interesante de ampliar es la inicialización del hardware disponible: quizá en tu equipo existe más de un micrófono (el de una cámara web y el que de unos auriculares), sería interesante ofrecer al usuario la posibilidad de escoger cuál se utiliza. Así también el nombre del fichero donde se dejan el resultado de la adquisición podría dejarse a la elección del usuario, etc.

Asentado esto, como trabajos futuros, se puede destacar la posibilidad de elección para el usuario de los datos de configuración o la manipulación de la forma de onda para, por ejemplo, pintarla, procesarla, etc. También se puede integrar con OpenALSoft[5] que ofrece un dispositivo virtual para capturar la señal de audio que se genera por el propio motor de OpenAL.

No quiero acabar sin antes agradecer a I. Salvador su colaboración en la elaboración del código que acompaña a este artículo como resultado de su trabajo en la asignatura de Integración de Medios Digitales.

# 6 Bibliografía

[1] Creative Labs: Connect:: OpennAL. http://connect.creativelabs.com/openal/default.aspx

[2] OpenAL 1.1 Specification and Reference. 2005. Version 1.1, <http://connect.creativelabs.com/openal/Documentation/OpenAL 1.1 Specification.pdf>

[3] Garin Hiebert et al. OpenAL Programmer's Guide, OpenAL Versions 1.0 and 1.1. Creative Technology Limited, 2006

[4] The OpenAL Utility Toolkit (ALUT). http://connect.creativelabs.com/openal/Documentation/The OpenAL Utility Toolkit.htm

[5] Libsndfile <http://www.mega-nerd.com/libsndfile/> (último acceso : mayo de 2012).

[6] WAV. Wikipedia, The Free Encyclopedia. <http://en.wikipedia.org/w/index.php?title=WAV&oldid=487823772> (último acceso mayo de 2012).

[5]Véase <http://kcat.strangesoft.net/openal.html>.