Sistemas Multimedia Interactivos e Inmersivos

Práctica de procesamiento de audio:

Audio posicional con OpenAL

Manuel Agustí
Curso 2k18/2k19
Grado de Ingeniero en Informática
Escola Tècnica Superior d'Enginyeria Informàtica
DISCA - UPV

En la presente práctica, se aborda el uso de OpenAL para realizar ejemplos de audio 3D. En PoliformaT, junto al boletín está el código de los ejemplos y una pequeña colección de archivos de audio para hacer pruebas copiado de la distribución de *Linux Ubuntu 16.04* del laboratorio. Se obtendrá una serie de resultados como respuesta a los ejercicios y las actividades propuestas en la misma. Recuerde que hay una actividad que ha de guardar su resultado para añadirlo al porfolio.

1 Introducción

OpenAL es [1] una interfaz de programación multiplataforma[1] para audio multicanal 3D apropiada para el uso en aplicaciones de videojuegos y otras relativas al tratamiento de audio.

La librería permite modelizar una colección de fuentes de audio moviéndose en un espacio tridimensional que son oídos por un único oyente en algún lugar de ese espacio. Los objetos básicos en OpenAL son un oyente (*Listener*), una fuente de audio (*Source*) y una zona de memoria (*Buffer*) que contiene la información de audio. Cada *buffer* puede ser asignado a una o mas fuentes que representan posiciones (definidas por coordenadas en un espacio tridimensional) de donde "brota" el audio. Siempre hay un oyente, que representa el punto donde se "escuchan" los sonidos que generan las fuentes (*rendering*).

La funcionalidad de OpenAL se estructura [2] en base a estos objetos:

- Una fuente (*source*) contiene un puntero a una zona de memoria (*buffer*), la velocidad, posición y dirección e intensidad del sonido.
- El oyente (*listener*) representa la velocidad, posición y dirección del mismo, así como la ganancia asociada a todos los sonidos. Aunque se pueden definir varios oyentes sólo uno puede estar activo.
- Los *buffers* contienen audio en formato PCM, en muestras de ocho o dieciséis bits, tanto en monofónico como en formato estéreo.

El motor de OpenAL realiza todos los cálculos necesarios como la atenuación debida a la distancia, el efecto *Doppler*, etc. El resultado final de todo esto para el usuario final es que las aplicaciones realizadas con OpenAL recrean un escenario aural cercano mientras el usuario se mueve en un espacio tridimensional.

Entre estos existen unas relaciones o asociaciones, por lo que se habla de otros dos objetos más:

1. Un contexto (context) es el conjunto de oyente y fuentes de sonido, que modela un conjunto de propiedades relativas a la propagación del audio y que define una determinada "escena" a representar de manera sonora.
2. Un dispositivo (*device*) que es la abstracción que representa el conjunto de hardware y manejador del mismo que finalmente se encargarán de generar el sonido.

La figura 1 muestra los objetos básicos de OpenAL y sus relaciones con los objetos de contexto y dispositivo.

Resumiendo: cuando se inicializa OpenAL, como poco, un dispositivo ha de estar presente. Dentro de ese dispositivo, al menos un contexto ha de crearse. En un contexto hay implícitamente definido un oyente y una serie de fuentes que pueden crearse en él. Cada fuente tiene uno o más *buffers* asociados a ella. Estos *buffers* se comparten entre todos los contextos que se definan sobre un mismo dispositivo.

1 Actualmente está disponible [4] en Mac OS X, iPhone, GNU/Linux (tanto para OSS como para ALSA), BSD, Solaris, IRIX, Microsoft Windows, Xbox, Xbox 360 y MorphOS.

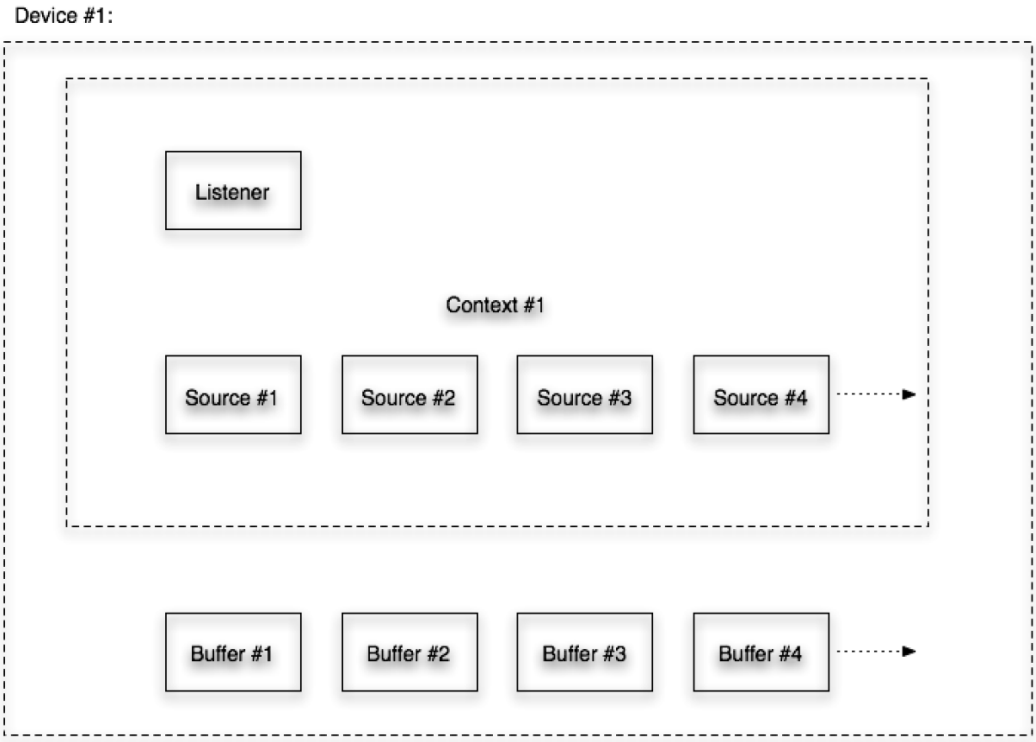

Figura 1: Un ejemplo de jerarquía entre objetos de OpenAL [1]

Desde el punto de vista de la utilización de OpenAL en una aplicación, puede abstraerse como [3] un conjunto de instrucciones que permite especificar las fuentes de sonido y el oyente bajo un sistema de coordenadas espaciales en tres dimensiones, junto a órdenes que permiten controlar cómo serán *renderizados* esos sonidos . El efecto de las instrucciones de OpenAL no está garantizado que sea inmediato, puesto que existen latencias dependiendo de la implementación, pero idealmente estas no deberían ser apreciadas por el usuario.

Un típico programa que utiliza OpenAL empieza con instrucciones para abrir un dispositivo de sonido del sistema, que procesará el sonido y lo reproducirá mediante el hardware asociado (altavoces o auriculares). Después se crea un contexto y se asocia con un dispositivo, dentro del cual también se definirá un oyente. A partir de ese momento, sobre el contexto activo, se disponen fuentes de sonido, en las que se pueden modificar las propiedades relativas a sus coordenadas espaciales y a su orientación; junto a operaciones de procesado del audio que afectan a cómo se modifica la señal de audio original en la escena (por ejemplo la atenuación). Las fuentes se inicializan con sonidos que se obtienen a partir de los *buffers*, que almacenan la descripción de un sonido desde un fichero o partir de una función de síntesis de una señal básica. El buffer se asignará a una fuente y desde ese momento puede "activarse", empezará a reproducir el sonido en cuestión con las modificaciones que la ejecución del resto del programa impongan. Terminada la aplicación hay que liberar los recursos creados antes de salir de la misma.

La especificación de OpenAL incluye dos subsecciones en la interfaz del programador (API): las de núcleo (AL) y las de contexto (ALC, *Audio Library Context*) que se utiliza para gestionar un contexto y el uso de recursos en las diferentes plataformas en que se puede utilizar. Sobre estas, existe un nivel superior de agrupación de funciones básicas y de otras funciones para facilitar el uso de formatos de ficheros de audio: ALUT (*The OpenAL Utility Toolkit*). Se pueden ver agrupadas por estos niveles en el anexo "Funciones de OpenAL".

2 Ejemplos básicos

Los siguiente ejemplos tienen un interfaz en modo texto para centrarnos en la temática del audio. Son ejemplos de código para ver las cuestiones básicas. Se pretende obtener un esquema mínimo sobre el que después aplicar diferentes ejercicios, así como las actividades que se propondrán, por separado sobre esta misma base.

2.1 'Hello, world' utilizando ALUT

Este es el tradicional primer programa que se busca realizar al empezar a aprender cómo funciona cualquier lenguaje de programación o librería. En este caso, en lugar de escribirlo en pantalla, dice "Hello, world!". Se ha obviado la gestión de errores por simplicidad. Lo muestra el listado 1 y corresponde con el fichero *helloWorld_ALUT.c* de los ejemplos que acompañan a esta memoria.

```
#include <stdlib.h>
#include <AL/alut.h>

int main (int argc, char **argv)
{
 ALuint helloBuffer, helloSource;

 alutInit (&argc, argv);

 helloBuffer = alutCreateBufferHelloWorld ();
 alGenSources (1, &helloSource);
 alSourcei (helloSource, AL_BUFFER, helloBuffer);
 alSourcePlay (helloSource);
 alutSleep (1);

 alDeleteSources( 1, &helloSource );
 alDeleteBuffers( 1, &helloBuffer );
 alutExit ();
 return EXIT_SUCCESS;
}
```

Listado 1: Código del ejemplo helloWorld_ALUT.c.

Para obtener un ejecutable, por ejemplo de nombre *helloWorld*, se puede compilar el ejemplo con la orden:

```
$ gcc helloWorld_ALUT.c  -lalut -lopenal -o helloWorld_ALUT
```

Para ejecutarlo bastará invocar el nombre del fichero resultante, si todo ha ido bien:

```
$ helloWorld_ALUT
```

Ejercicio 1.1. Probar este código: compilarlo y ejecutarlo.

Ejercicio 1.2. Anotar la descripción breve de las funciones *alGenSources*, *alSourcei* y *alSourcePlay* de la referencia de OpenAL [4].

Ejercicio 1.3. Anotar la descripción breve de las funciones *alutInit*, *alutCreateBufferHelloWorld* y *alutExit* de la referencia de ALUT [5].

El ejemplo *openal_info.c* muestra cómo averiguar, en tiempo de ejecución, las posibles extensiones instaladas de OpenAL, drivers y los posibles dispositivos hardware que hayan instalados en un equipo para poder dar a escoger o comprobar si es posible realizar una acción.

Ejercicio 1.4. Ejecute el ejemplo openal_info en el equipo del laboratorio y en el suyo propio. Anote la salida y compruebe las diferencias.

2.2 Generando señales simples

Se pueden generar, aparte de la síntesis de "Hello world", otros tipos de ondas básicas como: senoidales, cuadradas, triangulares, impulsioanales y ruido blanco. El fichero *openal_senyales.c* de los ejemplos que acompañan a esta memoria contiene el código para este apartado. Para ello recurriremos a la llamada *AlutCreateBufferWaveform* que podemos ver descrita en la documentación de ALUT [5] y que tiene el interfaz:

ALuint alutCreateBufferWaveform (ALenum waveshape,
ALfloat frequency,

```
...
int main (int argc, char **argv)
{
 ALuint buffers[6], fuente;
…
 buffers[0] = alutCreateBufferHelloWorld();
 buffers[1] = alutCreateBufferWaveform(ALUT_WAVEFORM_SINE, 440.0, 0.0, 1.0);
 buffers[2] = alutCreateBufferWaveform(ALUT_WAVEFORM_SQUARE, 440.0, 0.0, 1.0);
 buffers[3] = alutCreateBufferWaveform(ALUT_WAVEFORM_SAWTOOTH, 440.0, 0.0, 1.0);
 buffers[4] = alutCreateBufferWaveform(ALUT_WAVEFORM_WHITENOISE, 440.0, 0.0, 1.0);
 buffers[5] = alutCreateBufferWaveform(ALUT_WAVEFORM_IMPULSE, 440.0, 0.0, 1.0);
  alGenSources (1, &fuente);
 printf("'h'ellow, 's'ine, sq'u'are, sa'w'tooh, white'n'oise, 'i'mpulse\n");

 do {
  switch ( c )
  {
    case 'h': alSourcei(fuente, AL_BUFFER, buffers[0];  break;
    case 's': alSourcei(fuente, AL_BUFFER, buffers[1];  break;
    case 'u': alSourcei(fuente, AL_BUFFER, buffers[2];  break;
    case 'w': alSourcei(fuente, AL_BUFFER, buffers[3];  break;
    case 'n': alSourcei(fuente, AL_BUFFER, buffers[4];  break;
    case 'i': alSourcei(fuente, AL_BUFFER, buffers[5];  break;
  } // switch

  alSourcePlay (fuente);
  alutSleep (1);
…
 }while ((c!='q') && (c!='Q'));
…
 alDeleteBuffers(6, buffers);
 alutExit ();
 return EXIT_SUCCESS;
}
```

Listado 2: Generando señales simples (código del ejemplo openal_senyals.c).

ALfloat phase,
ALfloat duration);

Los parámetros permiten especificar la forma de la onda, la frecuencia, ... Con lo que podemos hacer variaciones sobre el ejemplo anterior cambiando la función que genera la onda a reproducir. El listado 2 muestra parte del código de *openal_senyales.c* para generar tipos de ondas diferentes y reproducirlos como respuesta a la pulsación de una tecla.

Ejercicio 2.1. Realizar una variación de este código que haga sonar una escala musical. Para ello se habrá de generar una onda básica del mismo tipo pero con diferentes valores de frecuencias. Por ejemplo, se puede probar con los valores [8] (en *Hz*) siguientes*: 262, 294, 330, 349, 392, 440, 494.*

2.3 Trabajando con ficheros de audio

Siguiendo con nuestra aproximación vamos a ver una variación del holaMundo que ahora tomará la señal de audio a hacer sonar desde un fichero con la información extraída a partir de la documentación sobre ALUT [5]. Observar como en este caso no se deja sonar un tiempo fijo, sino que se va consultando y, mientras queden datos, se mantiene el programa en espera.

```
…
int main (int argc, char **argv)
{
...
  buffer = alutCreateBufferFromFile( argv[1] );
…
   alGenSources (1, &fuente);
   alSourcei(fuente, AL_BUFFER, buffer);

   alSourcePlay (fuente);
   // Lo vamos a dejar sonar mientras hayan datos.
 alGetSourcei( fuente, AL_SOURCE_STATE, &sourceState);
 while (sourceState == AL_PLAYING)
  alGetSourcei( fuente, AL_SOURCE_STATE, &sourceState);
...
}// main
```

Listado 3: Extracto de código del ejemplo openal_fitxer.c.

Ejercicio 3.1: Probar este código: compilarlo y ejecutarlo pasándole un nombre de fichero de audio, p. ej. con

```
$ openal_fitxer Audios/hello16000.wav
```

Ejercicio 3.2. Comprobar que soporta la reproducción de ficheros WAVE mono y estereofónicos. A partir de los ejemplos adjuntos al código en el directorio *Audios*, pruebe con la orden

```
$ file Audios/*
```

busque, de entre estos ejemplos, ficheros que contengan valores diferentes de número de canales (monofónico y estéreo), frecuencias de muestreo (8000, 16000, 22050, 44100 y 48000) y tamaño de

las muestras (8 y 16 bits). Anote los que encuentre en una tabla y diga si ha podido reproducirlos no.

3 Audio posicional

Para ponerlo en un ambiente visualmente tridimensional (3D) recurriremos a la plataforma que nos proporciona el proyecto PIGE (*Platform Independent Game Engine*) [7] y que se puede ver en la figura 2. En el apartado de tutoriales de este proyecto se encuentra uno sobre OpenAL[2] que tiene este interfaz. De este ejemplo cabe destacar la función *main*, *init*, *keyboard* y *specialKeys*.

Ejercicio 3.1. Obtener el código del enlace "*Audio posicional: ejemplo del proyecto PIGE (openal.c)*" junto al boletín, compilarlo y ejecutarlo. Para compilarlo se utilizará una línea de órdenes como:

```
$ make pige
```

o

```
$ gcc openal.c -o openal -lalut -lopenal -lglut -lGLU -lGL -lm
```

Se puede ejecutar con la orden[3]:

```
$ openal Audios/Front_Left.wav Audios/Front_Right.wav Audios/Front_Center.wav
```

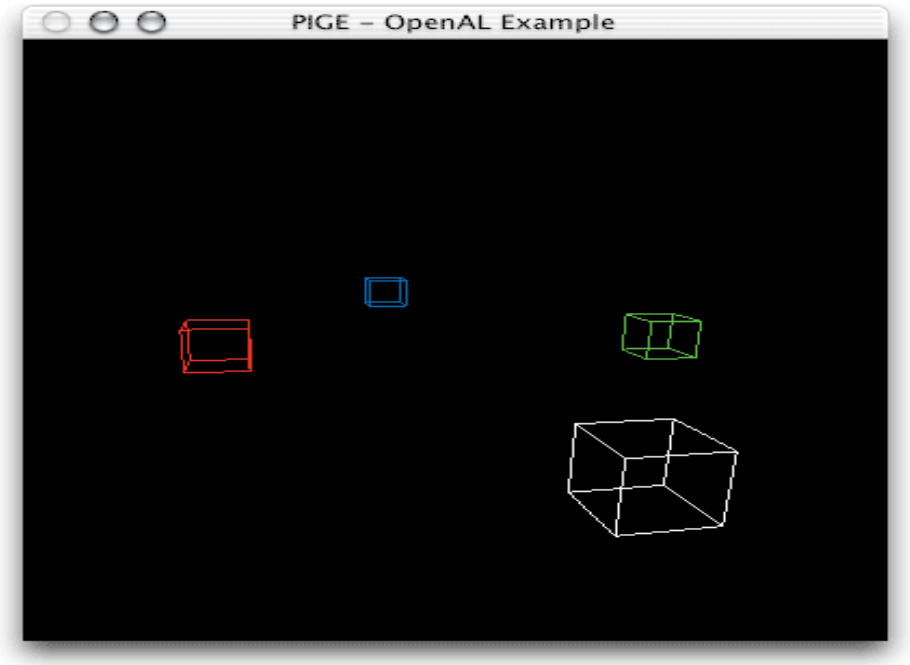

Figura 2: Un momento de la ejecución de PIGE.

Durante la ejecución del mismo la fuente de sonido 1 (la de la izquierda) se activa pulsando la tecla '1' sobre la ventana activa del programa y se desactiva con '4'. Igualmente sucede con la fuente 2 (la del centro) y las teclas '2' y '5; y, por último, a la derecha está la fuente 3, que se maneja con las teclas '3' y '6'. La posición del cubo que muestra la del oyente se controla con las teclas del cursor.

Puede utilizar la orden *file* para comprobar las propiedades un fichero y la orden *sox* para convertir

2 Se puede encontrar en la URL <http://www.edenwaith.com/products/pige/tutorials/openal.php>.

3 Siendo *fichero?.wav* un fichero monofónico en formato WAVE, p. ej., alguno de los disponibles en la máquina del laboratorio (utilizando la orden *locate* para encontrarlos) o en la red (siempre que su licencia lo permita) o del que ya disponga.

a monofónico cualquier otra versión del fichero WAV.

Ejercicio 3.2. Indicar si es posible identificar cual de los objetos de los que aparecen en la "escena" es el que tiene asignado cada fichero que se le pasa como parámetro en la llamada y cómo se puede hacer durante la ejecución de la aplicación, escuchando la misma.

Ejercicio 3.3. ¿Cómo afecta a la ejecución del ejemplo que se modifiquen las coordenadas de orientación del oyente? Aplique la respuesta al caso concreto de que se produzca el siguiente cambio de valores: de

ALfloat listenerOri[]={0.0,0.0,1.0, 0.0,1.0,0.0};

a

ALfloat listenerOri[]={0.0,0.0,-1.0, 0.0,1.0,0.0};

4 Actividades

Se propone experimentar con el código, la funcionalidad relativa a elementos o propiedades de la especificación de OpenAL, permitiendo que se varíen valores de los parámetros utilizados. Para ello, se propone una extensión del ejemplo anterior de PIGE que se denomina *openal_escena.c* (véase Figura 3).Este ejemplo muestra la posición del oyente y la de cuatro fuentes de sonido. De estas fuentes, tres permanecen estáticas durante la reproducción del sonido que tienen asignado, pero cambian de posición aleatoriamente cada vez que aparecen en pantalla. La cuarta fuente se puede mover y podemos controlar la velocidad de la misma.

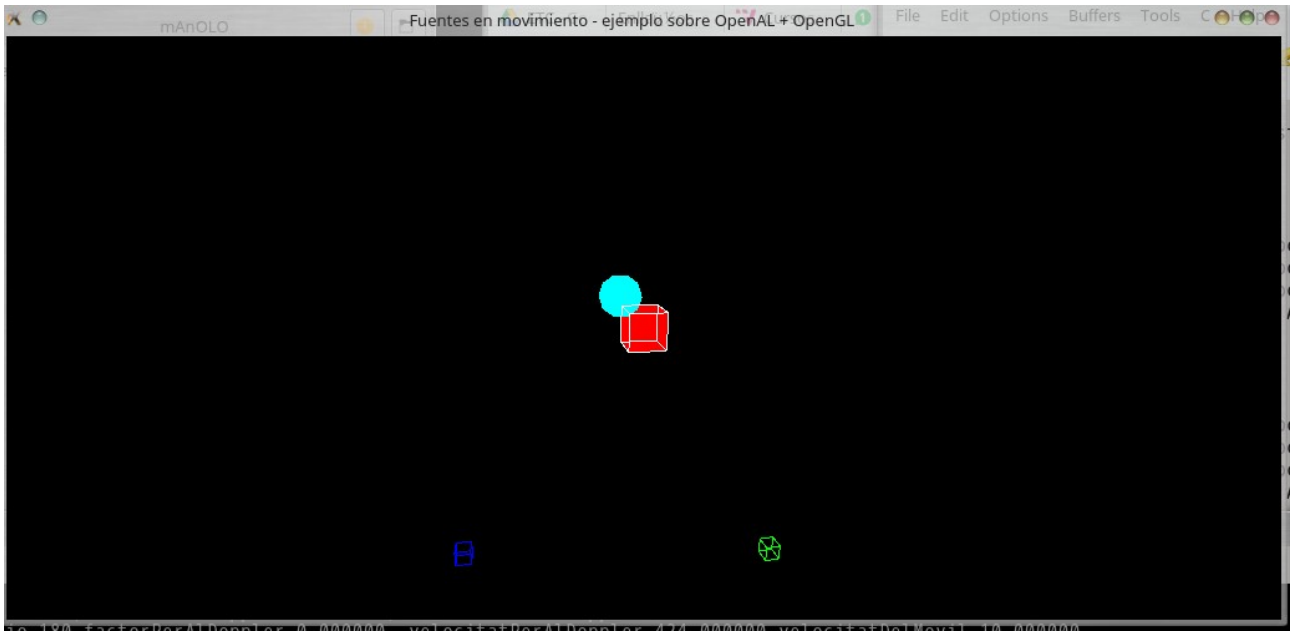

Figura 3: Un momento de la ejecución del ejemplo openal_escena.

La operativa de este ejemplo se basa en el uso de las siguientes teclas:

- Del '1' al '4' activan/desactiva la reproducción del sonido por parte de una fuente de sonido diferente: un sonido de batalla (BATTLE), dos de disparos (GUN1 y GUN2) y uno de ambulancia (VEHICLE).

- Con las teclas ‘f’ o ’F’ se pasa del modo ventana a pantalla completa y viceversa,
- Con las teclas ‘a’, ‘s’, ‘q’ y ‘z’ (o sus respectivas mayúsculas ‘A’, ‘S’, ‘Q’ y ‘Z’) se controla la posición del oyente en el sentido del los ejes X (horizontal) y Z (profundidad). También se puede utilizar las flechas de cursor para controlar estos movimientos del oyente.
- Con las teclas ‘v’ y ‘V’ se modifica la velocidad del objeto que se quiere simular que tiene un comportamiento móvil en la escena, el que se denomina VEHICLE.
- Las teclas ‘d’ y ‘D’ modifican el factor de escala con que se aplica el efecto Doppler.
- Las teclas ‘e’ y ‘E’ modifican la velocidad de propagación del sonido, permitiendo simular otros medios[4].
- El programa finaliza con el uso de la tecla ESC (ordinal 27).

Sobre este ejemplo habrá de introducir las siguientes ampliaciones:

- En la documentación de GLUT[5] aparecen otros siete métodos para dibujar objetos geométricos 3D aparte del cubo y la esfera que se utilizan aquí. Utilice los seis que quiera de ellos para incorporar en esta escena los seis “sonidos básicos” del ejemplo *openal_senyales.c*.
- El uso de las teclas para que la pulsación de ‘5’ al ‘0’ activen/desactiven la reproducción del sonido asociado a estas seis nuevas fuentes de sonido.

4 Puede consultar algunos valores en “Velocidad del sonido” <https://es.wikipedia.org/wiki/Velocidad_del_sonido>.
5 Veasé “The OpenGL Utility Toolkit (GLUT) Programming Interface API Version 3” <https://www.opengl.org/resources/libraries/glut/spec3/node80.html>.

5 Anexo: funciones de OpenAL

Este anexo recoge las funciones de esta librería agrupadas por su nivel para facilitar la localización de las mismas.

AL (núcleo de OpenAL):

Relativas a los *buffers*
- alGenBuffers
- alDeleteBuffers
- alIsBuffer
- alBufferf
- alBuffer3f
- alBufferfv
- alBufferi
- alBuffer3i
- alBufferiv
- alGetBufferf
- alGetBuffer3f
- alGetBufferfv
- alGetBufferi
- alGetBuffer3i
- alGetBufferiv

Relativas a las fuentes (*source*)
- alGenSources
- alDeleteSources
- alIsSource
- alSourcef
- alSource3f
- alSourcefv
- alSourcei
- alSource3i
- alSourceiv
- alGetSourcef
- alGetSource3f
- alGetSourcefv
- alGetSourcei
- alGetSource3i
- alGetSourceiv
- alSourcePlayv
- alSourceStopv
- alSourceRewindv
- alSourcePausev
- alSourcePlay
- alSourceStop
- alSourceRewind
- alSourcePause
- alSourceQueueBuffers
- alSourceUnqueueBuffers

Relativas al oyente (*listener*)
- alListenerf
- alListener3f
- alListenerfv
- alListeneri
- alListener3i
- alListeneriv
- alGetListenerf
- alGetListener3f
- alGetListenerfv
- alGetListeneri
- alGetListener3i
- alGetListeneriv

Relativas a propiedades y estados
- alEnable
- alDisable
- alIsEnabled
- alGetString
- alGetBooleanv
- alGetIntegerv
- alGetFloatv
- alGetDoublev
- alGetBoolean
- alGetInteger
- alGetFloat
- alGetDouble
- alDopplerFactor
- alDopplerVelocity
- alSpeedOfSound
- alDistanceModel

Relativas a gestión de errores
- alGetError

Relativas a las extensiones
- alIsExtensionPresent
- alGetProcAddress
- alGetEnumValue

ALC (*Audio Library Context*)

alcCreateContext
alcMakeContextCurrent
alcProcessContext
alcSuspendContext
alcDestroyContext
alcGetCurrentContext
alcGetContextsDevice
alcOpenDevice
alcCloseDevice
alcGetError
alcIsExtensionPresent
alcGetProcAddress
alcGetEnumValue
alcGetString
alcGetIntegerv
alcCaptureOpenDevice
alcCaptureCloseDevice
alcCaptureStart
alcCaptureStop
alcCaptureSamples

Funciones de la interfaz de ALUT The OpenAL Utility Toolkit

Error Handling
- alutGetError
- alutGetErrorString

Initialization / Exit
- alutInit
- alutInitWithoutContext
- alutExit

Sound Sample File Loading
- alutCreateBufferFromFile
- alutCreateBufferFromFileImage
- alutCreateBufferHelloWorld
- alutCreateBufferWaveform
- alutLoadMemoryFromFile
- alutLoadMemoryFromFileImage
- alutLoadMemoryHelloWorld
- alutLoadMemoryWaveform
- alutGetMIMETypes

Version Checking
- alutGetMajorVersion
- alutGetMinorVersion

Sleeping
- alutSleep

6 Bibliografía y referencias

En el PoliformaT de la asignatura, junto al boletín de prácticas, hay copias de estos documentos, junto al SDK y el instalador para Windows. En el caso de Linus/macOS se aconseja recurrir a los repositorios.

[1] OpenAL: Cross Platform 3D Audio. <http://www.openal.org>.

[2] OpenAL. En Wikipedia, The Free Encyclopedia. Retrieved 13:25, Consultada el 12 de Agosto de 2008, URL <http://en.wikipedia.org/w/index.php?title=OpenAL&oldid=231434910>

[3] OpenAL 1.1 Specification and Reference. 2005. Version 1.1, <https://www.openal.org/documentation/openal-1.1-specification.pdf>

[4] Garin Hiebert et al. OpenAL Programmer's Guide, OpenAL Versions 1.0 and 1.1. Creative Technology Limited, 2006. <https://www.openal.org/documentation/OpenAL_Programmers_Guide.pdf >

[5] The OpenAL Utility Toolkit (ALUT). <http://distro.ibiblio.org/rootlinux/rootlinux-ports/more/freealut/freealut-1.1.0/doc/alut.html >

[6] Sound and Music Programming , Devmaster, <http://www.devmaster.net/articles.php?catID=6> (<accedido 17 Sept. 2008).

[7] *Platform Independent Game Engine (PIGE)* <http://www.edenwaith.com/products/pige/tutorials/openal.php > (accedido 17 Sept. 2008)

[8] *Note*. En *Wikipedia, The Free Encyclopedia*. Consultada el 20 de Octubre 2008, 11:07h., from <http://en.wikipedia.org/w/index.php?title=Note&oldid=246433913 >.