UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ALURE: interfaz de alto nivel para OpenAL. Servicios relacionados con el hardware de audio

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 está formado [1] por un núcleo básico de objetos (dispositivo, contexto, fuentes, *buffer* y oyente) que permiten el desarrollo [3] de escenas de audio 3D. OpenAL proporciona funciones para reproducción de audio en un entorno tridimensional virtual, atenuación con la distancia, efecto Doppler y fuentes de sonido direccionales. Tiene soporte para streaming, audio multicanal y grabación de audio (desde micrófono). La extensión *Effects Extension* (EFX) incorpora parámetros ambientales (características del aire), así como las que introducen los objetos y sus materiales (oclusiones y reverberaciones).

La Figura 1 muestra el bloque de OpenAL 1.1 (la última versión publicada de este estándar) con las capas que lo forman, esto es la arquitectura interna de OpenAL. El nivel más bajo es el que forma el núcleo de operaciones básicas que se conoce como *Audio Layer* (AL) y que se encarga de gestionar los *buffers* (que contienen el sonido en memoria sin modificar), las fuentes (que son los objetos que asociados a un *buffer* le confieren propiedades de posición y velocidad en la escena) y el oyente (que es el objeto que representa la posición y velocidad del usuario). Sobre este, el *Audio Library Context* (ALC) que implementa las abstracciones de dispositivo (acceso al hardware de audio) y contexto (la definición de los límites y características de la escena sonora). Por encima de estos dos niveles, *The OpenAL Utility Toolkit* [2] (ALUT) proporciona capacidades de importación de ficheros, generación de señales básicas y alguna otra función auxiliar.

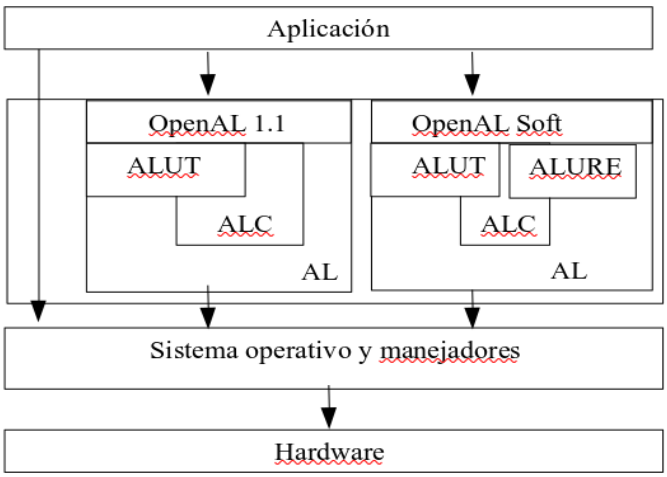

Figura 1: Correspondencia de los niveles del estándar OpenAL con la implementación de OpenAL Soft.

El otro bloque de la Figura 1, OpenAL Soft, es [6] la implementación (totalmente software) del API del estándar de OpenAL para audio 3D, realizado por Chris Robinson. Esta implementación da soporte a toda la jerarquía de niveles de OpenAL y ha ido actualizando sus servicios. Además, añade un bloque, al mismo nivel que ALUT, que se denomina *AL Utilities Retooled* (ALURE) y con el que el autor de *OpenAL Soft* puede tomar decisiones respecto a su evolución, independientemente de ritmo de cambios en el estándar. *ALURE* [7] expande las funcionalidades de ALUT en temas relacionados con la carga de ficheros, la decodificación de sus contenido o la gestión de *streaming*.

La Figura 1 muestra el bloque de *OpenAL Soft* al lado del de OpenAL para que podamos hacernos la composición de lugar: *OpenAL Soft* es, seguramente, la versión que tendremos instalada, ¿por qué no utilizamos las aportaciones de esta implementación que si ha tenido un progreso, unas actualizaciones y unas actualizaciones, que podemos utilizar en nuestros desarrollos?

En este artículo vamos a presentar las opciones que tenemos si queremos aprovechar que en la instalación que podemos tener actualmente de OpenAL en nuestros equipos viene ya, seguramente, preinstalado ALURE, que es una mejora sustancial de ALUT.

2 Objetivos

Este artículo está enfocado a explorar los servicios que proporciona ALUT como API de alto nivel, por lo que a partir del estudio del ejemplo de código que se aborda en este documento, el lector será capaz de:

- Identificar en el código las funciones que pertenecen al nivel de ALUT.
- Identificar en el código las funciones que pertenecen a los niveles ALURE.
- Instalar y compilar una aplicación que hace uso de la librería OpenAL, con el API de ALUT o de ALURE.

No es objetivo de ese artículo explicar el funcionamiento de audio 3D de OpenAL, queremos explorar el papel de ALURE como posible (y hasta recomendable) sustituto de ALUT. Tampoco lo es ser un mero listado exhaustivo de operaciones, sino mostrar las más habituales y proporcionar referencias para entender las operaciones y descubrir otras similares.

3 Introducción

OpenAL Soft proporciona ([6] y [8]) una implementación software de OpenAL, multiplataforma y publicada bajo licencia LGPL. Nace como un *fork* de la versión disponible del repositorio de SVN de openal.org que era de código abierto y estaba disponible solo para entorno Windows.

Sobre OpenAL Soft, ALURE [7] ofrece un interfaz en lenguaje C que se basa en OpenAL y lo expande en temas relacionados con la carga de ficheros, la decodificación de sus contenidos y la gestión de *streaming*. Desde la versión 1.1 de ALURE, está publicado bajo licencia X11/MIT[1], lo que permite su uso en aplicaciones de código abierto o cerrado, *freeware* o comercial. El propósito de esta librería (como lo es el de ALUT al que equivale) es proporcionar funcionalidades que se usen de forma habitual y que varían en función de la plataforma para la que se realiza el desarrollo, como por ejemplo, la carga de ficheros de sonido en los *buffers* de OpenAL o la gestión de la cola de *buffers* para ofrecer una reproducción en continuo (*streaming*).

Con esta idea, ALURE incluye un lector y decodificador nativo básico de WAVE y AIFF, así como también se apoya en librerías externas para la lectura y decodificación de otros formatos populares como: extensiones de WAVE con *libSndFile*, Ogg Vorbis con *VorbisFile*

[1] Puede consultarse en <https://en.wikipedia.org/wiki/MIT_License> al respecto.

y *OpusFile*, Ogg FLAC con *dr_flac*, MP3 con *minimp3. Además, incluye la opción de enlazar con otras librerías externas en tiempo de ejecución lo que permite seguir extendiendo sus capacidades sin tener que modificar el código de ALURE y la opción de deshabilitar estas librerías en tiempo de compilación*.

De entre todas las opciones que ofrece *ALURE*, en este artículo, nos centramos en cómo proporciona a una aplicación funciones para el acceso al hardware de audio: para listar o enumerar las opciones disponibles en un equipo y para inicializarlo.

Acompañaremos esta presentación de sendos ejemplos de código que nos permitirán identificar las operaciones de ALUT vs ALURE en el código. Para ello mostraremos ejemplos de instalación y compilación en plataforma Linux, que es fácilmente reproducible (también en línea de órdenes) en otros sistemas operativos.

Si está preparado, estimado lector, empecemos con la instalación que es breve, con las órdenes:

```
$ sudo apt install libalure-dev
$ sudo apt install libopenal-dev libvorbis-dev libopusfile-dev libsndfile1-dev
```

y comprobaremos su instalación, preguntado por la versión instalada o las dependencias para compilar un programa con esta librería con las siguientes órdenes y el resultado que pueden/deben mostrar:

```
$ pkg-config alure --modversion
1.2
$ pkg-config alure --cflags
-I/usr/include/AL
$ pkg-config alure --libs
-lalure -lopenal
```

4 Desarrollo

Desde la especificación del estándar OpenAL ([3]] y [5]), la secuencia de inicialización de OpenAL consiste en:

- Abrir un dispositivo, esto es acceder al hardware disponible e incializarlo.
- Inicializar un contexto, esto es asociar una escena sonora con un oyente a ese dispositivo.
- Crear los *buffers* que contendrán el sonido y rellenarlos con muestras de sonido.
- Crear la/s fuente/s para posicionar el sonido y asociar el/los *buffer*/s que correspondan a cada una.
- Reproducir el audio.

4.1 API de ALUT para acceso al hardware de audio

El API de ALUT [2] para este cometido está repartido en varios bloques, en los que podemos encontrar:

- Gestión de errores con *alutGetError* y *alutGetErrorString*.
- Inicialización y terminación con *alutInit*, *alutInitWithoutContext* y *alutExit*.
- Identificación de versión con *alutGetMajorVersion* y *alutGetMinorVersion*.
- Pausas con *alutSleep*.

El Listado 1 nos muestra un ejemplo de este grupo de operaciones que ofrece ALUT. Algunas partes están sacadas de [4] y así están los comentarios originales en inglés, otras son propias. Hemos recalcado cuándo aparecen las operaciones propias de ALUT para facilitar su identificación al lector. Le sugerimos que pruebe a ejecutar el código mostrado y, así, la experimentación le llevará a entender el desarrollo mostrado. Para compilarlo habrá de ejecutar la línea:

```
$ gcc openal_init_info_alut.c -o openal_init_info_alut $(pkg-config
freealut openal --cflags --libs)
```

¿Lo tiene? Veamos qué sale.

```
$ openal_init_info_alut
La versio de ALUT instalada es 1.1
OpenAL Renderer is 'OpenAL Soft'
OpenAL Version is '1.1 ALSOFT 1.19.1'
OpenAL Vendor is 'OpenAL Community'
OpenAL Extensions supported are :
AL_EXT_ALAW AL_EXT_BFORMAT AL_EXT_DOUBLE AL_EXT_EXPONENT_DISTANCE
AL_EXT_FLOAT32 AL_EXT_IMA4 AL_EXT_LINEAR_DISTANCE AL_EXT_MCFORMATS
AL_EXT_MULAW AL_EXT_MULAW_BFORMAT AL_EXT_MULAW_MCFORMATS AL_EXT_OFFSET
AL_EXT_source_distance_model AL_EXT_SOURCE_RADIUS AL_EXT_STEREO_ANGLES
AL_LOKI_quadriphonic AL_SOFT_block_alignment AL_SOFT_deferred_updates
AL_SOFT_direct_channels AL_SOFTX_events AL_SOFTX_filter_gain_ex
AL_SOFT_gain_clamp_ex AL_SOFT_loop_points AL_SOFTX_map_buffer
AL_SOFT_MSADPCM AL_SOFT_source_latency AL_SOFT_source_length
AL_SOFT_source_resampler AL_SOFT_source_spatialize
Extensions disponibles per separat:
 1 – AL_EXT_ALAW
...
OpenAL hardware presente:
 OpenAL Soft
Drivers disponibles d'audio:
 1 - OpenAL Soft
Per defecte:
OpenAL Soft
Available Capture Devices are:
 1 - QuickCam Pro 9000 Mono
 2 - Monitor of USB Audio Device Estéreo digital (IEC958)
 3 - USB Audio Device Multicanal
 4 - Monitor Source of Simultaneous output to USB Audio Device Estéreo
digital (IEC958)
```

```c
#include <stdio.h>
#include <stdlib.h>     // exit
#include <AL/alut.h>
#include <string.h>     // strlen

int main (int argc, char **argv) {
  int i = 1;
  const ALchar *pDeviceList, *llista;

  alutInit (&argc, argv);
  printf("OpenAL Renderer is '%s'\n", alGetString(AL_RENDERER) );
  printf("OpenAL Version is '%s'\n", alGetString(AL_VERSION) );
  printf("OpenAL Vendor is '%s'\n", alGetString(AL_VENDOR) );
  printf("Versió ALUT: %d.%d\n", alutGetMajorVersion(),
                                 alutGetMinorVersion() );
  printf("OpenAL Extensions supported are :\n%s\n",
         alGetString(AL_EXTENSIONS) );
  llista = alGetString(AL_EXTENSIONS);
  if (llista)     {
      printf("\nExtensions disponibles per separat:");
      printf("\n%2d - ", i);
      while (*llista)         {
          if (*llista != ' ' )    printf("%c", *llista);
           else                    printf("\n%2d - ", ++i);
        llista++;
        }
    }
  printf("\nOpenAL hardware present:\n %s\n",
          alcGetString(NULL, ALC_DEVICE_SPECIFIER) );
  pDeviceList = alcGetString(NULL, ALC_DEVICE_SPECIFIER);
  i=1;
  if (pDeviceList)     {
      printf("\nDrivers disponibles d'audio:\n");
      while (*pDeviceList)         {
          printf("%2d - %s\n", i++, pDeviceList);
          pDeviceList += strlen(pDeviceList) + 1;
        }
    }
 printf("Per defecte:\n%s\n",
         alcGetString(NULL, ALC_DEFAULT_DEVICE_SPECIFIER) );

  pDeviceList = alcGetString(NULL, ALC_CAPTURE_DEVICE_SPECIFIER);
  i=1;
  if (pDeviceList)     {
      printf("\nAvailable Capture Devices are:\n");
      while (*pDeviceList)         {
          printf("%2d - %s\n", i++, pDeviceList);
          pDeviceList += strlen(pDeviceList) + 1;
        }
    }
  printf("\n\n");
  // Ací podria sonar ja alguna cosa, si ho demanem ...
  alutExit();
  return EXIT_SUCCESS;
} // fi de main
```

Listado 1: Ejemplo de inicialización e identificación del hardware de audio con ALUT: openal_init_info_alut.c.

Como podrá comprobar, con ligeras diferencias en su propio equipo, el código empieza inicializando el componente de ALUT (línea 10), dándonos información de las versiones instaladas (líneas 11 a la 17) de ALUT y OpenAL; también las extensiones de OpenAL (estas son listadas separadas por espacios en la línea 16 y después en modo de lista ordenada, líneas 18 a a 27. El lector observará que hemos abreviado la salida por brevedad de la exposición y porque esta última información se obtiene no con funciones de ALUT, sino del componente AL. Lo he echo así para comparar con el siguiente apartado.

Las líneas de la 28 a la 38, consultan y muestran el dispositivo (a través del manejador asociado). En mi caso solo aparece OpenAL Soft, lo que quiere decir que estoy utilizando la implementación software que esta proporciona y que no tengo aceleración por hardware de mi equipamiento de sonido. Y, claro, las líneas 39 y 40 dicen que el dispositivo por defecto es ese único que ha encontrado.

Entre las líneas 42 a la 50 podemos ver que se pregunta por los dispositivos de captura de sonido (micrófonos) y responden cuatro opciones que corresponden, por orden numérico con: mi cámara web, el conector delantero de micrófono, el micro de los auriculares USB que estoy utilizando y el conector de la parte de atrás del equipo. Todas estas identificaciones las hace también el componente de AL, no ALUT.

Finalmente, la línea 53 cerrará la sesión de ALUT y es el momento para terminar el ejemplo.

4.2 API de ALURE para acceso al hardware de audio

Ya hemos visto lo que hace y cómo ALUT, veamos ahora cómo lo llevaríamos a cabo con ALURE [9]. De la documentación de ALURE[2] podemos ver que el módulo de *Main and Miscellanious* ofrece funciones para

- Gestión de errores con *alureGetErrorString*.
- Inicialización y terminación con *alureInitDevice*,alureGetDeviceNames, alureFreeDeviceNames y alureShutdownDevice.
- Identificación de versión con *alureGetVersion*.
- Pausas con *alureSleep*.
- Otras acciones: alureGetSampleFormat, alureInstallDecodeCallbacks y alureGetProcAddress. Estas permiten extender la funcionalidad de ALURE y no las vamos a ver aquí por brevedad en la exposición

El Listado 2 y el Listado 3 nos muestran un ejemplo de este grupo de operaciones que ofrece ALURE. Hemos recalcado cuándo aparecen para facilitar su identificación al lector. Le sugerimos que pruebe a ejecutar el código mostrado y, así, la experimentación le llevará a entender el desarrollo mostrado. Para compilarlo habrá de ejecutar la línea:

[2] La podemos encontrar en el sistema de archivos local si hemos instalado ALURE y también en <https://kcat.tomasu.net/alure-docs/files/alure-cpp.html>.

```
$ gcc openal_init_info_alure.c -o openal_init_info_alure $(pkg-config alure
openal --cflags –libs)


La versio de ALURE instalada es 1.2

OpenAL Renderer is 'OpenAL Soft'

OpenAL Version is '1.1 ALSOFT 1.19.1'

OpenAL Vendor is 'OpenAL Community'

OpenAL Extensions supported are :

AL_EXT_ALAW AL_EXT_BFORMAT AL_EXT_DOUBLE AL_EXT_EXPONENT_DISTANCE
AL_EXT_FLOAT32 AL_EXT_IMA4 AL_EXT_LINEAR_DISTANCE AL_EXT_MCFORMATS
AL_EXT_MULAW AL_EXT_MULAW_BFORMAT AL_EXT_MULAW_MCFORMATS AL_EXT_OFFSET
AL_EXT_source_distance_model AL_EXT_SOURCE_RADIUS AL_EXT_STEREO_ANGLES
AL_LOKI_quadriphonic AL_SOFT_block_alignment AL_SOFT_deferred_updates
AL_SOFT_direct_channels AL_SOFTX_events AL_SOFTX_filter_gain_ex
AL_SOFT_gain_clamp_ex AL_SOFT_loop_points AL_SOFTX_map_buffer
AL_SOFT_MSADPCM AL_SOFT_source_latency AL_SOFT_source_length
AL_SOFT_source_resampler AL_SOFT_source_spatialize


Dispositius? disponibles d'audio: 2
 0 - USB Audio Device Estéreo digital (IEC958)
 1 - Simultaneous output to USB Audio Device Estéreo digital (IEC958)
Inicialitzar dispositius
 0 - USB Audio Device Estéreo digital (IEC958)
Init : USB Audio Device Estéreo digital (IEC958)
 1 - Simultaneous output to USB Audio Device Estéreo digital (IEC958)
Init : Simultaneous output to USB Audio Device Estéreo digital (IEC958)
Inicialitzar dispositiu per defecte
Inicialitzat disp. per defecte
```

Empezando con el Listado 2, la línea 14 inicializa la sesión de ALURE (*alureInitDevice*) y nos muestra cómo comprobar si hay algún error (esto mismo se puede hacer tras cada llamada a OpenAL) con *alureGetErrorString*, línea 15 y 16.

Entre las líneas 20 a la 27 encontramos la identificación de las versiones instaladas de ALURE, OpenAL y las extensiones.

De las líneas 29 (*alureGetDeviceNames*) a la 37 preguntamos por los dispositivos disponibles. Entre las 40 y la 48 vemos cómo inicializar uno en concreto (con *alureInitDevice*) y cómo podemos tener varios inicializados al tiempo (varias tarjetas de audio se podrían utilizar de forma simultánea). En la línea 49 (*alureFreeDeviceNames*) todos son liberados.

En el Listado 3, empezamos por ver que pasar a trabajar con el dispositivo por defecto en el sistema en la línea 52. Y, entre las líneas 59 a la 63, cerramos la sesión de ALURE y podemos terminar la aplicación de forma “ordenada”.

```
1.  #include <stdlib.h>
2.  #include <stdio.h>
3.  #include <AL/alure.h>
4.  #include <string.h>
5.  #include <AL/al.h>
6.  #include <AL/alext.h>
7.
8.  int main(int argc, char **argv) {
9.    ALuint ALURE_major, ALURE_minor,elBuffer, laFont;
10.   int i, contaor;
11.   const ALCchar **pDeviceList, *llista;
12.   ALenum error;
13.
14.   if(!alureInitDevice(NULL, NULL))  {
15.         fprintf(stderr, "ALURE:: Failed to open OpenAL device: %s\n",
16.                     alureGetErrorString());
17.         return( -1);
18.   }
19.
20.   alureGetVersion(&ALURE_major, &ALURE_minor);
21.   printf ("La versio de ALURE instalada es %d.%d \n",
22.            ALURE_major, ALURE_minor);
23.   printf("OpenAL Renderer is '%s'\n", alGetString(AL_RENDERER) );
24.   printf("OpenAL Version is '%s'\n", alGetString(AL_VERSION) );
25.   printf("OpenAL Vendor is '%s'\n", alGetString(AL_VENDOR) );
26.   printf("OpenAL Extensions supported are :\n%s\n",
27.           alGetString(AL_EXTENSIONS) );
28.
29.  pDeviceList = alureGetDeviceNames( AL_TRUE, &contaor );
30.   if( contaor == 0 )
31.     printf(" ALURE:: error al buscar dispositius d'audio\n");
32.     return( -2 );
33.   }else{
34.     printf("\nLlistar dispositius disponibles d'audio: %d\n", contaor);
35.     for( i= 0; i < contaor; i++)        {
36.       printf("%2d - %s\n", i, pDeviceList[i]);
37.    }
38.   }
39.
40.   printf(Inicialitzar dispositius\n");
41.   for( i= 0; i < contaor; i++)       {
42.     printf("%2d - %s\n", i, pDeviceList[i]);
43.     if( alureInitDevice( pDeviceList[i], NULL ) == AL_FALSE )  {
44.         printf(" ALURE:: error al inicialitzar dispositiu %s.\n",
45.                 pDeviceList[i]);
46.         return( -3 );
47.     } else printf("Init : %s\n", pDeviceList[i]);
48.   }
49.   alureFreeDeviceNames( pDeviceList );
...
```

Listado 2: Ejemplo de inicialización e identificación del hardware de audio con ALURE: openal_init_info_alure.c (parte 1).

```
    ...
50.
51.   printf("Inicialitzar dispositiu per defecte\n");
52.   if( alureInitDevice( NULL, NULL ) == AL_FALSE )  {
53.    printf(" ALURE:: error al inicialitzar dispositiu per defecte.\n");
54.    return( -4 );
55.   } else printf("Inicialitzat disp. per defecte \n");
56.
57.   // Ací, podriem comprovar que sona alguna cosa ...
58.
59.   if( alureShutdownDevice() == AL_FALSE )    {
60.        printf(" ALURE:: error al lliberar el dispositiu.\n" );
61.        return( -5 );
62.   }
63.    alureFreeDeviceNames( pDeviceList );
64.
65.    return 0;
66. }
```

Listado 3: Ejemplo de inicialización e identificación del hardware de audio con ALURE: openal_init_info_alure.c (parte 2).

5 Conclusión y cierre

OpenAL está dividido en tres componentes. ALUT, la de más alto nivel, es vista como una parte de la que se podría prescindir o que, al menos, precisa ser actualizada. En este artículo hemos explorado su planteamiento y las operaciones que provee. Nuestro objetivo era analizar (con un desarrollo práctico) si es posible utilizar ALURE que viene proporcionado por OpenAL Soft para tal fin, ya que esta es la implementación que más probablemente tendrá instalada, de OpenAL, en su equipo .

La respuesta rápida es que sí se puede sustituir ALUT con ALURE, al menos a nivel de inicialización de los recursos hardware y software, que es lo que nos habíamos propuesto explorar en este artículo. Aunque con salvedades, observe que hemos remarcado las diferencias en resultados obtenidos en la ejecución de los dos ejemplos de código propuestos.

Espero que, a estas alturas, tenga ejecutándose el código propuesto y comprobando que puede oír el contenido básico de OpenAL con y sin ALUT ¿No es así? Pues descárguelo del repositorio creado a tal efecto en *GitHub*[3]. Y no cierre el documento sin haberlo comprobado antes.

¿Que no se escucha ningún sonido? Claro, no lo hemos pedido, pero lo haremos en otra ocasión, estimado lector. ¡Prometido!

[3] Disponible en la URL <https://github.com/magusti/OpenAL_examples/OpenAL_ALURE>.

6 Bibliografía

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

[2] Panne, S. (2006). "The OpenAL Utility Toolkit (ALUT)". Disponible en <http://distro.ibiblio.org/rootlinux/rootlinux-ports/more/freealut/freealut-1.1.0/doc/alut.html >.

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

[4] Peacock, D, Harrison, P., Hiebert. G . (2007). OpenAL Programmer's Guide. OpenAL Versions 1.0 and 1.1 Disponible en <https://www.openal.org/documentation/OpenAL_Programmers_Guide.pdf>.

[5] Peacock, D, Harrison, P., D'Orta, A., Carpentier, V., Cooper, E. (2006). Effects Extension Guide v 1.1. Disponible en <http://kcat.strangesoft.net/misc-downloads/Effects%20Extension%20Guide.pdf>.

[6] OpenAL Soft – Software 3D Audio. Disponible en <https://openal-soft.org/>.

[7] C. Robinson. ALURE Homepage. Disponible en < https://kcat.tomasu.net/alure.html>.

[8] OpenAL Soft – repositorio en Github. Disponible en <https://github.com/kcat/openal-soft>.

[9] ALURE – repositorio en Github. Disponible en <https://github.com/kcat/alure>.