UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Extensiones para OpenAL: efectos ambientales

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 la especificación de un estándar para “renderización” o recreación de audio tridimensional y las funciones que lo componen permiten el desarrollo de aplicaciones interactivas, en lo que respecta al tratamiento del audio. Su característica de independencia de fabricantes de *hardware* de audio, hace que los desarrollos sobre OpenAL sean llevados desde equipos de escritorio a dispositivos móviles.

Es habitual ver OpenAL en la creación de aplicaciones que utilicen el posicionamiento de audio en 3D: es decir donde la posición de las fuentes de sonido respecto al oyente, las características del medio de transmisión del sonido, así como la dirección y la velocidad relativa entre ellos, sea tenida en cuenta para calcular el sonido que llega al oyente. Pero suele quedar fuera del tratamiento de la documentación disponible el uso de otros efectos, no menos importantes, que ayudan a crear la idea de cómo es la escena donde se desarrolla la acción. Haciendo referencia al tamaño de la misma y a los materiales que la forman, modificando el sonido que llega hasta el oyente. Estos efectos se suelen denominar ambientales (*environmental effects*) y se refieren. p. ej., a la reverberación, las reflexiones y oclusiones del sonido que generan los objetos presentes en una escena real.

La extensión de OpenAL que se describe aquí permite añadir, en OpenAL, efectos ambientales y filtros, aunque por la brevedad en la exposición, me limitaré a los primeros.

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.

2 Objetivos

El presente documento está encaminado a ofrecer una perspectiva inicial de cómo averiguar si está disponible alguna de las extensiones de OpenAL y utilizar un efecto de la lista que corresponde a una instalación determinada.

A partir del estudio de los ejemplos que se abordan, el lector será capaz de:

- Explorar la extensión “Effects Extension” (abreviadamente EFX) que permite utilizar efectos ambientales realísticos 3D (o *aural worlds*).soportados por la especificación del motor de audio 3D OpenAL.
- Explorar el uso del efecto reverberación (*reverb)* de EFX.
- Compilar una aplicación sobre OpenAL y ejecutarla sobre la plataforma GNU/Linux. Esto no es ninguna limitación, puesto que el código de OpenAL es totalmente portable a otras plataformas.

3 Introducción

OpenAL está formado [1] por un núcleo básico de objetos (dispositivo, contexto, fuentes y oyente) que permiten el desarrollo [3] de escenas de audio 3D. Además, existen una serie de extensiones que han permitido ir haciendo crecer las opciones del motor básico en función del hardware y sistema operativo disponible. Por ejemplo, las extensiones [3] ALC_ENUMERATION_EXT (que permite listar el *hardware* de audio de salida instalado en una máquina) o

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

ALC_EXT_CAPTURE (que permite listar conjunto de dispositivos de entrada de audio que puedan estar presentes en un equipo dado).

Ambas son muy dependientes del sistema operativo sobre el que se trabaja. De hecho, algunas otras extensiones, solo están disponibles en alguna de las plataformas de trabajo, como EXT_vorbis (que ofrece el manejo de este formato en Linux), ALC_EXT_MAC_OSX (que hace referencia a opciones disponibles en macOS) o EAX2.0 (que Creative Labs. Implementó solo sobre Windows).

```
Available playback devices:
 Audio Interno Estéreo analógico
Available capture devices:
 Audio Interno Estéreo analógico
 Monitor of Audio Interno Estéreo analógico
 QuickCam Pro 9000 Mono analógico
Default device: OpenAL Soft
Default capture device: Audio Interno Estéreo analógico
ALC version: 1.1
ALC extensions:
    ALC_ENUMERATE_ALL_EXT, ALC_ENUMERATION_EXT, ALC_EXT_CAPTURE,
    ALC_EXT_DEDICATED, ALC_EXT_disconnect, ALC_EXT_EFX,
    ALC_EXT_thread_local_context, ALC_SOFTX_device_clock, ALC_SOFTX_HRTF,
    ALC_SOFT_loopback, ALC_SOFTX_midi_interface, ALC_SOFT_pause_device
OpenAL vendor string: OpenAL Community
OpenAL renderer string: OpenAL Soft
OpenAL version string: 1.1 ALSOFT 1.16.0
OpenAL extensions:
    AL_EXT_ALAW, 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_MCFORMATS, AL_EXT_OFFSET, AL_EXT_source_distance_model,
    AL_LOKI_quadriphonic, AL_SOFT_block_alignment, AL_SOFT_buffer_samples,
    AL_SOFT_buffer_sub_data, AL_SOFT_deferred_updates,
    AL_SOFT_direct_channels, AL_SOFT_loop_points, AL_SOFT_MSADPCM,
    AL_SOFT_source_latency, AL_SOFT_source_length
EFX version: 1.0
Max auxiliary sends: 4
Supported filters:
    Low-pass, High-pass, Band-pass
Supported effects:
    EAX Reverb, Reverb, Chorus, Distortion, Echo, Flanger, Ring Modulator,
Compressor, Equalizer
```

Listado 1: Salida de la ejecución de la orden info-openal.

En este trabajo me voy a centrar en la “Effects Extension” (EFX) que es la más genérica y ofrece a una aplicación el uso de efectos “ambientales”)como las reverberaciones o los ecos) y el uso de filtros (como el de “paso bajo”). Aunque, es importante insistir en que el número final de efectos y filtros disponibles depende de la versión de la implementación de OpenAL que se utilice. Para comprobarlo, el listado 1 muestra la salida de la aplicación *openal-info* que acompaña a OpenAL y que permite obtener un listado de las capacidades de la

implementación del estándar, en este caso por parte de *OpenAL Soft*[2], disponible en un equipo concreto. En la misma máquina, con la versión 1.14 de *OpenAL Soft*, solo están disponibles los valores remarcados con subrayado. Con la versión 1.16 de OpenAL Soft, como se observa, hay algunas opciones más.

3.1 Arquitectura de OpenAL

El esquema básico de trabajo del motor de OpenAL para "renderizar" una escenera sonora tridimensional se basa en la idea [4] que muestra la fig. 1. En ella se observa que:

- Existen unos objetos, los "buffer", que contienen los diferentes sonidos sin compresión y obtenidos desde ficheros o generados de manera sintética.
- Estos se asignan a los objetos de tipo "source". Estos objetos reciben como entrada el sonido y los parámetros que definen su situación, orientación y velocidad, así como también la atenuación, la direccionalidad de la fuente, etc. De esta forma se puede calcular cómo se van a comportar estas fuentes de sonido.
- Finalmente, todos los sonidos con mezclados en el "output mix", para proporcional al oyente todos los sonidos parametrizados con los parámetros globales de la escena como la velocidad de transmisión del sonido, si se tiene en cuenta el efecto *Doppler*, etc.

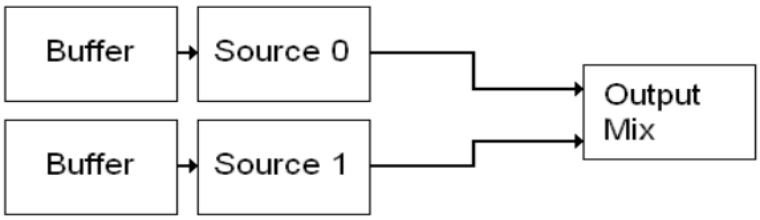


Figura 1: Arquitectura básica de OpenAL. Imagen extraída de [4].

Como se dice en [4], en esta expresión de la arquitectura interna de un motor de audio 3D no hay cabida para operaciones de procesado de audio (*Digital Signal Processing* o DSP) necesarias para realizar simulaciones interactivas de audio que se utilizan en las aplicaciones actuales. Estos cálculos incluyen los correspondientes a:

- Los efectos de absorción y reflejado de las ondas de sonido por parte de los objetos en la escena, como oclusiones y obstrucciones. Estos efectos vienen dados por las características de los materiales y por la geometría de los cuerpos físicos que se quieren simular en la escena. Obviamente, en lo que respeta a cómo influyen o modifican el sonido que llega a ellos, si los atraviesa, si lo modifican y si lo hacen en términos de intensidad o de frecuencia del sonido.
  Estas acciones requieren de operaciones de filtrado ambiental (denominado *filtering*) para cada fuente por separado.
- Los efectos ambientales "auxiliares" (denominados *Auxiliary effect sends*), como la reverberación, el eco, los coros, etc. Algunos también dependen de la geometría de la escena, otros no. Pero todos ellos, a

2 El código está disponible en el GitHub de este proyecto en <https://github.com/kcat/openal-soft/blob/master/utils/openal-info.c>. Y se puede compilar con:

```
$ gcc info-openal.c -lalut -lopenal -o info-openal
```

diferencia del grupo anterior de efectos, se aplican a todo un conjunto de las fuentes de sonido que existen en la escena.

La extensión de efectos (Effects Extension) se diseñó [4] con el propósito de dar solución a estos efectos. El esquema de funcionamiento ampliado se muestra gráficamente en la fig. 2. Con lo que la arquitectura del motor ahora incluye una expansión interna que, de forma independiente a plataforma, ofrece capacidades de efectos DSP o de procesado de la señal de audio.

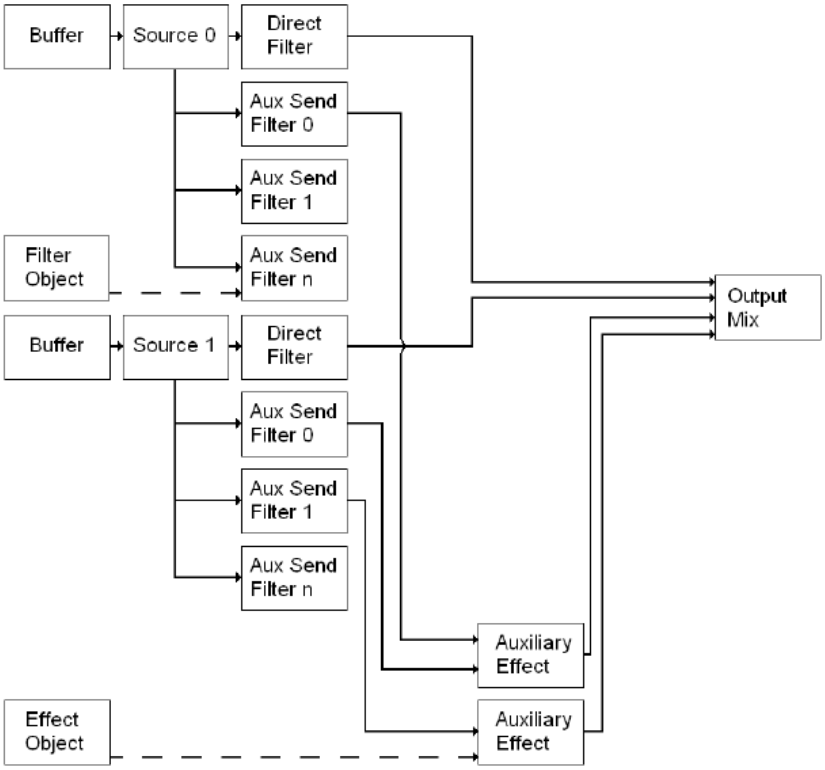


Figura 2: Arquitectura ampliada de OpenAL para dar soporte al uso de efectos y filtros. Imagen extraída de [4].

Para ello, se han creado nuevos objetos que se han introducido en la ruta del audio desde las fuentes hasta el mezclador final y que son:

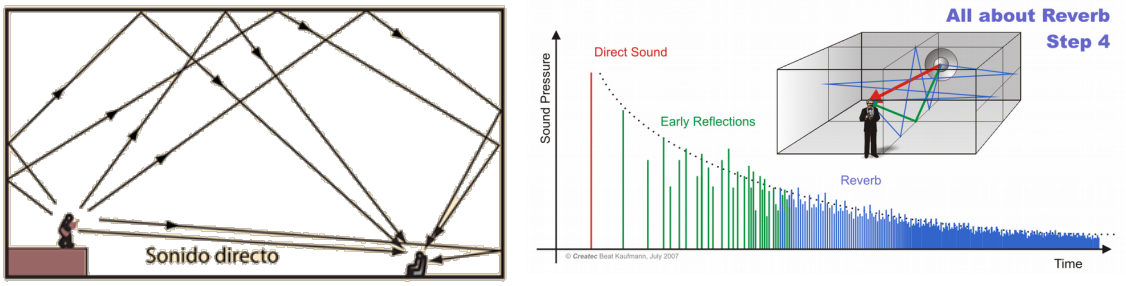
- Los filtros, que son efectos aplicados individualmente a una fuente y que están compuestos por dos partes.
  - Por un lado, cada fuente puede tener un número máximo de "salidas filtradas" (denominadas "Auxiliary Send") que dependen de la implementación de OpenAL que se utilice. En el caso observado en el listado 1 se dice "`Max auxiliary sends: 4`". Lo que se corresponde con lo que muestra la fig. 2, en tanto que se observa que de, cada "source", salen hasta cuatro flechas. La primera es "Direct filter", lo que supone que no se aplica filtrado alguno.
  - Por otro lado, el resto (tres) de flechas, toman la salida de la fuente y la filtran de acuerdo a la definición de parámetros de un objeto "Filter Object" que se asigna a una de los bloques "Auxiliary Send".
- Los efectos ambientales se definen con otra pareja de objetos y se aplican a un grupo de fuentes a la vez. Están compuestos por
  - Los objetos "Auxiliary Effect", que son como un mezclador de todas las entradas que recibe y a las que aplica un efecto.

- Los objetos (“Effect”) que son los que define el tipo y parametriza el efecto. Lo cual define el comportamiento de los “Auxiliary Effect”.

Por brevedad en la exposición, vamos a ver cómo utilizar estos efectos ambientales en ejemplos prácticos. Desde el punto de vista de la EFX, los objetos son el resultado de la selección de un tipo de efecto y los parámetros que lo definen.

4 Ejemplo de efecto: reverberación

Voy a comentar el ejemplo de reverberación (*reverb*) y plantear otros dos como son el el coro (chorus) y eco (*echo*). Todos estos efectos tienen[3] una definición física, unas implicaciones en el campo de la acústica arquitectónica y otras en el campo de la postproducción de audio por parte de los estudios de grabación.



a) *b)*

Figura 3: Definición de reverberación: a) sonido directo vs reflexiones (de <http://hyperphysics.phy-astr.gsu.edu/hbasees/Acoustic/reverb.html>) y b) descomposición de la reverberación en componentes (de <https://cursodesonido.webnode.com.co/curso/teoria-del-sonido/modulo-05-acustica-ii-reverberacion-eco-/>).

La reverberación es un efecto sonoro debido a la reflexión del sonido en los objectos que hay en la escena, fig. 3a . Al oyente un sonido le llega, fig. 3b, en forma de:

- Sonido directo (*direct sound*), que se refiere al que llega en primera instancia al oyente, sin chocar con ninguna superficie del recinto, proporcionando información acerca de la ubicación, tamaño y color (este se refiere a las componentes de frecuencia del sonido) de la fuente.
- Primeros rebotes (*early reflections*) o reflexiones tempranas, que se producen en una superficie y llegan al oyente con retardos de tiempo de 10 a 30 ms. Proporciona información acerca del tamaño del recinto acústico donde ocurre y “dan calidez y cuerpo” al sonido. Quiere decir que estamos tan acostumbrados a escucharlo que, si no está presente, puede ser apreciado como un sonido muy artificial o “poco natural”.
- Últimos rebotes (*late reflections*) o reflexiones secundarias, debidas al rebote en varias superficies hasta llegar al oyente, con retardos de tiempo mayores que 50 ms. A partir de 100 ms ya se habla de eco, en lugar de reverberación. Informan de la reflexión y absorción de los materiales y de la cercanía o lejanía del oyente a la fuente sonora.

---

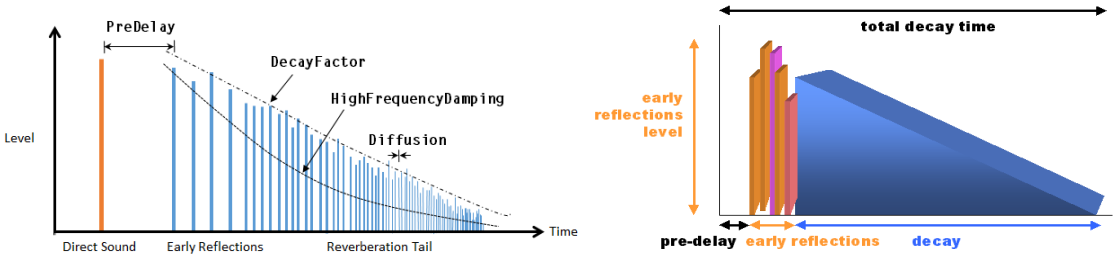
3 Por ejemplo, para la reverberación, la definición del fenómeno sonoro se puede ver en <https://en.wikipedia.org/wiki/Reverberation> y las implicaciones arquitectónicas y de postproducción, respectivamente, en <http://www.acusticaintegral.com/reverberacion.htm> y <https://www.audioproduccion.com/basicos-de-reverberacion/>.

Por ello, en OpenAL, la EFX, define [4] la reverberación a partir de un identificador de efecto (AL_EFFECT_REVERB) y un conjunto de parámetros que se muestran en la fig. 4, que refleja las unidades, los rangos y los valores por defecto de cada parámetro de este efecto.

AL_EFFECT_REVERB

Version	Parameter Name	Units	Range	Default
1.0	AL_REVERB_DENSITY		[0.0, 1.0]	1.0
1.0	AL_REVERB_DIFFUSION		[0.0, 1.0]	1.0
1.0	AL_REVERB_GAIN		[0.0, 1.0]	0.32
1.0	AL_REVERB_GAINHF		[0.0, 1.0]	0.89
1.0	AL_REVERB_DECAY_TIME	Seconds	[0.1, 20]	1.49
1.0	AL_REVERB_DECAY_HFRATIO		[0.1, 2.0]	0.83
1.0	AL_REVERB_REFLECTIONS_GAIN		[0.0, 3.16]	0.05
1.0	AL_REVERB_REFLECTIONS_DELAY	Seconds	[0.0, 0.3]	0.007
1.0	AL_REVERB_LATE_REVERB_GAIN		[0.0, 10.0]	1.26
1.0	AL_REVERB_LATE_REVERB_DELAY	Seconds	[0.0, 0.1]	0.011
1.0	AL_REVERB_AIR_ABSORPTION_GAINHF		[0.892,1.0]	0.994
1.0	AL_REVERB_ROOM_ROLLOFF_FACTOR		[0.0. 10.0]	0.0
1.0	AL_REVERB_DECAY_HFLIMIT	ON / OFF	[AL_FALSE, AL_TRUE]	AL_TRUE

Figura 4: Definición del efecto reverberación (AL_EFFECT_REVERB) en OpenAL. Datos extraídos de [4].



a) *b)*

Figura 5: Componentes de la curva de reverberación: versión (a) detallada y (b) simplificada. Imágenes extraídas respectivamente de<https://es.mathworks.com/help/audio/ref/reverberator-system-object.html> y de <http://www.harmonycentral.com/articles/understanding-digital-reverb-parameters>.

En lugar de repetir las definiciones de cada uno de los parámetros (fig. 5a), insistiré en la idea de "presets" (o conjunto de valores predefinidos) que muchas aplicaciones ofrecen en forma de un nombre "coloquial" que identifica al recinto (en este caso) que se describe con esos valores, como por ejemplo[4], de menor a mayor tamaño: *Room*, *Chamber* y *Hall*. Estos, se pueden ver en la fig. 5b y se controlan básicamente con:

4 Jon, H. (2017). "Los 5 Principales Tipos de Reverberación". Disponible en: <https://www.audioproduccion.com/los-5-principales-tipos-reverberacion/>. También identifica la producida artificialmente por los "dispositivos": *plate* y *spring*.

- “Decay Time”. Es es periodo de tiempo en que se dan las reflexiones (todas ellas, las tempranas y las secundarias). Toma valores típicos desde 0.1 segundos para una habitación pequeña y de superficies poco reflectantes (*dead*), a valores de 20.0 s. que corresponden a grandes espacios con materiales muy reflectantes (*live surfaces*).
- “Reflection Delay”. Hace referencia al tiempo que pasa entre las reflexiones tempranas, esto es, de lo cercanas que están en el tiempo entre sí. Con ello también define la cantidad de reflexiones tempranas respecto a las secundarias, puesto que ambas se reparten el tiempo total de “decay”. Cuanto más “pegadas” están simulan un espacio más pequeño (cerrado)

Solo insistir en que hay parámetros para establecer las propiedades globales (como *AL_REVERB_GAIN* y *AL_REVERB_REFLECTIONS_DELAY*) y la parte de estas que corresponde a las reflexiones secundarias (*AL_REVERB_LATE_REVERB_GAIN* y *AL_REVERB_LATE_REVERB_DELAY*), de este modo quedan implícitamente definidas las tempranas.

5 Reverberación: caso práctico de uso de EFX en OpenAL

Ya está bien de teoría y conceptos, ¿no? Voy a mostrarte un ejemplo de código. He dejando algunos detalles fuera por no alargar la exposición, pero si estás interesado, me puedes pedir el proyecto completo por correo electrónico.

¿Cómo se pone en un ejemplo todo esto? Desde la guía del programador [3] y la de extensiones [4] tenemos una buena introducción a la secuencia de pasos a dar. Pero en ellas falta un ejemplo que genere sonido, aplique el efecto y lo modifique mientras suena; por eso he hecho esta versión.

La secuencia propuesta por la documentación citada de OpenAL:es la que utilizo de base para completar posibilitar la “renderización” de la frase “Hello, World!” tal cual la sintetiza OpenAL y, a continuación, con el efecto de reverberación.

El primer paso corresponde a la inicialización de OpenAL y la EFX. Se utilizará la función `alutInit` y se habrá de comprobar si está disponible la extensión deseada con `alIsExtensionPresent`. Si todo va bien se procederá a crear el máximo un número de salidas de las fuentes (*Auxiliary Sends*) para conectarlas a los contenedores de efectos (*Auxiliary Effect*) con `alcCreateContext`. Y ya se puede cargar el audio, se crearán los `buffers` y se llenarán con sonido con la función `alutCreateBufferHelloWorld` y, por si quieres probar a utilizar un fichero de disco, he dejado una instrucción `alutCreateBufferFromFile`.

```
int main (int argc, char **argv) {
  ALuint buffers[6], fuente, fuenteConEfecto;

  //----------------- 1: Inicializa OpenAL y EFX -----------------
  alutInit( &argc, argv );
  pDevice =  alcGetContextsDevice(alcGetCurrentContext());
  if (!pDevice)     {
      printf(" pDevice no topetat!\n");       return 1;     }

  if(alcIsExtensionPresent(pDevice, "ALC_EXT_EFX"))    {
      printf( "Podem gastar les extensions EFX!\n" );    }
  else     {
      fprintf(stderr, "Error: EFX not supported\n");
      alcCloseDevice( pDevice );
```

```
      alutExit();
      return (1);    }

  // Request 4 Auxiliary Sends per Source
  attribsEfectos[0] = ALC_MAX_AUXILIARY_SENDS;
  attribsEfectos[1] = 4;
  pContextEfectos = alcCreateContext(pDevice, attribsEfectos);
  if (!pContextEfectos)
    {
      printf(" pContextEfectos no creat!\n");
      return 2;
    }
  alcMakeContextCurrent(pContextEfectos);   // Activate the context

  // The actual number of Aux Sends available on each Source?
alcGetIntegerv(pDevice, ALC_MAX_AUXILIARY_SENDS, 1,
                &iSendsEfectos);
 printf("Device supports %d Aux Sends per Source\n",
          iSendsEfectos);

  if (argc > 1)
    buffers[0] = alutCreateBufferFromFile( argv[1] );
  else
    buffers[0] = alutCreateBufferHelloWorld();
```

El segundo paso es la creación de los contenedores de efectos o *Auxiliary Effect Slots* con alGenAuxiliaryEffectSlots. Tras esto, será el momento para crear el bloque de efectos (*Effect Object*) con alGenEffects y definir sus valores con alEffecti. Esto implica escoger el tipo de efecto AL_EFFECT_TYPE y sus parámetros, para lo cual hay que indicar a cual pertenece cada valor como, por ejemplo, AL_REVERB_DECAY_TIME. En principio solo sería necesario un efecto, pero he querido mantener la nomenclatura del ejemplo de la documentación para que sea fácil emparejar los ejemplos que están en [4] y el aquí presentado.

```
//
  alGenSources (1, &fuente);
  alGenSources(1, &fuenteConEfecto);
//-----------------2-Crear SLOTS y EFECTS ----------------------
  alGetError();
  for (uiLoop = 0; uiLoop < 4; uiLoop++)    {
      alGenAuxiliaryEffectSlots(1, &uiEffectSlot[uiLoop]);
      if (alGetError() != AL_NO_ERROR)  break;
    }
  printf("Generated %d Aux Effect Slots\n", uiLoop);
  for (uiLoop = 0; uiLoop < 2; uiLoop++)    {
      alGenEffects(1, &uiEffect[uiLoop]);
      if (alGetError() != AL_NO_ERROR)
    break;
    }
```

```
  printf("Generated %d Effects\n", uiLoop);
  alGetError();
  if (alIsEffect(uiEffect[0]))    {
      alEffecti(uiEffect[0], AL_EFFECT_TYPE, AL_EFFECT_REVERB);
      if (alGetError() != AL_NO_ERROR)
        printf("Reverb Effect not supported\n");
      else{
       alEffectf(uiEffect[0], AL_REVERB_DECAY_TIME, 15.0f);
      alEffectf(uiEffect[0], AL_REVERB_ROOM_ROLLOFF_FACTOR, 10.0f);
       alEffectf(uiEffect[0], AL_REVERB_REFLECTIONS_DELAY, 10.3f);
       alEffectf(uiEffect[0], AL_REVERB_REFLECTIONS_GAIN, 3.0f);
      }
    }
```

El tercer paso corresponde a la asignación de efectos a los contenedores, . Para ello, hay que asignar, con alAuxiliaryEffectSloti, al contenedor (*Slot*) cuyo identificador es conocido, el valor correspondiente a la propiedad AL_EFFECTSLOT_EFFECT. Este valor es el identificador del efecto.

```
  //----------------3-Asignar EFECTOS a SLOTS---------------
alAuxiliaryEffectSloti(uiEffectSlot[0],                //reverb
                       AL_EFFECTSLOT_EFFECT, uiEffect[0]);
  if (alGetError() == AL_NO_ERROR)
    printf("Successfully loaded effect into effect slot\n");
```

El cuarto paso es la conexión de las salidas de las fuentes con un bloque contenedor de efectos con alSource3i y la propiedad AL_AUXILIARY_SEND_FILTER.

```
  //------4-Configurar las salidas  Auxiliary Sends de la fuente –
  alSource3i(fuenteConEfecto, AL_AUXILIARY_SEND_FILTER,
        uiEffectSlot[0], // id del slot de efecto auxiliar Reverb
             0, AL_FILTER_NULL);
```

El resto de la aplicación ya es la que asigna el sonido a la fuente y la que pide a la fuente que suene, Ahí no hay instrucciones de EFX, así que no remarco ninguna.

```
  alSourcei( fuente, AL_BUFFER, buffers[0]);
  alSourcei( fuente, AL_LOOPING, AL_FALSE);
  alSourcei( fuenteConEfecto, AL_BUFFER, buffers[0]);
  alSourcei( fuenteConEfecto, AL_LOOPING, AL_FALSE);

  alSourcePlay (fuente);  alSourceStop( fuente); alutSleep (2);
  alSourcePlay (fuenteConEfecto); alSourceStop( fuenteConEfecto );
  alutSleep (2);

  printf("\nLiberando recursos para terminar ... \n");
  alDeleteSources(1, &fuente);   alDeleteBuffers(6, buffers);
  alDeleteAuxiliaryEffectSlots(4, uiEffectSlot);
  alDeleteEffects(2, uiEffect);  alDeleteFilters(1, uiFilter);
  alcDestroyContext( pContextEfectos );
```

```
  alcCloseDevice( pDevice );
  alutExit();
  return EXIT_SUCCESS;
} // Fin de la función “main”
```

¿Quieres probarlo? ¿Cómo se ejecuta? Puedes compilar el fichero (en mi caso se llama `helloWorld_extensions.c`), desde el terminal con

```
$ gcc helloWorld_extensions.c getch.c -o helloWorld_extensions -lalut `pkg-config openal --cflags --libs`
```

Lo ejecutas … Y el resto … ¡ya es cosa de escucharlo!

```
$ helloWorld_extensions
Podem gastar les extensions EFX!
Device supports 4 Aux Sends per Source
Generated 4 Aux Effect Slots
Generated 2 Effects
Successfully loaded effect into effect slot
```

6 Cierre y conclusiones

En el presente documento se ha expuesto el uso de las extensiones de OpenAL y, en concreto, el de la denominada “Effects Extension” o, abreviadamente, EFX. Se ha definido el efecto *reverb* y se ha el explorado su parametrización en EFX. Tras ello, se ha procedido a explicar una realización de código para que el lector pueda explorar por su propia cuenta. El lector que ha participado en la explicación, podrá compilar una aplicación sobre OpenAL y ejecutarla sobre la plataforma GNU/Linux. Aunque el código de OpenAL es totalmente portable a otras.

Para comprobar que realmente has aprendido qué es la reverberación, según OpenAL, es el momento de que te pongas manos a la obra e intentes elaborar tu propia versión. Ya verás que enriquecedor te resulta. ¡¡ÁNIMO!!

7 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] 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>.

[4] 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>