UNIVERSIDAD POLITECNICA DE VALENCIA

# Introducción al empleo de técnicas de audio posicional mediante OpenAL


| | |
|---|---|
| **Apellidos, nombre** | Agustí Melchor, Manuel[1] (magusti@disca.upv.es) |
| **Departamento** | [1]Dpto. De Ing. De Sistemas y Computadores |
| **Centro** | Universidad Politécnica de Valencia |

UNIVERSIDAD POLITECNICA DE VALENCIA

# 1 Resumen

En este artículo vamos a presentar una introducción al tema de procesado de audio, en concreto a la temática de audio posicional, también llamado: envolvente, espacial o 3D. Para ello se va a utilizar el API de OpenAL (véase [1], [2] y [3]) como vehículo para realizar ejemplos prácticos.

Se va a presentar un enfoque que permita trabajar en cualquier sistema operativo desarrollando sistemas de características multimedia que hagan uso de operaciones sobre audio, en este caso, como un medio característico del que extraer y sobre el que mostrar información.

# 2 Introducción

El presente desarrollo está encaminado a ver posibilidades de audio posicional apoyado en una interfaz visual que permita representar la posición del oyente y de las fuentes de sonido en una escena tridimensional: OpenGL ofrecerá el soporte visual para lo que es nuestro interés que son las funciones del API de OpenAL que permiten añadir al sistema la funcionalidad de audio en tres dimensiones.

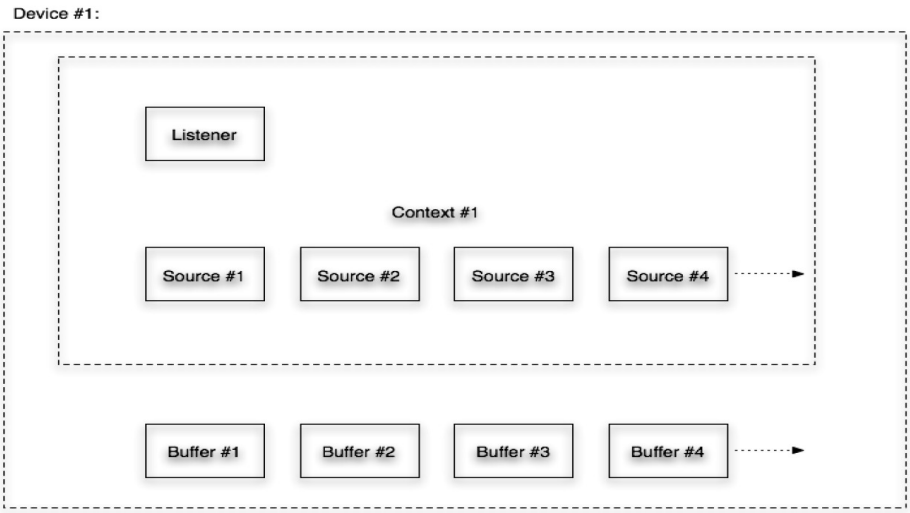


*Figura 1: Un ejemplo de jerarquía entre objetos de OpenAL [2]*

La funcionalidad de OpenAL se estructura [2] en base a los 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.

# 3 Objetivos

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

- Manipular sonido espacial en un computador, en términos de OpenAL, dando a conocer al alumno, de un modo participativo, el uso de OpenAL para llevar a cabo operaciones de audio posicional.
- Generar un ejecutable que utilice las bibliotecas de programación de OpenAL y las de OpenGL, sin instalar ningún entorno pesado, desde la línea de órdenes.
- Presentar un ejemplo que integra OpenAL para el audio con OpenGL para los gráficos, así se reafirma visualmente lo que se oye. Para la mayoría de las personas es así más claro: ver la escena y oírla. Ambos son ejemplos de estándares abiertos y multiplataforma que permita al lector proseguir su autoaprendizaje de forma independiente.

Atenderemos sólo a las cuestiones que hacen posible el sonido, prescindiendo de los detalles del entorno gráfico, aunque propondremos cómo llevarlo a cabo.

# 4 Desarrollo

El motor de OpenAL realiza todos los cálculos necesarios como la atenuación debida a la distancia, el efecto Doppler, etc. Nuestra tarea es modificar las propiedades de los objetos para recrear el resultado de la interacción del usuario, la evolución temporal de un sistema cerrado, o la mezcla de ambos.

## 4.1 Posicionamiento espacial del audio

Se pueden generar una señal de sonido (una fuente) y hacela sonar para el único oyente que puede tener una escena de audio en OpenAL. Por defecto, ambos estarán situados en una misma posición: el punto origen de un sistema de referencia de tres ejes.

El sonido sonará igual en todos los canales de salida que tengamos disponibles y, perceptualmente, parecerá provenir del punto central a todos ellos. Utilizando OpenAL podemos “mover” la posición del oyente o de las fuentes, o hacia dónde está “mirando” el oyente o las fuentes y el motor de OpenAL hará las operaciones oportunas para simular lo que se oiría en esas condiciones.

Para hacer una demostración haremos que una fuente primero y el oyente después, se muevan a lo largo de uno de los ejes que define su posición en un espacio tridimensional. De esta forma se apreciará el trabajo (efecto) de OpenAL incluso con unos auriculares o un sistema con dos altavoces ligeramente separados.

El código del listado 1 muestra un programa principal que espera recibir como parámetro un fichero WAV. Utiliza un fichero con un sólo canal (monofónico) puesto que es cuestión de oír cómo OpenAL lo hace “moverse”. Lo compilaremos con la línea de órdenes:

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

y lo ejecutaremos con la línea de órdenes:

```
$ ./ nombreFichero
```

```
#include <stdio.h>
#include <stdlib.h>
#include <AL/alut.h>
#include <unistd.h>
#include <math.h>

void fuente_moviendosePorLosEjes_oyente( ALuint fuente );
void oyente_moviendosePorLosEjes_fuente( ALuint fuente );

int main (int argc, char **argv)
{
 const ALchar *extensions;
 ALuint buffer, fuente;

 if (argc < 2)
   {
    printf("Utilitzacio: %s fitxer.wav \n", argv[0] );
    return -1;
   }
 else
   {
    alutInit (&argc, argv);
    extensions = alGetString(AL_VERSION);
    printf("OpenAL Version is '%s' \n",extensions);
    printf("ALUT version: %d.%d \n", alutGetMajorVersion (),  alutGetMinorVersion () );
    printf("Carregant %s\n", argv[1] );
    buffer = alutCreateBufferFromFile( argv[1] );

    alGenSources (1, &fuente);
    alSourcei(fuente, AL_BUFFER, buffer);
    alSourcei(fuente, AL_LOOPING, 1 );

    fuente_moviendosePorLosEjes_oyente( fuente );
    oyente_moviendosePorLosEjes_fuente( fuente );

    alDeleteSources( 1, &fuente );
    alDeleteBuffers( 1, &buffer );

    alutExit ();
    return EXIT_SUCCESS;
   }// if (argc < 2)

} // fi de main

#define PI 3.1415926

void oyente_moviendosePorLosEjes_fuente( ALuint fuente )
{
 int nSegons, i;

 printf("Oyente moviendosePorLosEjes de la fuente \n");
 alSourcePlay (fuente);
 nSegons = 30;

 for(i=0; i < nSegons; i+=2) // Conforme està orientat: esquerra
   {
    printf("x=%3d\r", i); fflush( stdout );
    alListener3f(AL_POSITION, i*1.0, 0.0, 0.0);
```

```
    usleep( 1000000 );  printf("      \r"); fflush( stdout );
   }

  for(i=-nSegons; i < 0; i+=2)// Venint de la dreta, conforme està orientat
   {
    printf("x=%3d\r", i); fflush( stdout );
    alListener3f(AL_POSITION, i*1.0, 0.0, 0.0);
    usleep( 1000000 ); printf("      \r"); fflush( stdout );
   }
  printf("\n");
}// fi de oyente_moviendose...


void fuente_moviendosePorLosEjes_oyente( ALuint fuente )
{
  int nSegons, i;

  printf("fuente_moviendosePorLosEjes_oyente\n");
  alSourcePlay (fuente);
  nSegons = 30;

  for(i=0; i < nSegons; i+=2) // Conforme està orientat: esquerra
   {
    printf("x=%3d\r", i); fflush( stdout );
    alSource3f(fuente, AL_POSITION, i*1.0, 0.0, 0.0);
    usleep( 1000000 ); printf("      \r"); fflush( stdout );
   }

  for(i=-nSegons; i < 0; i+=2)// Venint de la dreta, conforme està orientat
   {
    printf("x=%3d\r", i); fflush( stdout );
    alSource3f( fuente, AL_POSITION, i*1.0, 0.0, 0.0);
    usleep( 1000000 ); printf("      \r"); fflush( stdout );
   }
  printf("\n");
}
```

*Listado 1. Generación de señales de audio básicas con OpenAL.*

Ambas dos funciones se llevan el sonido hacia el canal izquierdo en primer lugar, para acercarse después por el otro canal. El significado absoluto de cada eje depende de las posiciones y orientaciones de partida del oyente y de la fuente.

Para detallar sus posibles valores y cómo utilizarlos es necesario un “entorno”, una escena, más elaborada. En el siguiente apartado se propone utilizar una aplicación existente y experimentar sobre el código fuente de esta para oír y ver la escena.

## 4.2 Discusión del ejemplo de OpenAL en PIGE

Para ponerlo en un ambiente visualmente tridimensional (3D) recurriremos a la plataforma que nos proporciona Chad Armstrong dentro del proyecto PIGE (*Platform Independent Game Engine*) [5] y que se puede ver en la figura 2. En el

apartado de tutoriales de este proyecto se encuentra uno sobre OpenAL[1] que tiene este interfaz. Descárgate el código, para compilarlo se utilizará una línea de órdenes como:

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

Se puede ejecutar cualesquiera tres ficheros WAV, por ejemplo con la orden:

```
$ openal /usr/share/sounds/KDE_Beep_Bottles.wav
/usr/share/sounds/KDE_Beep_Beep.wav /usr/share/sounds/KDE_Beep_ Ahem.wav
```

Durante la ejecución del mismo (véase fig. 2) la fuente de sonido 1 se activa pulsando la tecla '1' sobre la ventana activa del programa y se desactiva con '4'. Igualmente sucede con la fuente 2 ('2' y '5') y la 3 ('3' y '6'). La posición del cubo que muestra la posición del oyente se controla con las teclas del cursor.

¿Serías capaz de 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? Ejecuta el programa y prueba.

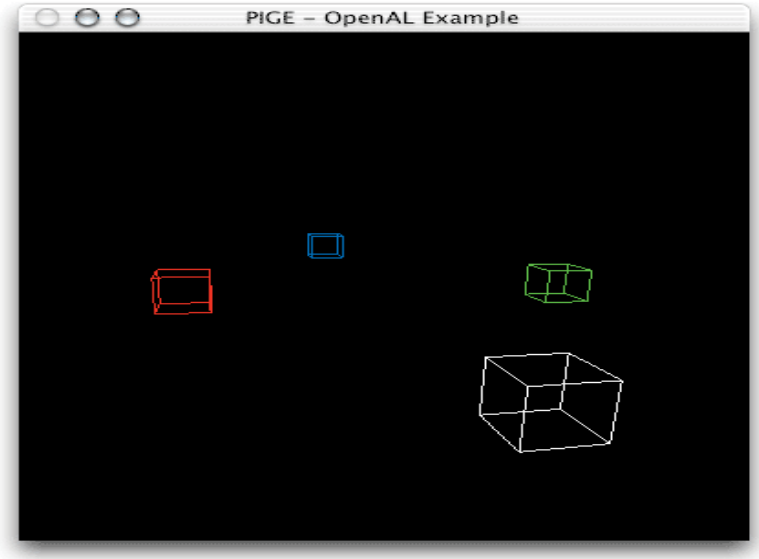


*Figura 2: Un momento de la ejecución de PIGE*

El código original, necesita pequeños retoques para compilar con la versión actual de OpenAL, seguro que lo verás en cuanto veas los errores de compilación. Lo veremos en este artículo. De este ejemplo, dejando de lado todo lo que es puramente gráfico, destacaremos las funciones *main*, *init*, *keyboard* y *specialKeys*.

El programa principal (*main*) está resumido en el listado 2, donde apenas se inicializa y cierra el sistema de audio con las llamadas de nivel ALUT de OpenAL. El resto de las operaciones relativas al audio son la carga de los ficheros de audio a utilizar y las respuestas a la interacción con el usuario. Estás últimas están dentro de las funciones que ofrece OpenGL para acceso a los eventos de teclado y ratón.

```
...
  //initialise glut
```

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

```
  glutInit(&argc, argv) ;
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH) ;
  glutInitWindowSize(400,400) ;

  //initialise openAL
  alutInit(&argc, argv);

  GLwin = glutCreateWindow("PIGE - OpenAL Example") ;
  init(argv[1], argv[2], argv[3]) ;
  glutDisplayFunc(display) ;
  glutKeyboardFunc(keyboard) ;
  glutSpecialFunc(specialKeys);
  glutReshapeFunc(reshape) ;

  glutMainLoop() ;

  alutExit();
...
```

*Listado 2. Función "main" de PIGE.*

El autor centraliza en la función "init" todo lo que respecta a la carga de sonidos, creación de fuentes, posicionamiento de los objetos, ... que tiene que ver con OpenAL. Yo he comentado la instrucción repetida de *alutInit* y he cambiado las funciones de carga de ficheros por las actuales *alutCreateBufferFromFile*.

```
void init( char *s1, char *s2, char *s3) //void)
{
 //alutInit(0, NULL);

 alListenerfv(AL_POSITION,listenerPos);
 alListenerfv(AL_VELOCITY,listenerVel);
 alListenerfv(AL_ORIENTATION,listenerOri);

 alGetError(); // clear any error messages
 if(alGetError() != AL_NO_ERROR)
   {
    printf("- Error creating buffers !!\n");
    exit(1);
   }
 else
   {
    printf("init() - No errors yet.");
   }

 // Generate buffers, or else no sound will happen!
 alGenBuffers(NUM_BUFFERS, buffer);

 printf( "Assignant buffer 1 a %s\n", s1 );
 buffer[0] = alutCreateBufferFromFile( s1 );
 printf( "Assignant buffer 2 a %s\n", s2 );
 buffer[1] = alutCreateBufferFromFile( s2 );
 printf( "Assignant buffer 3 a %s\n", s3 );
 buffer[2] = alutCreateBufferFromFile( s3 );
```

```
  alGetError(); /* clear error */
  alGenSources(NUM_SOURCES, source);

  if(alGetError() != AL_NO_ERROR)
    {
     printf("- Error creating sources !!\n");
     exit(2);
    }
  else
    {
     printf("init - no errors after alGenSources\n");
    }

  alSourcef(source[0],AL_PITCH,1.0f);
  alSourcef(source[0],AL_GAIN,1.0f);
  alSourcefv(source[0],AL_POSITION,source0Pos);
  alSourcefv(source[0],AL_VELOCITY,source0Vel);
  alSourcei(source[0],AL_BUFFER,buffer[0]);
  alSourcei(source[0],AL_LOOPING,AL_TRUE);

  alSourcef(source[1],AL_PITCH,1.0f);
  alSourcef(source[1],AL_GAIN,1.0f);
  alSourcefv(source[1],AL_POSITION,source1Pos);
  alSourcefv(source[1],AL_VELOCITY,source1Vel);
  alSourcei(source[1],AL_BUFFER,buffer[1]);
  alSourcei(source[1],AL_LOOPING,AL_TRUE);

  alSourcef(source[2],AL_PITCH,1.0f);
  alSourcef(source[2],AL_GAIN,1.0f);
  alSourcefv(source[2],AL_POSITION,source2Pos);
  alSourcefv(source[2],AL_VELOCITY,source2Vel);
  alSourcei(source[2],AL_BUFFER,buffer[2]);
  alSourcei(source[2],AL_LOOPING,AL_TRUE);
}
```

*Listado 3. Función "init" de PIGE.*

Inicializado todo, sólo hay que dejar que el programa evolucione actualizando valores de las propiedades de las fuentes para que se oiga lo que se ve en la ventana. Básicamente, lo que permite el programa es poner en marcha las tres fuentes y paralas por separado. Véase como en el listado 4, están las funciones alSourcePlay y alSourceStop a diferentes pulsaciones del teclado: del '1' al '6'.

```
void keyboard(unsigned char key, int x, int y)
{
  switch(key)
    {
    case 'h': case 'H': // Menú: opciones
      printf(" 1 on 1, 4 off 1\n 2 on 2, 5 off 2\n 3 on 3, 6 off 3\n La posició del oient es canvia en
les tecles del cursor o en 'asqz'\n");
      break;

    case '1':
      alSourcePlay(source[0]); break;
    case '2':
```

```
    alSourcePlay(source[1]); break;
   case '3':
    alSourcePlay(source[2]); break;

   case '4':
    alSourceStop(source[0]); break;
   case '5':
    alSourceStop(source[1]); break;
   case '6':
    alSourceStop(source[2]); break;

   case 'a': case 'A':
    listenerPos[0] -= 0.1 ;
    alListenerfv(AL_POSITION,listenerPos); break ;

   case 's': case 'S':
    listenerPos[0] += 0.1 ;
    alListenerfv(AL_POSITION,listenerPos); break ;

   case 'q': case 'Q':
    listenerPos[2] -= 0.1 ;
    alListenerfv(AL_POSITION,listenerPos); break ;

   case 'z': case 'Z':
    listenerPos[2] += 0.1 ;
    alListenerfv(AL_POSITION,listenerPos); break ;

   case 27:
    alSourceStop(source[2]);
    alSourceStop(source[1]);
    alSourceStop(source[0]);

    alutExit();
    glutDestroyWindow(GLwin) ;
    exit(0) ;
    break ;

   default: break;
   }
  glutPostRedisplay() ;
}
```

*Listado 4. Función "keyboard" de PIGE.*

También permite mover la posición del oyente, por lo que alguna de estas entradas actualiza valores de la posición del mismo con *alListenerfv*, por lo que al final de esta función se llama a OpenGL para que redibuje la escena, por si hay cambios, puesto que las variables globales de posición se utilizan tanto para pintar como para situar las fuentes de audio en la escena 3D. La tecla ESC (entrada con valor 27) hará que termine la aplicación rápidamente.

La posición del oyente representada por uno de los cubos en la escena se hace evolucionar con las teclas 'a', 's', 'q' y 'z'. También con las teclas del cursor que controla la función del listado 5 que permite los mismos movimientos: a derecha e izquierda y hacia dentro o fuera de la escena.

```
void specialKeys(int key, int x, int y)
```

```
{
 switch(key)
   {
  case GLUT_KEY_RIGHT:
    listenerPos[0] += 0.1 ;
    alListenerfv(AL_POSITION,listenerPos); break;
  case GLUT_KEY_LEFT:
    listenerPos[0] -= 0.1 ;
    alListenerfv(AL_POSITION,listenerPos); break;
  case GLUT_KEY_UP:
    listenerPos[2] -= 0.1 ;
    alListenerfv(AL_POSITION,listenerPos); break;
  case GLUT_KEY_DOWN:
    listenerPos[2] += 0.1 ;
    alListenerfv(AL_POSITION,listenerPos); break;
   }
 glutPostRedisplay() ;
}
```

*Listado 5. Función "specialKeys" de PIGE.*

# 5 Concluyendo

A lo largo de este objeto de aprendizaje hemos visto como OpenAL permite implementar operaciones de posicionamiento de audio en un espacio 3D. La experimentación sobre el código propuesto le permitirá al lector:

- Experimentar con las instrucciones mínimas de manera aislada.
- Observar como se mueven objetos en una escena visual, al tiempo que oír como varía el sonido asociado a cada uno de ellos

Ahora toca experimentar, empieza por cambiar cosas y después añade nuevas funcionalidades. Sorpréndete y sorpréndenos.

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