UNIVERSIDAD POLITECNICA DE VALENCIA

Efectos de audio básicos 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 al uso de las propiedades del audio para generar efectos básicos: volumen, *pitch*, polifonía y coros. 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 desglosar pequeños ejemplos de audio en los que las funciones de API de OpenAL se utilizan modificar las propiedades de las fuentes de sonido y ello el sonido que generan.

La librería permite modelizar una colección de fuentes de audio que son oídas por un único oyente en algún lugar de ese espacio. Los objetos básicos en OpenAL, el oyente (*Listener*) y las fuentes de audio (*Source*), comparten una serie de propiedades que se pueden consultar y modificar durante la ejecución del programa.

El motor de OpenAL hará esos cambios efectivos. 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 a lo que se escucha en el mundo real, mejor dicho, donde simular los efectos que suceden.

3 Objetivos

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

- Dar a conocer al alumno, de un modo participativo, el uso de OpenAL para llevar a cabo efectos básicos de procesado de audio.
- Manipular sonido en un computador, en términos de OpenAL.
- 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.
- Utilizar una plataforma de experimentación abierta 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.

4 Desarrollo

Vamos a ver cómo variar el volumen, la frecuencia percibida de un sonido (*pitch*) y la generación simultánea de sonidos en OpenAL. Lo cual es una casi una descripción completa del sonido

4.1 Programa Principal

Utilizaremos un pequeño programa para ir lanzando funciones que desarrollen por separado cada efecto que vamos a crear. Se puede ver el listado 1. Donde a parte de imprimir las versiones de las librerías de OpenAL y ALUT instaladas se invoca a los procedimientos mencionados.

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

void volumen();
void gravesAgudos();
void polifonic();
void coro();

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

 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 () );

    volumen();
    gravesAgudos();
    polifonic();
    coro();

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

} // fi de main
```

Listado 1. Programa principal de efectos básicos de audio con OpenAL.

Para ejecutarlo se compila rápidamente desde la línea de órdenes con

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

Atención a las comillas, que son las de ejecución: es como poner una tilde abierta a un espacio en blanco. Se ejecuta con la orden

$ ./nombreEjecutable

Pero necesitamos implementar esas funciones para que funciones ... Vamos a ello, simplemente avanzar que todas siguen el esquema siguiente:

- Obtener los sonidos de disco (*alutCreateBufferFromFile*) o generarlos con las primitivas de OpenAL, como, por ejemplo, *alutCreateBufferHelloWorld*..
- Crear las fuentes de sonido con *alGenSources, a*sociarles un sonido con *alSourcei* y ponerlas a "sonar" con al *SourcePlay*.
- Ir variando la/s propiedad/es que corresponda con *alSourcei, alSource3i, alSourcef o alSource3f*, según lo que corresponda.
- Liberar recursos utilizados con *alDeleteSources* y *alDeleteBuffers*

4.2 Volumen del audio

Se pueden subir la amplitud de la señal de sonido emitida y con ello el volumen, aumentando la propiedad AL_GAIN. El listado 2 muestra un ejemplo donde se hace sonar de manera continuada una misma señal en bucle, variando la mencionada propiedad para que se aprecie el detalle. Dejaremos un segundo entre cambio y cambio para que se aprecie.

```
void volumen()
{
 ALuint buffer, fuente;
 int i;
 ALint error;

 printf("Volumen\n");
 buffer = alutCreateBufferHelloWorld ();
 alGenSources( 1, &fuente );
 alSourcei(fuente, AL_BUFFER, buffer);
 alSourcei(fuente, AL_LOOPING, 1 );
 alSourcePlay(fuente);

 for (i=0; i < 180; i+=10)
   {
    alSourcef(fuente, AL_GAIN, sin((float)i*2*PI/360.0)  );
    printf("."); fflush( stdout );
    usleep( 1000000 );
   }
 alSourceStop( fuente );

 alDeleteSources( 1, &fuente );
 alDeleteBuffers( 1, &buffer );
 printf("\n");
}// Fi de volumen
```

Listado 2. Modificación del volumen del audio con OpenAL.

4.3 Graves y agudos

El *pitch* es el término, con pequeñas diferencias en el contexto donde se utiliza, por ejemplo, entre matemáticos y músicos. Hablamos aquí, como en psicoacústica[1], de la percepción del tono (frecuencia) de un sonido. Ésta es la que determina el nombre de las notas y que en algunos casos se conoce cono altura.

Siguiendo con nuestra aproximación vamos a ver una variación del ejemplo anterior, véase listado 3, donde cambiaremos la característica por esta, que se denomina AL_PITCH en OpenAL.

```
void gravesAgudos()
{
  ALuint buffer, fuente;
  int i;
  ALint error;

  printf("Graves y agudos.\n");
  buffer = alutCreateBufferHelloWorld ();
  alGenSources( 1, &fuente );
  alSourcei(fuente, AL_BUFFER, buffer);
  alSourcei(fuente, AL_LOOPING, 1 );
  alSourcePlay(fuente);

  for (i=0; i < 180; i+=10)
    {
     alSourcef(fuente, AL_PITCH, sin((float)i*2*PI/360.0)  );
     printf("."); fflush( stdout );
     usleep( 2000000 );
    }
  alSourceStop( fuente );

  alDeleteSources( 1, &fuente );
  alDeleteBuffers( 1, &buffer );
  printf("\n");

}// Fi de graves y agudos
```

Listado 3. Accediendo a ficheros de audio con ALUT en OpenAL.

4.4 Polifonía

Se habla de instrumentos polifónicos, en tanto que son capaces de generar varias notas (sonidos) ante la pulsación de varias teclas (botones o lo que corresponda a cada instrumento). Se aplica a instrumentos electrónicos para diferenciarlos de los que no son capaces: instrumentos monofónicos.

¿Podemos hacer esto en OpenAL? Para comprobarlo generaremos dos sonidos, los haremos sonar por separado y después los haremos sonar juntos para observar esta característica. Debe quedar claro que mientras haya datos se escuchará el sonido, hasta que se pare explícitamente y la ejecución continúa en la instrucción siguiente.

[1]Véase la definición del término en la wikipedia <http://en.wikipedia.org/wiki/Pitch_%28music%29

```
void polifonic()
{
#define NNOTES 2
  ALuint buffer[NNOTES], fuente[NNOTES];
  int i,j;
  ALint error;

  printf( "Polifonic\n" );

  alGenBuffers( NNOTES, fuente );
  buffer[0] =  alutCreateBufferWaveform( ALUT_WAVEFORM_SINE, 200.0, 0.0, 1.0);
  buffer[1] =  alutCreateBufferWaveform( ALUT_WAVEFORM_SINE, 800.0, 0.0, 1.0);
  alGenSources( NNOTES, fuente );
  if ((error = alGetError()) != AL_NO_ERROR)
    printf("Error: %s\n", alGetString(error));

  for (j=0; j < NNOTES; j++)
  {
    alSourcei(fuente[j], AL_BUFFER, buffer[j]);
    alSourcei(fuente[j], AL_LOOP, 1);
  }

  alSourcePlay(fuente[0]);
  usleep( 1000000 );
  alSourceStop(fuente[0]);
  alSourcePlay(fuente[1]);
  usleep( 1000000 );
  alSourceStop(fuente[1]);

  alSourcePlay(fuente[0]);
  usleep( 1000000 );
  alSourcePlay(fuente[1]);
  usleep( 4000000 );

  printf("borrant\n");
  alDeleteSources( NNOTES, fuente );
  alDeleteBuffers( NNOTES, buffer );
  printf("\n");

}// Fi de polifonic
```

Listado 4. Ejemplos de reproducción simultánea de varios sonidos con OpenAL.

El ejemplo se puede evidenciar aún más si se cargan dos ficheros de disco con contenidos claramente diferenciados, de modo que se escucharán ambos a la vez.

4.5 Coros

Se define[2] un efecto de coros (*chorus*) como el que hace sonar una única señal de sonido como si fuera producido por varias fuentes. En la práctica se utilizan algoritmos más complejos, para hacer más real el efecto, haciendo que cada fuente tenga una componente aleatoria que la haga mínimamente diferente a las demás en: en un coro de personas reales cada uno tiene una voz diferente y se ajusta más o menos a las indicaciones del director. No pretendemos llegar a tanto, pero es cuestión de documentarse y añadir más instrucciones ...

El listado 5 muestra como se genera un único sonido y un número de fuentes de sonido a las que se les asocia el mismo sonido. Podría haber sido una nota generada con una función al uso. Si se sitúa cada fuente en una posición ligeramente diferente se obtendrá una sensación más real. En cualquier caso, se comprobará que al ir aumentando el número de "voces" que participan se percibe un aumento del volumen y de la frecuencia básica.

```
void coro()
{
#define NVEUS 10
  ALuint helloBuffer, fuente[NVEUS];
  int i,j;
  ALint error;

  printf( "Coro\n" );
  helloBuffer = alutCreateBufferHelloWorld();
  alGenSources( NVEUS, fuente );
  if ((error = alGetError()) != AL_NO_ERROR)
    printf("Error: %s\n", alGetString(error));
  for (i=0; i < NVEUS; i++)
    {
      alSourcei(fuente[i], AL_BUFFER, helloBuffer);
    }

  for (j=0; j < NVEUS; j++)
    {
      for (i=0; i < j; i++)
            {
              alSourcePlay(fuente[i]);
              printf("\r %d", i); fflush( stdout );
            }
      usleep( 1000100 );
      printf("\r     "); fflush( stdout );
    }

  alDeleteSources( NVEUS, fuente );
  alDeleteBuffers(1, &helloBuffer );
  printf("\n");
}// Fi de coro
```

[2]De la ayuda de Audacity <http://audacity.sourceforge.net/manual-1.2/effects_chorus.html>. Se puede obtener más información al respecto de *Physical audio signal processing for virtual musical instruments and audio effects* (*Center for Computer Research in Music and Acoustics* (CCRMA)), <https://ccrma.stanford.edu/~jos/pasp/Chorus_Effect.html>.

Listado 5. Ejemplo de efecto "coral" con con OpenAL.

5 Concluyendo

A lo largo de este objeto de aprendizaje hemos visto operaciones básicas de manipulación de la señal de audio utilizando el API de OpenAL. La experimentación sobre el código propuesto le permitirá al lector manipular sonido en un computador, en términos de OpenAL:

- Volumen.
- La tonalidad principal o *pitch*.
- La generación simultánea de sonidos.

Es momento de experimentar con el código anterior. Prueba a cambiar los sonidos que se generan por los que tengas disponible en disco. Puedes hacer un sencillo reproductor con las opciones vistas o tu propio sintetizador .

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)