UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 5
ENTORNOS VIRTUALES

SoundToolKit

Módulo Para el Manejo de Sonidos en Sistemas de Realidad Virtual

Trabajo para optar por el Título de Ingeniería en Ciencias Informáticas

Autores: Lazaro Abreu Reche

Dagoberto Marrero López

Tutor: Ing. Yanoski Camacho Román

Ciudad de la Habana

Mayo 2007

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo, y autorizamos al Proyecto Herramientas de Desarrollo para Sistemas de Realidad Virtual de la Facultad 5 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Autores:

Lazaro Abreu Reche

_____

Dagoberto Marrero López

_____

Tutor:

Yanoski Camacho Román

_____

Datos de contacto

Nombre y Apellidos: Yanoski Rogelio Camacho Román

Edad: 26 años.

Ciudadanía: cubano.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Informática.

Categoría Docente: Profesor Instructor.

E-mail: rcamacho@uci.cu

Graduado de la CUJAE, con tres años de experiencia en el tema de la Gráfica Computacional, y líder de un proyecto de Realidad Virtual en la Universidad de las Ciencias Informáticas.

Dedicatoria

A mi madre y mi padre que siempre me apoyan en mis locuras, a mi hermana para que en unos años me dedique su tesis, a Rolo por meterme en el mundo de la Computación desde niño y a Ariangna por soportarme y apoyarme todos estos años.

Lazaro.

A mi madre, a mi padre, a mi hermano, a ellos le debo lo que soy.
A mi familia en general y demás seres queridos.

Dagoberto.

Agradecimientos

A nuestro tutor por el tiempo que nos dedicó.

A Fernando que siempre se mantuvo al tanto del trabajo y aclaró muchas dudas.

A Liudmila, Leticia, Karel, Yonnier, por su ayuda.

A Jesús Gumbau Portalés desarrollador de “Sandra Engine”, por su colaboración vía e-mail.

A Jose Luis Fernandez Fasani, por su colaboración vía e-mail.

Y a todos aquellos que han ayudado o al menos han soportado el ruido de las pruebas y a los que desde el comienzo nos han servido de guía en este mundo de la Realidad Virtual.

¡Gracias!

Resumen

La Realidad Virtual se ha expandido considerablemente a diferentes sectores de la sociedad en los últimos años. Puede encontrarse en aplicaciones de la defensa, la medicina, la educación y el entretenimiento. Esto ha provocado la evolución del hardware, y del software usado para ello, tanto sistemas de visualización, modelos físicos-matemáticos, modelos de inteligencia artificial como sistemas de sonido que se suman para dar realismo.

El sonido es una pieza importante para lograr la aproximación a la realidad, pues el mundo real está lleno de ondas sonoras, y hay que lograr simularlas con la mejor calidad posible.

Este trabajo propone una herramienta para el manejo de Sonidos en Sistemas de Realidad Virtual, surgida a partir de que se comienzan a realizar productos de esta rama de la informática dentro de la Universidad de las Ciencias Informáticas. Esta herramienta, SoundToolKit pretende brindar funcionalidades para el manejo de sonido de forma cómoda a los programadores, abstrayéndolos de los bajos niveles de programación, con una arquitectura orientada a objetos, configurable y multiplataforma, desarrollada usando el Proceso Unificado de Software (RUP) e implementada en C++, basada en OpenAL, y compatible con varios formatos y con posibilidades de adicionarle otros nuevos en futuras versiones.

Palabras Clave

Sonido, realidad virtual, OpenAl, DirectX, DirectXSound, efecto, 3D.

Contenido

Introducción

Desde el surgimiento de los Sistemas de Realidad Virtual (SRV), se trabaja en función de poder simular procesos o situaciones reales con tal nivel que sea difícil distinguir entre lo real y lo virtual. Para esto se han desarrollado motores de visualización y herramientas gráficas capaces de obtener entornos virtuales con gran calidad.

Los componentes por los que generalmente está formado un SRV los podemos agrupar en: gráfico, red, módulo físico-matemático, y sonido. [1] Cada componente juega un papel protagónico en el buen funcionamiento del sistema, pero el que ocupa esta investigación es el sonido, que junto al gráfico son con los que el usuario interactúa de forma más directa. Un mundo totalmente silencioso sería vacío, aburrido, y un SRV sin sonido carecería de realismo.

Si bien en los inicios de la computación no se hablaba de sonidos o solo se limitaban a producir algunos tonos, con el desarrollo de las tarjetas de sonidos ya se habla de reproducción y creación de música, sonidos 3D, video juegos, o SRV que crean un ambiente acústico tan real e impresionante que hace olvidar la realidad. Todos estos resultados se han logrado a partir de la creación de librerías de alto nivel, capaces de complementar los resultados en la parte gráfica y satisfacer los requerimientos en los SRV. (Ver **Tabla 3**)

Cuba no se ha quedado atrás en estos menesteres y actualmente incursionan en el desarrollo de gráficos por computadora la empresa cubana de software ”Simuladores Profesionales” (SIMPRO) y la Universidad de las Ciencias Informáticas (UCI). Ambas trabajan de conjunto en el desarrollo de SRV para su exportación y uso interno en el país. Actualmente disponen de una herramienta básica para el manejo gráfico. Sin embargo esta no cuenta con la capacidad de manipular sonidos.

La presente investigación parte del siguiente **problema**, ¿Cómo -a partir de la carencia que presenta la herramienta -, y empleando de forma óptima los recursos disponibles, obtener sonidos de la mayor calidad y realismo posibles?

Para solucionar esta interrogante se plantea como **objeto de estudio** el manejo de sonidos en computadoras y como **campo de acción** los sistemas de sonido para SRV.

Este trabajo se plantea como **objetivo general** obtener un módulo para el manejo de sonido en SRV.

Se definen como **objetivos específicos**:

- Acoplar el módulo a la herramienta.
- Facilitar la labor de los programadores que utilicen el módulo.
- Obtener sonido tridimensional en tiempo real.
- Cargar y manipular formatos de sonido de uso extendido (.wav, .au, .ogg).
- Garantizar que el nuevo módulo sea flexible a futuras actualizaciones.

Por ende, para cumplir este objetivo, se plantean las siguientes **tareas**:

- Estudiar y analizar las principales Bibliotecas de Sonido existentes
- Analizar las principales tendencias en cuanto al uso de sonidos en aplicaciones SRV.
- Estudiar la arquitectura y características de la Biblioteca Básica desarrollada por el proyecto Investigación y Desarrollo de Herramientas Básicas.
- Plantear soluciones técnicas.
- Realizar el análisis y diseño para obtener una arquitectura de clases que responda a los objetivos planteados.

En sentido general con este trabajo se espera de forma sencilla poder configurar y utilizar el sonido, para lograr una calidad acústica comparable con el mundo real.

Capítulo 1 Fundamentación teórica

Introducción

El sonido puede brindar información alternativa o complementaria a un usuario en general y esto no es menos válido en un mundo virtual, donde además está demostrado que un buen sonido 3D ayuda a compensar pequeños retrasos en la visualización, ya que éste usualmente, puede ser generado con mayor velocidad que el gráfico.

Existen por tanto determinadas características que un buen sistema de sonido debe cumplir sobre todo en cuanto a la forma de generar el sonido, al uso de formatos, a la manipulación de los sonidos y la variación de sus parámetros. Además debe tener funcionalidades que permitan una mayor integración al entorno gráfico.

En el presente capítulo se hace un análisis de las características básicas del sonido digital, así como bibliotecas, algoritmos, métodos, formatos y tendencias para su uso actualmente en el mundo de la realidad virtual y los videojuegos.

1.1 Conceptos fundamentales.

Para trabajar con el sonido digital hay que analizar varios conceptos y características, a modo de cimientos que servirán de base para los siguientes capítulos de este trabajo.

1.1.1 Sonido Digital. Características.

El sonido es la interpretación que hace nuestro cerebro de las variaciones de presión que genera un objeto vibrante en determinado medio. Por su naturaleza ondulante, se propaga en forma de ondas analógicas desde el objeto que lo produce. Para que esta onda o vibración sea audible por los seres humanos, el objeto debe oscilar aproximadamente entre 20 y 20 000 veces por segundo, ya que las frecuencias de hondas que somos capaces de escuchar van de 20 Hz a 20 KHz. Si se representa en un gráfico un sonido, se obtendrá lo siguiente (Figura 1). [2]

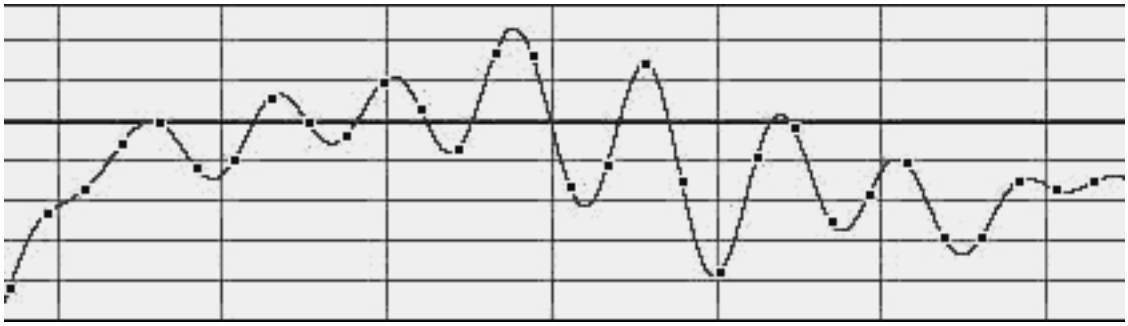

Figura 1 Representación gráfica de un sonido.

Pero en el mundo informático no se trabaja con datos analógicos, se hace con datos digitales, formados por estados binarios. Por lo tanto, para representar un sonido, desde el punto de vista informático, es preciso capturarlo en una naturaleza binaria. Para lograr esto se hacen dos procesos primero se hace un muestreo y después el valor de la muestra debe ser cuantificado y codificado como una cadena de bits. Tomando determinados valores de las ondas y representandolos en formato digital. Si se grafica se obtendrá lo siguiente (Figura 2).

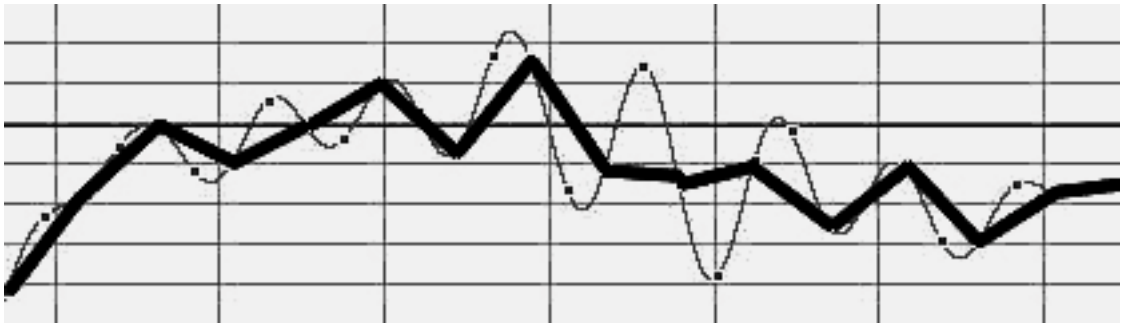

Figura 2 Digitalización de un Sonido

Entre más muestras se tomen más fiel será el sonido capturado respecto al original, por lo que tendrá más calidad.

Muestrear una señal analógica contínua en el tiempo significa medir a intervalos regulares (período Ts) dicha señal.

De este modo, la versión muestreada de una señal contínua f(t) se representa mediante sus valores:

$$f(t) = \{f(0), f(Ts), f(2Ts), f(3Ts), \ldots, f(nTs)\}$$

Donde Ts es el período de muestreo (intervalo temporal entre muestras y f = 1/Ts la frecuencia de muestreo.

Según el Teorema de Nyquist [3], para obtener una representación digital adecuada de una señal analógica, la frecuencia de muestreo debe ser el doble de la frecuencia más alta presente en la señal analógica.

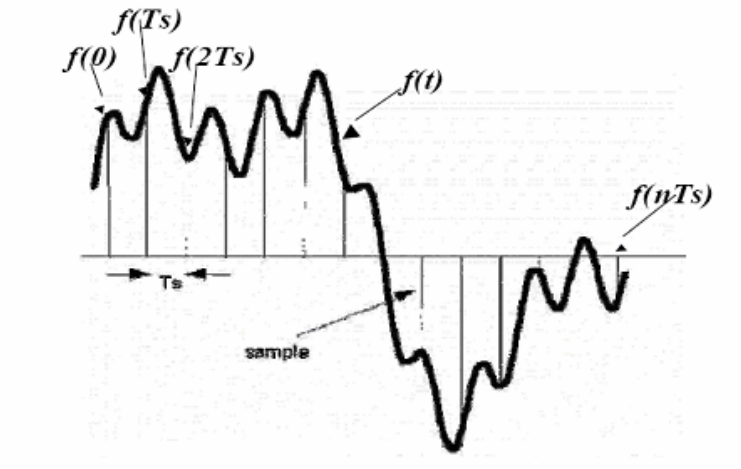

Figura 3 Frecuencia de muestreo.

Después que una señal analógica ha sido muestreada, el valor de la muestra debe ser cuantificado y codificado como una cadena de bits. La cuantificación implica la representación de los valores de las muestras en términos de un conjunto discreto de valores de amplitud de la señal analógica.

El número de bits utilizados para representar el valor de las muestras determina la precisión del proceso de cuantificación/codificación.

El proceso de cuantificación aproxima el nivel de la señal analógica al nivel discreto más cercano. Cuanto mayor sea el número de niveles (bits utilizados por muestra) más calidad tendrá la muestra, pero mayor espacio ocupará su representación digital.

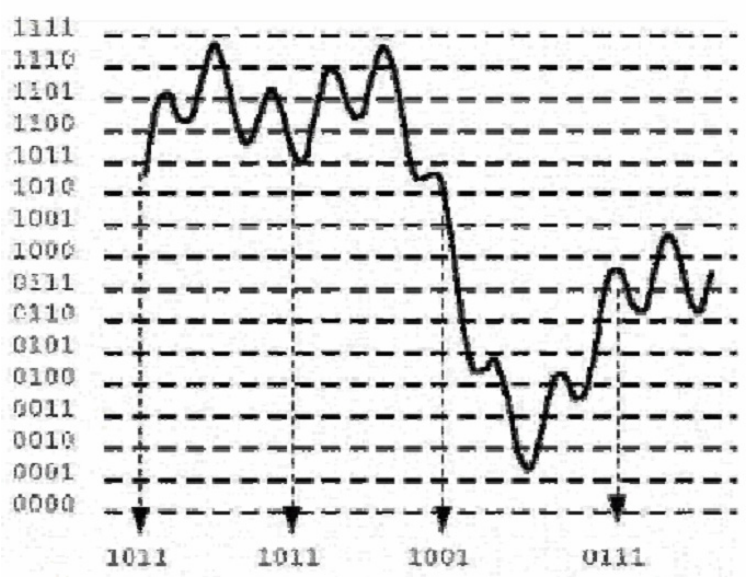

Figura 4 Cuantificación del Sonido

Las características fundamentales del sonido [4].

Frecuencia: Es el número de vibraciones por segundo. La frecuencia de sonido se mide en Hercios (Hz). Un sonido que vibra una vez por segundo tiene una frecuencia de 1 Hz. Las frecuencias se escriben normalmente en kilohercios (Khz.), unidad que representa 1.000 Hz.

Una persona normal sin problemas auditivos puede oír sonidos que estén entre los 20 Hz y 20 Khz. (aunque la mayoría de nosotros raramente sobrepasamos los 16 Khz.).

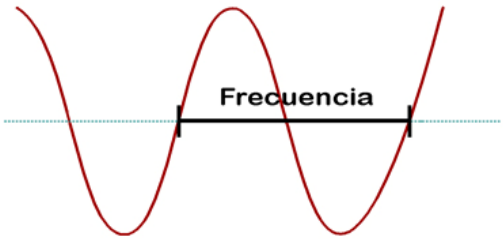

Figura 5 Frecuencia

Frecuencia de muestreo: es la cantidad de muestras de sonido capturadas en cada segundo. Su valor puede oscilar entre 8 Khz. (8.000 muestras en cada segundo) y 48 Khz.

Amplitud: Es la cantidad de fuerza o energía de un sonido. Se mide en decibelios (db). Es la que influye en la intensidad o volumen con que se escucha el sonido.

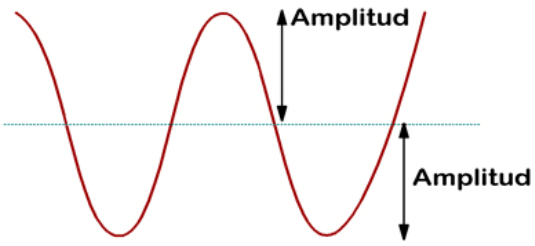

Figura 6 Amplitud de Onda

Por cada 3 db que se le aumente al sonido se duplica la intensidad del mismo. Por ejemplo si un sonido tiene 43 db y se le aumenta a 46 db se duplica su intensidad, y si se aumenta a 49 la intensidad será 4 veces mayor.

Precisión de las muestras: indica la escala de bits que se ha utilizado para guardar el sonido. Pueden ser 8 bits (256 valores posibles) o 16 bits (más de 65.000 valores posibles).

Mono / Estéreo: El sonido pude grabarse en un solo canal (mono) o en dos (estéreo).

1.1.2 Formatos de Sonido, compresión.

Los formatos de fichero indican la estructura con la que el audio es almacenado, si bien en los comienzos del audio digital aparecieron múltiples formatos de audio. Con el paso del tiempo y la aparición de formatos más flexibles y eficientes fue reduciéndose el conjunto, quedando entre los más estandarizados un pequeño grupo [5].

En general un mismo formato de fichero permite contener diversas codificaciones, tasas de muestreo, etc. En este sentido se distingue entre dos grupos de formatos de ficheros de audio [5]:

Formatos autodescriptivos: Contienen de forma explícita los parámetros del dispositivo y la codificación en algún punto del fichero.

La cabecera suele comenzar por lo que se conoce como un número mágico, que no es más que un valor fijo que permite identificar el fichero como un fichero del formato buscado. Esta cabecera suele contener la tasa de muestreo, el número de bits por muestra, si las muestras tienen signo o no, y otro tipo de información como descripción del sonido que contiene o notas de *copyright*.

La siguiente tabla muestra una relación de algunos de los formatos de fichero de audio autodescriptivos más habituales e indica algunos de los parámetros que permiten modificar.

Tabla 1 Formatos autodescriptivos

Extensión	**Nombre**	**Origen**	**Parámetros modificables**				**Comentarios**
			Tasa	***Canales***	***Codific***	***Otra info***	
.au, .snd		NeXT, SUN	Sí	Sí	Sí	Sí(1)	
.aif, .aiff	AIFF	Apple, SGI	Sí	Sí	Sí(2)	Sí	
.aif, .aiff	AIFC	Apple, SGI	Sí	Sí	Sí(2)	Sí	AIFF con compresión
.iff	IFF/8SVX	AMIGA	Sí	Sí		Sí	Info. de instrumento, 8 bits
.mp2, .mp3	MPEG	MPEG	Sí	Sí	Sí		
.ra	Real Audio	Real Networks	Sí	Sí	Sí		
.sf	IRCAM		Sí	Sí	Sí	Sí	
.smp		Turtle Beach					16 bits/1 canal,

							bucles
.voc		SoundBlaster	Sí				8 bits/1canal, puede detectar el silencio
.wav	WAVE	Microsoft	Sí	Sí	Sí(2)	Sí	
.wve		Psion					8k sps, 1 canal, ley-A, 8 bits
(ninguna)	HCOM	Mac	Sí				8 bits/1 canal, comp. Huffman

(1) Una cadena de información

(2) Sólo longitud de muestra

Formatos sin cabecera o tipo raw: Los parámetros del dispositivo y codificación empleada son fijos.

Estos formatos definen un único esquema de codificación y no permiten la variación de los parámetros salvo, en algunos casos, la tasa de muestreo. De hecho, muchas veces no se puede conocer de ninguna forma la tasa de muestreo empleada a menos que se escuche el sonido.

Estos formatos son menos importantes que los autodescriptivos, por ser menos flexibles. Hoy en día están prácticamente en desuso, aunque en el pasado fueron los primeros en aparecer.

Tabla 2 Formatos sin cabecera o tipo raw.

Extensión	**Origen**	**Características**
.snd, .fssd	Mac, PC	Tasa variable, 1 canal, 8 bits
.ul	US Telephony	8ksps, 1 canal, 8 bits, Ley-m
.snd	Amiga	Tasa variable, 1 canal, 8 bits con signo

En este trabajo se prestará mayor atención a los Formatos descriptivos por su flexibilidad y eficiencia, además de ser los más usados actualmente, entre ellos los formatos *.WAV*, .AU, *.MP3* y *.OGG*.

Los archivos WAV almacenan todos los bits obtenidos de la digitalización, por lo tanto el espacio que ocupará en disco dependerá directamente de la duración del sonido, la frecuencia de muestreo y la resolución utilizada. Como consecuencia estos archivos ocupan mucho espacio en el disco duro pero conservan toda la calidad obtenida en la digitalización [2].

El formato *AU* creado por Sun Microsystems es soportado por todas las diferentes plataformas, entre ellas Windows, UNIX, Linux y Macintosh. Este formato permite varios tipos de codificación de sonido, aunque el más popular es la codificación de 8 bits conocida como *u-law* por lo que en ocasiones el formato AU se llama directamente formato *u-law*. Los ficheros AU tienen una calidad aceptable. [5]

Debido a la necesidad de manejar archivos de sonido más pequeños (en cuanto a tamaño en bytes) se diseñaron otros formatos de almacenamiento, de los cuales el MP3 es uno de los más populares. Este formato aprovecha el hecho de que ciertos sonidos no son habitualmente distinguidos por el oído humano y a esto se le suma la utilización de complejas técnicas de compresión para lograr archivos varias veces más pequeños que los correspondientes WAV, en los cuales la diferencia de calidad es prácticamente imperceptible [2]. El MP3 usa un algoritmo de compresión MPEG Layer 3 (Moving Picture Experts Group). [6]

Otro formato comparable con el MP3 es el OGG pero con la característica particular de que es completamente código abierto, libre de patentes [7], este formato usa un algoritmo de compresión ADPCM (Acronym for Advanced Differential Pulse Code Modulation) [6] y en poco tiempo ha ganado mucha popularidad por la calidad y la compresión lograda por este formato.

Tanto el MP3 como el OGG usan algoritmos de compresión con pérdida “Lossy Compression”, en los que se pierde una parte de la información del fichero original, sin afectar totalmente la calidad del fichero pero como resultado se obtiene una reducción significativa del espacio en disco.

1.2 Sonidos en Sistemas de Realidad Virtual.

1.2.1 Precedentes históricos.

Historia del Proceso Digital [8]

- 1950 Max Mathews sintetiza el sonido digital en un ordenador.
- 1985 Commodore Amiga inicia el proceso digital del sonido: Estudio 16
- 1989 dos plataformas Apple y NeXT inician el proceso digital con un costo muy elevado.
- A finales de 1988 Ad Lib crea las primeras tarjetas de sonido para pc.
- 1990 Creative Labs lanza al mercado el primer Sound Blaster.

Actualmente las tarjetas para pc tienen características parecidas al original con dos canales de 16 bits y calidad 44.100 Hz.

1.2.2 Tecnologías Usadas.

La implementación de sonido en juegos de video, se realiza hoy en día bajo dos tecnologías diferentes: el sonido envolvente 5.1 y el audio posicionado en 3D. Si bien la calidad de sonido de ambos sistemas puede ser comparada debido a las características propias de cada uno de ellos, no existe una opinión única acerca de que sistema es más eficiente o de cuál presenta mejores resultados en cuanto a la calidad del sonido reproducido. [9]

Así como se conoce que los sistemas 5.1 presentan irregularidades en cuanto a la reproducción de sonido en ciertos rangos dentro de los 360º que rodean al oyente, los sistemas de audio 3D disponibles para juegos de video, si bien atentan a reproducir sonido desde cualquier posición del espacio, no alcanzan, hoy por hoy, un nivel de realismo tal, que convenzan tanto a los desarrolladores de juegos como al público en general de ser la mejor opción para el sonido en este género. [9]

¿Qué es sonido envolvente 5.1?

Sonido envolvente o sistema surround esta diseñado para crear una sensación en la que parezca que la “imagen sonora” rodea al oyente, para crear un efecto más realista [10] Como tal el sonido 5.1 es un arte determinado, ya que debe cumplir con ciertos requisitos para que sea tanto mezclado como escuchado. El uso del subwoofer busca separar el manejo de las frecuencias bajas, permite la reducción en tamaño y en coste de los demás altavoces, además de reducir la distorsión resultante del sistema. Entre sus principales configuraciones [9]:

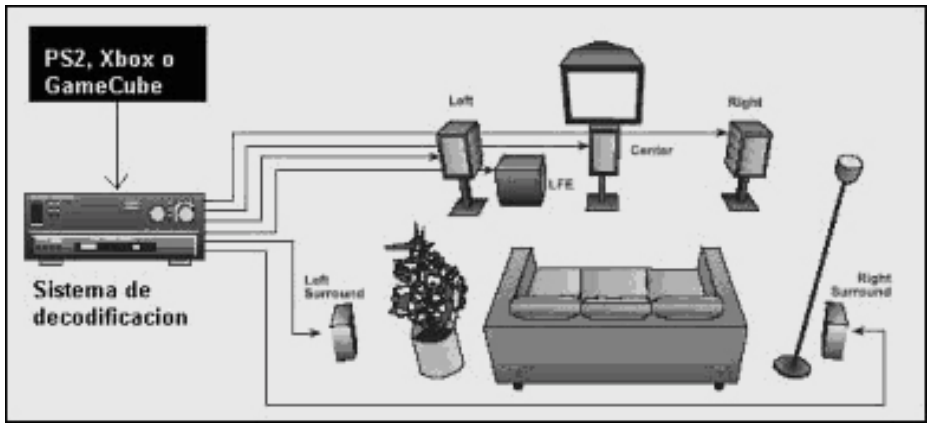

Figura 7 Configuración para Audio 5.1

Esta Figura 7 lo que puede ser una configuración típica de una estación de video juego 5.1.

En el sonido 5.1 se deben considerar múltiples altavoces que apuntan en diferentes direcciones y que son afectados por factores tales como el tamaño de la sala de mezcla y el del ambiente en cual se reproduce el sonido, para lo cual debe tomarse en consideración la geometría, instalación eléctrica y acústica del recinto. Las condiciones están dadas por el tamaño de la sala (chica, mediana y grande). Idealmente se espera que tengan paredes paralelas, incluso entresuelo y techo y una altura mínima de 3 metros, y que el volumen de la pieza tenga unos 300 m^3 con un área de piso de 30 m^2. Debido a estas condiciones es que se tiene que tener en cuenta, que no siempre (o mejor: nunca) en una casa se va a logar obtener un sonido 5.1 de calidad.

¿Qué es audio 3d?

La síntesis del audio 3D es ampliamente utilizada en sonido en video juegos como en la música en algunos de los casos. Ya se ha dicho que el audio 3D funciona con la codificación HRTF, Funciones de Transferencia Relativas (Head Related Transfer Functions HRTFs) que simula procesos acústicos que ocurren naturalmente en un espacio determinado el cual va a producir una interacción entre la atmósfera recreada por el video juego y nuestros oídos. Figura 8.

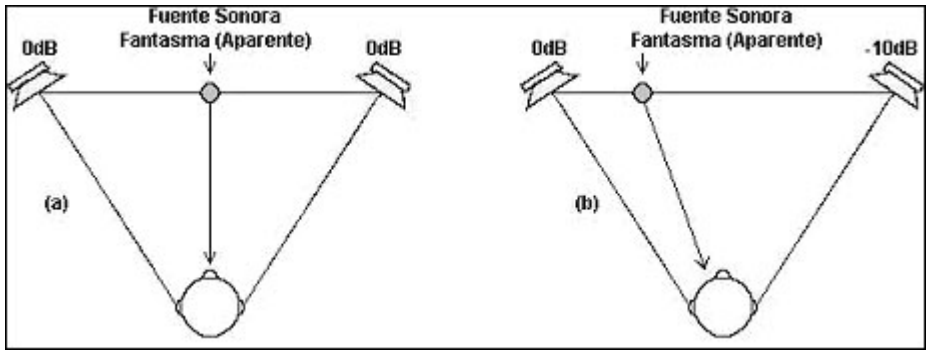

Figura 8 Audio 3D Posicionado

Este fenómeno es el resultado de reverberaciones y reflexiones con todas las variaciones que pueden llegar a tener. Muchas formas se han recreado para reproducir el sonido 3D, una de ellas es por medio de audífonos, los cuales han sido descartados de este análisis ya que la cercanía de los monitores no permiten sentir con real dimensión los espacios tratados de recrear con los algoritmos 3D, ya que se sentirá un sonido muy cerca, que gana en posicionamiento pero pierde en realismo creando una imagen de sonido en nuestra cabeza y no en un espacio en el cual uno se encuentre dentro [11].

En la Figura 9 se representa en el proceso HRTF el porque en esta ocasión, el sistema de audífonos ha quedado fuera de nuestro análisis comparativo con el sistema de 5.1 [11].

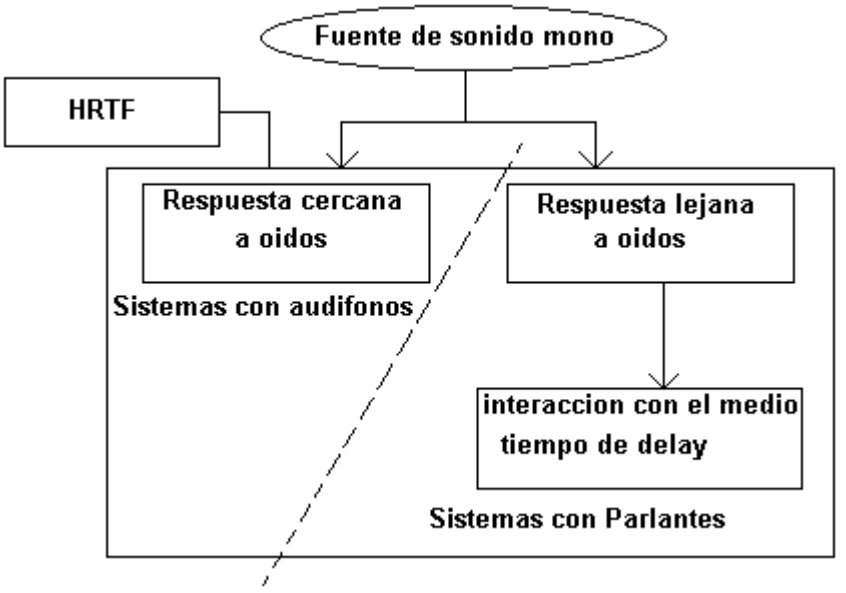

Figura 9 Proceso HRTF.

Porque 3D es mejor que 5.1, según fabricantes:

Los métodos convencionales que reproducen reflexiones y reverberaciones de forma externa (a través de más monitores y sin los algoritmos) como el clásico 5.1, dan una imagen fallida por ser un modelo muy simplista. La relación análoga entre el modelo de imagen y la línea de *delay* (demora) que dan las reverberaciones deben ser dirigidas desde el mundo irreal (mundo video juegos) a nuestro ambiente.

El hecho de que los jugadores pratiquen en sus propias habitaciones (por lo general ocupadas por muebles), y de que las salas de juego son habitualmente pequeñas , provoca que en estos lugares falle la difusión acústica.

Esto destruye la verdadera imagen que debería exponer un juego 3D. La figura 10 muestra las 6 reflexiones de primer orden que han sido calculadas para una sala normal de 7 m de largo por 5m de ancho. Ha sido trazado un esquema cuando el sonido 3D es reproducido por un demo con codificación de la empresa Sensaura. Se puede observar que las primeras reflexiones (que son las del techo a 2.8 m/s), y las segundas (provenientes del suelo a 3.2 m/s), por su intensidad son las que

más podrían molestar al oyente. Pero el oído humano no está capacitado para diferenciar señales con menos de 30 m/s de diferencia. Así que es el sonido directo el que domina sin importar donde se reproduzca [11].

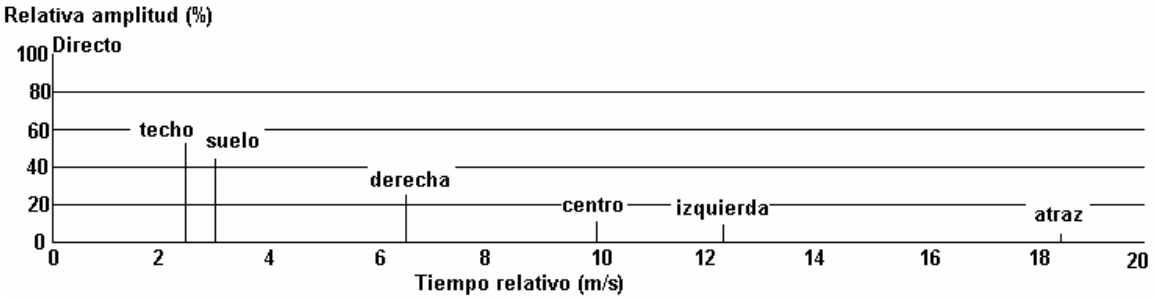

Figura 10 Reflexiones de Onda.

1.3 Actualidad de los engines de sonido.

1.3.1 Manipulación del audio a bajo nivel.

Hay varias opciones para manipular el sonido a bajos niveles, se analizarán dos de las variantes que reúnen las prestaciones y características necesarias para lograr un producto de calidad.

1.3.1.1 DirectX/DirectXSound.

A la hora de elegir que API (Application Programmer Interface) utilizar en la programación de audio se encuentran varias posibilidades. Por un lado están las soluciones de bajo nivel como DirectSound, componente de audio de DirectX. En teoría si se desea exprimir al máximo las posibilidades de sonido de la PC (y consta de recursos necesarios para brindarle) sería la opción ideal, pero lo cierto es que los tiempos de desarrollo no son eternos y es necesario buscar opciones con una relación esfuerzo/resultado más favorable. [12]

El API DirectMusic, también componente de DirectX pero de más alto nivel, automatiza la carga de varios formatos populares de sonido, así como la gestión de los buffers y otras tareas de bajo nivel, pero como desventaja, en ciertos casos aumenta la latencia (tiempo entre la ejecución de un sonido y el mismo sonando en los parlantes).[12]

1.3.1.2 OpenAL.

OpenAL es una API libre y multiplataforma para sonido tridimensional, orientada inicialmente a los videojuegos, pero actualmente multipropósito [13]. Creada por Loki Entertainment a finales de los 90 como herramienta para facilitar los portes de juegos con sonido 3D a Linux. Adoptada por Creative Labs, nVidia y otros fabricantes como estándar de audio para sus productos. Adoptada por Raven Software, Epic Megagames, Exelweiss Entertainment y otros como base del sonido de sus juegos. En la siguiente figura se muestra un gráfico del flujo de la OpenAL dentro de una aplicación general [6]. (Figura 11)

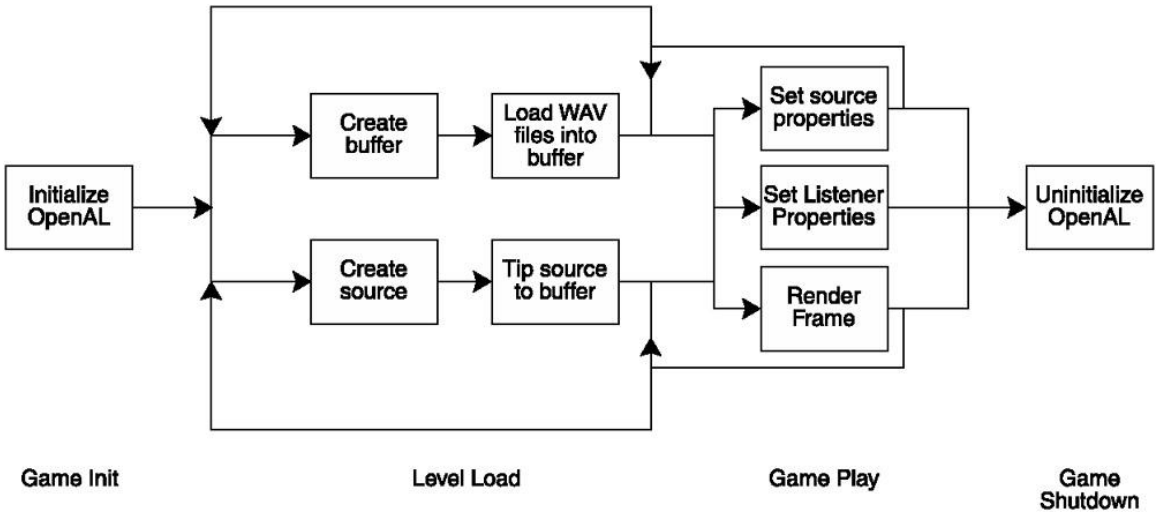

Figura 11 Flujo de Trabajo de OpenAL.

¿Qué proporciona OpenAL? (teoría) [14]

- Especialización del sonido.
- Fuentes de sonido posicionadas en 3D.
- Efectos físicos realistas de comportamiento del sonido.
- Múltiples contextos simultáneos de sonido.

¿Qué proporciona OpenAL? (práctica)

- API básica estable, simple, clara y multiplataforma.
- Interfaz en C++.
- Interoperabilidad con OpenGL, diseñada acordemente con ésta.
- Posibilidad de construir extensiones y usar las EAX, (*environmental audio extensions)* Extensión de audio ambiental, tecnología propia de Creative y Soundblaster, que proporciona un nivel nunca antes alcanzado en simulación acústica y sonido de juegos [15] (Figura 12).

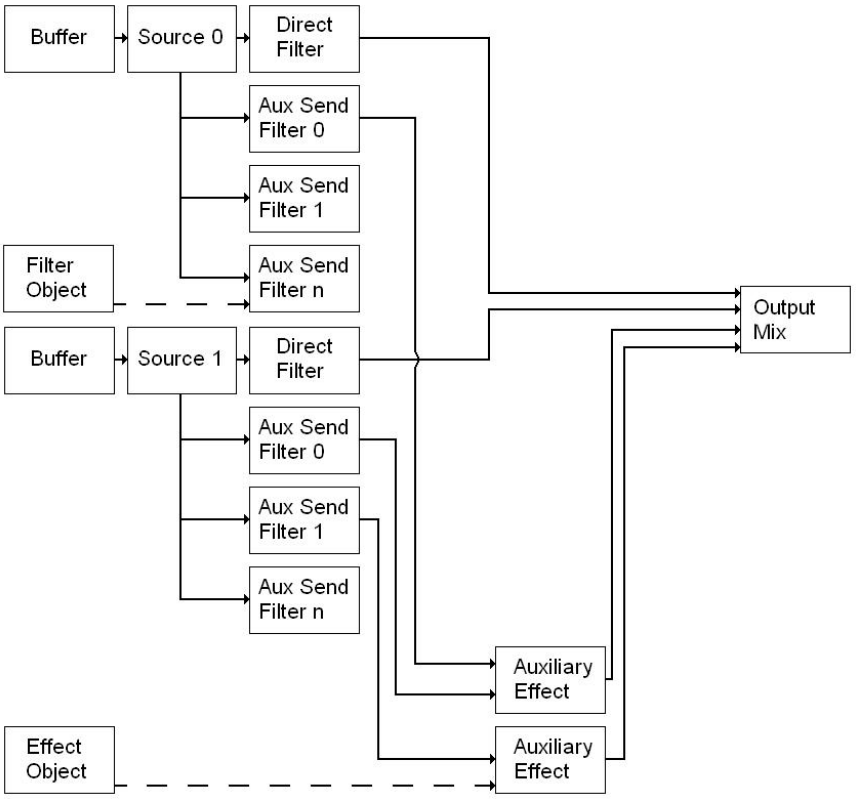

Figura 12 Arquitectura de OpenAL + EAX.

¿Qué abstracciones proporciona?

Dispositivos: Flujo de salida del sonido. Puede ser un dispositivo físico (tarjeta de sonido) o un demonio de sonido (esd)

Buffers: Depósitos de sonidos en memoria.

Contexto: Marco en el que "viven" varios objetos (fuentes, oyente) diferenciado y acotado.

Fuente: Objeto tridimensional que emite sonido. Posee diversas propiedades como cono de orientación, ganancia de sonido o bucle de reproducción.

Oyente: (*Listener*) Objeto tridimensional respecto al que se calcula el posicionamiento de sonido de las fuentes. (Figura 13)

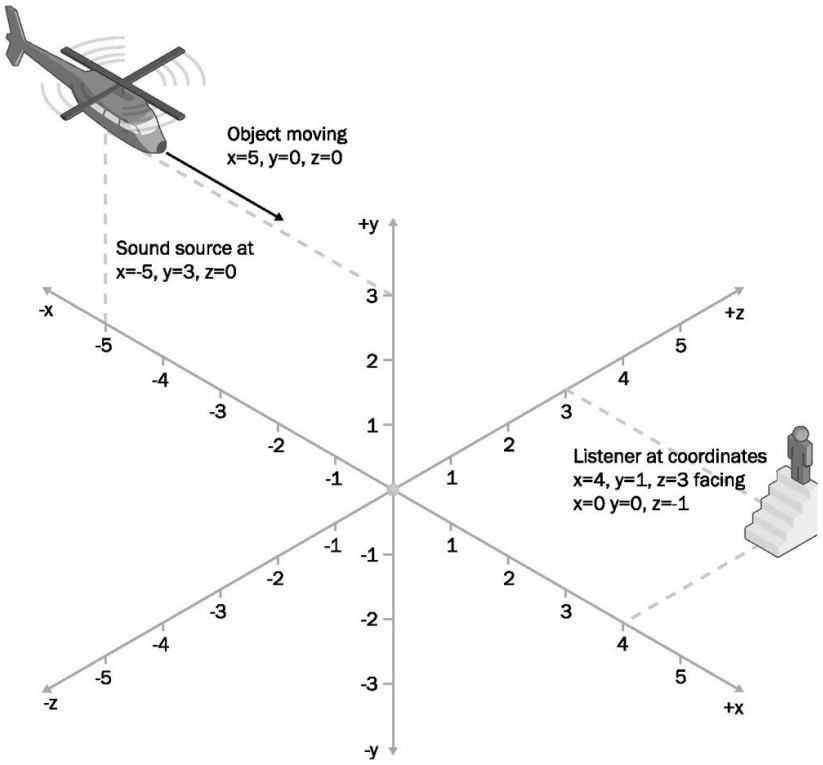

Figura 13 Oyente o Listener en un mundo 3D

¿Qué funcionalidades proporciona?

Efecto Doppler: Modificación de la frecuencia de un sonido según su velocidad y posición relativas al oyente. Está activado automáticamente y se aplica a todas las fuentes. Puede desactivarse para consumir menos recursos. (Figura14)

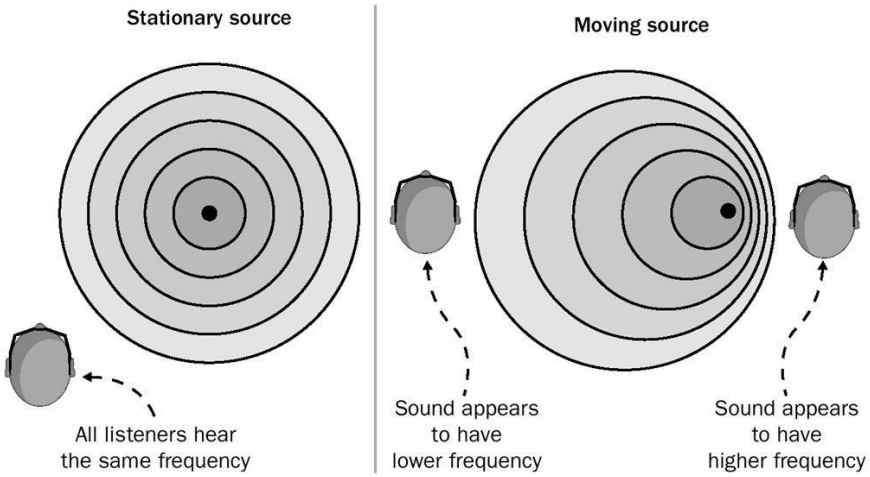

Figura 14 Effecto Doppler

Atenuación logarítmica del sonido por distancia: Simula el comportamiento realista no lineal del sonido. Se le pueden aplicar dos modelos de cálculo o desactivarlo, según se quiera gastar más o menos recursos. (Figura 15)

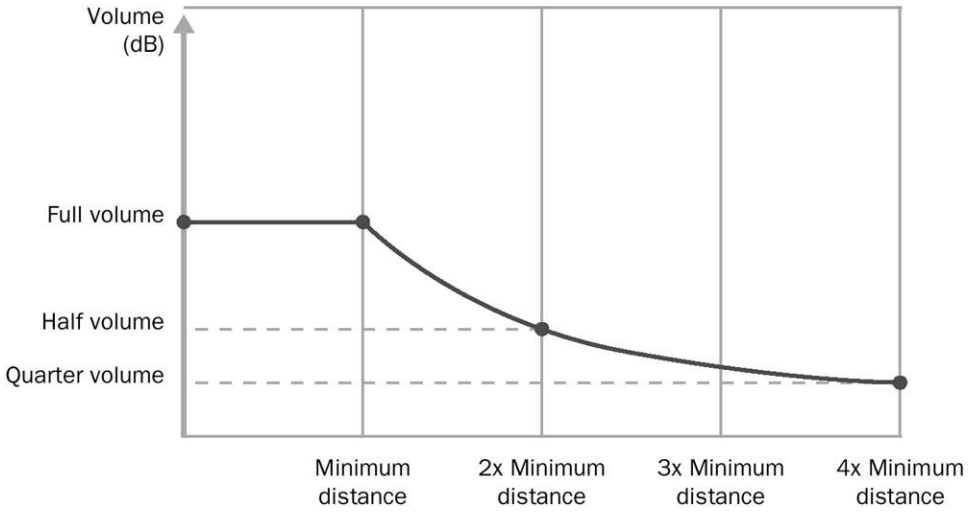

Figura 15 Atenuación logarítmica por Distancia

Ventajas

- API estable y fácil.
- Abierta.
- Extensible.
- Escrita en C.
- Multiplataforma.
- Modelo físico inherente.

1.3.2 Librerías de alto nivel de Terceras Partes. Características.

Cuando un proyecto posee un alto nivel de complejidad a veces no es factible desarrollar una librería de sonido partiendo desde cero, el tiempo u otros factores no le permitirían realizarla, o quizás el proyecto a realizar no necesita grandes particularidades en cuanto a sonido. Por estas razones hay varias compañías que se han dedicado a desarrollar librerías de alto nivel con gran número de funcionalidades y de fácil manejo para personal no especializado en el tema, en la siguiente tabla podrán ver algunos ejemplos de las más usadas, precios y características fundamentales.

Tabla 3 Principales Librerías de Terceras Partes. Características.

Librería	Plataforma	Licencia	Fuentes	Descripción
FMod http://www.fmod.org	Win32, Win64, WinCE, Linux, Linux64, Macintosh (os8/9/10/x86) PS2, PSP, PS3, Xbox, Xbox 360,GameCube, Wii.	Libre para uso no comercial $100 USD para shareware en PC $6000 USD para título AAA en PC .	No	Librería muy popular utilizada en diversos títulos. Se jacta de ser la librería que hace menor uso de CPU (compara contra BASS, MILES, Seal).
BASS http://www.un4seen.com	Win32, Linux, MacOSX, PocketPC.	Libre para uso no comercial €100 para shareware en PC €950 para un título AAA en PC. €2450 para uso ilimitado	No	Librería popular. No posee Implementaciones en consolas.
X-Audio http://www.xaudio.com	Win32, Linux, MacOS, PocketPC	Libre para uso no comercial	No	Requiere permiso Explícito para uso no comercial y descarga del SDK.

MILES http://www.radgametools.com/miles.htm	Win32, MacOS, XBOX (próximamente GameCube)	Comercial para todo tipo de uso $3,000.00	No	Tal vez la librería Más ampliamente utilizada. No es posible probarla sin Una licencia (es posible solicitar un testeo vía mail).
Audiere http://audiere.sourceforge.net	Win32, Linux	Libre	Si	Libre para todo tipo de uso. Interfaz Sencilla pero con pocas funcionalidades.

Estas librerías en general brindan funcionalidades como:

- Uso de Oyentes múltiples.
- Efecto Doppler.
- Atenuación de Sonidos por distancia.
- Manipulación de Conos de sonido.
- Obstrucción y oclusión.
- Sonidos estéreos de varios canales en 3D.
- Soporte para audio 5.1 y 7.1.
- Uso de Filtros y Efectos.
- Soporte para formatos de audio como AIFF, ASF, ASX, DLS, FLAC, FSB, M3U, MID, MOD, MP2, MP3, OGG, PLS, RAW, S3M, VAG, WAV, WMA entre otros.

El uso de estas librerías le puede traer varias ventajas al proyecto: muchas de ellas son multiplataforma, se ahorra tiempo de desarrollo (y dinero), son fáciles de usar y poseen documentación, tutoriales y ejemplos, además a su funcionalidad básica se puede acceder en pocas líneas de código. Pero también hay varias desventajas importantes que destacar, por ejemplo, usualmente se requiere el pago de una licencia, no se posee control de sus características o el modo en que funcionan, se suele "pagar" por funcionalidades innecesarias que consumen recursos de memoria o CPU, y si se encuentran errores solamente es posible reportarlos y esperar al lanzamiento de una nueva versión. [12].

Conclusiones

En este capitulo se enunciaron los principales conceptos que son la base para el entendimiento del tema que se abordará en este proyecto. Además se mostraron las principales características y formatos de los sonidos así como su compresión. También se dió a conocer los tipos de tecnologías usadas y se presentaron algunas de las librerías existentes en el mundo que servirán de cimientos en el producto final.

Con el estudio realizado del tema se tienen todas las condiciones para continuar con el desarrollo de la siguiente fase del proyecto donde se analizaran las características del sistema a construir definiendo de forma más específica el objeto de estudio.

2

Capítulo 2 Descripción de la Solución Propuesta.

Introducción

En el presente capítulo se proponen soluciones técnicas para el funcionamiento del motor de sonidos y soluciones específicas para lograr su integración a la biblioteca gráfica utilizada, así como para lograr una manipulación y reproducción eficiente de los sonidos en la escena. Además se comienza a tener una visión del sistema a realizar y se dan los primeros pasos en su concepción práctica, basándose en las necesidades y dificultades.

2.1 Soluciones Técnicas.

Dada la necesidad de integración del módulo de sonido a la herramienta básica de visualización (SceneToolkit) y a otras herramientas gráficas o de multimedia, así como la creciente tendencia hacia el uso de la plataforma Linux y la alternativa de software libre, se propone el uso de la OpenAL, como API (Application Programmer Interface) a bajo nivel para el desarrollo de este módulo, teniendo en cuenta las ventajas y la facilidad de cambiar de plataforma y además la facilidad del uso de los componentes de efectos "*environmental audio extensions"* (EAX) que esta trae integrado a partir de las más recientes versiones. Se trabajará básicamente orientado a un sistema de sonido 3D posicionado, disminuyendo así los costos en cuanto a hardware necesario y haciéndolo más compatible con los sistemas de audio convencionales dado que el efecto de tridimensionalidad obtenido con este sistema alcanza un alto nivel de realismo y calidad acústica.

Después del análisis de los diferentes formatos y formas de compresión del sonido se ha planteado el uso del los formatos auto descriptivos. En un inicio se usarán específicamente el *.WAV*, .AU, pues permiten la manipulación de sus parámetros en tiempo real, algo muy necesario para lograr un efecto realista, además de que se encuentran muchos bancos de sonidos gratis con estos formatos con posibilidad de descarga gratuita, estos formatos son compatibles tanto en Windows, Linux y Macintosh. En el caso del formato OGG será usado para las pistas de sonido ambientales, bandas sonoras u otras que por su duración serían demasiado grandes si se usa el WAV, como se dijo en el capítulo anterior este formato (.Ogg) esta totalmente libre de patentes, esto será un paso más en la portabilidad del módulo hacia otras plataformas.

En este módulo se usará como estructura de datos básica la *Vector* de la *STL (Standard Template Library)* de C++, por las facilidades que brinda dicha estructura y además se le ha dado mucho uso dentro de la Herramienta SceneToolkit. El uso de esta lista de sonidos permite una cómoda manipulación y acceso, aprovechando también la gestión dinámica de la memoria que hace, optimizando de esta forma el módulo.

Muchas veces será muy útil o necesario trabajar los sonidos de forma grupal o de capas, esto permitiría modificar un parámetro como por ejemplo el volumen o la frecuencia de varios sonidos en un momento determinado sin tener que ir uno a uno, por eso el módulo implementará un sistema de tratamiento por Niveles de Jerarquía o Capas. Esto a pesar de que las librerías comerciales actuales no lo implementan directamente es muy usado sobre todo en momentos dentro de un juego o un simulador que se desea destacar un evento acústico. Por ejemplo en un juego de militar o de comandos en primera persona seria muy importante el sonido de un tiro o incluso de pasos a una distancia determinada para identificar donde se encuentra el enemigo y estos se oirán a pesar de la música y sonido ambiental. Otro ejemplo: cuando se entra a un túnel o una cueva los sonidos externos podrán ser atenuados todos sólo con modificar el parámetro al nivel o niveles en los que se encuentran.

En tiempo de ejecución pueden aparecer errores tan sencillos como un fichero de audio dañado o que no está en la ubicación esperada. Para esto el módulo implementará un sistema de tratamiento de errores, facilitando su manejo.

2.2 Modelo del Dominio.

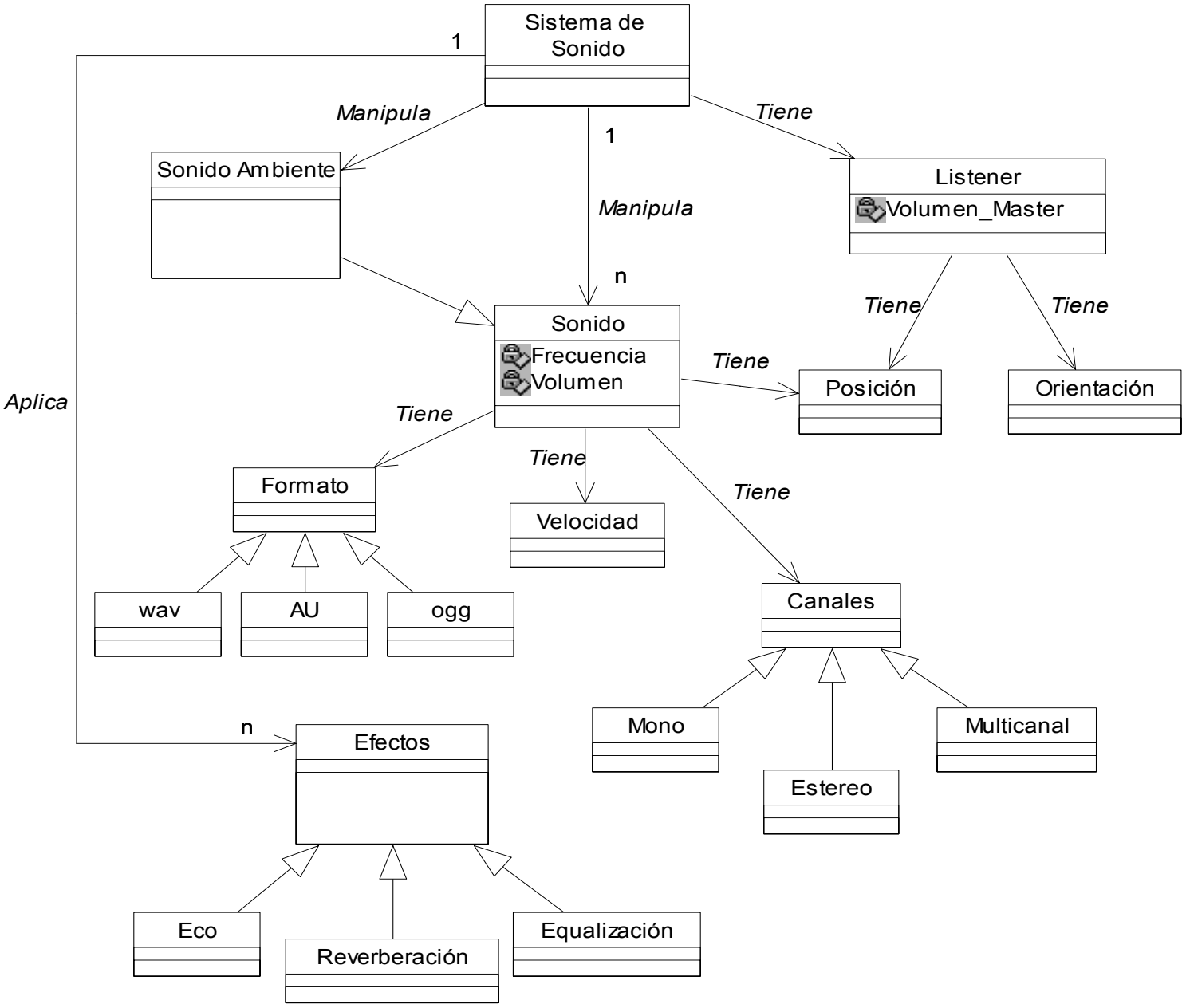

Figura 16 Modelo de Dominio.

2.2.1 Glosario de términos del dominio.

Sistema de sonido: controlador o manipulador de los sonidos y sus propiedades.

Listener: Termino anglosajón estandarizado en esta rama de audio que referencia un punto donde se centra la percepción del audio.

Sonido ambiente: Son sonidos Estéreos o Multipistas que complementan el sonido de una escena.

Sonido: fichero de sonido grabado en un ambiente real.

Formato: En el mundo de la informática, es el conjunto de reglas o especificaciones mediante las cuales se pueden organizar datos de diversa naturaleza, para poder acceder posteriormente a estos a través de los intérpretes adecuados.

Wav: Formato de sonido digitalizado que puede ser reproducido por el ordenador.

Ogg: Formato de audio de propósito general con compresión.

Posición: Ubicación vectorial en un ambiente virtual(x, y, z).

Velocidad: Rapidez de cambio de la posición.

Orientación: componente del estado geométrico de los cuerpos, que describe la variación en ángulo respecto a los ejes X, Y y Z.

Mono: Sistema de registro y reproducción que utiliza solo un canal de información sonora.

Multipista / MultiCanal: Múltiples canales de audio (generalmente más de dos), que se reproducen por diferentes Bocinas para obtener un efecto de surround.

Estéreo: Sistema de registro y reproducción que utiliza dos canales de información sonora: izquierdo y derecho.

Eco: La repetición de un sonido retardada en el tiempo, un mínimo de 50 milisegundos en relación al sonido original.

Ecualización: Componente de audio que resalta o atenúa ciertas frecuencias de sonido (bandas) de la señal original; existen varios tipos como gráficos, paramétricos, gráficos electrónicos.

Reverberación: es un fenómeno de la reflexión del sonido, consistente en la prolongación del mismo una vez se ha extinguido la fuente debido al reflejo de las vibraciones por el choque con un obstáculo cercano.

2.3 Reglas del negocio

- Los sonidos que se cargan serán de formato *wav*, *au* y *ogg*.
- Cada fichero se carga una sola vez.
- Instanciar un fichero cargado varias veces.
- Posición en datos vectoriales respecto al mundo.
- Los ficheros de sonidos antes de ser utilizados deben pasar previamente por un proceso de edición.
- Los efectos de sonidos *wav* y *au* que serán utilizados como sonidos 3D deben ser mono. Solo serán estereo los sonidos destinados a ambientes o bandas musicales, que serán codificados como *OGG*.
- La calidad de la reproducción depende directamente de la calidad de la grabación original, el procesamiento solo será en parámetros como la frecuencia o el volumen.
- La variación de los parámetros no afectará directamente el fichero de sonido.

2.4 Requerimientos del Sistema.

2.4.1 Requisitos funcionales.

1. Crear un contexto de sonido.
    1.1- Definir factor Doppler
    1.2- Definir Velocidad Sonido.
    1.3- Definir Modelo de Distancia.
2. Cargar fichero de sonido.
    2.1- Cargar datos del fichero.
    2.2- Crear *Buffer*.
    2.3- Cargar Datos al *Buffer*.
3. Crear Fuente Sonora.
    3.1- Asignar *Buffer* a una fuente.
    3.2- Definir el rango acción al de una fuente de sonido.
    3.3- Conocer el estado en que se encuentra una fuente sonido.
    3.4- Ejecutar fuente sonido.
    3.5- Parar fuente sonido.
    3.6- Darle pausa a una fuente sonido.
    3.7- Variar frecuencia de fuente de sonido.
    3.8- Variar volumen de fuente de sonido.
    3.9- Variar posición de una fuente de sonido.
    3.10- Variar velocidad de una fuente de sonido.
    3.11- Definir el ángulo de incidencia de una fuente de sonido.
    3.12- Definir una fuente de sonido como relativa.
    3.13- Definir si la fuente se ejecutara de forma cíclica.
    3.14- Asignar *Slot* de efectos a una fuente de sonido.
    3.15- Deshabilitar *Slot* de efecto a fuente de sonido.

3.16- Definir parámetro de Atenuación de Sonido (RollOffFactor).

4. Agregar Fuente de Sonido a una capa o nivel.
   - 4.1- Modificar volumen a una capa o nivel.
   - 4.2- Modificar frecuencia a una capa o nivel.
   - 4.3- Buscar Fuentes de una capa.
   - 4.4- Ejecutar todas las fuentes de sonidos de un nivel o capa.
   - 4.5- Detener todas las fuentes de sonidos de un nivel o capa.
   - 4.6- Pausar todas las fuentes de sonidos de un nivel o capa.
5. Crear *Listener*
   - 5.1- Variar posición del *Listener*.
   - 5.2- Variar orientación del *Listener*.
   - 5.3- Variar velocidad de *Listener*.
   - 5.4- Variar volumen del *Listener* (volumen master).
   - 5.5- Definir factor de Unidades de distancia al *Listener*.
6. Crear un *Slot* de Efectos.
   - 6.1- Definir efecto.
   - 6.2- Asignar efecto a un *Slot*.
   - 6.3- Asignar propiedades de efecto predefinidas.
   - 6.4- Definir Filtro.
   - 6.5- Asignar Filtro a un *Slot*.
7. Destruir Contexto.
8. Destruir *Buffer*.
9. Destruir Fuentes.

2.4.2 Requisitos no funcionales.

Usabilidad.

- Fácil para el programador.

Rendimiento.

- Haga uso óptimo de la memoria.

Portabilidad.

- Multiplataforma.

Ayuda y documentación en línea.

- Va a contar con Ayuda y especificaciones de uso.

2.5 Diagrama de Casos de Uso.

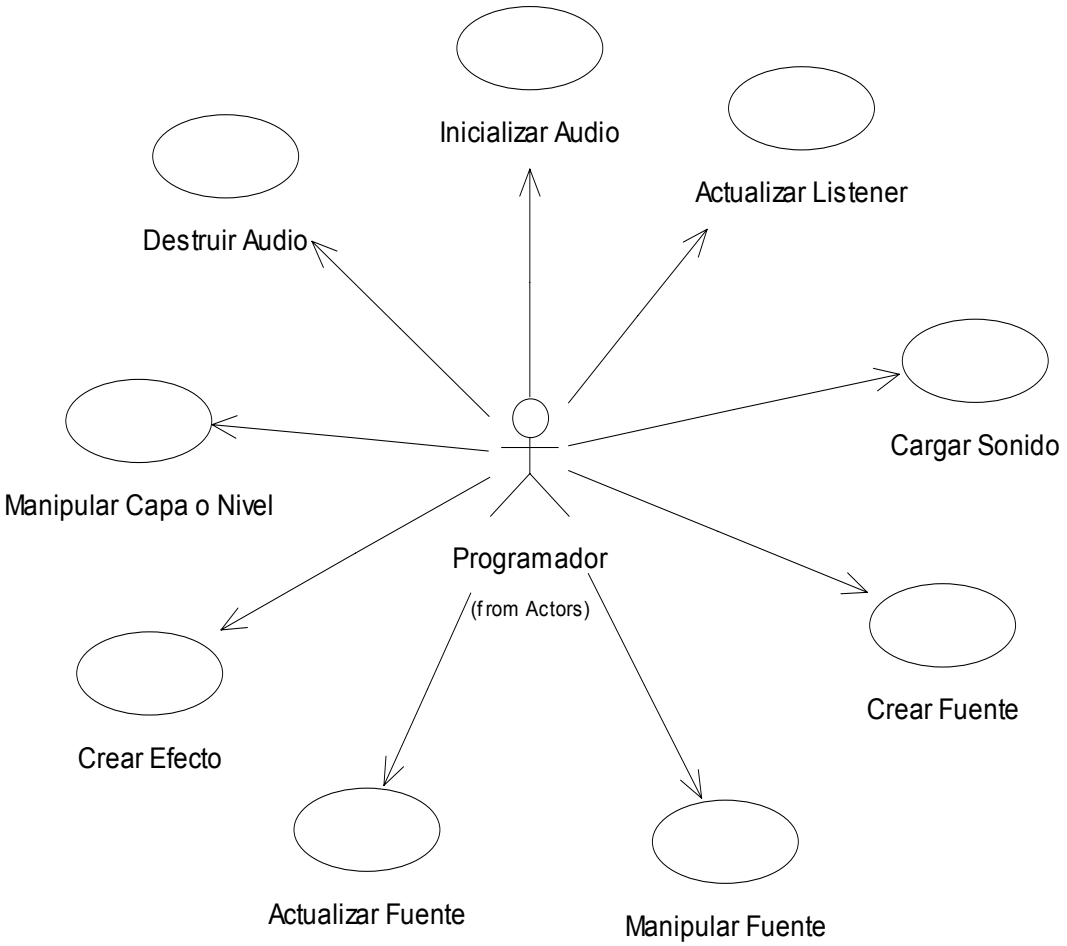

Figura 17 Diagrama de Casos de Uso.

2.6 Expansión de los Casos de Uso.

2.6.1 Definición de los actores.

Tabla 4 Definición de los actores.

Actores	**Justificación**
Programador	Es quien se beneficia con las funcionalidades que brinda el módulo de sonido y es el que de manera directa interactúa con el sistema

2.6.2 Listado de casos de uso

Tabla 5 CU-1 Inicializar Audio.

CU-1	**Inicializar Audio**
Actor	Programador
Descripción	Inicializa el hardware, crea el contexto y el *Listener* inicializando parámetros necesarios, inicializa los *Slot* de efectos,
Referencia	

Tabla 6 CU-2 Actualizar Listener.

CU-2	**Actualizar *Listener***
Actor	Programador
Descripción	Actualiza la velocidad, posición, orientación y volumen del *Listener*.
Referencia	

Tabla 7 CU-3 Cargar Sonido.

CU-3	**Cargar Sonido**
Actor	Programador
Descripción	Se Cargan los datos del fichero de sonido, se crea el *buffer*, y se cargan los datos a este.
Referencia	

Tabla 8 CU-4 Crear Fuente.

CU-4	**Crear Fuente**
Actor	Programador
Descripción	Crear fuente de sonido asignándole un *buffer* existente, y se le asigna el identificador del nivel al que pertenece dicha fuente.
Referencia	

Tabla 9 CU-5 Manipular Fuente.

CU-5	**Manipular Fuente**
Actor	Programador
Descripción	Trabajo con los parámetros del sonido, ejecutar, parar, volumen, frecuencia …
Referencia	

Tabla 10 CU-6 Actualizar Fuente.

CU-6	**Actualizar Fuente**
Actor	Programador
Descripción	Actualiza la posición y velocidad de la fuente de sonido dentro del mundo.
Referencia	

Tabla 11CU-7 Crear Efecto.

CU-7	**Crear Efecto**
Actor	Programador
Descripción	Asigna un Efecto o Filtro a un *Slot* existente.
Referencia	

Tabla 12 CU-8 Manipular Capa o Nivel.

CU-8	**Manipular capa o Nivel**
Actor	Programador
Descripción	Facilitar el manejo de varios sonidos de forma conjunta, una vez asignados a un nivel.
Referencia	

Tabla 13 CU-9 Destuir Audio.

CU-9	**Destruir Audio**
Actor	Programador
Descripción	Destruye *buffer*s, fuentes, *Slot*s de efectos, y libera los recursos de hardware.
Referencia	

2.6.3 Casos de uso por ciclo.

Tabla 14 Casos de uso por ciclo.

Cód	**Nombre de caso de uso**	**Paquete**	**Justificación de la selección.**
CU-1	Inicializar Audio	1	
CU-2	Actualizar *Listener*	1	
CU-3	Cargar Sonido	1	
CU-4	Crear Fuente	1	
CU-5	Manipular Fuente	1	
CU-6	Actualizar Fuente	1	
CU-7	Crear Efecto	1	
CU-8	Manipular capa o Nivel	1	
CU-9	Destruir Audio	1	

2.6.4 Casos de uso expandidos.

Tabla 15 Expansión CU-1.

Caso de uso		
CU-1	Inicializar Audio	
Propósito	Inicializar dispositivos de audio	
Actores: Programador.		
Resumen: se inicia cuando el programador solicita inicializar el dispositivo de sonido y crea el contexto, el *Listener* y los *Slot* de Efectos, se inicializan parámetros: factor Doppler, Velocidad de Sonido, Modelo de Distancia, Define factor de Unidades de distancia al *Listener*. El caso de uso termina cuando se el dispositivo esta inicializado.		
Referencias	1, 1.1, 1.2, 1.3, 5, 5.5, 6	
Acción del actor		**Respuesta del sistema**
1- Solicita inicializar el audio.		2- Se crea el Contexto.
		3- Inicializa factor doppler.
		4- inicializa Velocidad de Sonido.
		5- Define Modelo de Distancia.
		6- Se Crean los *Slot* de Efectos
		7- Se Crea el *Listener*.
		8- Se Define factor de Unidades de distancia al *Listener*
Precondiciones		
Poscondiciones		• Queda inicializado el dispositivo de audio.

Tabla 16 Expansión CU-2.

Caso de uso	
CU-2	Actualizar *Listener*
Propósito	Actualizar posición , velocidad, orientación, y volumen del *Listener*
Actores: Programador.	
Resumen: se inicia el caso de uso cuando el programador solicita actualizar el Listener. Entonces se procede a variar la posición, orientación, velocidad o volumen. El caso de uso termina cuando se halla actualizado el estado del *Listener*.	
Referencias	5.1, 5.2, 5.3, 5.4

Acción del actor	**Respuesta del sistema**
Sección: Posición.	
1- Se solicita actualizar la posición pasando el valor de x, y, z (coordenadas).	2- Se varía la Posición del *Listener* a los nuevos valores.
Sección: Velocidad	
1- Se solicita actualizar la Velocidad pasando el valor de x, y, z (Vector de Velocidad).	2- Se varía la Velocidad del *Listener* a los nuevos valores.
Sección: Orientación	
1- Se solicita actualizar la Orientación pasando vectores de Orientación y Dirección (Ox,Oy,Oz);(Upx,Upy,Upz)	2- Se varía la Orientación del *Listener* a los nuevos valores.
Sección: Volumen	
1- Se solicita actualizar el Volumen pasando el nuevo valor.	2- Se varía el Volumen del *Listener* al nuevo valor.
Precondiciones	• Que se haya creado el *Listener*
Poscondiciones	• El *Listener* queda Actualizado con los nuevos Valores.

Tabla 17 Expansión CU-3.

Caso de uso	
CU-3	Cargar Sonido
Propósito	Cargar un sonido deseado.
Actores: Programador.	
Resumen: El caso de uso se inicia cuando el programador solicita cargar un sonido, entonces se procede a la cargar los datos del fichero de sonido, se crea el *buffer* y se cargan los datos a este. El caso de uso termina cuando el *buffer* esta listo para usarse.	
Referencias	2, 2.1, 2.2, 2.3

Acción del actor	**Respuesta del sistema**
1- Solicita cargar un sonido	2- Se cargan los Datos del Fichero
	3- Se Crea un nuevo *Buffer*
	4- Se cargan los Datos al *Buffer*.
Precondiciones	
Poscondiciones	• Quedan cargados los datos en el *Buffer*.

Tabla 18 Expansión CU-4.

Caso de uso	
CU-4	Crear Fuente
Propósito	Crear fuente de sonido a partir de un *buffer*
Actores: Programador.	
Resumen: El CU se inicia cuando se va a crear una fuente de sonido, se le asigna un identificador de una capa a la que se quiere que pertenezca, se le asigna un *buffer* existente, termina cuando ha sido creada la fuente.	
Referencias	3, 3.1, 4

Acción del actor	**Respuesta del sistema**
1- Solicita crear fuente pasando el *buffer* que se le asignara y el Id del nivel.	2- se crea la Fuente
	3- Se le asigna el *buffer*.
	4- Se le asigna el Id del nivel.
Precondiciones	• Que se haya creado un *Buffer*.
Poscondiciones	• Fuente lista para reproducirse.

Tabla 19 Expansión CU-5.

Caso de uso	
CU-5	Manipular Fuente
Propósito	Trabajo con los parámetros de la fuente de sonido
Actores: Programador.	
Resumen: se inicia el caso de uso cuando el programador solicita cambiar parámetros o estados del sonido. Entonces se procede a variar los parámetros deseados del sonido, el caso de uso termina cuando se varíen los parámetros deseados.	
Referencias	3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16
Acción del actor	**Respuesta del sistema**
Sección: Volumen.	
1- Se solicita variar el volumen de la fuente de sonidos pasando el valor del parámetro volumen.	2- Se varía el volumen de la fuente de sonidos.
Sección: Frecuencia.	
1- Se solicita variar la frecuencia de la fuente de sonidos pasando el valor del parámetro frecuencia.	2- Se varía la frecuencia de la fuente de sonidos.
Sección: Play.	

1- Se solicita ejecutar la fuente de sonidos.	2- Se ejecuta la fuente de sonido.
Sección: Stop.	
1- Se solicita detener la reproducción de la fuente de sonido.	2- Se detiene la reproducción de la fuente de sonido.
Sección: Pausa.	
1- Se solicita pausar la reproducción de la fuente de sonido.	2- Se ponen en pausa la reproducción de la fuente de sonido.
Sección: Angulo.	
1- Se desea definir el ángulo de incidencia de la fuente de sonido pasando el ángulo como parámetro.	2- Se establece el valor del ángulo de incidencia a la fuente.
Sección: Relativa.	
1- Se desea poner relativa una fuente de sonido.	2- Se pone relativa la fuente de sonido.
Sección: Cíclica.	
1- Se desea que la fuente de sonido se reproduzca de forma cíclica.	2- Se establece el parámetro de la fuente para reproducción cíclica.
Sección: Rango.	
1- Se desea definir el rango de acción de la fuente y se le pasa como parámetros la distancia mínima y distancia máxima.	2- Se define el rango de acción de la fuente de sonido.
Sección: Estado.	
1-Se desea conocer el estado en que se encuentra la fuente de sonido.	2- Se devuelve el estado en que se encuentra la fuente.
Sección: RollOfFactor	
1-Se desea definir el parámetro de atenuación del sonido (RollOfFactor)	2- Se define el RollOFFactor.
Sección: DistanciaReferencia	

1-Se desea definir la Distancia de Referencia (ReferenDistanc)	2- Se define la ReferenDistanc.
Sección: Asignar Efecto	
1- Se solicita asignarle un efecto a la fuente.	2- Se Asigna el *Slot* de Efecto a la fuente.
Sección: Deshabilitar Efecto	
1- se desea desactivar el efecto a la fuente	2-Se deshabilita el *Slot* a la fuente.
Precondiciones	• Inicializado el Sonido. • Fuente creada. • Efecto creado. (Sección: Asignar Efecto, Deshabilitar Efecto)
Poscondiciones	• La fuente toma el nuevo valor asignado.

Tabla 20 Expansión CU-6.

Caso de uso	
CU-6	Actualizar Fuente
Propósito	Actualizar posición y velocidad de la fuente.
Actores: Programador.	
Resumen: se inicia el caso de uso cuando el programador solicita actualizar el estado del sonido. Entonces se procede a variar o no los parámetros posición o velocidad de la fuente, el caso de uso termina cuando se halla actualizado el estado de la fuente de sonido.	
Referencias	3.9, 3.10
Acción del actor	**Respuesta del sistema**
Sección: Posición.	
1- Se solicita posicionar una fuente de sonido pasando el valor de x, y, z (coordenadas).	2- Se posiciona la fuente de sonido en las coordenadas deseada.

Sección: Velocidad	
1- Se solicita actualizar los parámetros de velocidad de la fuente.	2- Se asignan los nuevos parámetros de velocidad a la fuente.
Precondiciones	• Inicializado el Sonido. • Fuente creada.
Poscondiciones	• La fuente toma el nuevo valor asignado.

Tabla 21 Expansión CU-7.

Caso de uso	
CU-7	Crear Efecto
Propósito	Crea un efecto y lo asigna un Efecto o Filtro a un *Slot* existente.
Actores: Programador.	
Resumen: se inicia el caso de uso cuando el programador solicita crear un Efecto o un filtro, Entonces se procede a crearlo y asignarlo a un *Slot* existente, el caso de uso termina cuando queda asignado al *Slot* el efecto o filtro deseado.	
Referencias	6.1, 6.2, 6.3, 6.4, 6.5

Acción del actor	**Respuesta del sistema**
Sección: Filtro	
1- Se solicita crear un Filtro pasando el tipo (EFilterType)	2- Se crea el Filtro.
	3- Se le asigna a un *Slot*.
Sección: Efecto	
1- Se solicita crear un Efecto	2- Se crea el Efecto.
	3- Se le asigna a un *Slot*.
4- Se le pasan las propiedades al efecto	5- Se le asignan las propiedades al Efecto.

Precondiciones	• Inicializado el Audio.
Poscondiciones	• Retorna un *Slot* con un Filtro o Efecto cargado

Tabla 22 Expansión CU-8.

Caso de uso		
CU-8	Manipular Capa o Nivel	
Propósito	Facilitar el manejo de varios sonidos de forma conjunta, una vez asignados a un nivel.	
Actores: Programador.		
Resumen: Se inicia el caso de uso cuando el programador solicita el trabajo con una capa o nivel de sonido, entonces se procede a variar el parámetro deseado de todos los sonidos miembros de la capa. El caso de uso finaliza cuando se varía el parámetro de la capa o nivel.		
Referencias	4.2, 4.3, 4.4, 4.5, 4.6, 4.7	
Acción del actor		**Respuesta del sistema**
Sección: Volumen.		
1- Se solicita variar el volumen de una capa o nivel de sonidos pasando el valor del factor volumen.		2- Se buscan todas las fuentes de sonido que pertenecen a la capa.
		3- Se varía el volumen de todos los sonidos de la capa.
Sección: Frecuencia.		
1- Se solicita variar la frecuencia de una capa o nivel de sonidos pasando el valor del factor frecuencia.		2- Se buscan todas las fuentes de sonido que pertenecen a la capa.
		3- Se varía la frecuencia de todos los sonidos de la capa.
Sección: Play.		
1- Se solicita ejecutar todos los sonidos de la		2- Se buscan todas las fuentes de sonido que

capa.	pertenecen a la capa.
	3- Se ejecutan todos los sonidos de la capa.
Sección: Stop.	
1- Se solicita detener todos los sonidos de la capa.	2- Se buscan todas las fuentes de sonido que pertenecen a la capa.
	3- se detienen todos los sonidos de la capa.
Sección: Pausa.	
1- Se solicita pausar todos los sonidos de la capa.	2- Se buscan todas las fuentes de sonido que pertenecen a la capa.
	3- Se ponen en pausa todos los sonidos de la capa.
Precondiciones	• Fuentes estén Creadas.
Poscondiciones	• Parámetros modificados del grupo de Fuentes.

Tabla 23 Expansión CU-9.

Caso de uso		
CU-9	Destruir Audio	
Propósito	Destruye *buffer*s, fuentes, *Slot*s de efectos, y libera los recursos de hardware.	
Actores: Programador.		
Resumen: se inicia cuando el programador desea liberar el dispositivo de audio y destruir los *buffer* y fuentes existentes. Termina cuando se liberó el dispositivo de audio y todos los recursos que este ocupaba.		
Referencias	7, 8, 9	
Acción del actor		**Respuesta del sistema**
1- Solicita Destruir Audio.		2- Se destruyen la Fuentes
		3- Se Destruye el *Buffer*.

	4- Se destruye Contexto.
Precondiciones	• Audio inicializado.
Poscondiciones	• Liberan los recursos de memoria y hardware.

Conclusiones

Con este capítulo quedan sentadas las bases técnicas por las que se regirá el sistema. Además se definió qué es exactamente lo que espera el usuario con este sistema. Para ello quedaron establecidos los requisitos funcionales de este, y descritos los casos de uso que le permitirán a su futuro usuario obtener los resultados. Partiendo de todo esto se diseñará una estructura de clases en correspondencia con las técnicas citadas y para lograr los objetivos del proyecto.

3

Capítulo 3 Diseño e Implementación.

Introducción

En el presente capítulo se presenta ya el diseño del sistema propuesto, con un diagrama de clases, donde se definen las responsabilidades de estas y sus relaciones. Además se presentan otros artefactos involucrados en el diseño como los diagramas de secuencia por casos de uso, facilitando con esto el entendimiento del módulo.

Esta etapa del proyecto además el paso del diseño de clases a la creación de componentes físicos, que se traducen en ficheros .cpp correspondiente a la implementación en C++, y se enuncian los estándares para la nomenclatura y el control de versiones que se usarán.

3.1 Diagrama de Diseño de clases.

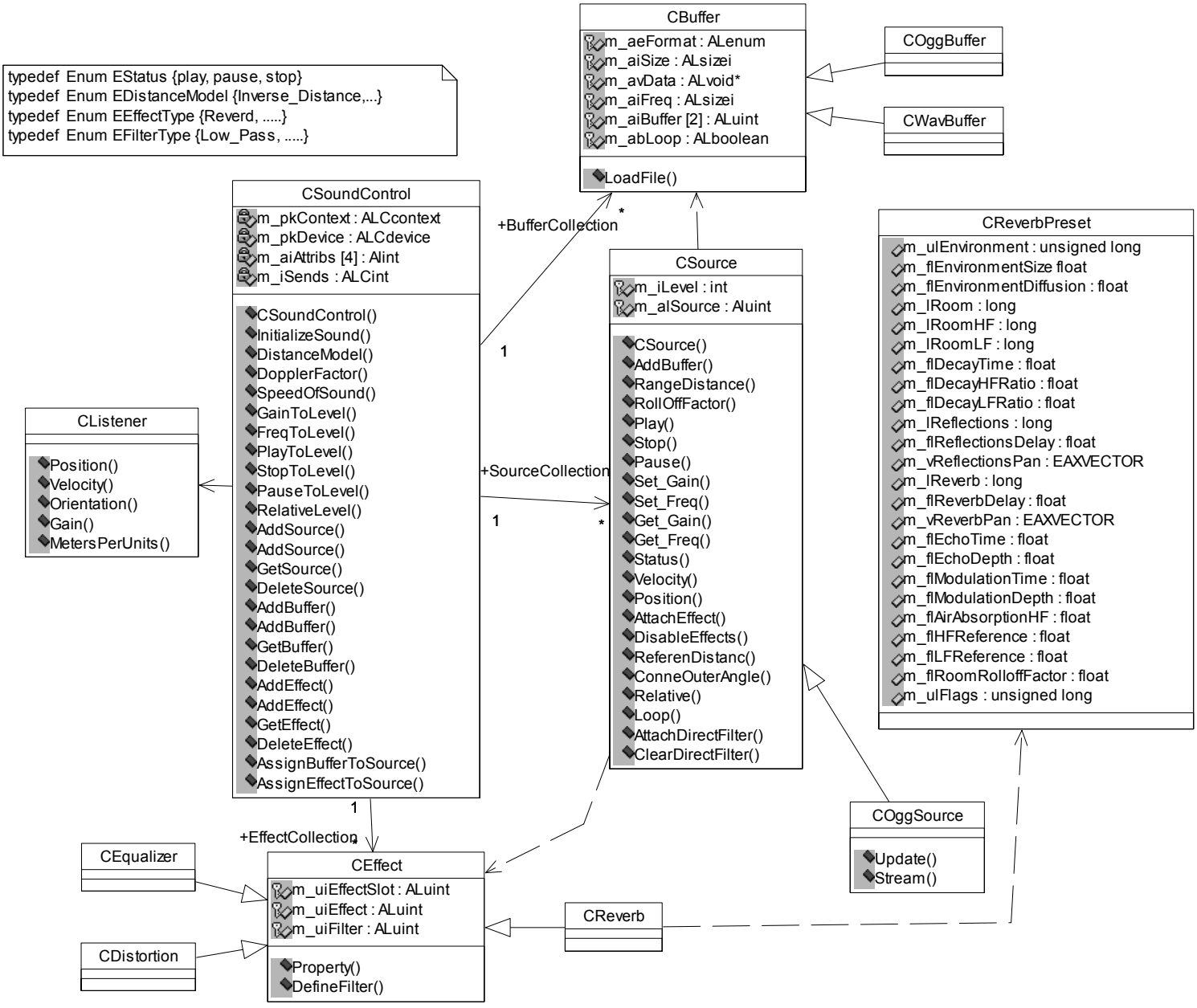

Figura 18 Diagrama de Clases

3.2 Descripción de las clases.

Tabla 24 Descripción CSoundControl.

Nombre: CSoundControl	
Controladora	
Atributo	**Tipo**
m_pkContext	ALCcontext
m_pkDevice	ALCdevice
m_aiAttribs[4]	Alint
m_iSends	ALCint
Para cada responsabilidad:	
Nombre:	InitializeSound()
Descripción:	Inicializa los dispositivos de sonido.
Nombre:	DistanceModel(arg_eDistance : EDistanceModel)
Descripción:	Define el modelo de distancia a usar.
Nombre:	DopplerFactor(arg_fDoppler : float)
Descripción:	Define la constante de factor doppler.
Nombre:	SpeedOfSound(arg_fSpeed : float)
Descripción:	Define la constante de velocidad de sonido.
Nombre:	GainToLevel(arg_iLevel : int, arg_fGain : float)
Descripción:	Aumenta o disminuye el volumen de un nivel en un porciento definido por arg_fGain.
Nombre:	FreqToLevel(arg_iLevel : int, arg_fFreq : float)
Descripción:	Aumenta o disminuye la frecuencia de un nivel en un porciento definido por arg_fFreq.
Nombre:	PlayToLevel(arg_iLevel : int)
Descripción:	Ejecuta todos los sonidos miembros de la capa.
Nombre:	StopToLevel(arg_iLevel : int)

Descripción:	Detiene la ejecución de todos los sonidos miembros de la capa.
Nombre:	PauseToLevel(arg_iLevel : int)
Descripción:	Pone en pausa a todos los sonidos miembros de la capa.
Nombre:	AddSource(arg_pkSource : CSource*)
Descripción:	Agrega una fuente a la lista y devuelve su posición.
Nombre:	GetSource(arg_uiPos : unsigned int)
Descripción:	Retorna un puntero a una fuente.
Nombre:	DeleteSource(arg_uiPos : unsigned int)
Descripción:	Elimina una fuente de la lista.
Nombre:	AddBuffer(arg_pkBuffer : CBuffer*)
Descripción:	Adiciona un bufer a la lista y retorna su posición.
Nombre:	GetBuffer(arg_uiPos : unsigned int)
Descripción:	Retorna un puntero a un *buffer*.
Nombre:	DeleteBuffer(arg_uiPos : unsigned int)
Descripción:	Elimina un *buffer* de la lista.
Nombre:	AddEffect(arg_pkEffect : CEffect*)
Descripción:	Adiciona un efecto a la lista y retorna su posición.
Nombre:	GetEffect(arg_uiPos : unsigned int)
Descripción:	Retorna un puntero a un efecto.
Nombre:	DeleteEffect(arg_uiPos : unsigned int)
Descripción:	Elimina un efecto de la lista.
Nombre:	AssignBufferToSource(arg_iBufferIndex: int, arg_iSourceIndex : int, arg_bLoop : bool, arg_bRelative : bool)
Descripción:	Asigna un *buffer* a una fuente miembros ambos de la lista y se define si será relativa y cíclica.
Nombre:	AssignEffectToSource(arg_iEfectIndex: int, arg_iSourceIndex : int)
Descripción:	Asignar un efecto a una fuente miembros ambos de la lista.

Tabla 25 Descripción CSource.

Nombre: CSource	
Controladora	
Atributo	**Tipo**
m_iLevel	int
m_alSource	Aluint
Para cada responsabilidad:	
Nombre:	AddBuffer(arg_pkBuffer : CBuffer*, arg_BLoop : bool, arg_bRelative : bool)
Descripción:	Adiciona un *buffer* a la fuente y define si será relativa y cíclica.
Nombre:	RangeDistance(arg_fMaxDist : float, arg_fMinDist : float)
Descripción:	Define la distancia mínima y máxima a la que se oirá la fuente.
Nombre:	RollOffFactor(arg_RollOffFactor : float)
Descripción:	Factor para el modelo de atenuación por distancia.
Nombre:	Play()
Descripción:	Ejecuta la fuente.
Nombre:	Stop()
Descripción:	Detiene la fuente.
Nombre:	Pause()
Descripción:	Pone en pausa a la fuente.
Nombre:	Set_Gain(arg_fGain : float)
Descripción:	Le pasa el volumen a la fuente.
Nombre:	Set_Freq(arg_fFreq : float)
Descripción:	Le pasa la frecuencia a la fuente.
Nombre:	Get_Gain()
Descripción:	Devuelve el volumen de la fuente.
Nombre:	Get_Freq()
Descripción:	Devuelve la frecuencia de la fuente.
Nombre:	Status()
Descripción:	Devuelve el estado en que se encuentra la fuente (play, stop, pausa).

Nombre:	Velocity(arg_fX : float, arg_fY : float, arg_fZ : float)
Descripción:	Le pasa la velocidad a la fuente.
Nombre:	Position(arg_fX : float, arg_fY : float, arg_fZ : float)
Descripción:	Le pasa la posición a la fuente.
Nombre:	AttachEffect(arg_CEffect : CEffect*)
Descripción:	Le asigna el efecto a la fuente.
Nombre:	DisableEffects()
Descripción:	Deshabilita el efecto a la fuente.
Nombre:	ReferenDistanc(arg_fRefernce : float)
Descripción:	Es la distancia efectiva donde se escuchará la fuente, a partir de esta comienza la atenuación.
Nombre:	ConneOuterAngle(arg_fAngle : float)
Descripción:	Le pasa la ampitud del ángulo emision de la fuente.
Nombre:	Relative(arg_bRelative : bool)
Descripción:	Pone relativa o no a la fuente.
Nombre:	Loop(arg_bLoop : bool)
Descripción:	Pone cíclica o no a la fuente.
Nombre:	AttachDirectFilter(arg_efilterType: EfilterType, arg_fGain: float, arg_fGainHF: float)
Descripción:	Activa un filtro por el *slot* directo a la fuente, configurando el tipo y los parámetros de ganancia.
Nombre:	ClearDirectFilter()
Descripción:	Desactiva el filtro por el *slot* directo.

Tabla 26 Descripción CListener

Nombre: CListener	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Position(arg_fX : float, arg_fY : float, arg_fZ : float)
Descripción:	Le pasa la posición donde va a estar ubicado el *listener*.
Nombre:	Velocity(arg_fX : float, arg_fY : float, arg_fZ : float)
Descripción:	Le pasa la velocidad a la que se mueve el *listener*.
Nombre:	Orientation(arg_fX: float, arg_fY: float, arg_fZ: float, arg_fUpX : float, arg_fUpY : float, arg_fUpZ : float)
Descripción:	Le pasa orientación hacia donde va a estar mirando el *listener*.
Nombre:	Gain(arg_fGain : float)
Descripción:	Volumen general.
Nombre:	MetersPerUnits(arg_fMerterPerUnits : float)
Descripción:	Factor de conversión de unidades de distancia.

Tabla 27 Descripción CBuffer

Nombre: CBuffer	
Entidad	
Atributo	**Tipo**
m_aeFormat	ALenum
m_aiSize	ALsizei
m_avData	ALvoid*
m_aiFreq	ALsizei
m_aiBuffer [2]	ALuint
m_abLoop	ALboolean

Para cada responsabilidad:	
Nombre:	LoadFile(arg_File : char*)
Descripción:	Carga el fichero de sonido según el formato y lo almacena en el *buffer*.

Tabla 28 Descripción CEffect

Nombre: CEffect	
Entidad	
Atributo	**Tipo**
m_uiEffectSlot	ALuint
m_uiEffect	ALuint
m_uiFilter	ALuint
Para cada responsabilidad:	
Nombre:	Property(arg_RevProp : CReverbPreset) : void
Descripción:	Asigna los parámetros al efecto.
Nombre:	DefineFilter(arg_efilterType: ALenum, arg_fGain: float, arg_fGainHF: float)
Descripción:	Define un filtro como efecto configurando el tipo y los parámetros de ganancia.

Tabla 29 Descripción CReverbPreset

Nombre: CReverbPreset	
Entidad	
Atributo	**Tipo**
m_ulEnvironment	unsigned long
m_flEnvironmentSize	float
m_flEnvironmentDiffusion	float
m_lRoom	long
m_lRoomHF	long

m_flDecayTime	float
m_flDecayHFRatio	float
m_flDecayLFRatio	float
m_lReflections	long
m_flReflectionsDelay	float
m_vReflectionsPan	EAXVECTOR
m_lReverb	long
m_flReverbDelay	float
m_vReverbPan	EAXVECTOR
m_flEchoTime	float
m_flEchoDepth	float
m_flModulationTime	float
m_flModulationDepth	float
m_flAirAbsorptionHF	float
m_flHFReference	float
m_flLFReference	float
m_flRoomRolloffFactor	float
m_ulFlags	unsigned long
Para cada responsabilidad:	
Nombre:	
Descripción:	

3.3 Diagramas de Secuencia.

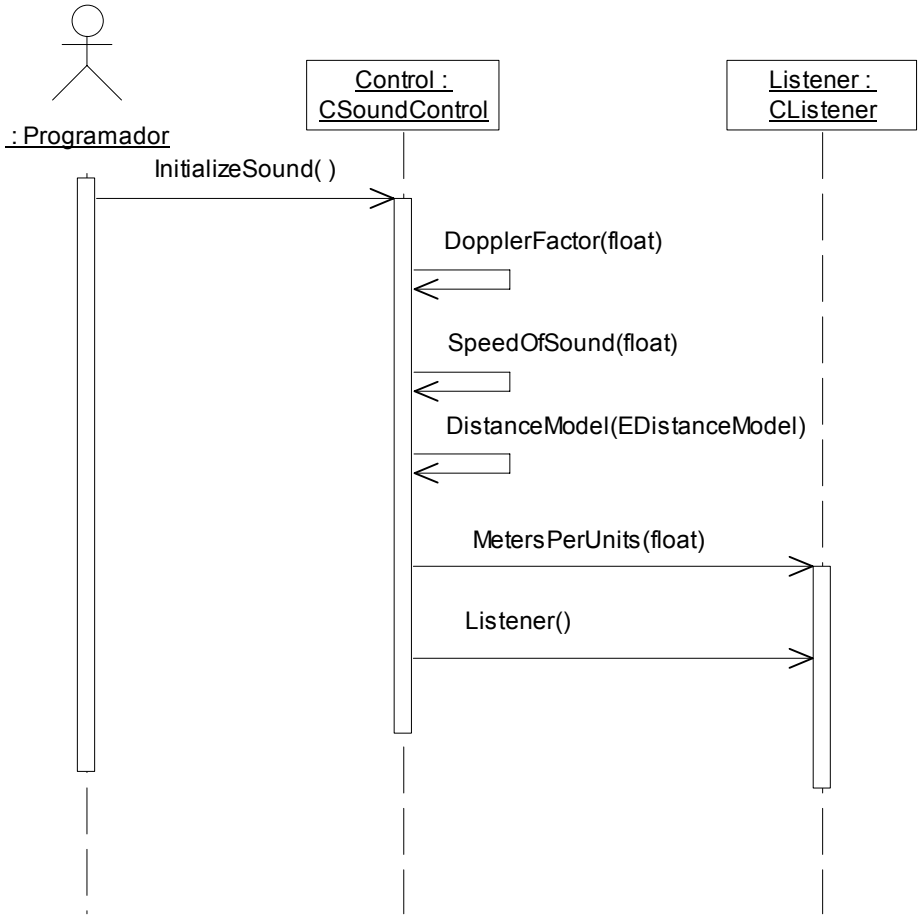

Figura 19 Diagrama de Secuencia "Inicializar Audio"

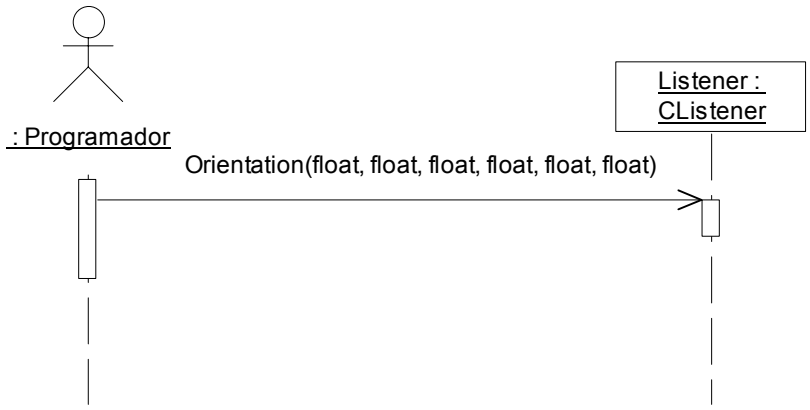

Figura 20 Diagrama de Secuencia "Actualizar Listener" (Sesión Orientación).

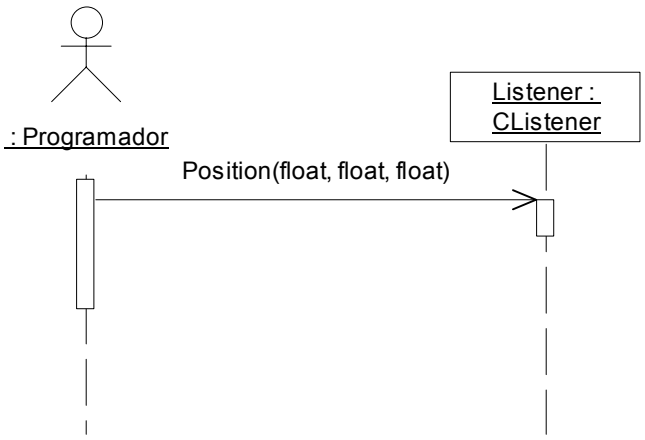

Figura 21 Diagrama de Secuencia "Actualizar Listener" (Sesión Posición).

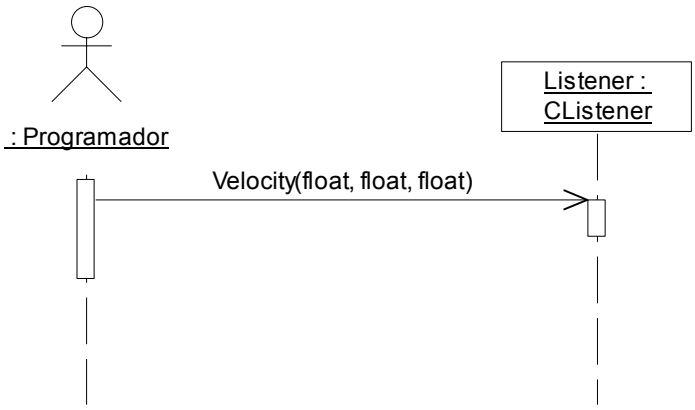

Figura 22 Diagrama de Secuencia "Actualizar Listener" (Sesión Velocidad).

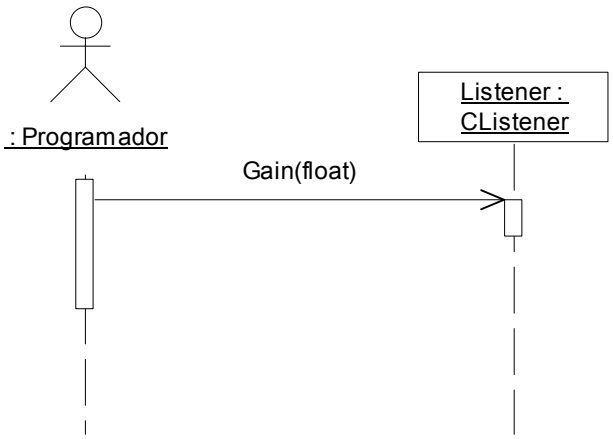

Figura 23 Diagrama de Secuencia "Actualizar Listener" (Sesión Volumen).

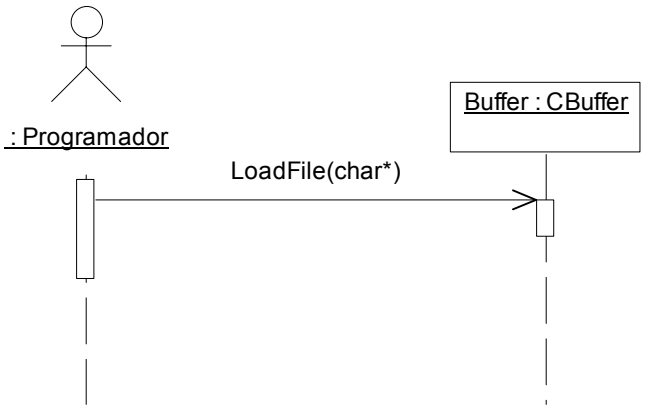

Figura 24 25 Diagrama de Secuencia "Cargar Sonido".

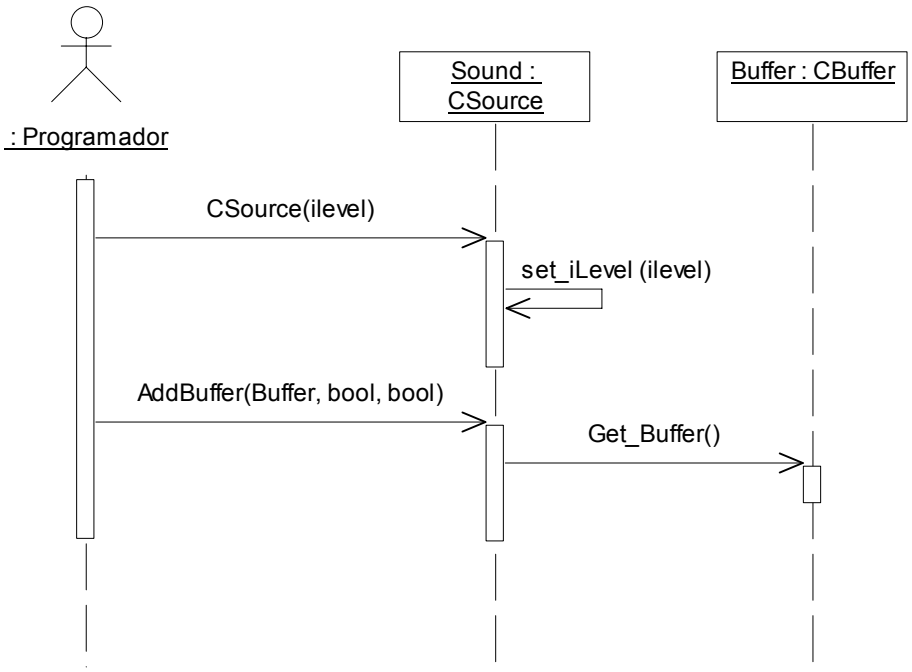

Figura 26 Diagrama de Secuencia "Crear Fuente".

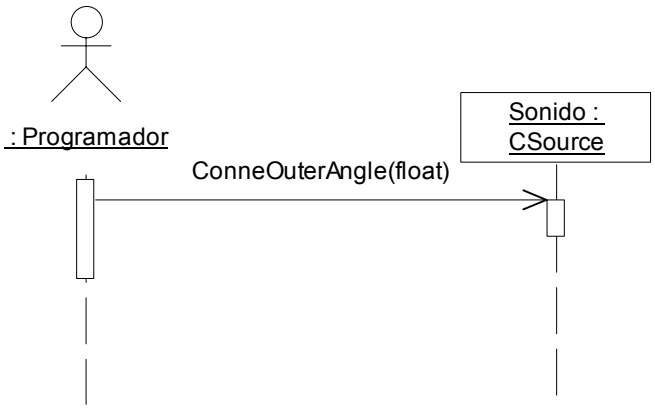

Figura 27 Diagrama de Secuencia "Manipular Fuente" (Sesión Ángulo).

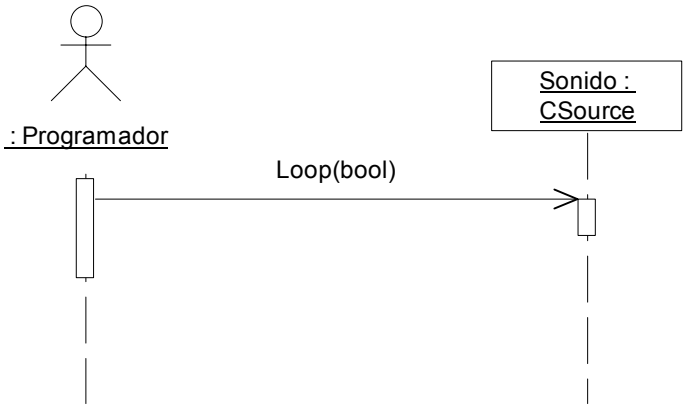

Figura 28 Diagrama de Secuencia "Manipular Fuente" (Sesión Cíclica).

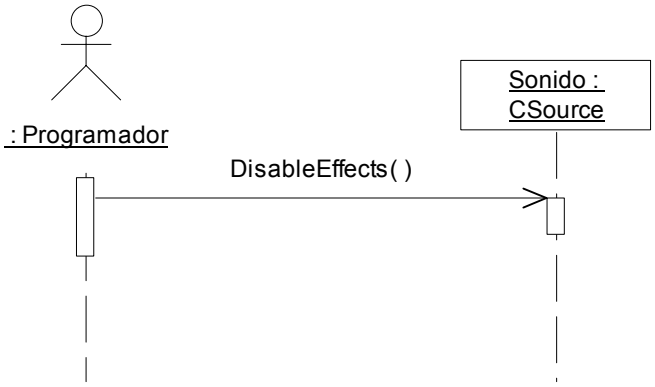

Figura 29 Diagrama de Secuencia "Manipular Fuente" (Sesión Deshabilitar Efecto).

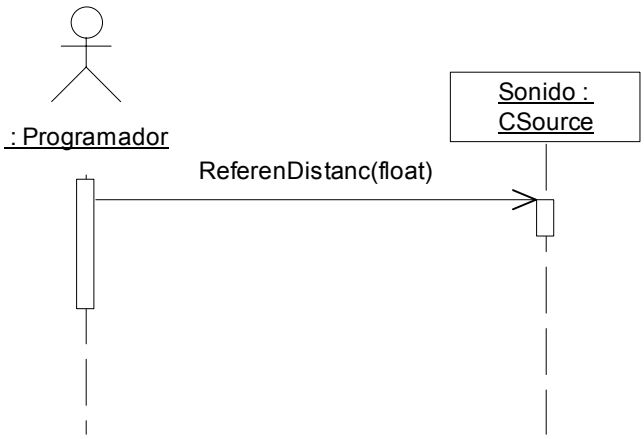

Figura 30 Diagrama de Secuencia "Manipular Fuente" (Sesión DistanciaReferencia).

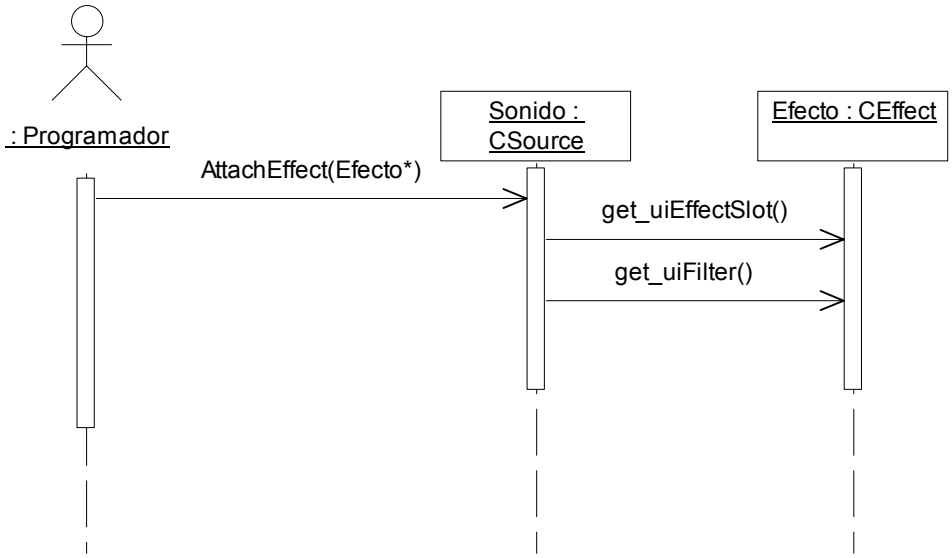

Figura 31 Diagrama de Secuencia "Manipular Fuente" (Sesión Asignar Efecto).

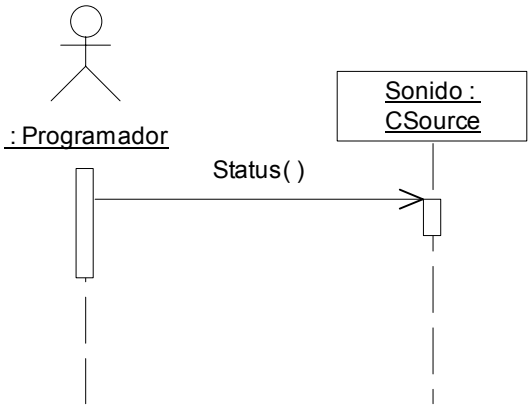

Figura 32 Diagrama de Secuencia "Manipular Fuente" (Sesión Estado).

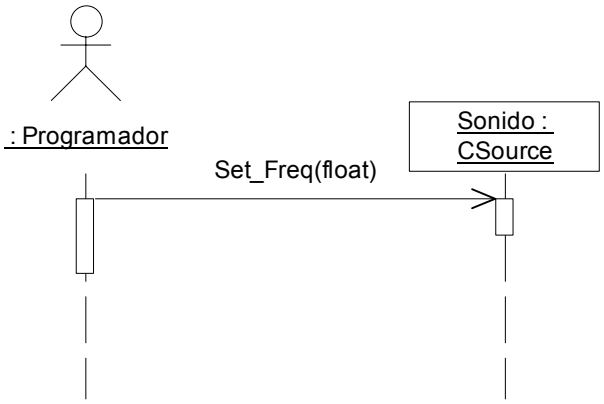

Figura 33 Diagrama de Secuencia "Manipular Fuente" (Sesión Frecuencia).

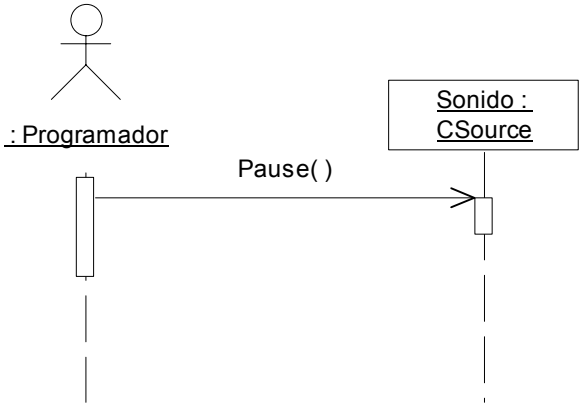

Figura 34 Diagrama de Secuencia "Manipular Fuente" (Sesión Pausa).

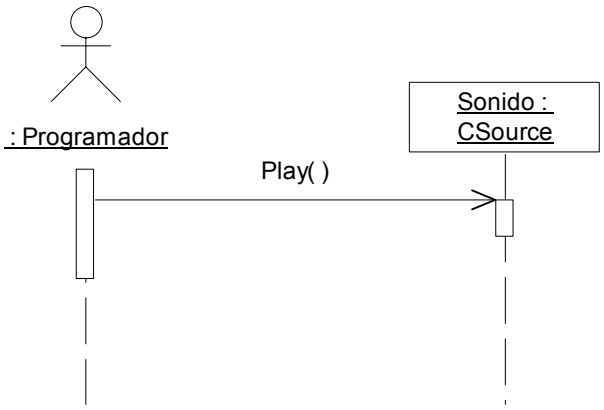

Figura 35 Diagrama de Secuencia "Manipular Fuente" (Sesión Play).

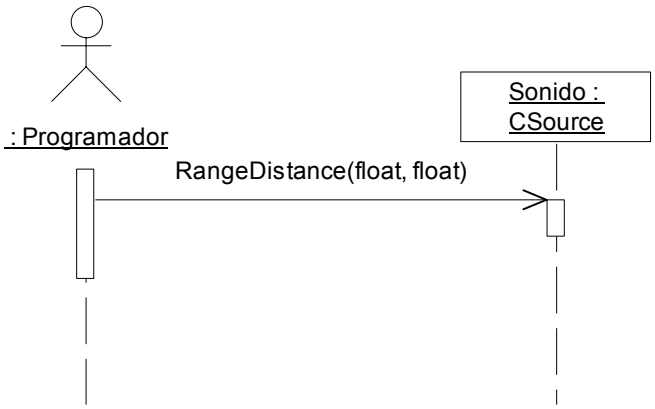

Figura 36 Diagrama de Secuencia "Manipular Fuente" (Sesión Rango).

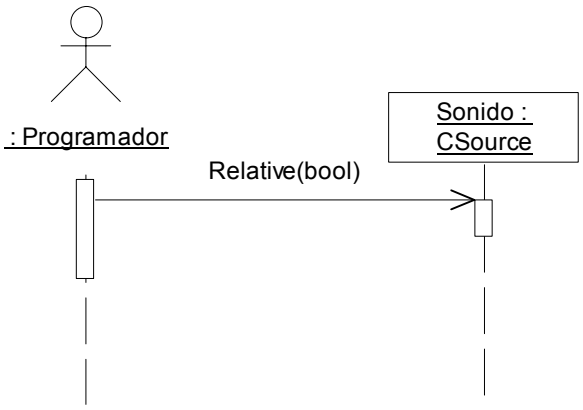

Figura 37 Diagrama de Secuencia "Manipular Fuente" (Sesión Relativa).

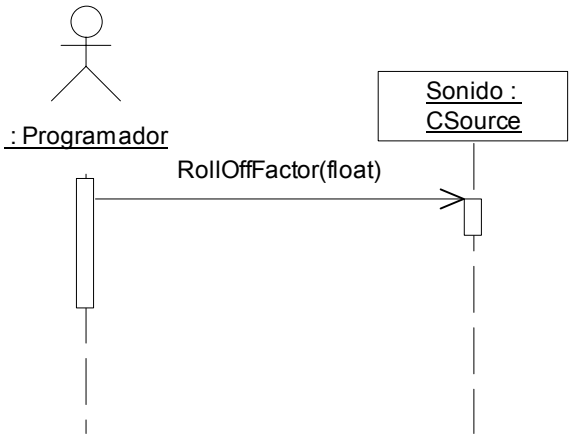

Figura 38 Diagrama de Secuencia "Manipular Fuente" (Sesión RollOfFactor).

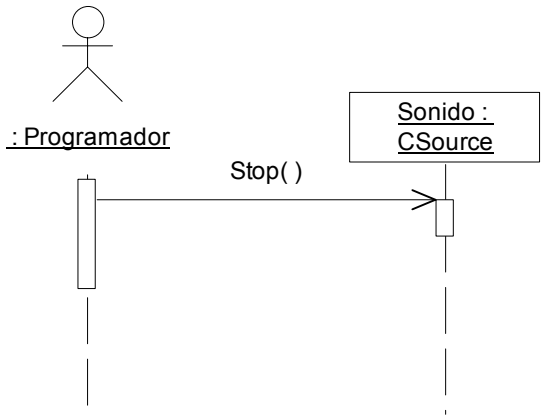

Figura 39 Diagrama de Secuencia "Manipular Fuente" (Sesión Stop).

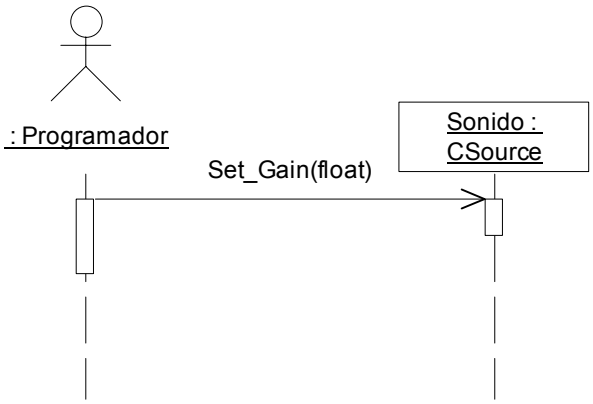

Figura 40 Diagrama de Secuencia "Manipular Fuente" (Sesión Volumen).

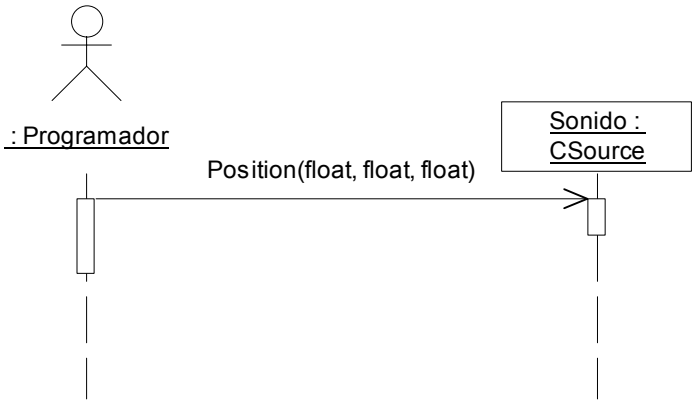

Figura 41 Diagrama de Secuencia "Actualizar Fuente" (Sesión Posición).

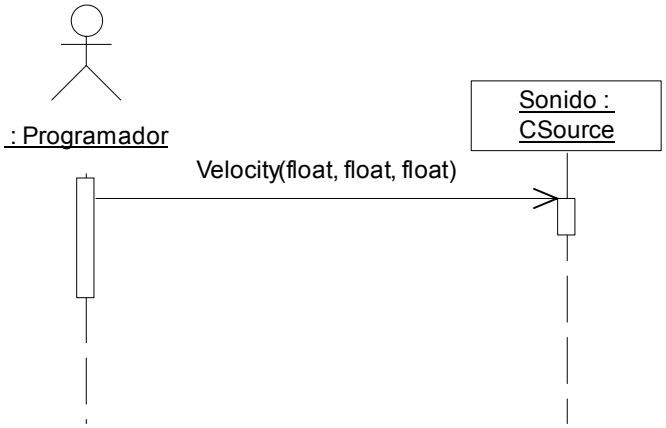

Figura 42 Diagrama de Secuencia "Actualizar Fuente" (Sesión Velocidad).

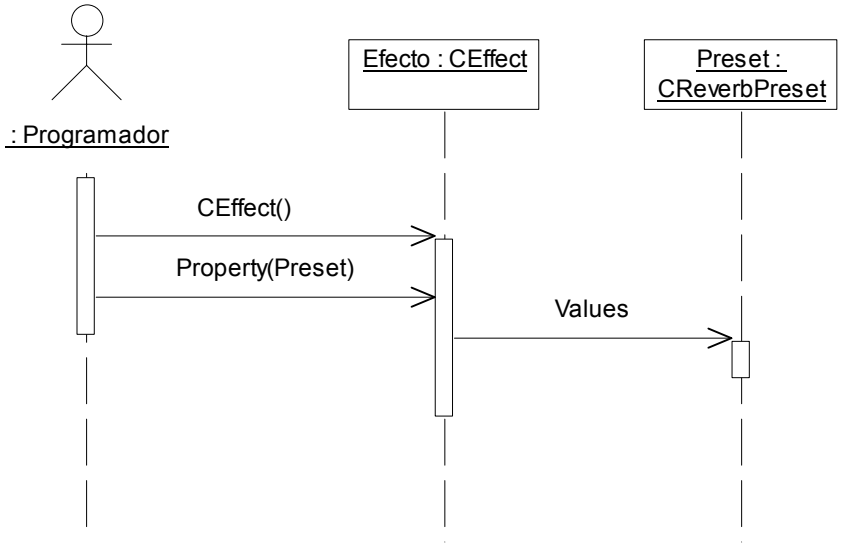

Figura 43 Diagrama de Secuencia "Crear Efecto" (Sesión Efecto).

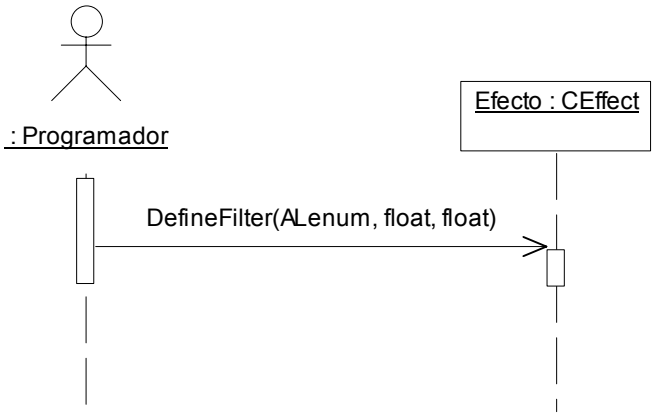

Figura 44 Diagrama de Secuencia "Crear Efecto" (Sesión Filtro).

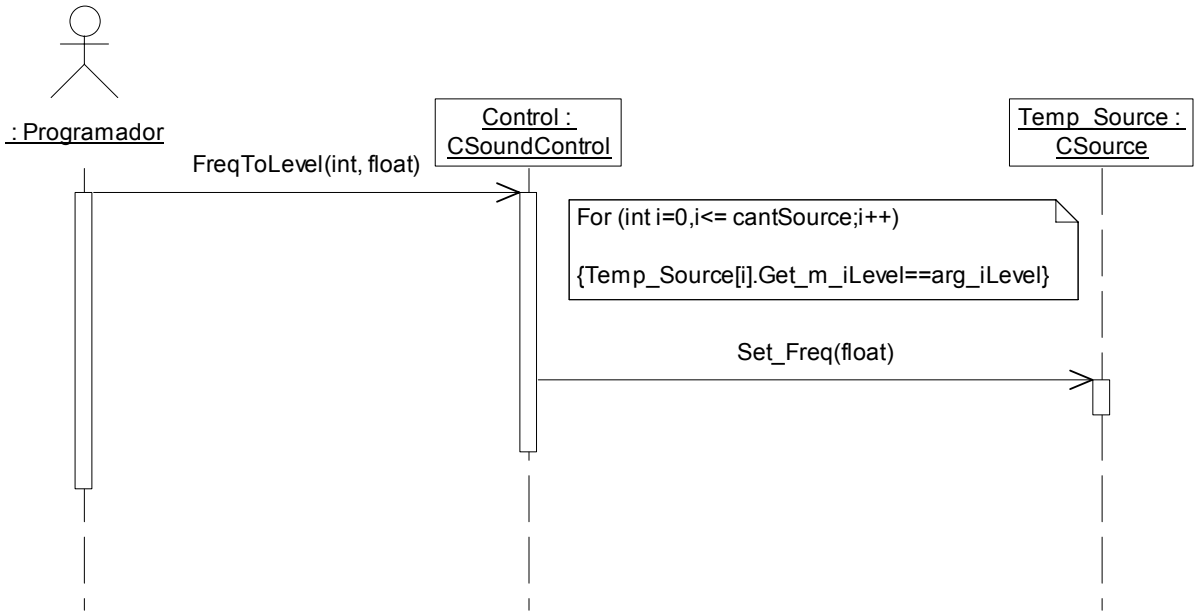

Figura 45 Diagrama de Secuencia "Manipular capa o Nivel" (Sesión Frecuencia).

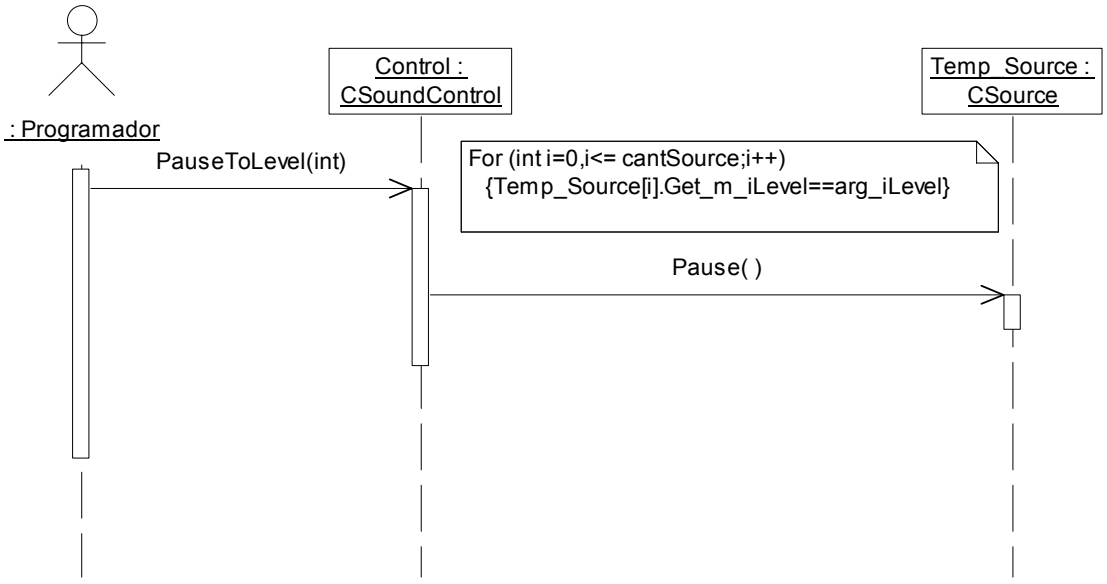

Figura 46 Diagrama de Secuencia "Manipular capa o Nivel" (Sesión Pausa).

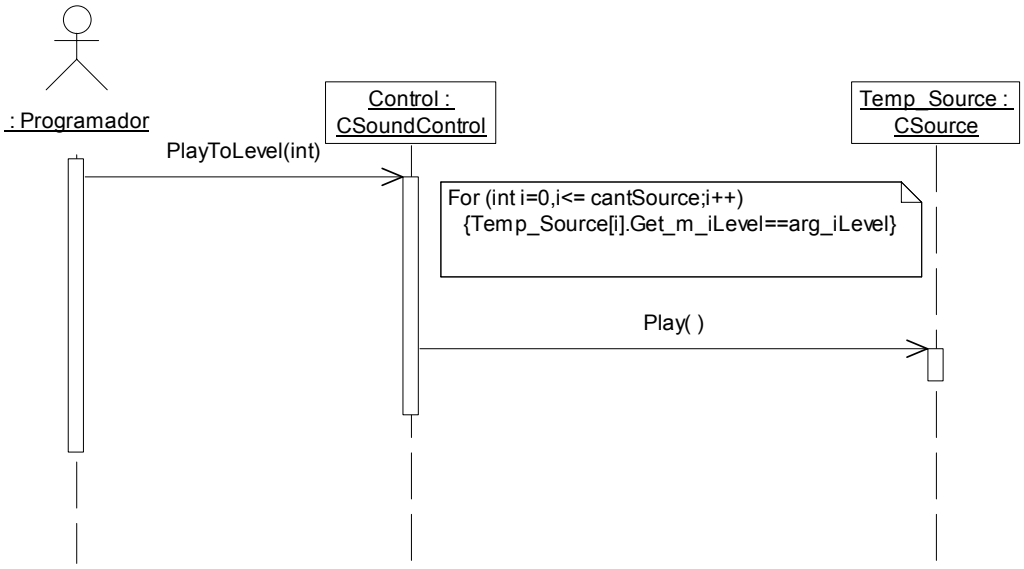

Figura 47 Diagrama de Secuencia "Manipular capa o Nivel" (Sesión Play).

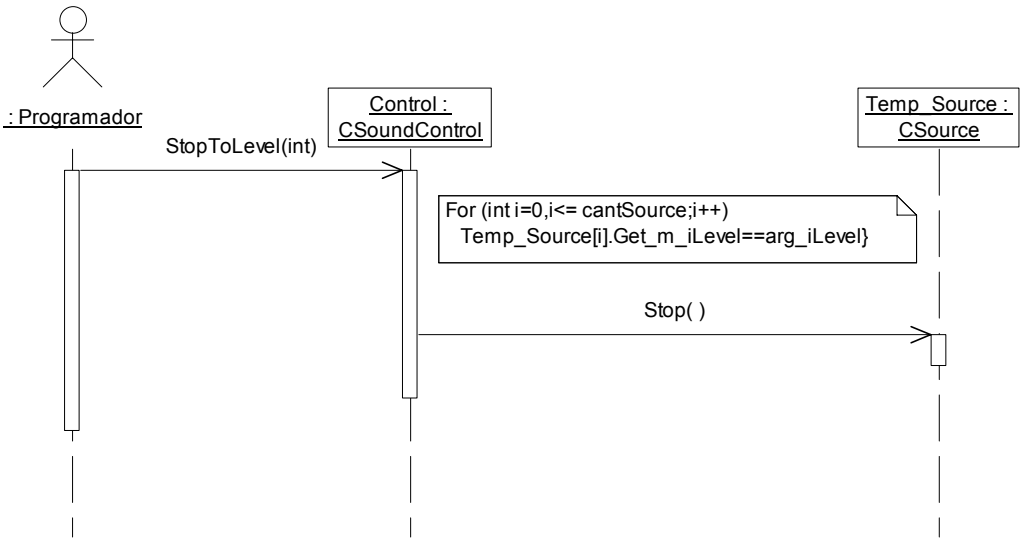

Figura 48 Diagrama de Secuencia "Manipular capa o Nivel" (Sesión Stop).

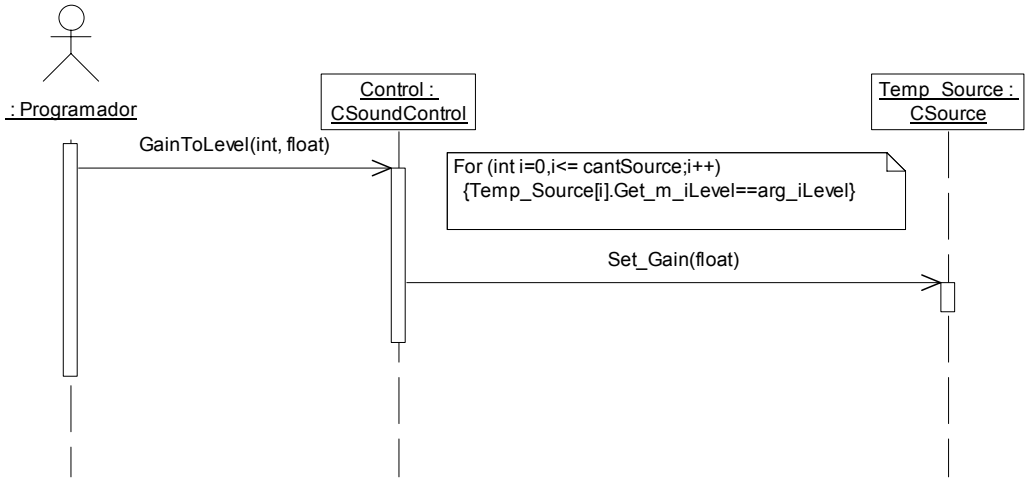

Figura 49 Diagrama de Secuencia "Manipular capa o Nivel" (Sesión Volumen).

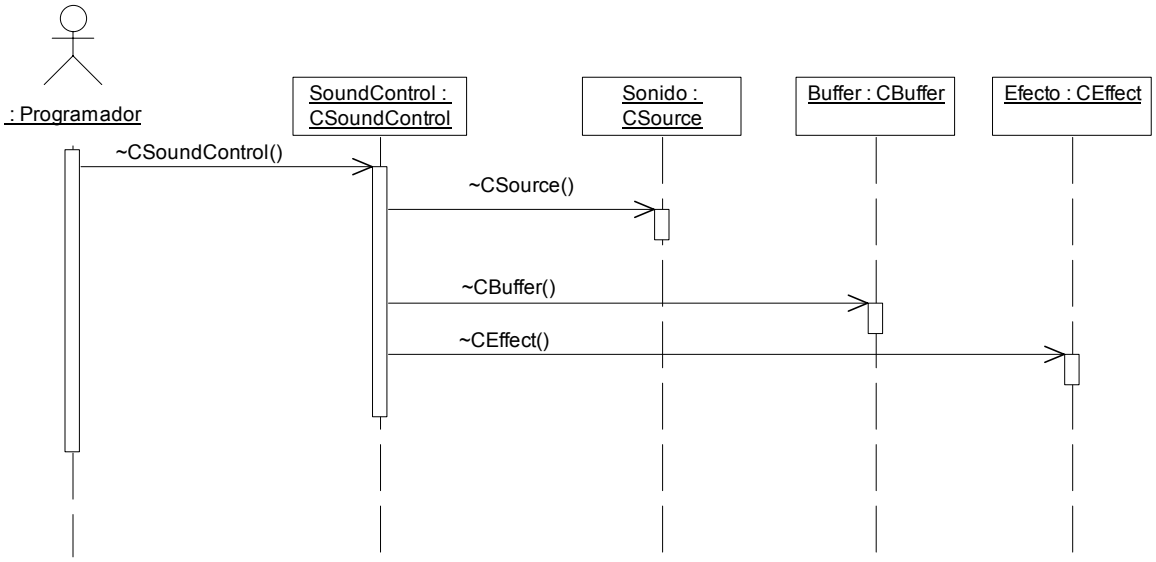

Figura 50 Diagrama de Secuencia "Destruir Sonido"

3.4 Estándares de codificación.

3.4.1 Nomenclatura de las versiones.

Este módulo SoundToolKit se rige por las mismas políticas para el control de versiones que la Herramienta SceeneToolKit. Tiene como característica que está en una fase de desarrollo primario, y quedan elementos y funcionalidades esenciales o importantes que no han sido añadidos, por lo que todavía no se ha llegado a una versión completamente estable. De esta manera, cada *release* nuevo que salga se considera Beta, y por demás sobrescribe completamente las versiones anteriores. Solamente se considera oficial la última versión liberada, y las anteriores se consideran como disfuncionales.

Es responsabilidad del líder del presente proyecto tener al tanto a los usuarios de la herramienta de la liberación de la nueva versión así como la información de los cambios añadidos.

La nomenclatura de las versiones presenta la siguiente notación:

- ***.0.0**: Los cambios en la primera parte representan cambios estructurales arquitectónicos de gran impacto, así como versiones más terminadas. Constituyen las liberaciones oficiales de la herramienta, a no ser que por necesidad se requiera liberar alguna versión menos estable.

- **0.*.0**: los cambios en la segunda parte representan adición de funcionalidades o módulos de menor impacto.

- **0.0.*:** los cambios en la tercera parte representan refinamiento de módulos y correcciones. Constituyen versiones internas al proyecto.

Beta: versiones para prueba por parte de los clientes.

3.4.2 Estándares de codificación.

El código de la biblioteca sigue algunos estándares propuestos por el grupo de desarrollo (respetando los estándares de codificación para C++ (identado, uso de espacios y líneas en blanco, etc.)). Está programado en inglés, debido que las palabras son simples, no se acentúan y es un idioma muy difundido en el mundo informático.

El conocimiento de los estándares seguidos para el desarrollo de la misma permitirá un mayor entendimiento del código, y es una exigencia de los autores de la misma que cualquier módulo que se añada debe estar codificado siguiendo estos estándares.

Nombre de los ficheros:

Se nombrarán los ficheros .h y .cpp de la siguiente manera:

STKNameOfUnits.cpp

Constantes:

Las constantes se nombrarán con mayúsculas, utilizándose el "_" para separar las palabras: MY_CONST_ZERO = 0;

Tipos de datos:

Los tipos se nombrarán siguiendo el siguiente patrón:

Enumerados*:* enum **E**MyEnum {**ME**_VALUE, **ME**_OTHER_VALUE};

Indicando con "**E**" que es de tipo enumerado. Nótese que las primeras letras de las constantes de enumerados son las iniciales del nombre del enumerado. Véase otro ejemplo:

enum **E**NodeType {**NT**_GEOMETRYNODE,…};

Estructuras*:* struct **S**MyStruct {…};

Indicando con "**S**" que es una estructura. Las variables miembros de la estructura se nombrarán igual que en las clases, leer más adelante.

Clases*:* class **C**ClassName;

Indicando con "**C**" que es una clase

Interfaces*:* **I**MyInterface

Indicando con "**I**" que es una interfaz.

Listas e iteradores STD*:*

vector<> **T**Name**List**;
TNameList::iterator **T**Name**ListIter**;

map<> **T**Name**Map**;
TNameMap::iterator **T**Name**MapIter**;

multimap<> **T**Name**MultiMap**;
TNameMultiMap::iterator **T**Name**MultiMapIter**;

Declaración de variables:

Los nombres de las variables comenzarán con un identificado del tipo de dato al que correspondan, como se muestra a continuación. En el caso de que sean variables miembros de una clase, se le

antepondrá el identificador “” (en minúscula), si son globales se les antepondrá la letra “g”, y en caso de ser argumentos de algún método, se les antepondrá el prefijo “arg_”.

Tipos simples*:*

bool **b**VarName;

int **i**Name;

unsigned int **ui**Name;

float **f**Name;

char **c**Name;

char* **ac**Name; // arreglo de caracteres

char* **pc**Name; // puntero a un char

char** **aac**Name; // bidimensional

char** **apc**Name; // arreglo de punteros

bool **m_b**MemberVarName; //variable miembro

char **g*C***GlobalVarName; //variable global, no se le antepone “”

short **s**Name;

Instancias de tipos creados*:*

EMyEnumerated **e**Name;

SMyStructure **k**Name;

CClassName **k**ObjectName;

CClassName* **pk**Name; //puntero a objeto

CClassName* **ak**Name; //arreglo de objetos

CClassName* **ak**Name; // variable miembro de clase

IMyInterface* **pI**Name; //puntero interfaces

Métodos:

En el caso de los métodos, se les antepondrá el identificador del tipo de dato de devolución, y en caso de no tenerlo (void), no se les antepondrá nada. Solamente los constructores y destructores comenzarán con “”.

En el caso de los argumentos se les antepone el prefijo “arg_”

Constructor *y* destructor*:*

CClassName (bool **arg_**bVarName, float& **arg_**fVarName);
~**C**ClassName ();

Funciones*:*

bool **b**Function1 (…);

int* **pi**Function2 (…);
CClassName* **pk**Function3 (…);

Procedimientos*:*
void Procedure4 (…);

Métodos de acceso a miembros:

Los métodos de acceso a los miembros de las clases no se nombrarán “Gets” y “Sets”, sino como los demás métodos, pero **con el nombre** de la variable a la que se accede y sin “m_”:

int **i**MyVar; //variable

Obtención del valor*:*

```
int iMyVar();
{
   return iMyVar;
}
```

Establecimiento del valor*:*

```
void MyVar(char* arg_iMyVar)
{
  iMyVar = arg_iMyVar;
}
```

Obtención *y* establecimiento del valor*:*

```
int& iMyVar();
{
   return iMyVar;
}
```

3.5 Diagrama de componentes.

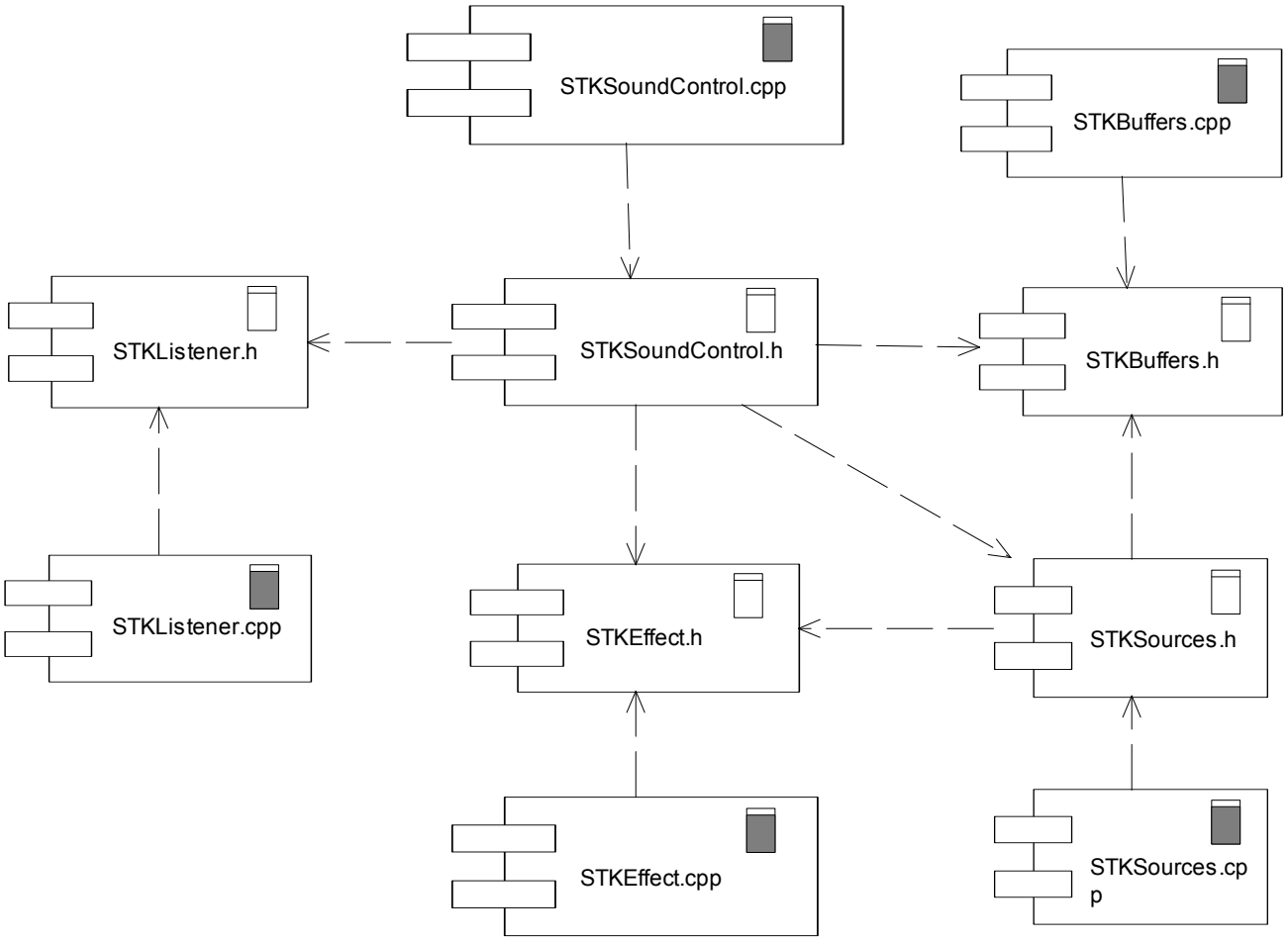

Figura 51 Diagrama de Componentes.

Conclusiones

Ya se encuentra todo preparado para pasar a la etapa de programación de los casos de uso. A partir de los diagramas hechos en Rational Rose y aprovechando la posibilidad que brinda este, se generará el código de fuente de los componentes.

Conclusiones

Para dar cumplimiento a los objetivos de este proyecto, -en correspondencia con las exigencias de la entidad cliente-, fue necesario primeramente hacer un estudio de las técnicas, tecnologías y tendencias en cuanto al uso de los sonidos en los Sistemas de Realidad Virtual, y en los sistemas informáticos en general. Se analizaron formatos, funcionalidades y características básicas, que brindan algunos sistemas existentes, teniendo en cuenta además las deficiencias de éstos que se requiere sean erradicadas en este trabajo.

A partir de estas premisas se obtuvo un módulo totalmente funcional, que cumple los requisitos, y que permite manipular los sonidos en los entornos virtuales de forma fácil al programador, dándole la posibilidad de variar parámetros en tiempo real y aplicarle efectos para lograr un alto nivel de realismo, siempre con la posibilidad de agregar funcionalidades y de adaptarlo a nuevas necesidades.

Recomendaciones

Se hacen las siguientes recomendaciones:

- Aumentar el número de formatos soportados.
- Ampliar los tipos de efectos y filtros que se pueden usar, en dependencia del hardware en que se use.
- Crear formato de sonido propio.
- Crear herramienta visual para configurar las posiciones que ocuparán los sonidos en un entorno.
- Crear módulo para la generación de sonidos a partir de algoritmos matemáticos.
- Crear herramienta visual para comprobar los parámetros de los ficheros.
- Crear documentación de ayuda y especificaciones.

Glosario de términos

A:

ADPCM: (acronym for Advanced Differential Pulse Code Modulation). Algoritmo de compresión de sonido, que según la velocidad, obtiene calidades muy altas.

API: Interfaz de programación de aplicaciones (Applications Programming Interface): una serie de funciones que están disponibles para realizar programas para un cierto entorno.

B:

*Buffer*s: espacio de memoria para almacenamiento temporal de datos.

C:

Copyright: Derecho exclusivo de un autor o de su editor de explotar durante varios años una obra literaria, artística o científica

C++: Lenguaje de programación orientado a objetos.

CPU: CPU (siglas de Central Processing Unit) o Unidad Central de Proceso (UCP) a la unidad donde se ejecutan las instrucciones de los programas y se controla el funcionamiento de los distintos componentes del ordenador. Suele estar integrada en un chip denominado microprocesador.

D:

Delay: retardo, demora.

DirectX/DirectXSound: Es un grupo de APIs (Application Program Interface) que permite a los programadores acceder de manera más directa a las capacidades gráficas y de audio de un

ordenador. Con ello se pueden crear Websites dinámicas con contenidos en 3D, gráficos, video y audio (Producto de Microsoft).

Doppler: consiste en la variación de la longitud de onda de cualquier tipo de onda emitida o recibida por un objeto en movimiento.

Difusión acústica: Dispersión, propagación, de audio.

E:

Estéreo: Sistema de registro y reproducción que utiliza dos canales de información sonora: izquierdo y derecho.

Equalizer: Componente de audio que resalta o atenúa ciertas frecuencias de sonido (bandas) de la señal original; existen varios tipos como gráficos, paramétricos, gráficos electrónicos.

Eco: La repetición de un sonido retardada en el tiempo, un mínimo de 50 milisegundos en relación al sonido original.

Esd / demonio de sonido: Enlightened Sound Daemon. Programa diseñado para mezclar varias secuencias de audio digital para ser reproducidos en un mismo dispositivo.

EAX: Environmental Audio extension. Extensiones de audio ambiental.

Engine: Motor.

H:

HRTF: Funciones de Transferencia Relativas.

I:

Interoperabilidad: Capacidad de un programa para acceder a múltiples sistemas diferentes.

L:

Listener: Punto donde se centra la percepción del audio.

Lossy Compression: Un tipo de compresión que sacrifica algunos de los datos originales para obtener a cambio mayores volúmenes de compresión facilitando el almacenamiento o la transmisión de los datos.

M:

Mono: Sistema de registro y reproducción que utiliza solo un canal de información sonora.

Multipista / MultiCanal: Multiples canales de audio (generalmente mas de dos), que se reproducen por diferentes Bocinas para obtener un efecto de surround.

Multiplataforma: Que la aplicación corra en varios sistemas operativos.

O:

OpenGL: es una biblioteca gráfica desarrollada originalmente por Silicon Graphics Incorporated (SGI).OpenGL significa Open Graphics Library, cuya traducción es biblioteca abierta de gráficos.

OpenAL: OpenAL significa Open Audio Library, lo que en castellano significa: Biblioteca Abierta de Audio. Pretende ser una extensión de OpenGL que provea herramientas para el manejo de audio.

R:

Reverb / reverberación: es un fenómeno de la reflexión del sonido, consistente en la prolongación del mismo una vez se ha extinguido la fuente debido al reflejo de las vibraciones por el choque con un obstáculo cercano.

Reflexiones: Fenómeno que ocurre cuando una onda incide sobre una superficie y es desviada por esta sin cambiar de medio.

S:

SceneToolkit: Motor de visualización 3D.

Surround: se refiere al uso de múltiples canales de audio para provocar efectos envolventes a la audiencia, ya sea proveniente de una película o de una banda sonora. El surround se puede conseguir mediante la colocación física de un conjunto de altavoces o introduciendo efectos al procesar la señal, de modo que produzcan una percepción psicoacústica de 3D.

Subwoofer: Conocido comúnmente como "bajo", es un parlante que reproduce las frecuencias muy bajas, es decir, las que dan "fuerza" a la música.

Sound blaster: Una de las marcas de tarjetas de sonido más conocida. Corresponde a toda una gama creada por la casa Creative Labs.

Glosario de Abreviaturas

ADPCM: (acronym for Advanced Differential Pulse Code Modulation).

API: Interfaz de programación de aplicaciones (Applications Programming Interface).

CPU: Siglas de (Central Processing Unit) o Unidad Central de Proceso (UCP).

EAX: Environmental Audio extension.

HRTF: Funciones de Transferencia Relativas.

m: Metros.

m/s: Metros por segundo.

PC: Computadora personal, computadora de escritorio y microcomputadora.

SRV: Sistema de realidad virtual.

STL: Standard Template Library.

Referencias bibliográficas

[1] G., J. B. G. *La realidad virtual*, Lucas Morea / Sinexi S.A., 1997. [2007]. Disponible en: http://www.monografias.com/trabajos4/realvirtual/realvirtual.shtml

[2] SACCO, A. *Apuntes sobre sonido digital*, [pdf]. 2003-2004. [2006]. Disponible en: http://www.antoniosacco.com.ar/

[3] ROMBERG, J. *Teorema de Nyquist*, 2005. [Disponible en: http://cnx.org/content/m12971/1.2/

[4] TIMONEDA, J. M. MÓDULO 3: Sonido digital. en: *Herramientas Básicas Multimedia*. Departamento de Educación y Ciencia, 2002.p.

[5] MARTÍN, A. L. *FORMATOS DE AUDIO DIGITAL* [web]. 2005-2006. [2006]. Disponible en: http://www.lpi.tel.uva.es/~nacho/docencia/ing_ond_1/trabajos_01_02/formatos_audio_digital

[6] MCCUSKEY, M. *Beginning Game Audio Programming*, PREMIER PRESS a division of Course Technology, 2003.

[7] *Vorbis.com main page*. Xiph.Org., 1994 - 2005. [2006]. Disponible en: http://www.vorbis.com/

[8] DUARTE, A. *Una Breve Historia del Proceso Digital*, 2006. [2006]. Disponible en: http://maravillososSonidos.blogspot.com/2006/09/una-breve-historia-del-proceso-digital.html

[9] FASANI, I. J. L. F. *Introducción a Sonido en video Juegos (1ª parte)*, CodePixel ©, 2004. [2006]. Disponible en: http://www.codepixel.com/tutoriales/sonido1/

[10] TURCAN, P. and M. WASSON. *Fundamentals of Audio and Video Programming for Games*, Microsoft Press

[11] FASANI, I. J. L. F. *Análisis Sonido 3D*, CodePixel ©, 2004a. [2006]. Disponible en: http://www.codepixel.com/tutoriales/sonido2/

[12] RUIZ, D. G. *Programación de Audio*, miniSites, 2006]. Disponible en: http://www.dedalus-software.com.ar/_tutoriales/sound/prog%20de%20audio.pdf

[13] HIEBERT, G.; K. CHARLEY*, et al. OpenAL Programmer's Guide*. LTD., C. T., 2006.

[14] PERIS, F. B. *Seminario OpenAL: sonido para videojuegos.*, 2003. [2006]. Disponible en: http://personales.alumno.upv.es/~ferblape/formacion

[15] RESERVED., C. T. L. A. R. *SoundBlaster*, 2007. [2007]. Disponible en: http://www.soundblaster.com/eax/

Bibliografía consultada

[1] AUTORES, R. D. Game Programming Gems. Charles River Media. Rockland Massachusetts, 2000. 614 p.

[2] RAGHUVANSHI, N. and M. LIN. *Interactive sound synthesis for large scale environments*. SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games, ACM, 2006. 101-108p.

[3] LOKI SOFTWARE, I. *Programming Linux Games*. San Francisco, William Pollock Phil Hughes, 2001. p. 1-886411-48-4 (pbk.).

[4] VINCENT, D.; M. DENIS*, et al.* Numerical simulations of xylophones. II. Time-domain modeling of the resonator and of the radiated sound pressure *The Journal of the Acoustical Society of America*, 1998, 104(3): 1633-1647.

Índice de figuras y tablas

Índice de figuras

Índice de tablas