Universidad de las Ciencias Informáticas

Facultad #5

Sistema de Efectos de Sonidos Avanzados

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Pavel Delgado Aguirre.

Pablo José Hernández Cartalla.

Tutor: Ing. Leoder Alemañy Socarrás.

Ciudad de la Habana, Julio de 2010

“Año 52 de la Revolución”

“Descubrir algo significa mirar lo mismo que está viendo todo el mundo, y percibirlo de manera diferente.”

Albert Szent Gyorgi

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Pavel Delgado Aguirre

Autor

Pablo José Hernández Cartalla

Autor

_____

Leoder Alemañy Socarrás

Tutor

Datos de Contacto

Ing. Leoder Alemañy Socarrás.

Ingeniero en Ciencias Informáticas, actualmente en adiestramiento, está pendiente a discutir la maestría en Informática Aplicada en diciembre - enero. Tiene 2 años de experiencia. Lleva 1 año de graduado.

E-mail: lalemany@uci.cu

Agradecimientos

De Pavel:

A mis padres por la educación dada, a mi novia por su apoyo todos estos años y a mis amigos por la amistad incondicional.

Agradecimientos

De Pablo:

A mi madre, por darme fuerzas para seguir adelante siempre, por ser el ejemplo a seguir de toda mi vida y por todo el amor que me ha dado.

A mi padre, por darme su apoyo en todo momento y ayudarme en los momentos difíciles y por el amor que me ha dado en todos estos años.

A mi hermana Rosi, por soportar todas mis pesadeces y tenerme como un ejemplo, por su apoyo y su cariño.

A mis tías y tíos, en especial a mi tío Landy y mi tía Mary por apoyarme siempre y ser incondicionales conmigo.

A mis abuelos, que han sido parte de mi vida y siempre me han apoyado en todas las decisiones que he tomado, por su amor infinito y comprensión.

A Yenia mi novia por mantenerse a mi lado en todos los momentos, por darme todo su amor, su ternura y por soportarme en todos estos años.

A mi compañero de tesis Pavel que más que mi compañero ha sido mi amigo y mi hermano todos estos años y siempre estuvo a mi lado en las buenas y en las malas.

A todos mis amigos por lo que no voy a mencionar nombres para que no se me quede nadie fuera, solo uno en especial, al que tengo que agradecerle el haber estado aquí, Fermín.

A todos aquellos que me ayudaron y apoyaron para poder realizar este sueño.

En fin a todas aquellas personas que de una forma u otra han estado a mi lado en todo este tiempo.

Dedicatoria:

A mis padres por siempre confiar en mí y en especial a mi amigo Fermín, que fue quien me embulló a estudiar en la UCI. A mi familia en general y demás seres queridos.

Pablo José.

A mis padres por la educación dada, a mi novia por su apoyo todos estos años y a mis amigos por la amistad incondicional.

Pavel Delgado Aguirre.

Resumen

La Realidad Virtual se ha expandido considerablemente a diferentes sectores de la sociedad en los últimos años. Esto ha provocado la evolución del hardware y software usado para ello, tanto sistemas de visualización, modelos físicos-matemáticos, modelos de inteligencia artificial, simuladores como sistemas de sonido que se suman para dar realismo y credibilidad. 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 la inmersión del usuario en el mundo virtual a través del sentido auditivo. En el Departamento de visualización y realidad virtual de la Universidad de las Ciencias Informáticas (UCI) existen dificultades con el manejo de sonidos 3D en entornos virtuales. La producción de la misma se ve afectada ya que los módulos de sonido integrado al motor gráfico Ogre 3D no cuentan con funciones de efectos de sonidos avanzados que le permitan un fácil manejo a los programadores.

El presente trabajo propone la adición de funciones a un módulo que trabaja con bibliotecas de Sonidos en Sistemas de Realidad Virtual. Con este trabajo se pretende brindar funcionalidades para el manejo de sonido de forma cómoda a los programadores con una arquitectura orientada a objetos, configurable y multiplataforma basada en OpenAL y compatible con varios formatos y con posibilidades de adicionar nuevas funcionalidades en futuras versiones. Luego de realizar un análisis a los diferentes formatos y formas de compresión del sonido se ha planteado el uso de los formatos autodescriptivos. Se usarán específicamente el .WAV y .AU ya que permiten la manipulación de sus parámetros en tiempo real, algo muy necesario para lograr un efecto realista. En 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 otro tipo de formato.

Con la incorporación de efectos de sonidos al módulo integrado a OGRE 3D se podrá incorporar sonido a los juegos y/o aplicaciones que utilicen esta herramienta.

Palabras clave: Sonidos 3D, OGRE 3D, Open AL.

Índice

Índice

Introducción

El desarrollo del software desde su surgimiento ha evolucionado a gran escala a nivel mundial. Empresas, instituciones y países han optado por el desarrollo de esta industria para favorecer sus activos económicos. En la actualidad las compañías dependen cada vez más del software para realizar sus funciones. Con el desarrollo del software también ha venido la evolución de las Tecnologías de la Información y las Comunicaciones que es un factor decisivo para el desarrollo de las empresas, de la economía y de la sociedad en el mundo. Todo esto trae consigo un proceso que busca lograr más eficacia, eficiencia, satisfacer las necesidades de información y conocimiento de todas las personas y esferas de la sociedad; a la vez agiliza los procesos en las instituciones, incrementa los beneficios y ganancias en el mercado mundial.

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 software usado para ello, tanto sistemas de visualización, modelos físicos-matemáticos, modelos de inteligencia artificial, simuladores como sistemas de sonido que se suman para dar mayor realismo y credibilidad a los entornos virtuales. 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 la inmersión del usuario en el mundo virtual a través del sentido auditivo.

Desde el surgimiento de los Sistemas de Realidad Virtual 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 Sistemas de Realidad Virtual se pueden agrupar en: gráfico, red, módulo físico-matemático, hardware, inteligencia artificial y sonido. Cada componente desempeña 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 Sistemas de Realidad Virtual 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 Sistemas de Realidad Virtual que crean un ambiente acústico tan real e impresionante que hace olvidar la realidad.

Se pretende brindar funcionalidades para el manejo de sonido de forma cómoda a los programadores, con una arquitectura orientada a objetos, configurable y multiplataforma, basada en una API y compatible con varios formatos y con posibilidades de adicionarle otros nuevos en futuras versiones.

En el Departamento de visualización y realidad virtual presente en la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI) existen dificultades con el manejo de sonidos 3D en entornos virtuales. La producción de la misma se ve afectada ya que los módulos de sonido integrado al motor gráfico OGRE 3D no cuentan con algunas funciones para un mejor manejo de los programadores. Por lo que se arriba al siguiente **problema científico:** ¿Cómo incorporarle nuevos efectos al módulo de sonido integrado al Motor Gráfico OGRE?

Constituyó **objeto de estudio** las bibliotecas de sonidos integradas al motor gráfico OGRE 3D, siendo el **campo de acción:** efectos de sonidos para sistemas de realidad virtual.

Para dar solución al problema científico se define como **objetivo de investigación**: Incorporar nuevos efectos al módulo de sonido integrado al Motor Gráfico OGRE.

Como **idea a defender** se puntualizó qué: Con la incorporación de efectos de sonidos al módulo integrado a OGRE 3D se podrá incorporar sonido a los juegos y/o aplicaciones que utilicen esta herramienta.

Para dar cumplimiento al objetivo de la investigación se acometieron las siguientes **tareas de investigación:**

1. Investigar las bibliotecas que existen para la generación de sonidos vinculadas al motor gráfico OGRE

2. Estudiar OGRE 3D para obtener un mejor conocimiento de su funcionamiento.
3. Estudiar el módulo integrado al Motor Gráfico OGRE 3D para mantener la estructura de clases.
4. Estudiar los efectos de sonidos.

Para desarrollar esta investigación se utilizaron algunos **métodos** científicos tantos teóricos como empíricos los cuales quedan planteados a continuación.

El **Análisis Histórico Lógico** permite realizar una serie de estudios basados en la evolución y las tendencias actuales de la librería de sonido OpenAL, lo cual ayuda a conocer la trayectoria histórica real de su desarrollo.

El **Analítico – Sintético** permite concretar y resumir el conocimiento reflejado en los materiales consultados sobre el tema de la librería de sonido OpenAL para utilizarlo en el desarrollo de la presente investigación.

El método de **Modelación** ayudará en la investigación, ya que se podrá realizar una buena modelación de los modelos y diagramas del diseño, para un mejor entendimiento de la situación en cuestión para la integración de la investigación.

La idea de adicionar otras funciones a este módulo de sonido OpenAL integrado al motor gráfico OGRE surge después de varias **entrevistas** con programadores y sonidistas de otros proyectos ya que tienen una excelente opinión del módulo antes planteado y fue recomendado por la mayoría de estas personas.

Observación: Hay que observar el comportamiento de los módulos de Sonidos existentes hasta la actualidad y las herramientas que han sido utilizadas para su desarrollo, para poder tener idea de cómo trabajar en él con esta nueva herramienta OGRE.

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, 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.

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

1.1 Sonido. Características.

Existen 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 ya 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.

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.

El **sonido** es un tipo de onda que se propaga únicamente en presencia de un medio que haga de soporte de la perturbación o es la interpretación que hace el 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 los seres humanos son capaces de escuchar van de 20 Hz a 20 KHz. Si se representa en un gráfico un sonido, se obtendrá lo siguiente: (Figura 1). (Loureiro, 2010)

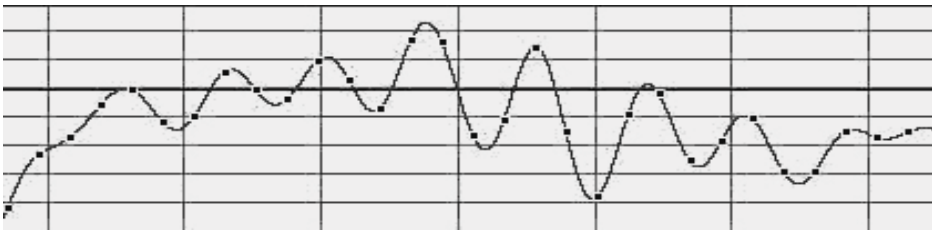

Figura 1: Representación gráfica de un sonido.

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. (Loureiro, 2010)

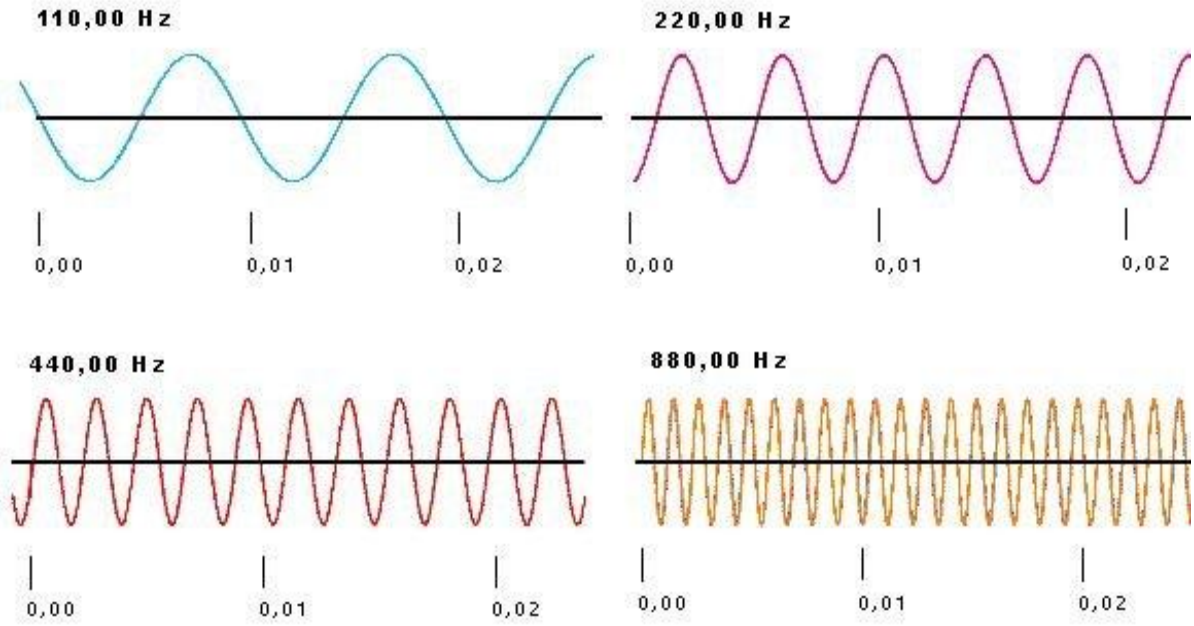

Figura 2: Frecuencia.

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. (Loureiro, 2010)

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. (Loureiro, 2010)

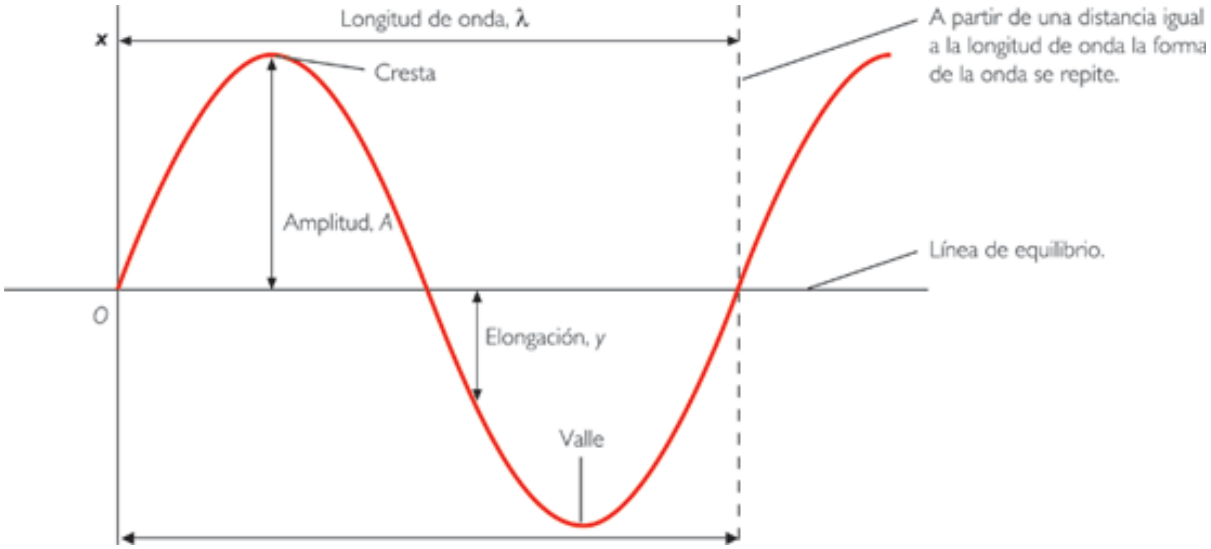

Figura 3: Amplitud de Onda.

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. (Loureiro, 2010)

Precisión de las muestras: indica la escala de bits que se ha utilizado para guardar el sonido. Pueden ser 8 bits (256 valores posibles) ó 16 bits (más de 65.000 valores posibles). (Loureiro, 2010)

Mono / Estéreo: El sonido puede grabarse en un solo canal (mono) o en dos (estéreo). (Loureiro, 2010)

Los efectos de audio 3D son un grupo de efectos de sonido que tratan de expandir la imagen estéreo producido por dos altavoces o auriculares estéreos, o crea la ilusión de las fuentes de sonido puesto en cualquier espacio tridimensional, incluyendo detrás, encima o debajo del detector. (WorldLingo, 2010)

Hay diversos tipos de efectos de sonido 3D: (WorldLingo, 2010)

- Aquellos que solo amplían la imagen de estéreo mediante la modificación de la información de fase.
- Aquellos que pueden poner los sonidos fuera de las bases estéreos.

- Los que incluyen una completa simulación 3D.

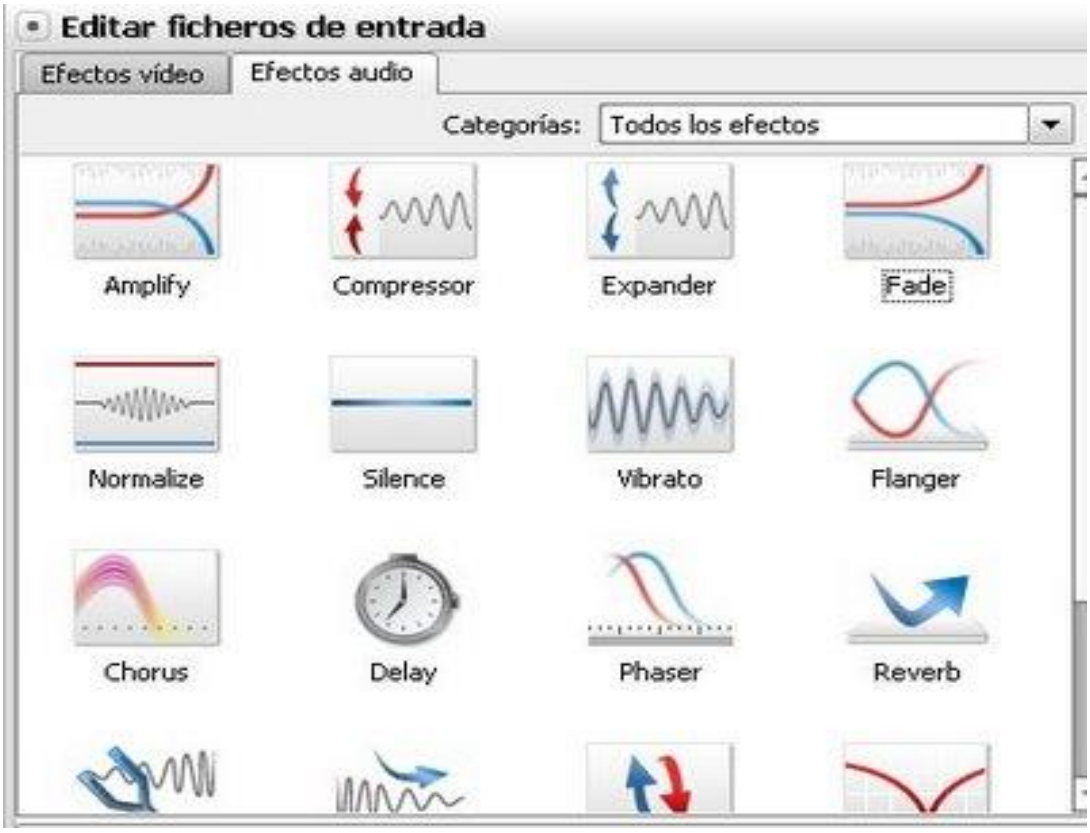

Figura 4: Efectos de audio 3D.

Las tarjetas de sonido utilizan chips procesadores de señales digitales o DSPs dedicados o chips ASP (Advanced Signal Processors), para generar efectos tales como reverberación, ecos, chorus de gran calidad. Los recursos disponibles y entregados por una tarjeta para el procesamiento de la información, dependen propiamente de la calidad de sus procesadores integrados. (Loureiro, 2010)

1.1.1 Sonido 3D Posicional Completo.

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. (Loureiro, 2010)

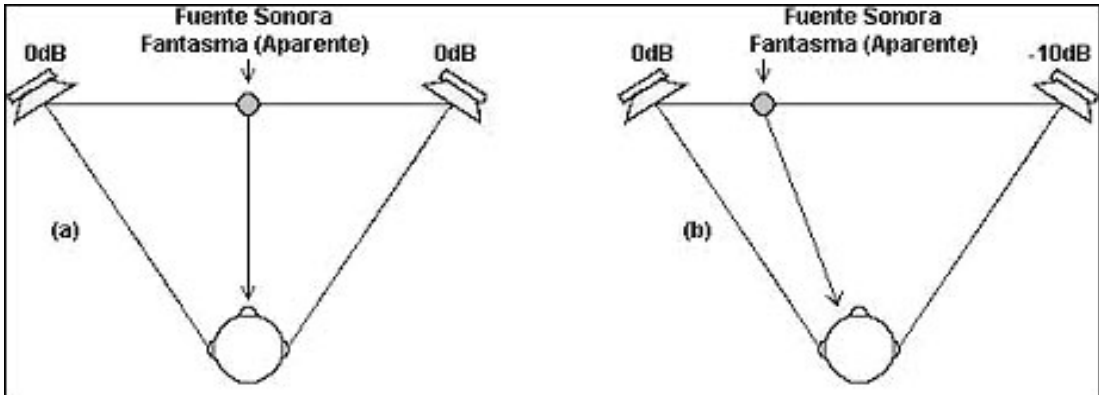

Figura 5: Audio 3D posicionado.

1.1.2 Reflexiones y Reverberaciones.

Las tecnologías para representar sonido virtual son ocupadas para recrear un espacio ficticio en una sala de entretención. Ejemplo: el poder simular que un jugador está dentro de una cueva y sentir el espacio de ella a través del sonido. La tecnología 3D es solo ocupar un sistema estéreo L y R y con los algoritmos recrear dos monitores mas virtuales. (Loureiro, 2010)

Recreación Virtual.

La función llamada Funciones de Transferencias Asociadas a Relacionar la cabeza (HRTF) es la encargada de relacionar el sonido originado de la fuente y poder localizarlo en un punto determinado. (Loureiro, 2010)

El sistema de HRTF tiene que comprender tres elementos fundamentales:

• Función de transferencia al oído izquierdo.

• Función de transferencia al oído derecho.

• Inter-aural tiempo de delay.

Simulación de reverberaciones y reflexiones.

Las reflexiones y las reverberaciones tienen gran influencia en la percepción de las propiedades del sonido y que el radio del sonido directo al radio del sonido reverberante es una poderosa herramienta para

saber a qué distancia se encuentra una fuente ubicada en algún punto del espacio. Una descripción de los que se habla está detallada en la figura 6. (Loureiro, 2010)

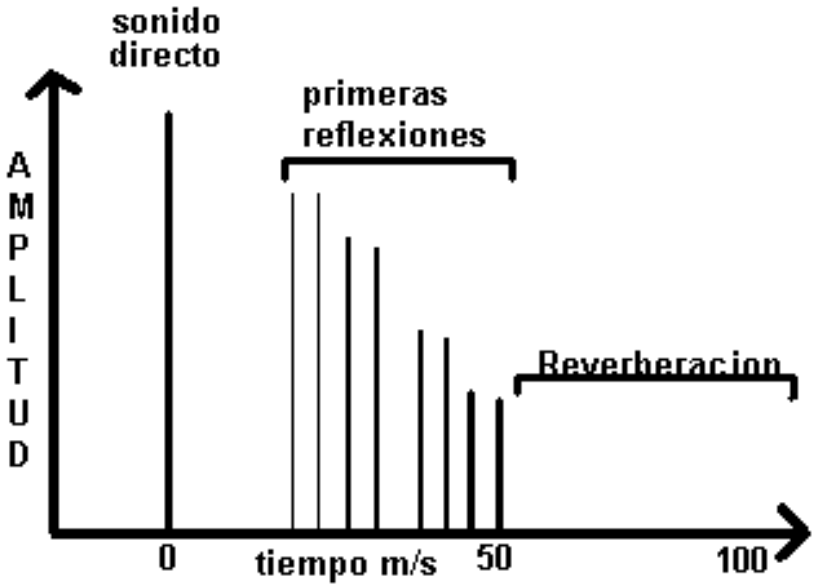

Figura 6: Modelo sonido directo e indirecto.

.

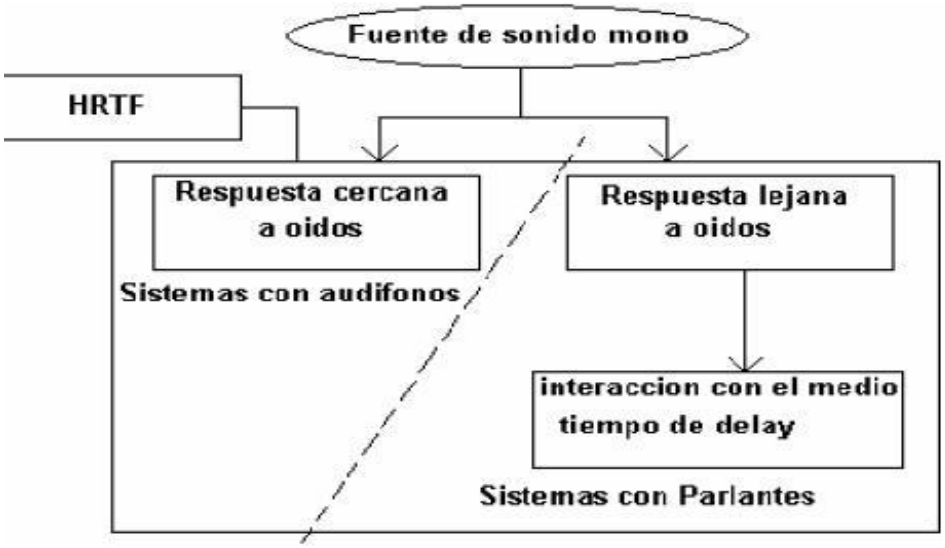

Figura 7: Proceso HRTF.

La simulación en 3D es el grupo más avanzado de efectos de sonido 3D. Utilizando la cabeza de funciones de transferencia relacionadas y reverberación, los cambios de sonido en su camino desde la fuente (incluida la reflexión de los muros y los suelos) a oídos del oyente pueden ser simulados. Estos efectos incluyen localizaciones de ruido a sus espaldas, arriba y abajo del oyente. (FASANI, 2004a. [2006].)

Algunas tecnologías 3D también convierten las grabaciones binarias para grabaciones estéreo. MorrowSoundTrue3D convierte binario, estéreo, 5.1 y otros formatos a 8.1 simple y multizonas con experiencias de sonido 3D en tiempo real. (WorldLingo, 2010)

Los efectos de audio posicional 3D surgieron en los años 1990 en PCs y juegos de consola. Como medios los juegos interactivos podrían beneficiar más que ningún otro. Algunas tecnologías al parecer trabajan mejor que otras, los sonidos 3D en juegos todavía son menos convincentes, especialmente sobre speakers (bocinas). Estas técnicas han incorporado música y estilos de videos de arte.

El proyecto investigativo Audioscape provee a los músicos con contenido audiovisual y renderizado de ambientes 3D en tiempo real, apropiado para aplicaciones de actuación en vivo. Un sito con animación y teoría de un sistema usando HRTF's para crear Audio 3D es: ISVR Virtual Acoustics. (WorldLingo, 2010)

Representaciones fidedignas del nivel de elevación para la reproducción en 3D Loudspeaker (bocinas altas) ha sido posible por el principio de "Ambisonics y síntesis de ondas de campo" y MorrowSound3D. (WorldLingo, 2010)

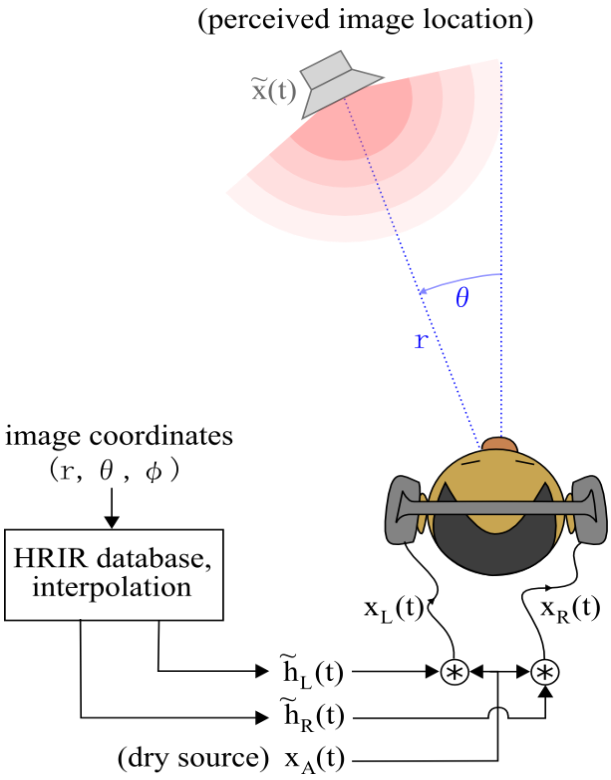

Figura 8: Ubicación de la imagen percibida.

Un sonido es situado en el plano horizontal procesando el sonido con respuestas de impulsos relacionados grabados.

1.1.3 Formatos de Sonido. Características.

Los formatos de fichero indican la estructura con la que el audio es almacenado, si 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. (MENCÍAS, 2009)

En general un mismo formato de fichero permite contener diversas codificaciones, tasas de muestreo. En este sentido se distingue entre dos grupos de formatos de ficheros de audio: (MENCÍAS, 2009)

- **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 auto descriptivo más habitual 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 el presente 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 y .*OGG*.

RIFF WAVE (*.wav) apócope de *WAVEform audio format*: es un formato de audio digital normalmente sin compresión de datos desarrollados y propiedad de Microsoft y de IBM que se utiliza para almacenar sonidos en el PC, admite archivos mono y estéreo a diversas resoluciones y velocidades de muestreo, su extensión es .wav.

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.

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.

Ogg Vorbis (*.ogg) es un formato de archivo contenedor multimedia, desarrollado por la Fundación Xiph.org y es el formato nativo para los códecs multimedia que también desarrolla Xiph.org. El formato es libre de patentes y abierto al igual que toda la tecnología de Xiph.org, diseñado para dar un alto grado de eficiencia en el "streaming" y la compresión de archivos. Como la mayoría de formatos contenedores, Ogg encapsula datos comprimidos (e incluso sin comprimir) y permite la interpolación de los datos de audio y de vídeo dentro de un solo formato conveniente. Los archivos terminados en la extensión ".ogg" pueden ser de cualquier tipo de archivo Ogg, audio o vídeo, aunque existe la recomendación de renombrarlos con la extensión ".oga" para audio y ".ogv" para video. La versión 1.0 fue lanzada el 29 de julio de 2002.

1.2 Módulos de sonido que se integran a OGRE.

OGRE (Object Oriented Graphics Engine) es un motor de gráficos en tres dimensiones multiplataforma, cuya principal ventaja radica en que es un proyecto de código abierto bajo licencia LGPL, lo tiene gran peso para nosotros al seleccionar las herramientas y por tanto distingue a OGRE sobre otros motores gráficos 3D. Con OGRE podemos hacer cualquier tipo de aplicación que requiera gráficos tridimensionales, no trae soporte nativo para sonido ni física lo que no es un problema gracias a la enorme comunidad existente en Internet; cuenta con módulos especialmente diseñados para OGRE que permiten tener estas facilidades como es el caso de FMOD SoundManager, Audiere, OgreOggSound o MogreFreeSL en el caso del sonido. (Junker, 2000-2009)

Existen muchas razones por las cuales es recomendable utilizar OGRE entre las que podemos mencionar: (Junker, 2000-2009)

- Facilita el desarrollo.
- Abstracción de la plataforma: si un Motor corre en varias plataformas la aplicación también
- Separación de Motor/Contenidos: facilita el trabajo en los diferentes módulos.

- Beneficios a terceros: los avances en el motor pueden beneficiar a varias aplicaciones.
- Permite enfatizar en la parte artística, ya que, al tener mayor grado de abstracción, se tiene menor carga de trabajo en programación.
- Mayor modularidad y reaprovechamiento.

OGRE permite la integración de módulos de sonidos tales como: OpenAL, Audiere, FMOD y MogreFreeSL que serán descritos en los epígrafes que se muestran a continuación.

1.2.1 OpenAL.

OpenAL es una API libre y multiplataforma para sonido tridimensional orientada inicialmente a los videojuegos, pero actualmente multipropósito. 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. La API está disponible para las siguientes plataformas: Mac OS, Linux (tanto para OSS como para ALSA), *BSD, Solaris, Irix, Microsoft Windows, Sony PlayStation 2, Microsoft Xbox y Nintendo GameCube. (Creative Labs, 2008)

Al contrario que la especificación de OpenGL, la especificación de OpenAL se divide en dos secciones: el núcleo, consistente en las llamadas a funciones, y la ALC API, utilizado para la gestión de contextos de renderizado, uso y bloqueo de recursos. Con la intención de agregar funcionalidades extra en el futuro, OpenAL utiliza un mecanismo basado en extensiones. Cada cual puede incluir sus propias extensiones en la distribución de OpenAL. El funcionamiento global de OpenAL se puede dividir en objetos fuente, oyentes y buffers de audio. Un objeto fuente contiene un puntero a un buffer, además de una serie de atributos como la velocidad, posición, dirección o intensidad del emisor de sonido. Un oyente contiene información sobre la velocidad, posición y orientación del sistema de referencia, además de la ganancia general aplicada a todo sonido. Sólo puede haber un oyente. Los buffers contienen la información del sonido en formato PCM, bien en 8 ó 16 bits, en formato mono o estéreo. El motor de renderizado se encarga de todos los cálculos necesarios como la atenuación, doppler. En la siguiente figura se muestra un gráfico del flujo de la OpenAL dentro de una aplicación general. (Creative Labs, 2008)

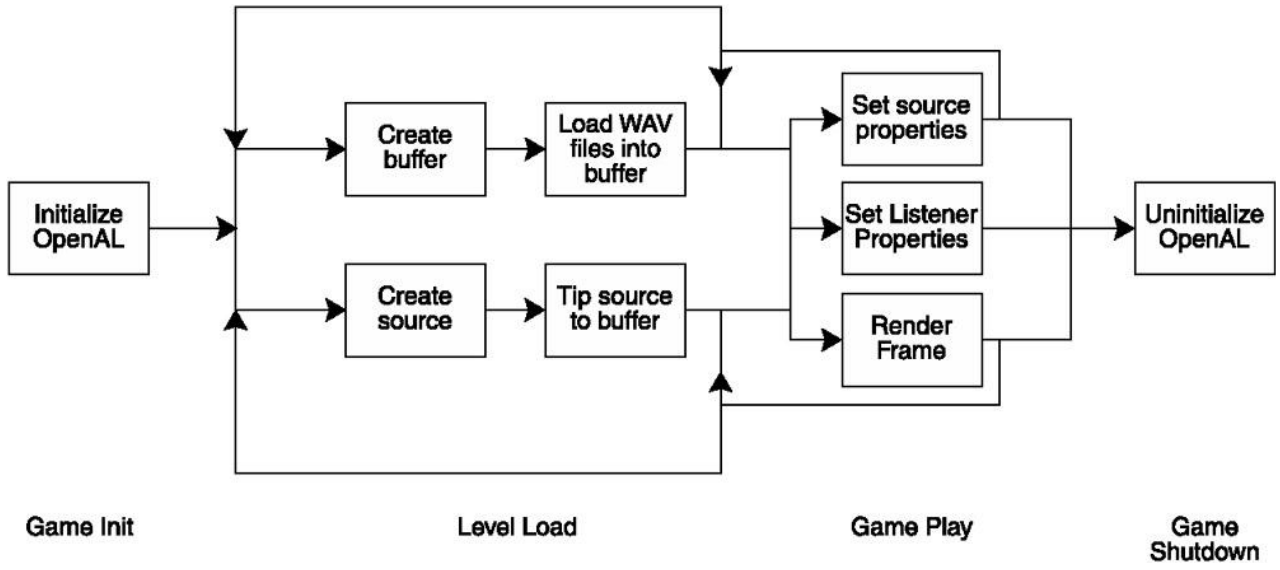

Figura 9: Flujo de Trabajo de OpenAL.

¿Qué proporciona OpenAL? (teoría). (Lazaro Abreu Reche, 2007)

- 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). (Lazaro Abreu Reche, 2007)

- 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.

Ventajas. (Lazaro Abreu Reche, 2007)

- API estable y fácil.
- Abierta.

- Extensible.
- Escrita en C.
- Multiplataforma.
- Modelo físico inherente.

Juegos que utilizan OpenAL. (WorldLingo, 2010)

- Basados en motores de Id Software.
    - Doom 3.
    - Jedi Knight II: Jedi Outcast.
    - Jedi Knight: Jedi Academy.
    - Quake 4.
    - Prey.
    - Tremulous.

- Basados en motores Unreal.
    - Unreal 2.
    - Unreal Tournament 2003.
    - Unreal Tournament 2004.
    - Unreal Tournament 3.
    - America's Army.
    - Hitman 2.

1.2.2 Audiere.

Audiere: es una API de audio de alto nivel, open source y bajo la licencia LGPL. Puede reproducir Ogg Vorbis, MP3, FLAC, WAV descomprimido, AIFF, MOD, S3M, XM e IT. Para la salida de audio compatible con DirectSound o Audiere WinMM en Windows, Linux y OSS en Cygwin y SGI IRIX en AL. (Junker, 2000-2009)

Ventajas. (Junker, 2000-2009)

- API fácil de usar.

- Formatos de ficheros compatibles: WAV sin comprimir, AIFF sin comprimir, Ogg Vorbis, FLAC, MP3, MOD, S3M, IT, XM (soporta búsquedas).
- Streaming de audio y protegido.
- Cambio en el volumen, pan, y cambiar el tono.
- Tono plano, onda cuadrada, ruido blanco, rosa y la generación de ruido.
- Ejecuta enumeración de tiempo el equipo y los formatos de archivo de audio.
- Enlaces a Python, Delphi, Java, XPCOM (JavaScript en Mozilla).

1.2.3 FMOD.

FMOD: es una biblioteca de audio propietario realizado por Firelight tecnologías que reproduce archivos de música de diversos formatos en diferentes plataformas. Se utiliza en los juegos y aplicaciones de software para proporcionar la funcionalidad de audio. (Junker, 2000-2009)

Antes de la versión 3.75, la biblioteca fue nombrada simplemente como FMOD. Desde entonces, el sistema de sonido FMOD ha sido rediseñado y ahora contiene tres partes principales: (Junker, 2000-2009)

* Ex FMOD, el motor de sonido de bajo nivel.
* FMOD de sucesos del sistema, más abstracto, más alto nivel de capa de aplicación para simplificar la reproducción de contenido creado con el Diseñador de FMOD.
* Diseñador de FMOD, la herramienta de diseño de sonido utilizado para la creación de complejos eventos de sonido y música para su reproducción.

El sistema de sonido FMOD tiene una arquitectura de plug-in avanzada, que puede ser utilizada para ampliar el apoyo de los formatos de audio o para desarrollar nuevos tipos de salida, por ejemplo, para el streaming.

FMOD está disponible bajo los regímenes de licencia múltiple. Está disponible de forma gratuita para aplicaciones no comerciales. FMOD puede ser utilizado comercialmente con un número de diferentes tipos de licencias, como la licencia comercial para el desarrollo comercial estándar, Shareware / Licencia de Aficionado para los programadores aficionados únicos y eDistributed juegos de consola para el

presupuesto de la consola de juegos. Cada licencia contiene diferentes restricciones, condiciones y costos. (Junker, 2000-2009)

FMOD sirve de apoyo a las siguientes plataformas: (Junker, 2000-2009)

* Microsoft Windows (32-bit y 64-bit).
* Mac OS 8/9/X.
* Mac OS, el iPhone.
* Linux (32-bit y 64-bit).
* Solaris.
* Nintendo GameCube, Wii.
* Xbox, Xbox 360.
* PlayStation 2, PlayStation Portable, PlayStation 3.

FMOD soporta los formatos de audio: AIFF, ASF, ASX, DLS, FLAC, FSB, IT, M3U, MID, MOD, MP2, MP3, OGG, PLS, RAW, S3M, VAG, WAV, WAX, WMA, XM, XMA.

Juegos que utilizan FMOD.

- Call of Duty 4.
- Cortex Command.
- Crysis.
- Darkfall.
- Batman: Arkham Asylum.
- Jurassic Park: Operation Genesis.
- Need for Speed: Shift.
- StarCraft II: Wings of Liberty.
- Tomb Raider: Underworld.
- World of WarCraft.

1.2.4 MogreFreeSL.

MogreFreeSL: es un “wrapper” MOGRE para el motor de sonido 3D FreeSL. Permite el uso de sonido 3D con su aplicación MOGRE. Es un puerto de la envoltura OgreFreeSl original. También incluye la colección de EAX 2.0 para entornos de simulación realista de audio junto con la oclusión y la obstrucción. Compilación MogreFreeSL es muy simple. Capacidad de carga hasta la solución dada para Visual Studio 2005 (no los archivos de proyecto para Visual Studio 2003 en el momento). (Junker, 2000-2009)

Ventajas. (Junker, 2000-2009)

* EAX 2.0 (puede simular entornos complejos de sonido como bajo el agua o cuevas).
* Sound Manager de Memoria de Objetos.
* Estática de WAV y Ogg.
* Flujo de reproducción Ogg y AVI (PCM).
* Soporte de archivos (puede cargar sonidos estáticos y scripts EAX de los paquetes. zip).
* Puede ser utilizado por Visual Basic y C + +.

Conclusiones.

En el presente capítulo 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. También se presentaron algunas de las Bibliotecas existentes en el mundo que servirán de cimientos en el producto final.

Con el estudio realizado del tema se tienen creadas 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.

Capítulo 2: Descripción de la Solución Propuesta.

Introducción.

En el presente capítulo se proponen soluciones técnicas para el funcionamiento del módulo de sonido y soluciones específicas para lograr su integración al Motor Gráfico OGRE, así como lograr una manipulación y reproducción eficiente de los sonidos en el entorno virtual. 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 que este posee.

2.1 Soluciones Técnicas.

Dada la necesidad de realizar mejoras al módulo de sonido de la biblioteca de audio OpenAL integrado al Motor Gráfico OGRE. Se propone la adición de efectos de sonido avanzados a dicho módulo. Básicamente se trabajará orientado a un sistema de sonido 3D posicionado, disminuyendo a su vez 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.

Luego de realizar un análisis a los diferentes formatos y formas de compresión del sonido se ha planteado el uso de los formatos autodescriptivos. Se usarán específicamente el *.WAV* y .AU ya que permiten la manipulación de sus parámetros en tiempo real, algo muy necesario para lograr un efecto realista. 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 cualquier otro formato de audio. Muchos de estos sonidos se encuentran gratis con estos formatos y posibilidad de descarga gratuita, estos formatos son compatibles tanto en Windows, Linux y Macintosh.

Para desarrollar el presente trabajo se utilizó la metodología RUP ya que sirve de guía para realizar el análisis y diseño de la aplicación. Es la metodología usada por los proyectos de la UCI, por lo que existe una amplia experiencia en su uso y posee una extensa documentación de referencia en la red y en soporte impreso. RUP hace énfasis en la documentación como elemento primordial que todo proyecto debe generar, esto propicia que en próximos ciclos de vida el equipo de desarrollo pueda retroalimentarse.

Como herramientas para el diseño e implementación se utilizará Visual Paradigm ya que es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones con mayor calidad y un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

OGRE (Object Oriented Graphics Engine) es un motor gráfico tridimensional multiplataforma, cuya principal ventaja radica en ser un proyecto de código abierto bajo licencia LGPL, lo que tiene gran peso a la hora de seleccionar las herramientas y por tanto distingue a OGRE sobre otros motores gráficos 3D. Con OGRE se puede hacer cualquier tipo de aplicación que requiera gráficos tridimensionales, no trae soporte nativo para sonido ni física lo que no es un problema, gracias a la enorme comunidad existente en Internet, cuenta con módulos especialmente diseñados para OGRE que permiten tener estas facilidades.

Como lenguaje de programación utilizaremos C++, que es un lenguaje de propósito general que se adapta a múltiples situaciones y especialmente indicado para la programación de sistemas por su flexibilidad y su potencia, además tiene una alta compatibilidad con OpenAL.

Se propone trabajar sobre la biblioteca de sonido OpenAL por ser una API a bajo nivel con facilidad de cambiar de plataforma y facilitar el uso de los componentes de efectos “environmental audio extensions” (EAX), para sonido tridimensional, orientada inicialmente a los videojuegos, pero actualmente multipropósito. 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. La API está disponible para las siguientes plataformas: Mac OS, Linux (tanto para OSS como para ALSA), *BSD, Solaris, Irix, Microsoft Windows, Sony PlayStation 2, Microsoft Xbox y Nintendo GameCube.

2.2 Modelo de dominio.

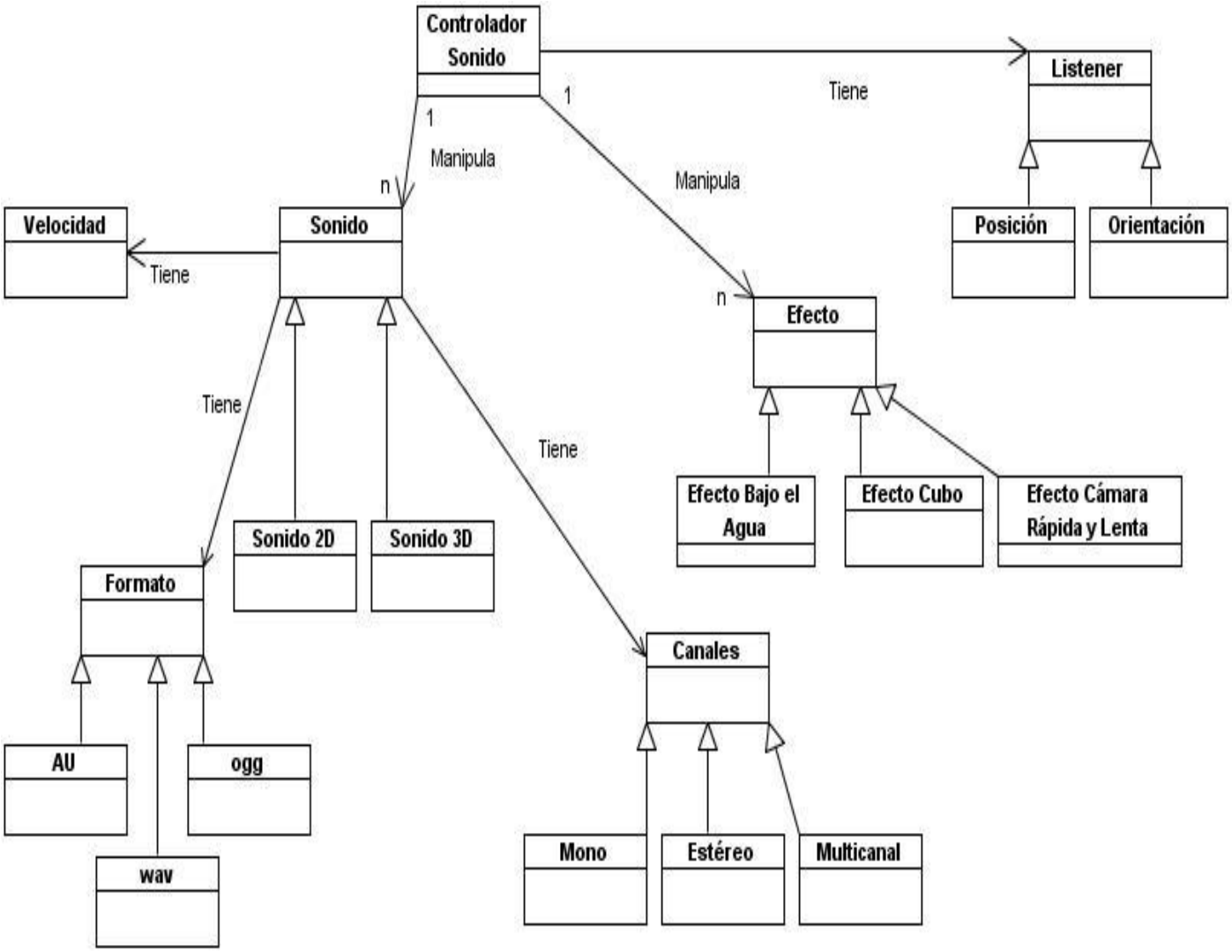

Figura 10: Modelo de clases del dominio.

2.2.1 Glosario de términos del dominio.

Controlador Sonido: Controlador o manipulador de los sonidos y sus propiedades.

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

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.

Sonido 2D: Detallada descripción de todos los elementos sonoros que el entorno necesita para su realización. Voces, sonidos ambientales, efectos sonoros y música.

Sonido 3D: Simula procesos acústicos que ocurren naturalmente en un espacio determinado el cual va a producir una interacción entre la atmósfera recreada por el entorno y nuestros oídos.

2.3 Reglas del negocio.

Los sonidos que se cargan serán de formato *wav*, *au* y *ogg*.

Cada fichero se carga una sola vez.

Posición en datos vectoriales respecto al mundo.

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.

2.4 Captura de requisitos.

2.4.1 Requisitos funcionales.

- RF1 Inicializar efectos de sonido.
- RF2 Actualizar efectos de sonido.
- RF3 Finalizar efectos de sonido.
- RF4 Posicionar sonido en espacios 3D.
- RF5 Cargar sonido 3D.
- RF6 Reproducir sonido ambiental.
- RF7 Cargar sonido ambiental.
- RF8 Cargar formatos de sonido de uso extendido (.wav, .au, .ogg).

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 del Sistema.

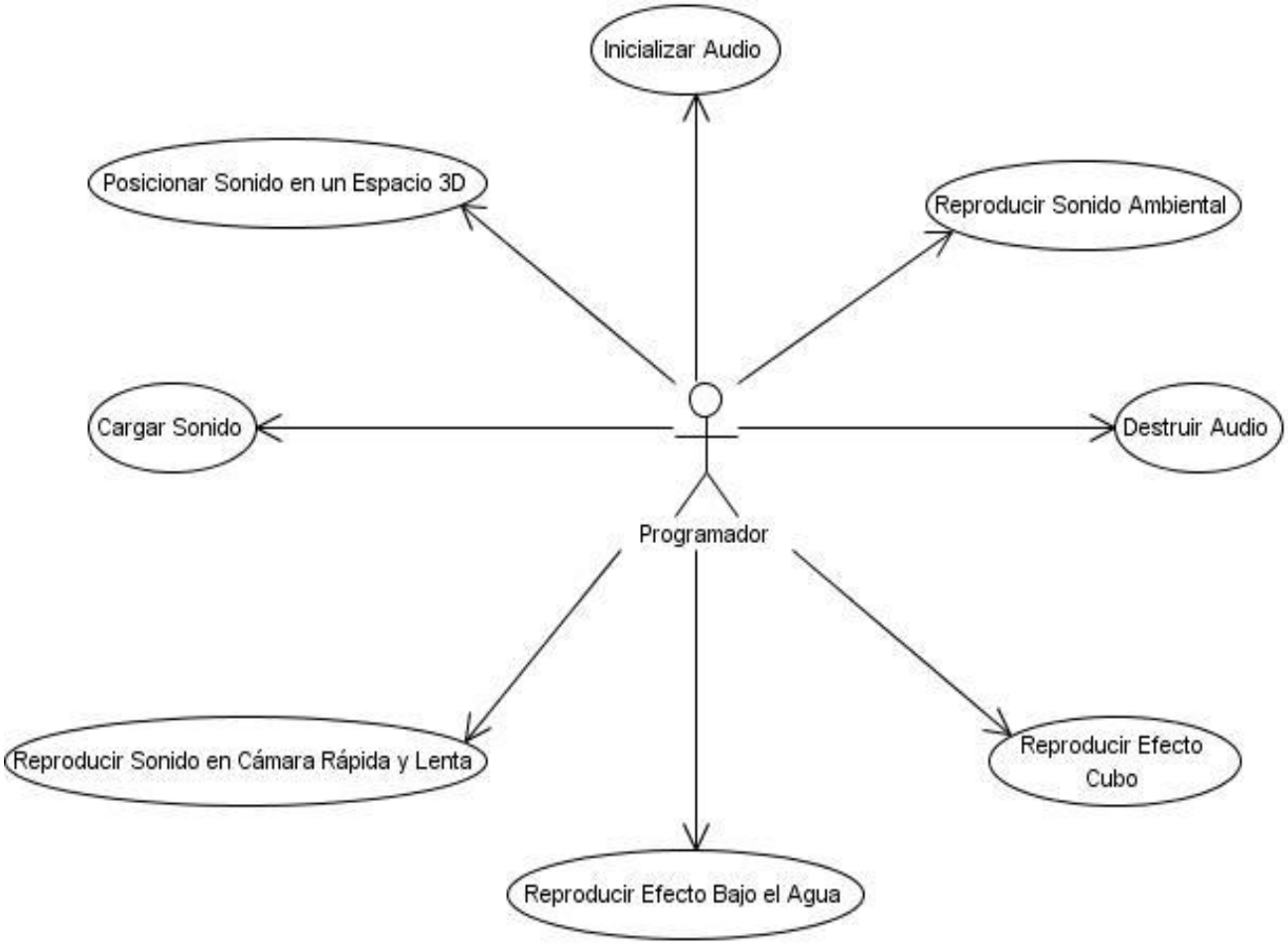

Figura 11: Diagrama de Casos de Uso del Sistema.

2.6 Expansión de los Casos de Uso.

2.6.1 Definición de los actores.

Actores	Justificación
Programador	Es quién 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 3: Caso de Uso "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 y actualiza la velocidad, posición, orientación y volumen del *Listener*.
Referencia	

Tabla 4: Caso de Uso "Cargar Sonido".

CU-2	Cargar Sonido.
Actor	Programador
Descripción	Se Cargan los datos del fichero de sonido, se crea el *buffer*, se cargan los datos a este y se crea la fuente de sonido asignándole un *buffer* existente y a este se le asigna el identificador del nivel al que pertenece dicha fuente.
Referencia	

Tabla 5: Caso de Uso "Reproducir Efecto Cubo".

CU-3	Reproducir Efecto Cubo.
Actor	Programador
Descripción	Asigna un sonido al efecto.
Referencia	

Tabla 6: Caso de Uso "Reproducir Efecto Bajo el Agua".

CU-4	Reproducir Efecto Bajo el Agua.
Actor	Programador

Descripción	Asigna un sonido al efecto.
Referencia	

Tabla 7: Caso de Uso "Reproducir Sonido en Cámara Rápida y Lenta".

CU-5	Reproducir Sonido en Cámara Rápida y Lenta.
Actor	Programador
Descripción	Asigna un sonido al efecto.
Referencia	

Tabla 8: Caso de Uso "Reproducir Sonido Ambiental".

CU-6	Reproducir Sonido Ambiental.
Actor	Programador
Descripción	Asigna un sonido.
Referencia	

Tabla 9: Caso de Uso "Posicionar Sonido en espacio 3D".

CU-7	Posicionar Sonido en espacio 3D.
Actor	Programador
Descripción	Asigna un sonido y una posición.
Referencia	

Tabla 10: Caso de Uso "Destruir Audio".

CU-8	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 expandidos.

Tabla 11: Descripción del caso de uso "Inicializar Audio".

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. El caso de uso termina cuando el dispositivo está inicializado.	
Acción del actor	Respuesta del sistema
1- Solicita inicializar el audio.	2- Inicializa velocidad de sonido.
	3- Inicializa la posición de sonido.
	4- Inicializa la orientación del sonido.
	5- Genera la fuente.
	6- Crea los buffers.
Precondiciones	
Poscondiciones	Queda inicializado el dispositivo de audio.

Tabla 12: Descripción del caso de uso "Cargar Sonido".

Caso de uso	
CU-2	Cargar Sonido.
Propósito	Cargar un sonido.
Actores: Programador.	
Resumen: El caso de uso se inicia cuando el programador solicita cargar un sonido, luego se procede a 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* está listo para usarse.	
Acción del actor	Respuesta del sistema
1- Solicita cargar un sonido.	2- Verifica dirección de archivo.
	3- Carga datos del archivo de audio.
	4- Se crea un nuevo *buffer.*

	5- Se cargan los datos al *buffer*.
Precondiciones	El fichero del sonido tenga la dirección valida.
Poscondiciones	Quedan cargados los datos en el *Buffer*.

Tabla 13: Descripción del caso de uso "Reproducir Efecto Cubo".

Caso de uso	
CU-3	Reproducir Efecto Cubo.
Propósito	Que se perciba el sonido en forma de eco.
Actores: Programador.	
Resumen: Se inicializa cuando el programador solicita reproducir efecto cubo, se carga el fichero del sonido al cual se le va aplicar el efecto. El caso de uso termina cuando se le aplica dicho efecto al sonido.	
Acción del actor	Respuesta del sistema
1- Solicita reproducir el efecto Cubo	2- Solicita cargar un sonido.
	3- Verifica datos del archivo de audio.
	4- Carga datos del archivo de audio.
	5- Crea contexto de audio.
	6- Reproduce sonido simulando el efecto.
Precondiciones	
Poscondiciones	Se simula el efecto de un eco.

Tabla 14: Descripción del caso de uso "Reproducir Sonido en Cámara Rápida y Lenta".

Caso de uso	
CU-5	Reproducir Sonido en Cámara Rápida y Lenta.
Propósito	Reproducir el sonido de forma rápida y lenta.
Actores: Programador.	

Resumen: Se inicializa cuando el programador quiere simular que se oiga un sonido en cámara rápida y lenta. Se asigna un valor que va a determinar de qué forma se va a reproducir el sonido; rápido o lento; luego se reproduce el sonido en dependencia del valor del parámetro. El caso de uso termina cuando se reproduce el efecto de sonido.	
Acción del actor	Respuesta del sistema
1- Solicita reproducir sonido en cámara rápida o lenta.	2- Solicita seleccionar opción.
	3- Selecciona opción deseada.
	4- Verifica opción deseada.
	5- Solicita cargar un Sonido.
6- Selecciona archivo de audio.	7- Carga datos del archivo de audio.
	8- Crea contexto de audio.
	9- Reproduce sonido simulando el efecto.
Precondiciones	
Poscondiciones	Se reproduce sonido simulando el efecto deseado.

Tabla 15: Descripción del caso de uso "Reproducir Sonido Ambiental".

Caso de uso	
CU-6	Reproducir Sonido Ambiental.
Propósito	Reproducir Sonido Ambiental.
Actores: Programador.	
Resumen: Se inicializa cuando el programador quiere reproducir un sonido ambiental. El caso de uso termina cuando se reproduce el sonido ambiental.	
Acción del actor	Respuesta del sistema
1- Solicita cargar un sonido ambiental.	2- Solicita cargar datos del archivo de audio.

	3- Verifica datos del archivo de audio.
	4- Carga datos del archivo de audio.
	5- Crea contexto de audio.
	6- Reproduce sonido ambiental.
Precondiciones	
Poscondiciones	Se reproduce sonido ambiental.

Tabla 16: Descripción del caso de uso "Posicionar Sonido en espacio 3D".

Caso de uso	
CU-7	Posicionar Sonido en espacio 3D.
Propósito	Posicionar Sonido en espacio 3D.
Actores: Programador.	
Resumen: El caso de uso se inicia cuando el programador solicita posicionar un sonido en un espacio 3D, entonces se cambia la posición vector de la cual se quiere colocar el sonido en el entorno virtual, el caso de uso termina cuando el sonido ha sido posicionado.	
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 17: Descripción del caso de uso "Destruir Audio".

Caso de uso	
CU-8	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 *buffers* y fuentes existentes. Termina cuando se liberó el dispositivo de audio y todos los recursos que este ocupaba.	
Acción del actor	Respuesta del sistema
1-Solicita Destruir Audio.	2- Se destruyen la fuentes.
	3- Se destruye el *buffer*.
	4- Se destruye el contexto.
Precondiciones	Audio inicializado.
Poscondiciones	Liberan los recursos de memoria y hardware.

Conclusiones.

En este capítulo queda planteada la situación técnica por la que se regirá el sistema, satisfaciendo las necesidades del usuario. Para ello se establecieron los requisitos funcionales del mismo y la descripción de los casos de uso para que los usuarios puedan interactuar fácilmente con el sistema. Partiendo de todo esto se diseñará una estructura de clases en correspondencia con las técnicas mencionadas anteriormente y para lograr los objetivos del proyecto.

Capítulo 3: Diseño e Implementación.

Introducción.

En el presente capítulo se muestra el diseño del sistema propuesto con el diagrama de clases donde se definen las responsabilidades de estas y sus relaciones. Se muestran otros artefactos involucrados en el diseño como los diagramas de secuencia por casos de uso facilitando con esto el entendimiento del sistema.

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

3.1. Diagrama de clases de diseño.

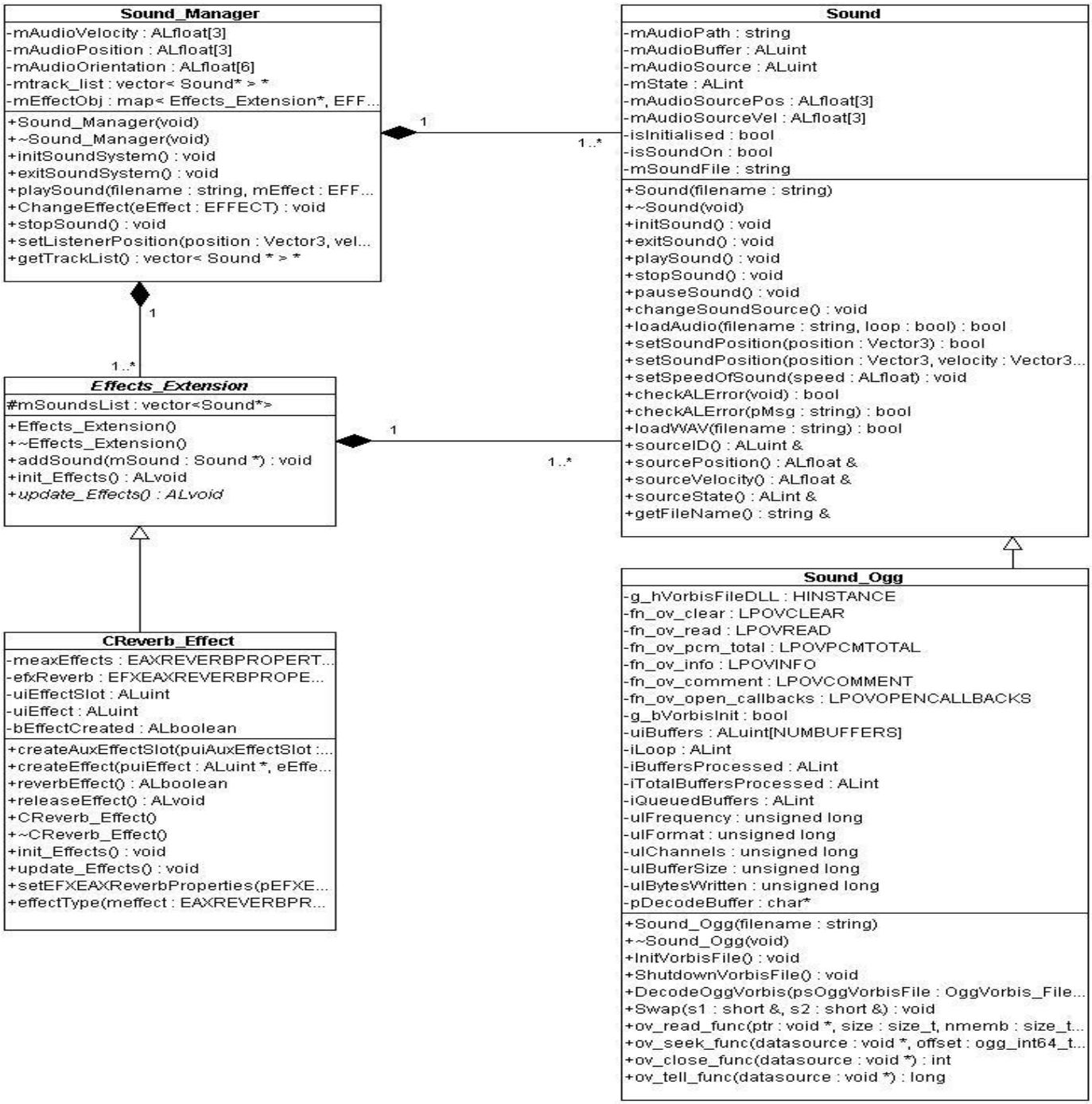

Figura 12: Diagrama de Clases.

3.2. Descripción de las clases de diseño.

Tabla 18: Descripción de Sound_Manager.

Nombre: Sound_Manager	
Controladora	
Atributo	Tipo
mtrack_list	vector<Sound*> *
mEffectObj	map<Effects_Extension*,EFFECT> *
mAudioVelocity [3]	ALfloat
mAudioPosition [3]	ALfloat
mAudioOrientation[6]	ALfloat
Para cada responsabilidad:	
Nombre:	~Sound_Manager(void)
Descripción:	Destructor de la clase OGG.
Nombre:	Sound_Manager(void)
Descripción:	Constructor de la clase OGG.
Nombre:	initSoundSystem()
Descripción:	Inicializa el sistema de sonido.
Nombre:	exitSoundSystem()
Descripción:	Finaliza el sistema de sonido.
Nombre:	PlaySound(std::string filename,EFFECT mEffect = EFFECT_NONE)
Descripción:	Reproduce un efecto de sonido.
Nombre:	ChangeEffect(eEffect : EFFECT)
Descripción:	Cambia el efecto en reproducción.
Nombre:	stopSound()
Descripción:	Detiene el sonido en reproducción.
Nombre:	setListenerPosition(position : Vector3, velocity : Vector3, orientation : Quaternion)
Descripción:	Cambia la posición del Listener.
Nombre:	getTrackList()

Descripción:	Devuelve la lista de sonidos.

Tabla 19: Descripción de Sound.

Nombre: Sound	
Controladora	
Atributo	Tipo
mAudioBuffers	ALint
mAudioSource	ALuint
mAudioVelocity[3]	ALfloat
mAudioSourcePos [3]	ALfloat
mAudioOrientation[6]	ALfloat
mAudioSourceVel[3]	ALfloat
mAudioPosition[3]	ALfloat
mSoundContext	ALCcontext*
mSoundDevice	ALCdevice*
isInitialised	bool
mSoundFile	string
isSoundOn	bool
Para cada responsabilidad:	
Nombre:	~Sound (void)
Descripción:	Destruye de la instancia de objeto creada de la clase.
Nombre:	initSound()
Descripción:	Inicializa el contexto de sonido.
Nombre:	exitSound()
Descripción:	Finaliza el contexto de sonido.
Nombre:	loadAudio(std::string filename, bool loop)
Descripción:	Abre un archivo de audio.
Nombre:	loadAudioInToSystem(std::string filename)
Descripción:	Abre un archivo de audio del sistema.

Nombre:	checkALError(void)
Descripción:	Chequea un error de código.
Nombre:	checkALError(std::string pMsg)
Descripción:	Chequea un error de código de tipo string.
Nombre:	ChangeSoundSource()
Descripción:	Cambia la fuente de un sonido.
Nombre:	loadWAV(std::string filename, ALuint pDestAudioBuffer)
Descripción:	Abre archivos de audio .wav.
Nombre:	pauseSound()
Descripción:	Pone en pausa el archivo de sonido que está siendo reproducido.
Nombre:	playSound()
Descripción:	Reproduce un archivo de audio.
Nombre:	setListenerPosition(Vector3 position, Vector3 velocity, Quaternion orientation)
Descripción:	Permite cambiar la dirección del oyente.
Nombre:	setSoundPosition(Vector3 position, Vector3 velocity, Vector3 direction)
Descripción:	Permite cambiar la posición de un sonido y sus propiedades.
Nombre:	setSoundPosition(Vector3 position)
Descripción:	Permite cambiar la posición de un sonido.
Nombre:	Sound(std::string filename)
Descripción:	Constructor de la clase Sound.
Nombre:	stopSound()
Descripción:	Detiene el archivo de sonido en reproducción y lo pone al comienzo de la pista.

Tabla 20: Descripción de Sound_Ogg.

Nombre: Sound_Ogg	
Entidad	
Atributo	Tipo
g_hVorbisFileDLL	HINSTANCE
fn_ov_clear	LPOVCLEAR

fn_ov_read	LPOVREAD
fn_ov_pcm_total	LPOVPCMTOTAL
fn_ov_info	LPOVINFO
fn_ov_comment	LPOVCOMMENT
fn_ov_open_callbacks	LPOVOPENCALLBACKS
g_bVorbisInit	bool
uiBuffers[NUMBUFFERS]	ALuint
iLoop	ALint
iBuffersProcessed	ALint
iTotalBuffersProcessed	ALint
iQueuedBuffers	ALint
ulFrequency	unsigned long
ulFormat	unsigned long
ulChannels	unsigned long
ulBufferSize	unsigned long
ulBytesWritten	unsigned long
pDecodeBuffer	char*
Para cada responsabilidad:	
Nombre:	~Sound_Ogg (void)
Descripción:	Destruye de la instancia de objeto creada de la clase.
Nombre:	InitVorbisFile()
Descripción:	Inicializa el contexto de sonido.
Nombre:	ShutdownVorbisFile()
Descripción:	Finaliza el contexto de sonido.
Nombre:	DecodeOggVorbis(psOggVorbisFile : OggVorbis_File *, pDecodeBuffer : char *, ulBufferSize : unsigned long, ulChannels : unsigned long)
Descripción:	Decodifica un archivo de audio con extensión .ogg.
Nombre:	Swap(s1 : short &, s2 : short &)
Descripción:	Intercambia los valores s1 y s2 de un archivo de audio.

Nombre:	ov_read_func(ptr : void *, size : size_t, nmemb : size_t, datasource : void *)
Descripción:	Función que lee la codificación de un archivo .ogg.
Nombre:	ov_seek_func(datasource : void *, offset : ogg_int64_t, whence : int)
Descripción:	Función que ofrece el seguimiento de un archivo con la codificación .ogg.
Nombre:	ov_close_func(datasource : void *)
Descripción:	Función que finaliza la lectura de un archivo .ogg.
Nombre:	ov_tell_func(datasource : void *)
Descripción:	Función que realiza llamadas a la codificación de los archivos .ogg.

Tabla 21: Descripción de CReverb_Effect.

Nombre: CReverb_Effect	
Entidad	
Atributo	Tipo
eaxBathroom	EAXREVERBPROPERTIES
eaxHangar	EAXREVERBPROPERTIES
efxReverb	EFXEAXREVERBPROPERTIES
uiEffectSlot	ALuint
uiEffect	ALuint
bEffectCreated	ALboolean
Para cada responsabilidad:	
Nombre:	~CReverb_Effect()
Descripción:	Destructor de la clase CReverb_Effect().
Nombre:	CReverb_Effect(EAXREVERBPROPERTIES eaxBathroom)
Descripción:	Constructor de la clase CReverb_Effect
Nombre:	init_Effects()
Descripción:	Inicializa el contexto de efectos.
Nombre:	exit_Effects()
Descripción:	Finaliza el contexto de efectos.
Nombre:	update_Effects(ALfloat timesteps)

Descripción:	Actualiza el contexto de efectos.
Nombre:	setEFXEAXReverbProperties(EFXEAXREVERBPROPERTIES *pEFXEAXReverb, ALuint uiEffect)
Descripción:	Cambia las propiedades de un efecto.
Nombre:	createAuxEffectSlot(ALuint *puiAuxEffectSlot)
Descripción:	Crea un Slot de efecto auxiliar.
Nombre:	createEffect(ALuint *puiEffect, ALenum eEffectType)
Descripción:	Crea un efecto.
Nombre:	reverbEffect(ALuint m_uiSource)
Descripción:	Crea el efecto reverberación.
Nombre:	releaseEffect(ALuint m_uiSource)
Descripción:	Detiene un efecto.

Tabla 22: Descripción de Effects_Extension.

Nombre: Effects_Extension	
Controladora	
Atributo	Tipo
mSondsList	vector<Sound*> *
Para cada responsabilidad:	
Nombre:	~Effects_Extension()
Descripción:	Destructor de la clase Effects_Extension.
Nombre:	Effects_Extension()
Descripción:	Constructor de la clase Effects_Extension.
Nombre:	addSound(Sound * mSound)
Descripción:	Adiciona un sonido.
Nombre:	init_Effects(ALint uiSource)
Descripción:	Inicializa el sistema de efectos.
Nombre:	exit_Effects()
Descripción:	Finaliza el sistema de efectos.

Nombre:	update_Effects(ALfloat timesteps)
Descripción:	Actualiza el sistema de efectos.

3.3. Diagramas de secuencia.

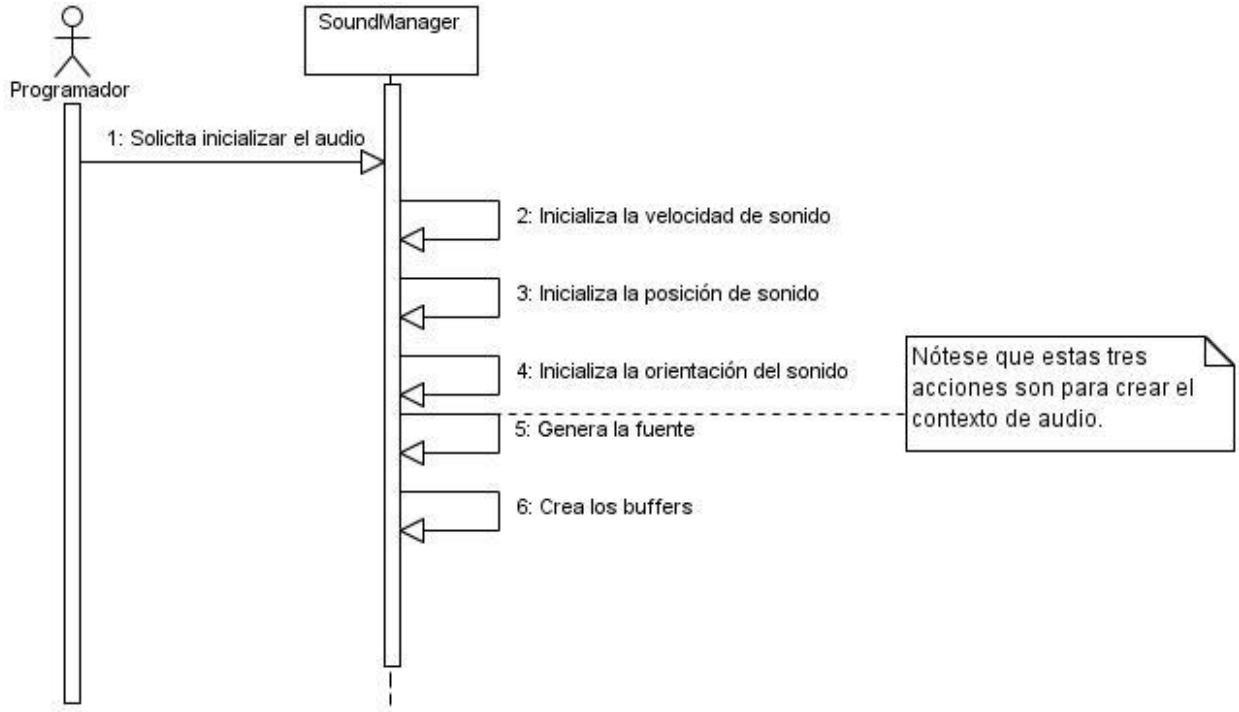

Figura 13: Diagrama de Secuencia Inicializar audio.

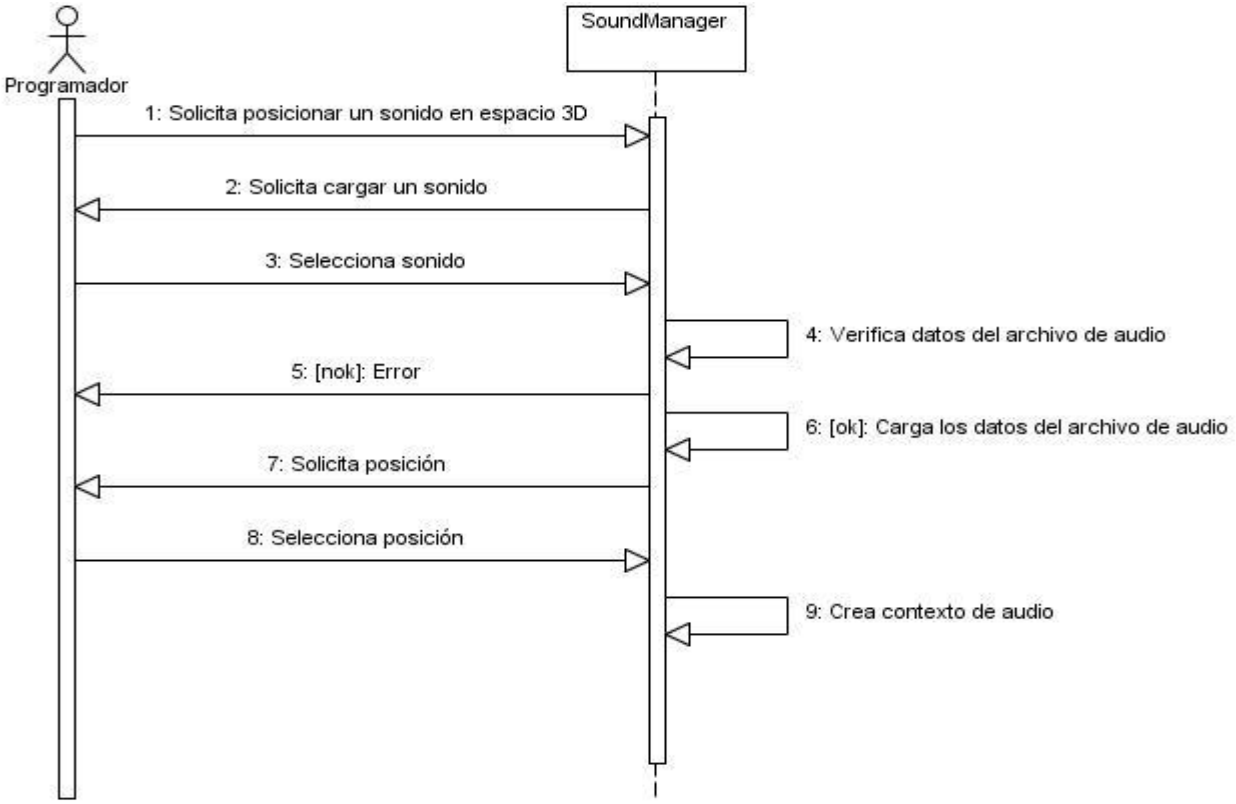

Figura 14: Diagrama de Secuencia Posicionar Sonido en espacio 3D.

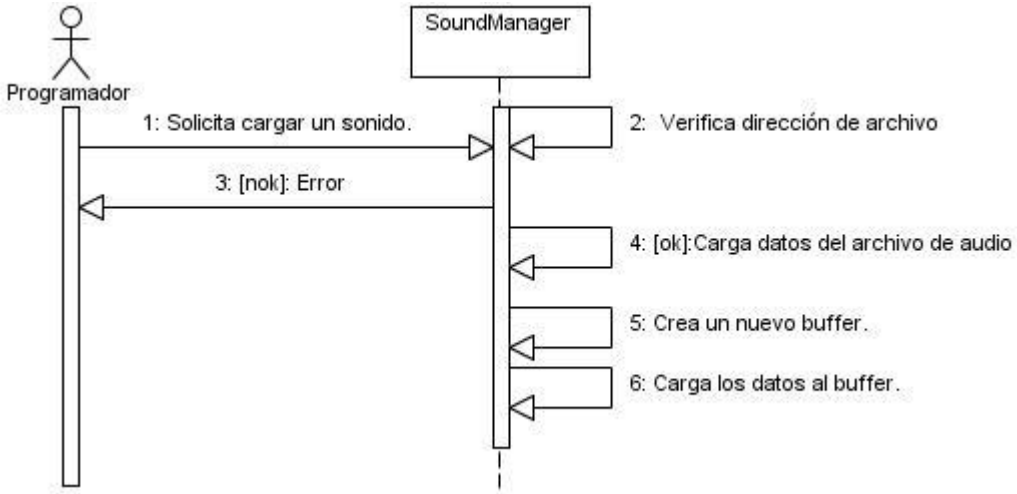

Figura 15: Diagrama de Secuencia Cargar Sonido

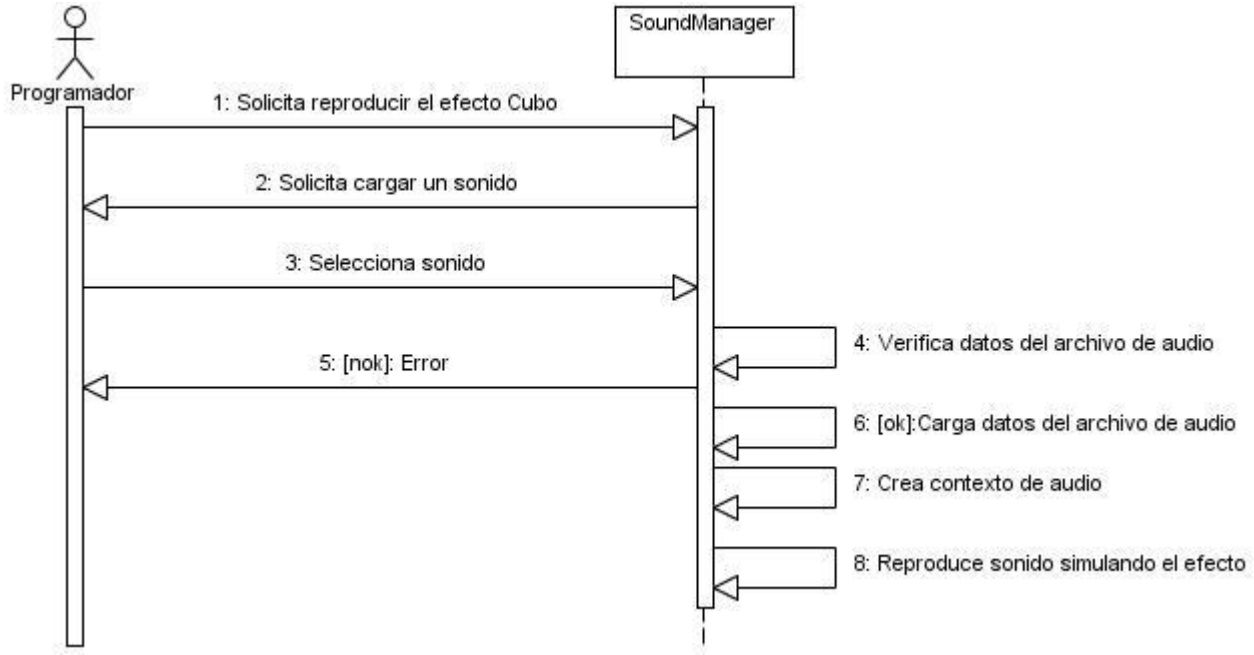

Figura 16: Diagrama de Secuencia Reproducir Efecto Cubo.

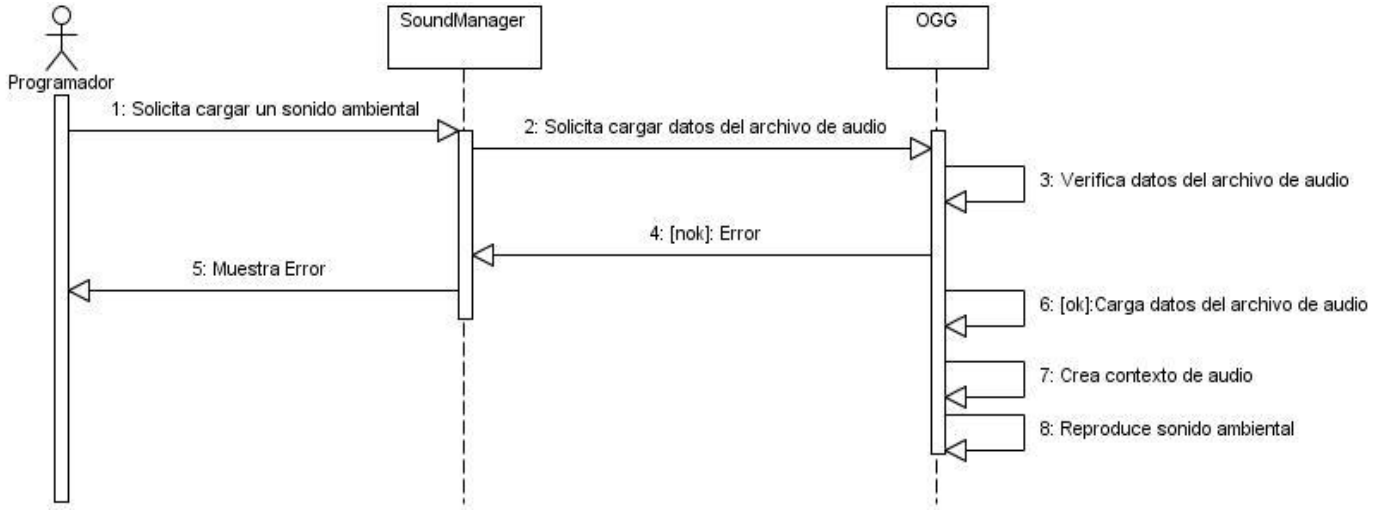

Figura 17: Diagrama de Secuencia Reproducir Sonido Ambiental.

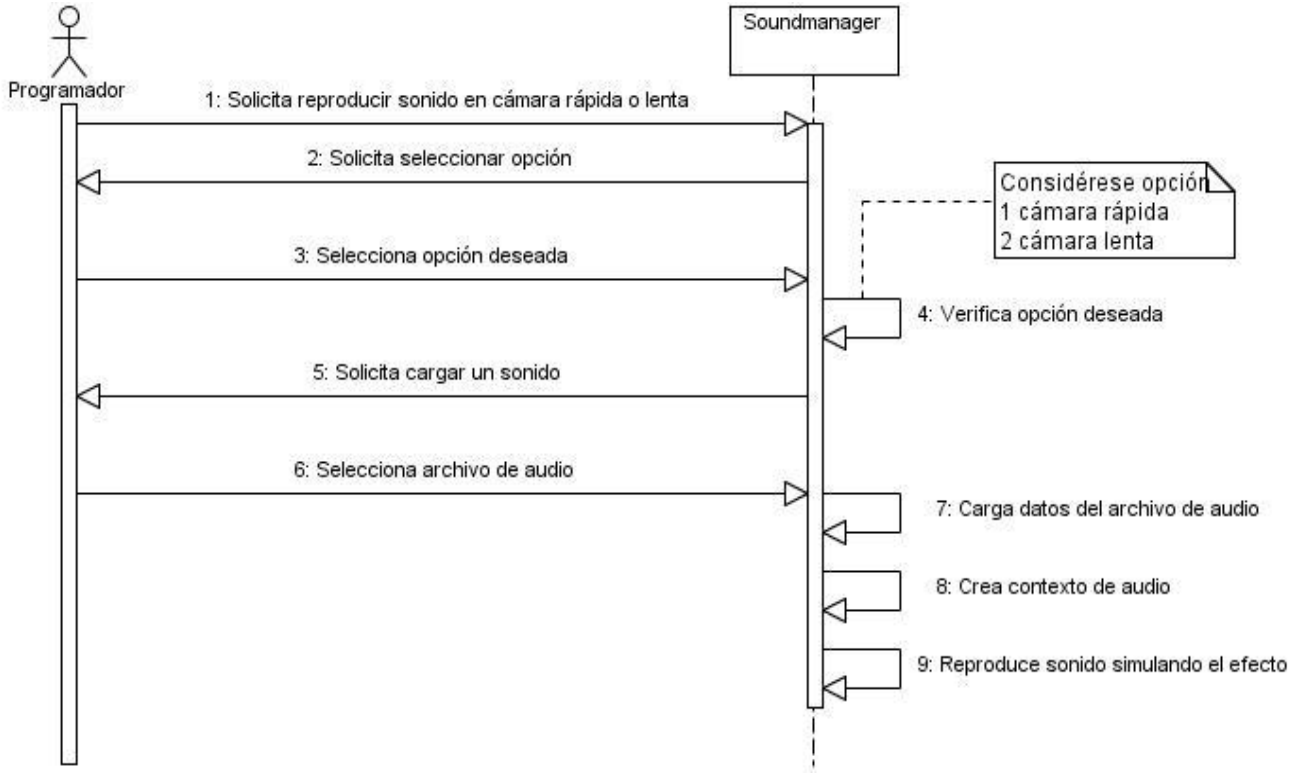

Figura 18: Diagrama de Secuencia Reproducir Sonido en Cámara Rápida y Lenta.

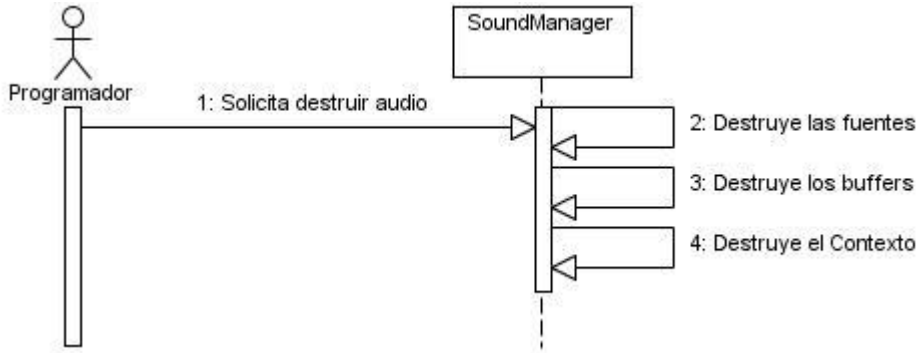

Figura 19: Diagrama de Secuencia Destruir Audio.

3.4. Diagrama de componentes.

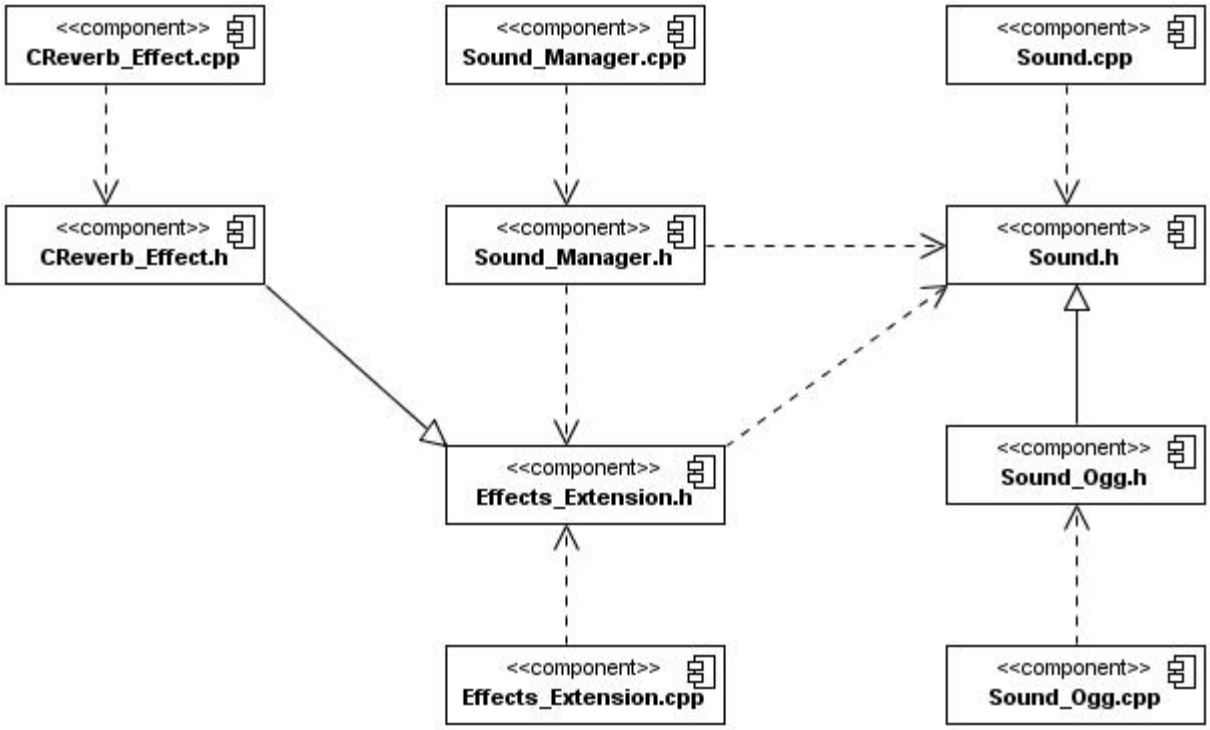

Figura 20: Diagrama de Componentes.

Conclusiones.

Durante el presente capítulo se mostraron los artefactos necesarios para el desarrollo del sistema de efectos de sonidos avanzados. Al finalizar el mismo se tiene concebido detalladamente el diseño completo del sistema y la mensajería entre las clases de los casos de usos a desarrollar que se hace en forma secuencial.

Conclusiones.

Se logró incorporar nuevas funcionalidades al módulo OpenALSoundManager, las cuales pueden ser usadas tanto para la elaboración de simuladores como para videojuegos desarrollados en la UCI. Las funcionalidades adicionadas son totalmente funcionales, cumplen los requisitos planteados y permite darle efectos a los sonidos en los entornos virtuales de forma fácil al programador para lograr un alto nivel de realismo.

Para satisfacer las necesidades planteadas se realizó un estudio acerca de los diferentes módulos de sonidos integrados al OGRE, además se trataron conceptos sobre los módulos de sonidos para tener un amplio conocimiento del tema. Se hizo un estudio de las técnicas, tecnologías y tendencias en cuanto al uso de los sonidos en los Sistemas de Realidad Virtual a pesar de que muchos de los módulos estudiados satisfacían las necesidades planteadas tenían como inconveniente que estaban bajo una licencia propietaria.

Recomendaciones.

Se recomienda:

• Aumentar el número de formatos soportados.

• Ampliar los tipos de efectos que se pueden usar en dependencia del hardware en que se utilice.

• Crear una herramienta visual para configurar las posiciones que ocuparán los sonidos en un entorno.

• Crear documentación de ayuda y especificaciones.

Referencia Bibliográfica.

(Díaz, 2010) Díaz, E. D. (28 de mayo de 2010). *Glosario de términos informáticos y otros*. Recuperado el 2010, de Glosario de términos informáticos y otros: http://www.joomla-gnu.com/glosario.htm

(Junker, 2000-2009) Junker, G. (2000-2009). *OGRE.* Recuperado el 2010, de OGRE: http://www.ogre3d.org/documentation

(Creative Labs, 2008) Labs, C. (2008). *OpenAL*. Recuperado el 2010, de OpenAL: http://connect.creativelabs.com/openal/default.aspx

(Lazaro Abreu Reche, 2007) Lazaro Abreu Reche, D. M. (mayo de 2007). *Biblioteca UCI.* Recuperado el 2010, de Biblioteca UCI: http://biblioteca.reduc.edu.cu/biblioteca.virtual/cgi/.../TrabajoUCIciencia.pdf

(Loureiro, 2010) Loureiro, J. (2010). *Codepixel*. Recuperado el 2010, de Codepixel: http://www.codepixel.com/content/category/4/57/32/

(MENCÍAS, 2009) MENCÍAS, F. J. (mayo de 2009). *e-archivo Universidad Carlos III de Madrid.* Recuperado el 2010, de e-archivo Universidad Carlos III de Madrid: http://e-archivo.uc3m.es/bitstream/10016/7462/2/PFC_FranciscoJavier_Martinez_Mencias.pdf

(WorldLingo, 2010) WorldLingo. (2010). *WorldLingo*. Recuperado el 2010, de WorldLingo: http://www.worldlingo.com/ma/enwiki/es/3D_audio_effect

Bibliografía Consultada.

FASANI, I. J. (2004a. [2006].). *codepixel*. Recuperado el 25 de enero de 2010, de codepixel: http://www.codepixel.com/tutoriales/sonido2/.

PERIS, F. B. *Seminario OpenAL: sonido para videojuegos.*, 2003. [2006]. Disponible en:

http://personales.alumno.upv.es/~ferblape/formacion

FASANI, I. J. L. F. *Análisis Sonido 3D*, CodePixel ©, 2004a. [2006]. Disponible en:

http://www.codepixel.com/tutoriales/sonido2/

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

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

Glosario de Abreviaturas.

ALC (Automatic Level Control): control automático de nivel, se usa en audio para mantener constante el nivel de volumen de sonido.

API: Interfaz de Programación de Aplicaciones o API (del inglés application programming interface).

HRTF: Funciones de Transferencia Relativas (Head Related Transfer Functions HRTFs).

EAX: Componentes de efectos (environmental audio extensions).

Glosario de Términos.

Biblioteca: en ciencias de la computación, una **biblioteca** (o **librería**) es un conjunto de **subprogramas** utilizados para desarrollar **software.**

Buffer: es una ubicación de la memoria en una computadora o en un instrumento digital reservada para el almacenamiento temporal de información digital, mientras que está esperando ser procesada.

Delay: es un efecto de sonido que consiste en la multiplicación y retraso modulado de una señal sonora. Una vez procesada la señal se mezcla con la original. El resultado es el clásico efecto de eco sonoro.

DirectX: es una colección de programas que acelera el sistema en las tareas gráficas. Un componente es "Direct-3D". Con él se puede, bajo Windows, jugar de un modo más rápido y cómodo. Otros programas son, por ejemplo, "Direct-Sound" para la reproducción de sonido, así como el "Direct-Draw" y "Direct-Video" para la representación de dibujos y vídeo.

Funciones: son una rutina de software independiente que realiza una tarea para el programa en que está escrita o para algún otro programa. La función ejecuta la operación y devuelve el control a la instrucción siguiente a la que la llamó o al programa que la llamó. Los lenguajes de programación proveen un conjunto de funciones estándares y permiten a los programadores definir otras. Por ejemplo, el lenguaje C está completamente construido alrededor de funciones.

Microsoft Corporation, Redmond, WA: compañía de software más grande del mundo. Microsoft fue fundada en 1975 por Paul Allen y Bill Gates, dos estudiantes universitarios que escribieron el primer intérprete BASIC para el microprocesador 8080 de Intel. Aunque también se conoce por sus lenguajes de programación y aplicaciones para computadores personales, el éxito sobresaliente de Microsoft se debe a sus sistemas operativos DOS y Windows.

Glosario de Términos

Programa: es un conjunto de instrucciones u órdenes que indican a la máquina las operaciones que ésta debe realizar con unos datos determinados. En general, todo programa indica a la computadora cómo obtener unos datos de salida, a partir de unos datos de entrada.

Procedimientos: se denomina procedimiento al conjunto de instrucciones, controles, etc. que hacen posible la resolución de una cuestión específica. La impresión es un procedimiento, como lo es la incorporación de una imagen a un texto predeterminado, etc.

Puntero: es una variable que referencia una región de memoria; en otras palabras es una variable cuyo valor es una dirección de memoria.

Software: se refiere al equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (hardware).

Sonido estereofónico (**estéreo):** es el sonido que está grabado y reproducido en dos canales , donde, al menos en parte, se reproduce la dirección (izquierda y derecha) de donde proviene cada fuente de sonido grabada.

Sonido monoaural (mono): es el sonido que sólo está definido por un canal (ya sea una grabación captada con un solo micrófono o bien una mezcla final) y que origina un sonido semejante al escuchado con un solo oído.

Subprograma: es un **programa**, el cual, es llamado desde otro programa o subprograma.

Tarjeta de los sonidos: es un dispositivo de hardware que sirve como expansión de las posibilidades que brindan las computadoras, permitiendo la salida o entrada de información en forma de audio.

Wrapper: contenedor o envoltura.

Glosario de Términos

API: es un interfaz de programación de aplicaciones, conjunto de funciones y procedimientos o métodos en la programación orientada a objetos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Usados generalmente en las bibliotecas.

DSSI: es un instrumento virtual (sintetizador del software) arquitectura plug-in para uso de secuenciador de la música. Fue diseñado para los usos que funcionaban bajo Linux, aunque no hay nada específico en Linux. se distribuye de conformidad con una combinación de GNU y algunos Licencias del DEB, que son licencias libres del software.

Índice de Tablas.

Índice de Figuras.