ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA DE TELECOMUNICACIÓ LA SALLE

TREBALL FINAL DE MÀSTER

MÀSTER EN ENGINYERIA DE XARXES I TELECOMUNICACIONS

AUDIO TRIDIMENSIONAL SOBRE INTERFAZ NATURAL

ALUMNE
Julià Vicens Bennasar

PROFESSOR PONENT
Carles Vilella i Parra

ACTA DE L'EXAMEN
DEL TREBALL FINAL DE MÀSTER

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D. Julià Vicens Bennasar

va exposar el seu Treball Final de Màster, el qual va tractar sobre el tema següent:

AUDIO TRIDIMENSIONAL SOBRE INTERFAZ NATURAL

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

AUDIO TRIDIMENSIONAL SOBRE INTERFAZ NATURAL

Abstract

En este trabajo se arma un interfaz natural de usuario, *table-top user interface*, capaz de detectar fiduciales y comunicar la información que representa cada uno de ellos a una aplicación de audio. Para ello se estudian diferentes técnicas hardware y software que permiten la construcción e interacción hombre – máquina y se desarrolla la más adecuada para alcanzar los objetivos propuestos. Así mismo se realizada un aplicación de audio tridimensional que reacciona en tiempo real a los eventos que se dan sobre la interfaz natural. Esta aplicación se implementa sobre los fundamentos del audio binaural.

El sistema al completo es sensible a objetos marcados con fiduciales (marcadores) que representan fuentes sonoras en un entorno virtual. Un receptor percibe la posición de las fuentes en ese espacio en relación al movimiento que los objetos ejercen sobre la interfaz física.

Julià Vicens Bennasar.

Junio 2011.

Resumen

En este trabajo se lleva a cabo la implementación de un sistema de audio tridimensional que responde a eventos sobre una interfaz natural de usuario.

La interfaz que se desarrolla es del tipo *table-top user interface* (*TUI*). Funciona mediante un software de visión por computador, *reacTIVision*, que detecta los fiduciales (marcadores) adheridos a objetos físicos situados sobre de la mesa. El protocolo *TUIO*, especifico para aplicaciones de audio, es el que permite la comunicación entre el software de reconocimiento y la aplicación cliente.

La aplicación cliente es un sistema de audio tridimensional, a priori se desarrolla utilizando librerías libres para el procesado de audio, concretamente audio 3D, como opeanAL, fmod... Debido a las limitaciones de estas para trabajar en multiplataforma con todas las funcionalidades, se opta por utilizar una serie de librerías que usan técnicas binaurales haciendo uso de la auralización, ITD/ILD, HTRF, reverberadores... De esta manera se modelan fuentes, entorno y receptor.

El sistema funciona en tiempo real mediante un lenguaje de programación orientado a objetos, Java. El hecho de trabajar en tiempo real con Java implica una serie de limitaciones en el procesado de audio directamente relacionadas con el coste computacional de este, todo ello se encuentra debidamente razonado en las páginas que siguen.

Índice

Índice de figuras y tablas

Figuras

Tablas

0. Introducción

Se explican los objetivos del proyecto en su conjunto, así como las principales puntos de enfoque y el planteamiento de la solución genérica del mismo.

0.1. Objetivos

El objetivo general de este proyecto es crear un entorno físico que interactúa con un sistema de audio digital en 3D. Por un lado se desarrolla una interfaz natural que permite, mediante objetos tangibles, modificar un entorno virtual y por tanto intangible. Este medio virtual es un sistema que permite reconocer la posición de las fuentes de audio en función de la situación de los objetos en el medio físico.

Para la implementación de la interfaz es necesario conocer y estudiar las diferentes tecnologías disponibles para llevar a cabo su implementación. Concretamente nos centramos en las interfaces naturales de usuario que permiten una interacción intuitiva y natural con la máquina. La idea es construir una mesa interactiva que sea capaz de reconocer una serie de marcadores o fiduciales enganchados a objetos que son el elemento tangible que relaciona el entorno físico con el entorno virtual. Es imprescindible por tanto evaluar la conveniencia e inconveniencia de toda nueva interfaz tangible y la viabilidad de su implementación bajo estas condiciones.

El medio virtual es un sistema de audio digital tridimensional donde se localizan varios sonidos. Por tanto, es necesario situar dos tipos de objetos encima de la superficie, un receptor y una serie de fuentes sonoras que interaccionan con este. El receptor, debidamente identificado, situado en un punto determinado de la superficie y por tanto del entorno virtual, recibirá los sonidos representados y localizados por la posición de los objetos que caracterizan las fuentes. A su vez, emisores y receptores se encuentran en una sala virtual. Cabe destacar que el hecho de situar objetos físicos encima de una superficie implica perder una dimensión, puesto que se mueven en el mismo plano.

La meta es que, en tiempo real, se cree un entorno de audio en el que se localiza la posición de cada fuente en relación con el receptor según

sea la situación de los objetos encima de la superficie tangible. Se pueden ir modificando las posiciones de todos los objetos e incluso añadirlos y eliminarlos del entorno.

Es primordial tener presente las limitaciones que implica el hecho de trabajar en tiempo real, tanto por lo que se refiere al rendimiento de la interfaz y reconocimiento de los objetos, como por el procesado del audio digital. Todo ello afecta a la latencia del sistema y limita las características del proyecto, será preciso conocer el uso de los recursos del sistema donde se implementa.

Se trabaja a nivel de interfaz y a nivel de procesado de audio. Por tanto, si extrapolamos ese concepto, también podremos hacer lo propio en otras interfaces y con las librerías apropiadas para cada una de ellas.

En definitiva, se trata de un proyecto conceptual en el que se pretende dar una visión de cómo puede funcionar en tiempo real un sistema de audio tridimensional en función de las órdenes que va recibiendo desde una interfaz natural.

0.2. Planteamiento

El desarrollo del proyecto se lleva a cabo en dos partes. En primer lugar todo lo que tiene que ver con las interfaces de usuario, y en segundo lugar lo referente al audio digital. Para unir la interfaz con el audio es necesario un protocolo que permita la comunicación, en tiempo real, de ambas partes.

Figura 0.1: Diagrama de bloques planteamiento genérico

Las nuevas interfaces de usuario se pueden implementar utilizando un conjunto muy diverso de tecnologías (microcontroladores, sensores, visión por computador, realidad aumentada...). La idea es construir una *table-top*, superficie tangible, en la que posicionar una serie de objetos que representen los sonidos en el medio virtual y que sea multiusuario, es decir, que pueda interactuar con la interfaz más de una persona. Este concepto implica decantarse por la utilización de un sistema que funcione con técnicas de visión por computador, donde se reconocen una serie de imágenes que se identifican de alguna manera con una serie de elementos virtuales.

Hay otras muchas interfaces que permiten realizar el mismo experimento, sin embargo en este momento esta interfaz dispone de software CreativeCommons para realizar la parte de visión por

computador y la comunicación con el ordenador. Además es muy intuitiva y representa de forma muy cómoda la posición y movimiento de los objetos, a este respecto la única limitación que tiene es que el audio siempre estará situado en la horizontal, lo que provoca la perdida de una dimensión.

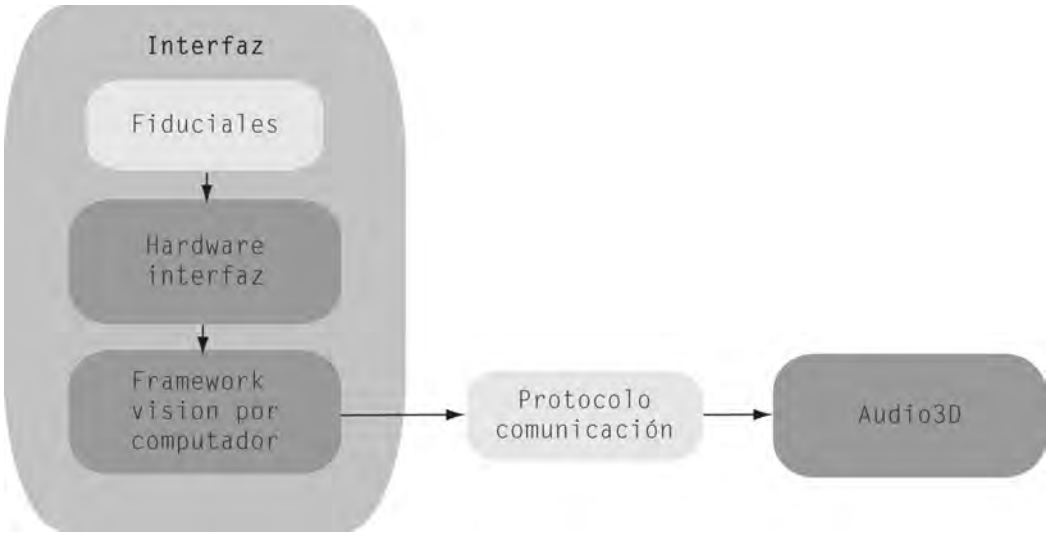


Figura 0.2: Diagrama de bloques planteamiento genérico interfaz

Es necesario estudiar las principales técnicas de implementación de NUI de una forma realista, así como la elección del software de visión por computador y los protocolos de comunicación. La principal misión del software es reconocer fiduciales, para esto se utilizará reacTIVision. La comunicación de este framework con la aplicación que recibirá los mensajes se realiza mediante el protocolo TUIO, construido adrede para la comunicación en aplicaciones musicales o de audio, una evolución del OSC que a su vez es la alternativa de MIDI.

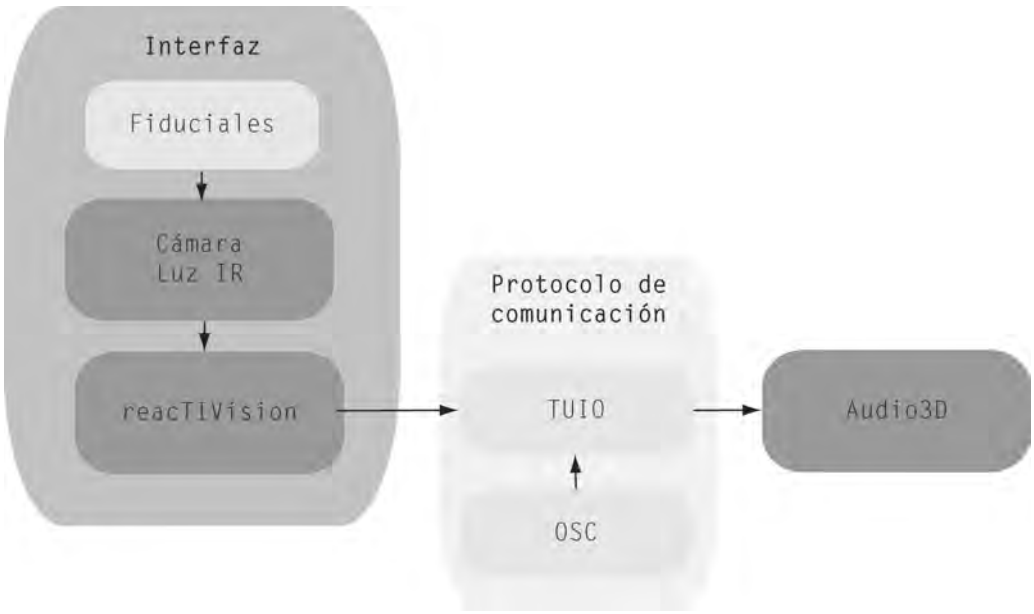


Figura 0.3: Diagrama de bloques planteamiento genérico, herramientas interfaz

En el medio virtual se desarrolla una aplicación de audio digital en 3D que permite localizar en un determinado espacio la posición de diversos sonidos. Se decide utilizar como lenguaje de programación Java por el reto que supone procesar audio con este lenguaje que tal vez no sea el más idóneo para este propósito. Aún así se dispone de varias librerías que permiten implementar audio 3D en este lenguaje.

Otros lenguajes de programación que pueden ser útiles para desarrollar la parte sonora son Processing, Python y sobretodo C++ para el cual hay varias librerías interesantes.

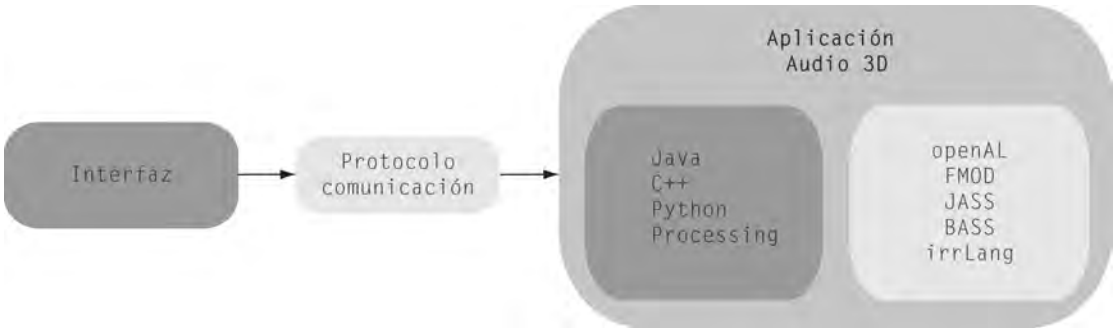


Figura 0.4: Diagrama de bloques planteamiento genérico bibliotecas audio 3D

Otra opción es realizar una librería propia que utilice la metodología habitual para crear un entorno en 3D; auralización, HRTF, cálculo de las ITD/ILD, reverberador, sala... Esta implementación implica una programación eficiente para que la latencia del sistema sea baja.

La información de los objetos, básicamente posición y orientación, se representará mediante un interfaz gráfica en la pantalla del ordenador para tener una referencia visual de lo que sucede encima de la mesa.

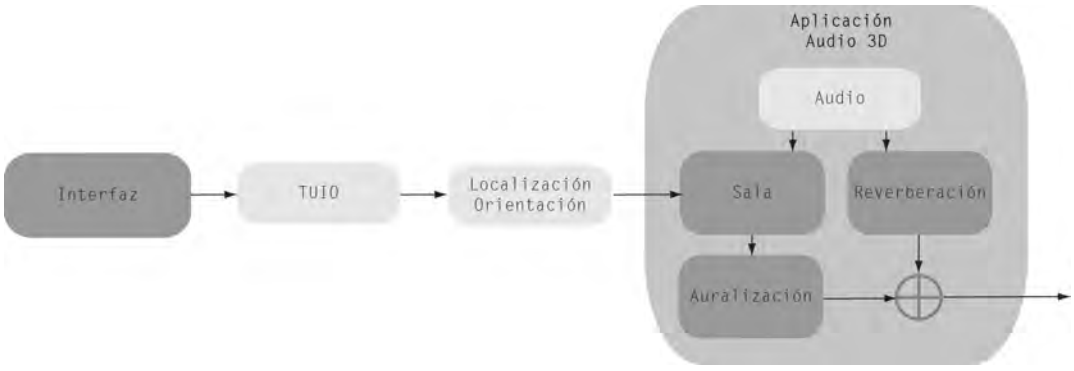


Figura 0.5: Diagrama de bloques planteamiento genérico algoritmo audio 3D

En conjunto el sistema reconoce una serie de fiduciales enganchados a objetos físicos que son identificados por una cámara conectada al ordenador. En este se ejecuta reacTIVision, un framework de visión por computador que mediante el protocolo TUIO se conecta a una aplicación cliente. La aplicación desarrolla un entorno de audio tridimensional. Para este cometido se probarán diversas librerías de audio 3D y las informaciones de los objetos se representan de forma gráfica en el ordenador.

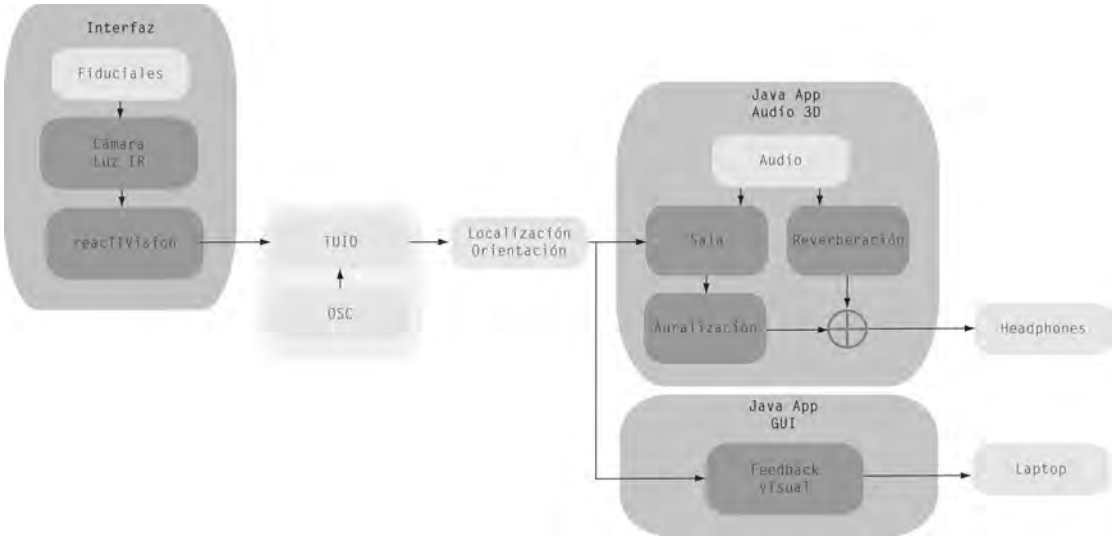


Figura 0.6: Diagrama de bloques planteamiento proyecto

1. Estado del arte

Introducción histórica al audio tridimensional, con los primeros sistemas y su evolución en este campo. Así como el estado del arte de las principales librerías que se utilizan en la actualidad para el desarrollo de aplicaciones, poniendo especial énfasis en JOAL (Java openAL).

1.1. Introducción

El audio tridimensional, desde el desarrollo de la realidad virtual, ha sido un reto. Fue en 1988 cuando se desarrollo el primer sistema de audio virtual , el Convolvotron, simulaba no solo el sonido de los objetos si no también su ubicación en el mundo virtual. Desde entonces la tecnología ha avanzado de una manera extraordinaria y hoy encontramos aplicaciones en dispositivos móviles, vehículos, TV, videojuegos…

Figura 1.1: Convolvotron

En los últimos años nos hemos vuelto a familiarizar con el término 3D. La evolución de los canales de audio ha sido muy lenta, del MONO al STEREO y de este a los 6 y 8 canales. Un paso definitivo y que cambia el paradigma de los sistema multicanal tal como lo conocemos es el audio 3D. Algunas empresas ya apuestan por la próxima implementación del sonido 3D en cines.

En la *International Conference on Audio, Language and Image Processing* del 2010 se realizo una presentación sobre las técnicas para aplicar el audio 3D en el cine 3D [1]. El sonido en los cines en general esta situado en la horizontal a la altura de los oídos del oyente. Para conseguir crear un medio sonoro adecuado se pueden utilizar múltiples técnicas, aunque no todas son recomendables para utilizarse en un ámbito en el que se tiene que renderizar el sonido en una sala de cine más o menos grande, esto aporta ciertas limitaciones. Los principales problemas a tener en cuenta en este caso son la diferencia de volumen entre el sonido directo y las imágenes, así como personalizar el campo sonoro para un sistema multiusuario. Las técnicas que se plantean para este fin son: Vector Base Amplitude Planning, técnicas binaurales utilizando altavoces y cascos, Wave Field Synthesis y Ambisonics. Solo la técnica binaural con cascos es capaz de proporcionar esta sensación de audio 3D en un cine. El coste computacional asociado al cálculo de las HRTF y el head-tracking se puede ver reducido usando el formalismo de Fourier-Bessel. Mientras que las técnicas binaurales pueden ser las que se usen como estándar en el cine (también se usan gafas), las técnicas transaurales [20] pueden ser las que se utilicen para la HDTV en casa. Estas últimas permiten emitir señales binaurales en los odios de un oyente con altavoces estereo. La idea básica es filtrar la señal binaural de tal manera que la presentación posterior en los oidos del oyente sea una señal binaural. La técnica se puso en marcha por Schroeder y Atal [21] y más tarde fue perfeccionada por Cooper y Bauck [22] quien la llamó técnica transaural.

En el *3D Audio and Appied Acoustic Lab* de la Universidad de Princeton el Prof. Edgar Choueiri estudia las aplicaciones de audio 3D. Ha realizado Pure Stereo [2], un sistema que permite con unos altavoces la reproducción de sonidos en 3D. Utiliza técnicas de crosstalk para crear el "muro" que separa los dos altavoces mediante los filtros c-BACCH implementados en el mismo laboratorio. Aplicando esta técnica a un sistema de altavoces hi-fi, mediante un procesado en tiempo real, es posible crear el entorno 3D. Este mismo departamento junto con la compañía Británica, Cambridge Mechatronics están diseñando un sistema de Audio 3D para televisión.

Se ha creado la 3D Audio Alliance, cuyo objetivo es promover y facilitar el desarrollo de las especificaciones para la transmisión de espacios de escucha en cines, grandes eventos, calles… Es una alianza similar a las que se crearon para las especificaciones de los nuevos formatos de registro y reproducción de datos multimedia.

Algunas móviles llevan incorporados chips que procesan audio en 3D, para simulación, o bien aplicaciones que funcionan en tiempo real. Yamaha Corp. crea el YMU786 [3] que permite efectos como reverb, delay, echo… además de crear fuentes sonoras virtuales y

posicionarlas en el espacio. Sonaptic está especializada en audio 3D para dispositivos móviles como teléfonos o consolas.

Telefónica presentó en el *Mobile World Congress* de este año un sistema de multiconferencia [4] para *iPhone* que permitía establecer una multiconferencia mejorando la calidad de voz y permitiendo identificar, con tecnología 3D, que participante estaba hablando en cada momento y desde que ubicación en un espacio de reunión virtual.

1.2. Librerías

Para el desarrollo de una aplicación de audio 3D hay una gran variedad de APIs que permiten definir escenas acústicas complejas. Están disponibles en varios lenguajes, a priori el interés se centra en librerías en Java, aunque la mayor parte de las mismas están en C++ debido al buen rendimiento que este lenguaje aporta trabajando con audio en tiempo real.

Podemos dividir las APIs en tres niveles, según la complejidad de la programación. Esto quiere decir que cuanto más avancemos en nivel, más recursos computaciones utilizaremos y en ocasiones será necesario un hardware específico para poder desarrollar la aplicación en cuestión. En el core, el nivel más básico, simplemente se implementa el esqueleto de la estructura y la escena sonora básico, esto involucra atenuación por distancia, efecto doppler, cambio posicional de las fuentes, uso de fuentes puntuales. En el nivel más común añadimos al core algunas propiedades acústicas de la sala, como la oclusión y obstrucción debida a obstáculos, la reverberación. Por último, en el nivel más completo y complejo se permite añadir propiedades al escenario sonoro como: reflexiones, fuentes direccionales, fuentes sonoras con propósitos especiales…

Para el caso que nos ocupa nos centramos en la APIs desarrolladas para el lenguaje de programación Java. En concreto las que permiten una implementación separada del entorno de gráfico y de audio. Las más utilizadas son FMOD y OpenAL, en su versión para este lenguaje. De la misma manera se comentarán las principales características de otras bibliotecas para diferentes lenguajes.

1.2.1. FMOD[1]

Es una librería propietaria realizada por Firelight Technologies que permite manipular archivos sonoros en diversos formatos y en múltiples plataformas. Es usada en juegos y aplicaciones software con diferentes funcionalidades.

---

[1] http://www.fmod.org/

Es una API para C++ (C#) muy útil, con un paquete de efectos muy extenso. El diseño básico de una aplicación de audio 3D con estas librerías permite realizar un gran número de configuraciones. Esto es, soporta salidas estero y multicanal, permite un número tanto de sources como de listeners diverso, las fuentes pueden ser direccionales, la sala geométricamente puede simular obstrucciones y oclusiones, se pueden realizar efectos especiales basados en software (osciladores, low pass filter con resonancia, high pass filter con resonancia, distorsion, EQ, reverb…), permite el trabajo en tiempo real con una baja latencia… Por tanto, es una librería completa que permite realizar trabajos muy interesantes. En principio una gran cantidad de estos efectos es posible implementarlos sin hardware dedicado, en caso de que la aplicación sea muy exigente (por ejemplo, utilizar una reverb de alta calidad) se puede recurrir a hardware acelerador creative (sound blaster) para implementar los efectos sin apenas limitaciones.

- Estructura FMOD.

La estructura básica de los datos en una implementación en tiempo real FMOD la podemos dividir en tres partes: coge muestras de audio de una entrada (recording), las empaqueta los datos y los transfiere (data packet) y por último los reproduce (playback).

  - Recording

    Cuando recibe una entrada de datos crea una FMOD_INSTANCE, para cada una de ellas le asignamos un input device (fuente), output driver (direct sound, OSS,…) y un output device (soundcard) usando cada driver. De esta manera tenemos varias fuentes que se pueden reproducir por varias tarjetas de sonido en caso de que creemos más de una FMOD_INSTANCE.

    En cada FMOD_INSTANCE creamos estructuras FSOUND_Sample, les asignamos varios parámetros como la frecuencia de muestreo, la resolución de bit, unit length… La FSOUND_Sample generalmente es utilizada para cargar datos de audio que tenemos en memoria.

  - Data packet transfer

    Mientras estamos grabando datos de diferentes fuentes se comprueba cuantos se han copiado a FSOUND_Sample, en caso de que esté lleno los siguientes datos pasan a un buffer secundario y así sucesivamente, de esta manera se asegura que no se pierden datos. Cada paquete de datos perteneciente a una misma fuente se introduce en FSOUND_Sample con una cabecera con la información correspondiente (volumen, posición 3D).

- Playback

  Para la reproducción FMOD dispone de una unidad básica llamada FSOUND_Channel. Cada FSOUND_Channel esta conectado con FSOUND_Sample para ser reproducido. El vector de posición en 3D y el volumen se extrae del paquete para ser reproducido con el efecto en 3D.

En caso de disponer del motor Creative, se puede utilizar la Environmental Audio extensión (EAX) para añadir reverberación y otros parámetros del medio para su correcto funcionamiento en tiempo real.

Como se ha comentado con anterioridad es una API para C++, pero hay una versión bastante más básica para el uso en Java. Las características de esta no son tan espectaculares. De hecho algunos efectos no están incluidos en la API, debido probablemente al alto coste computacional de los mismos, y a la alta latencia que estos pueden provocar con un lenguaje más lento como Java.

1.2.2. OpenAL[2]

OpenAL (Open Audio Library) es una interface software de hardware de audio. Consiste en un número de funciones que permiten al programador especificar objetos y operaciones produciendo salidas de audio de alta calidad. Especialmente salidas en 3D de fuentes de sonido alrededor de un receptor.

OpenAL incluye extensiones compatibles con IA-SIG 3D (Level 1 y Level 2) que permite modelizar la directividad y la atenuación por distancia de las fuentes, el efecto Doppler así como efectos del medio (reflexión, obstrucción, transmisión y reverberación).

Antes de pasar a enunciar los atributos genéricos de openAL es importante remarcar que algunas características de esta API que necesitan mucho procesado no es suficiente hacerlo por software, sino que necesitan un hardware dedicado que implemente EAX (Creative).

- Fundamentos OpenAL [5]

Básicamente renderiza audio de un buffer de salida que previamente a ido recolectando muestras de audio en un buffer de entrada. Sobretodo es usado para especializar el sonido, audio 3D.

OpenAL dispone principalmente de tres objetos: buffers, sources y un listener simple.

Las sources guardan la localización, dirección y otros atributos de un objetos 3D en el espacio, teniendo un buffer de salida asociado. Cuando se quiere reproducir un sonido la ejecución se controla

[2] http://connect.creativelabs.com/openal/default.aspx

mediante la source. Cada una de ellas es procesada de forma independiente.

Los buffers guardan datos de audio comprimidos o descomprimidos. Los buffers están referenciados por las sources. Los datos de las muestras de audio están asociados con el buffer.

Hay solo un listener por cada contexto de audio. Los atributos del listener son similares al de las sources, solo que representan desde donde el usuario esta escuchando los audios. Todas las fuentes están renderizadas (mezcladas y reproducidas) en relación con el listener.

- Tiempo, frecuencia y espacio.

OpenAL utiliza por defecto segundos y Hertz como unidades de tiempo y frecuencia. Un valor float o int especifica las duraciones, latencias, delays… Es decir, las frecuencias de muestreo, de corte o parámetros específicos de los filtros se expresan utilizan como unidad el Hertz.

Las unidades de espacio, para calcular distancias, no están definidas. Se puede usar metros, pulgadas… Para la simulación de la atenuación por distancia se utilizan distancias euclídeas relativas entre fuente y receptor Por tanto la aplicación deberá ajustar las ganancias según estas distancias relativas. Será necesario referencial las distancias utilizando alDistanceModel().

- Listener y sources.

Tanto los objetos listener como sources tienen una serie de atributos básicos que pueden representar su posición, velocidad y orientación en el espacio tridimensional. OpenAL usa el sistema de coordenadas cartesiano de la mano derecha.

Los atributos más importantes son:

AL_POSITION: especifica la localización actual del objeto en el sistema de coordenadas.

AL_VELOCITY: especifica la velocidad y dirección de los objetos en el sistema de coordenadas. No afecta a la posición, es decir, la posición y velocidad son independientes en lo que implica efecto sonoro. Este parámetro se utiliza para calcular el efecto Doppler.

AL_GAIN: define un multiplicador escalar de amplitud. Como atributo de un objeto fuente, este se aplica de forma particular a una sola fuente. Si se aplica al listener, el atributo afecta a todas las fuentes. Por defecto vale 1.0f, lo que implica que no hay atenuación, en caso de que el valor sea mayor a 1 el sistema se amplificará. Si el valor es 0.5f hay una perdida de 6dBs.

- Listener Object.

El listener define varias propiedades que afectan al procesado del sonido en la salida. El dispositivo de salida (altavoces, auriculares) y el método de posicionamiento 3D (HRTF) está implementado y es dependiente del hardware.

AL_ORIENTATION: esta definido por un par de vectores que marcan la dirección en la que el listener esta mirando.

- Source object

La fuente maneja atributos como la posición, la velocidad y un buffer con los datos. Para controlar los atributos de las fuentes la aplicación puede modificar y parametrizar los datos provenientes del buffer referenciados a la fuente. También son posibles funciones adicionales como la ejecución del estado de la fuente, esto es started, stoped, paused.

AL_SOURCE_RELATIVE: indica que la posición, velocidad, cono y direcciones están interpretadas de forma relativa respecto a la posición del listener.

AL_LOOPING: es un flag que al estar activo indica que cuando la source llegue al final del buffer no se pare la reproducción, si no que vuelva a AL_INITIAL y seguidamente a reproducirse AL_PLAYING.

AL_MIN_GAIN: es un umbral que indica el valor de amplitud mínimo que siempre dará la fuente. (Análogamente AL_MAX_GAIN)

AL_REFERENCE_DISTANCE: es usado para calcular la atenuación en función del modelo de distancia utilizado.

AL_MAXIM_DISTANCE: marca la distancia máxima. Para distancias relativas mayores el valor queda limitado a este máximo.

AL_PITCH: desplazamientos del pitch. Cada reducción del 50% implica una reducción de una octava (12 semitonos), obviamente si lo doblamos conseguimos un incremento de una octava.

AL_DIRECTION: si es diferente a cero estamos en presencia de una fuente direccional. La fuente se presume simétrica alrededor de la dirección que marca el vector.

AL_CONE_INNER_ANGLE: ángulo interior del sonido, en grados. Si es 360 estamos delante de una fuente omnidireccional. (Análogamente AL_CONE_OUTER_ANGLE).

AL_CONE_OUTER_GAIN: es el factor por el cual AL_GAIN es multiplicado para determinar la ganancia eficiente del cono definido por el ángulo exterior.

Aparte de estos parámetros que caracterizan lo más importante de las fuentes y receptor, hay otros que permiten modificaciones de buffers… Es importante también la configuración de las propiedades de la mezcla final que caracterizan los canales, la frecuencia de muestreo…

- Extensiones

Con la extensión EAX [8] creada por Creatives es posible implementar los niveles 1 y 2 del IA – SIG 3D (Interactive audio – sound interest group).

Nivel 1 (1998). [6]

- Reproducción de 8 fuentes simultaneas (16 streams).

  Sample rate mínimo 22050 Hz, 16 bit-out (recomendado 44.1 kHz).
  Source y listener con posiciones 3D.
  Velocidad de source y listener.
  Listener orientation
  Orientación y patrón de radiación de la source.

- Efectos

  Distancia
  Doppler
  Posiciones 3D reales.
  Modelo de radiación

- Capacidad de renderizado en tiempo real e interactivo sin defectos auditivos y latencia.

Nivel 2 (1999). [7]

- 16 fuentes simultaneas (32 streams)

  Sample rate de 33050 Hz (o mayor) a 16 bits
  Posición 3D de la source y el listener
  Velocidad de la source y el listener
  Orientación listener
  Orientación y patrón de radioactividad de la source
  Efectos dinámicos de reverberación de la source y el listener
  Low-pass individual y atenuación aplicada a la trayectoria directa para cada fuente.
  Low-pass individual y atenuación aplicado a los ecos de cada fuente.

- Efectos

  Distancia (atenuación y reverberación)
  Doppler
  Modelo de radiación
  Efectos sala (early reflections y reverberación)
  Oclusión y obstrucción.

- Renderizado en tiempo real.

  No tiene artificios sonoros (popping, clicking o otras distorsiones)
  Baja latencia.

Cabe destacar que estas adiciones pertenecen a Direct Sound 3D, es decir, parte del componente software DirectX de Microsoft. Funciona bajo Windows y con hardware específico.

- Implementación OpenAL.

Para desarrollar el sistema en tiempo real de audio tridimensional la primera alternativa es utilizar una API especifica para este fin. Las más completas son la FMOD y la OpenAL. Aún así se trata de desarrollar la aplicación en Java y bajo MacOS, con lo que la FMOD pierde gran parte de sus características puesto que funciona muy bien sobretodo la API de C++. Sin embargo openAL tiene una versión para Java bastante interesante JOAL (Java Open AL) el problema en este caso es que sobre MacOS no puede utilizar el software especifico Creative, ni por supuesto el Direct Sound 3D (DirectX). Aún así se hace una primera implementación de JOAL con lo básico de la API.

JOAL[3]

Es una librería que permite acceder a programadores de Java a openAL. Es una librería desarrollada por Sun Microsystems Game Technology Group. Esta desarrollada para múltiples plataformas.

El algoritmo

La implementación mediante JOAL de un sistema tridimensional con el core de la biblioteca OpenAL simplemente calcula la atenuación por distancia y la auralización. No tiene en cuenta nada que tenga que ver con el entorno, la sala…

Creamos la variable para acceder a la API.

```
private AL al;
```

[3] http://jogamp.org/

Accedemos a openAL

```
al = ALFactory.getAL();
```

Inicializamos el Listener

```
al.alListener3f(AL.AL_POSITION, xLis, yLis, zLis);
al.alListener3i(AL.AL_VELOCITY, 0, 0, 0);
al.alListenerfv(AL.AL_ORIENTATION, oriLis, 0);
```

Abrimos el WAV y gestionamos el buffer

```
ALut.alutLoadWAVFile(fnm, format, data, size, freq, loop);
al.alGenBuffers(1, buffer, 0);
al.alBufferData(buffer[0], format[0], data[0], size[0], freq[0]);
```

Generamos la fuente

```
al.alGenSources(1, source, 0);
al.alSourcei(source[0], AL.AL_BUFFER, buf[0]);
al.alSourcef(source[0], AL.AL_PITCH, 1.0f);
al.alSourcef(source[0], AL.AL_GAIN, 1.0f);
al.alSource3f(source[0], AL.AL_POSITION, 0.0f, 0.0f, 0.0f);
al.alSource3i(source[0], AL.AL_VELOCITY, 0, 0, 0);
al.alSourcei(source[0], AL.AL_LOOPING, AL.AL_TRUE);
```

Actualizamos la posición de las fuentes

```
al.alSource3f(source[0], AL.AL_POSITION, x, y, z);
```

Reproducimos

```
al.alSourcePlay(source[0]);
```

Paramos

```
al.alSourceStop(source[0]);
```

Actualizamos la posición del listener

```
al.alListenerfv(AL.AL_ORIENTATION, oriLis, 0);
```

1.2.3. Otras librerías

- irrKlang[4]

Es una potente API de alto nivel de sonido en 2D y 3D, para aplicaciones multimedia, juegos… Es de uso libre para aplicaciones no comerciales.

  - Soporta diferentes formatos de audio
    .wav, .ogg, .mp3, .flac… incluso de se pueden instalar plugins para extender el número de formatos.

[4] http://www.ambiera.com/irrklang/

- Multiplataforma
  - Windows X (DirectSound 3, DirectSound 8, WinNM)
  - Linux (ALSA)
  - MacOs X (CoreAudio)

- Sonido 3D

  Soporta el audio tridimensional en todas las plataformas y con cualquier driver de audio de los mencionados anteriormente. Permite reproducir archivos cortos y sonido en streaming. Es eficiente y utiliza pocos recursos de tiempo de CPU.

  El audio 3D es rápido con hardware no específico. Este tipo de audio se puede combinar con los efectos de que dispone, entre los que se encuentran eco y la reverb.

- Compiladores y lenguajes

  Soporta compiladores como MS Visual C/C++, GCC 3-4, y todos los lenguajes .NET (C#, VisualBasic.NET, Delphi.NET, IronPython…).

- Efectos de sonido

  Efecto Doppler, choros, compressor, distorsion, echo, reverb, parametric EQ…

• CLAM[5]

Es un Framework para investigar y desarrollar aplicaciones en el campo del audio y la música. Permite realizar complejos análisis de señales de audio, transformación y síntesis. Proporciona una interfaz uniforme para realizar tareas con dispositivos y ficheros de audio. Se puede utiliza CLAM como una librería para programar aplicaciones en C++, pero también se pueden usar herramientas gráficas para desarrollar aplicaciones sin tener que utilizar código.

- Multiplataforma

  Linux, MacOS y Windows. También puede usarse sobre otras plataformas como SuSe, Fedora…

- Soporta diferentes formatos de audio.

  Soporta todos los formatos de audio de libsndfile (wav, raw, flac…) así como el mp3 (via libmad).

- Control de eventos

  Puede recibir control de eventos mediante dispositivos MIDI o bien OSC (Open Sound Control).

---

[5] http://clam-project.org/

- Audio 3D
  - Se puede renderizar un escenario 3D mediante bases de datos con la respuesta impulsional punto a punto del espacio o bien mediante la simulación en tiempo real ray-tracing.
  - Codificación MONO a Ambisonics
  - Codificación Ambisonics a array de altavoces periphonic.
  - Codificación Ambisonics a binaural (HRTF)
  - Codificación MONO a binaural (HRTF)
  - Codificación MONO a array de altavoces

• BASS[6]

Es una librería de audio para usar con Windows y MacOS. Su propósito es dar a los desarrolladores una herramienta potente y eficiente para trabajar con audio, archivo o streaming.

En Windows requiere DirectX 3 y puede completar sus características y funciones con DirectAudio o DirectAudio 3D con hardware acelerador. En MacOS X usa CoreAudio. Esta disponible en C/C++ y lenguajes .NET. Así como también en plataformas iOS.

Las principales características de la librería son:

- Soporte de muestras WAV/AIFF/MP3/OGG
- Stream y sample data
- Streaming por Internet desde HTTP y FTP.
- Streaming multicanal.
- MOD music
- MO3 music
- Salidas múltiples.
- Sistemas de grabación flexibles
- Asignación de altavoces
- Sincronización de alta precisión
  - EfectosChorus
  - Compressor
  - Distorsion
  - Echo
  - Flanger
  - Gargle
  - EQ paramétrico
  - Reverb
- Sincronización de alta precisión
- DSP
- 3D sound

[6] http://www.un4seen.com/

	OpenAL	FMOD	BASS	irrKlang	CLAM
Posicionamiento 3D	✓	✓	✓	✓	✓
Efecto Doppler	✓	✓		✓	✓
Control de reverberación	✓	✓	✓	✓	✓
Grabación de audio personalizado	✗	✓	✓	✗	✗
Streaming de archivos grandes o desde redes	✗	✓	✓	✗	✗
Hardware específico[7]	✓	✓	✓	✓	✗
Varios lenguajes[8]	✓	✓	✓	✗	✗
Multiplataforma	✓	✓	✗ [9]	✓	✓
Licencia	GNU-LGPN	NC[10]	NC[10]	NC[10]	GNU-GPL

Tabla 1.1: Comparación entre librerías [9]

[7] Requerimiento de hardware específico para conseguir todas las funcionalidades.
[8] Está diseñado sobre un lenguaje concreto (C++), sin embargo se pueden encontrar versiones adaptadas para otros lenguajes pero con especificaciones similares.
[9] Solo funciona en plataformas MacOS y Windows.
[10] Non-commercial use. Dispone de otras licencias de pago para usos comerciales.

2. Fundamentos audio 3D.

En este capitulo se exponen los fundamentos teóricos del audio tridimensional. Los modelados de la fuentes, entorno y receptor. Desarrollando en cada uno de ellos los rasgos más característicos, como la teoría de imágenes, la teoría duplex, el uso de HRTF…

2.1. Introducción

Un sistema de audio 3D permite posicionar sonidos alrededor de un receptor. El receptor percibe que el sonido viene de algunos puntos arbitrarios de su alrededor. Es decir, un oyente evoca determinadas fuentes sonoras que se producen dentro de un escenario, y puede situar esas fuentes en el espacio.

	Cascos	Altavoces estereo	Array de altavoces	Dimensiones	Control interactivo de usuario	Percepción
Mono	Si	Si	-	0D	No (on/off)	Fuente puntual desde la localización del altavoz
Estereo	Si	Si	-	1D (L/R)	L/R panning	Fuente localizada en línea entre los altavoces
"3D" Stereo Expansion	Algunos	Mayoría	-	1D (añadiendo espacio)	No (on/off)	El sonido llena el área alrededor de los altavoces
Multi-speaker surround sound	No	No	Si	2D (L/R, F/B)	No (pistas pre-codificadas)	El sonido esta en circulo formado por los altavoces reales
Virtual surround sound	Algunos	Mayoría	-	2D (L/R, F/B)	No (pistas pre-codificadas)	El sonido esta en circulo formado por los altavoces virtuales
IASIG Interactive 3D audio	Mayoría	Mayoría	Algunos	3D (L/R, F/B, U/D)	3D completo, espacio XYZ	El sonido esta localizado a cualquier distancia y posición del receptor

Tabla 2.1: Cuadro comparativo técnicas de reproducción de audio [6]

La auralización, es el proceso de generación de la percepción de un sonido en 3D. Caracteriza el sonido en función de las propiedades del entorno en el que este se encuentra. El proceso de auralización incluye el modelado de tres aspectos sumamente importantes para la correcta percepción del sonido: el modelado de la fuente, modelado de la sala y el modelado de receptor.

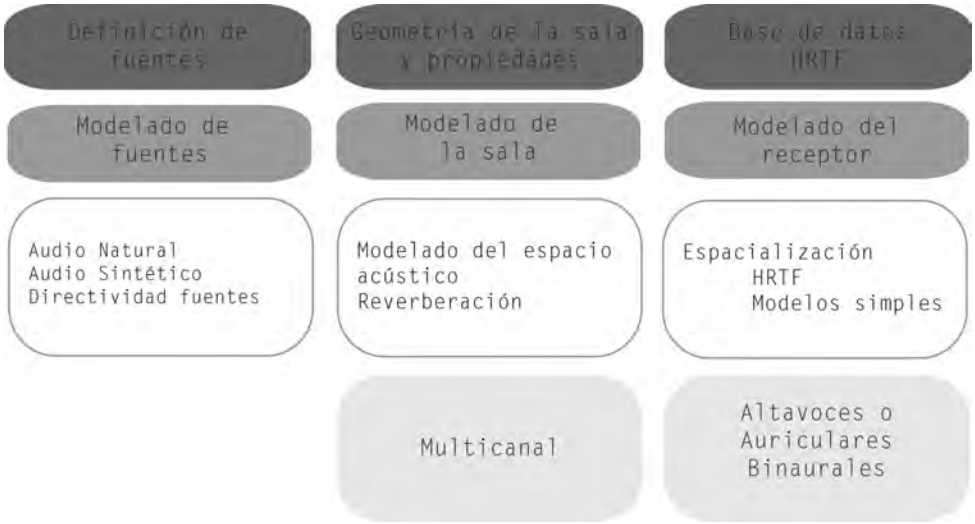


Figura 2.1: Diagrama de un sistema de audio 3D [10]

2.2 Modelado de la fuente.

La finalidad del modelado de la fuente es incorporar al sonido unas propiedades en un entorno, por ejemplo la directividad.

Los requerimientos cualitativos principales [10] que debe tener una fuente sonora son:

- Cada sonido debe ser seco, no contener reverberación ni propiedades direccionales.

- Las fuentes sonoras son tratadas como fuentes puntuales, por tanto deben ser monofónicas. En el caso de que sean estereofónicas cada canal será modelado como una fuente sonora.

- La calidad del sonido debe ser lo suficientemente alta como para no causar efectos indeseados.

El modelado de las propiedades de radiación y directividad de las fuentes sonoras es un tema al que normalmente no se le presta mucha atención, sin embargo tiene su importancia. A este respecto se estudian las propiedades direccionales de los instrumentos musicales, y de la cabeza humana. El modelado matemático de la directividad de las fuentes sonoras puede ser complejo, en muchas casos las medidas de directividad son usadas para obtener un representación de los datos numérica con propósitos de simulación.

2.3. Modelado del entorno.

Para modelar el entorno es necesario comprender como se comporta en sonido en un recinto y caracterizar todo parámetro que describa ese comportamiento. Los tres parámetros más importantes de una sala a la hora de virtualizarla son: *Early Reflections*, *Late Reverberation* y la absorción de las paredes. Por tanto la respuesta de

la sala se puede dividir en tres momentos concretos: el camino directo, las primeras reflexiones (dan idea de la coloración de la sala) y la reverberación (que le da sentido de profundidad y por tanto más o menos lejanía al sonido).

Cuando una fuente empieza a emitir un sonido dentro de un recinto se produce un frente de onda que llega al receptor en línea recta en caso de no encontrar ningún obstáculo, este camino se considera el camino directo. El frente de ondas que choca con obstáculos dentro del recinto y rebota hasta llegar al receptor, en este caso estamos hablando de reflexiones. Parte de la energía es absorbida por paredes y obstáculos, por esta razón las reflexiones tienen menor intensidad que el camino directo.

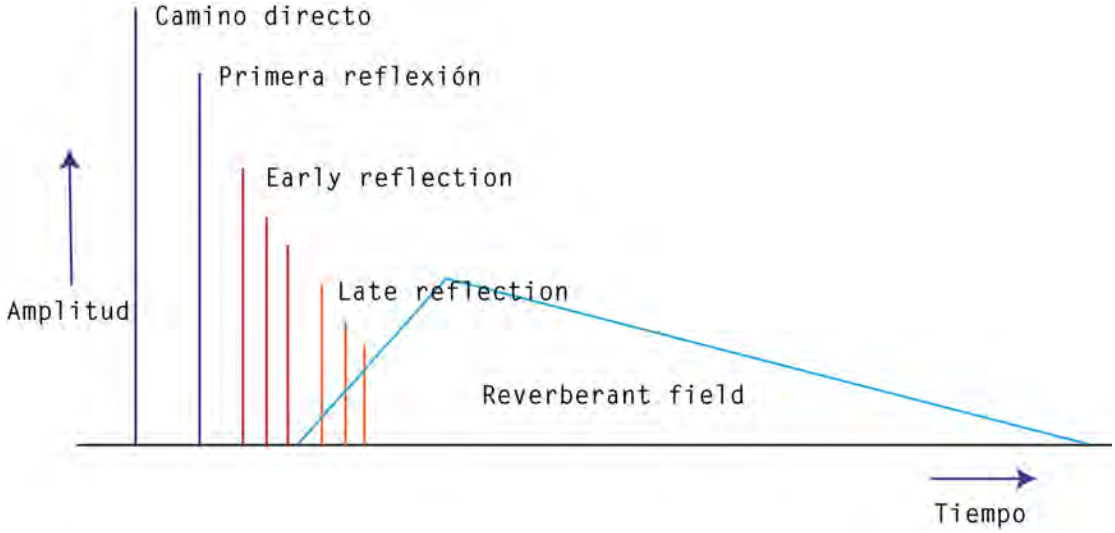


Figura 2.2: Parámetros modelado del entorno

Se consideran *Early reflections* las que se producen aproximadamente durante los 80ms siguientes al camino directo para origen frontal. Para los laterales, si los retardos son inferiores a 5ms se intuye un movimiento lateral de la fuente, mientras que para retardos mayores se produce un cambio en el tamaño de la fuente que da la sensación de envolvente.

Las reflexiones tempranas dan información del tamaño y forma de la sala, principalmente debido a las diferencias de tiempo y el ángulo de incidencia de las mismas. La distancia entre el receptor y la fuente sonora lo proporciona la relación entre la intensidad del sonido directo y la reverberación, esto se debe a que el nivel de la reverberación en una sala es constante mientras que la fuente sonora a medida que se aleja el nivel baja.

- Método físico

Se puede crear un entorno sonoro mediante métodos físicos, esto es, tomando medidas en una determinada sala, implica que dicha sala debe existir, lo que incluye las *early reflections* y la *late reverberation*. El principal problema de este método, dando por sentado que la sala

existe, es que se deben hacer gran cantidad de medidas fuente – receptor para que los resultados de la misma sean aceptables.

Al margen de tomar medidas, es posible la simulación utilizando métodos geométricos. Esto es altamente útil para las *early reflection*, se utiliza la teoría de rayos que funciona muy bien a alta frecuencia (cuando el sonido se comporta como un rayo), sin embargo para bajas frecuencias sería necesario utilizar métodos numéricos. A parte debería incorporar otros fenómenos como la difusión o difracción.

- Método perceptual

Es importante sobretodo para calcular la *late reverberation*, parte de parámetros medibles en toda sala como el tiempo de reverberación, *energy decay curve…* La *late reverberation* es la que permite que el sonido salga fuera de nuestra cabeza, el nivel de fondo es el que sirve de referencia para saber si la fuente esta lejos o cerca. El parámetro que tradicionalmente describe esta cualidad es *Clarity Index*.

Si nos centramos en el modelado computacional de las salas, nos encontramos con tres técnicas con las que podemos simular lo que ocurre con el sonido en su interior.

Figura 2.3: Modelado computacional

En el modelado basado en rayos, obviamos la longitud de onda del sonido en cuestión y consideramos que este actúa como un rayo. Todo fenómeno que se produce de forma natural en la onda (difracción, interferencia…) no se tiene en cuenta. Este método es bastante preciso en tanto en cuanto la longitud de onda del sonido es pequeña con respecto a los posibles obstáculos o superficies, y a su vez es grande con las rugosidades de los mismos. *Ray tracing* y *image source* son los métodos más utilizados en este campo. Se deben añadir las mejoras en cuanto a difracción y difusión.

Un método más preciso y fiel a la realidad física es utilizar la solución de la ecuación de onda. Es complicado poder llegar a la solución analítica en la mayoría de los casos, solo es posible en salas que tienen geometrías bastante simples. Para llegar a la solución es necesario hacer uso de métodos numéricos con un gran coste computacional, estos son FEM (*Finite Element Method*) o BEM (*Boundary element method*), cuyo funcionamiento en tiempo real es

inviable a día de hoy. Otra alternativa a estos métodos son FDTD (*Finite-difference time domain*).

Por otro lado se encuentran los métodos estadísticos, como el SEA (*Statical energy analysis*). Este da información de niveles de energía, por tanto para propósitos de auralización no funciona, se usa para medir los niveles de ruido en sistemas acoplados donde la transmisión del sonido en las estructuras es bastante importante.

El método de las imágenes aunque no sea el más preciso (sobretodo a baja frecuencia) es el que normalmente se implementa en casos de auralización.

2.3.1. Método de las imágenes.

Este método es útil para calcular las *Early Reflections*, se modela la propagación como un rayo normal al frente de onda. Donde la onda, se considera un rayo, por tanto tiene reflexión especular.

Básicamente consiste en crear salas virtuales a la original, es decir salas espejo por cada cara de la misma. Consideramos que las paredes no reflejan el sonido y por tanto el rayo sigue hacia las otras salas virtuales adyacentes que se han creado. Según el número de paredes que atraviese el rayo consideraremos la reflexión de un determinado orden (si atraviesa una pared será de orden uno, si atraviesa dos de orden dos, y así sucesivamente). De esta manera podemos calcular las distancias que recorren las reflexiones y por tanto calcular la atenuación por distancia (que es la relación logarítmica entre la distancia de la reflexión "i" con respecto al camino directo) y el retardo. Por otro lado, toda vez que una reflexión "atraviesa" una pared implica una perdida, en función del coeficiente de absorción de energía de la pared correspondiente.

En el caso de tener salas poliédricas [23], se pueden dar casos donde las fuentes virtuales que se crean no se puedan dar. Por tanto, hay que aplicar una serie de test.

- Test de validez: Para cada *virtual source* comprobar que ha estado generada por las caras reflectantes de las paredes, es decir, por las paredes que colindan con el interior de la sala.

- Test de proximidad: Es el criterio que limita el cálculo recursivo de las fuentes virtuales. Consideraremos que para una distancia determinada, que no orden, se trata de reverberación y no de *early reflections*.

- Test de visibilidad: La *virtual source* debe ser visible a través de la cara de la sala que la ha creado. En los lugares donde esta no sea visible no puede haber ningún rebote.

- Oclusiones: En caso de que la forma de la sala sea extraña se puede dar el caso de que un rayo es ocluido por una pared. Por tanto, si nos encontramos delante de formas no regulares, poliédricas, hay que testear cada una de las trayectorias.

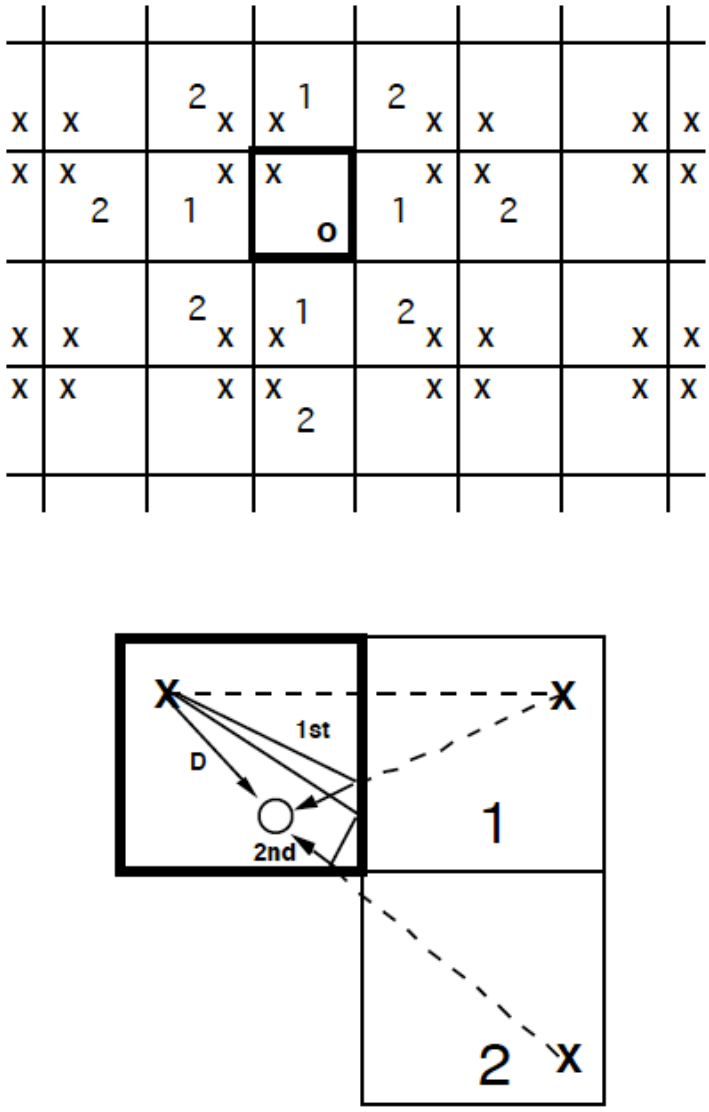


Figura 2.4: Método de las imágenes [10]

El método de las imágenes, como se ha ido comentando, tiene unas limitaciones y carencias:

- Solo es valida para altas frecuencias. A bajas frecuencias hay modos propios y por tanto hay que usar la solución de la ecuación de ondas o sus aproximaciones.

- La reflexión especular solo es valida si las dimensiones de los objetos que la onda sonora se encuentra son menores a las longitudes de onda. A altas frecuencias tendríamos difusión y por tanto hay que aplicar otros métodos, como el *ray tracing*.

- No contempla difracción. Aunque hay mejoras [12] que describen su implementación.

- Las atenuaciones por distancia y reflexión dependen de la frecuencia. Obviamente a altas frecuencias tenemos más atenuación que a bajas. Se puede solucionar modelando la atenuación con filtros.

2.3.2. Reverberación

La reverberación da información perceptual sobre el volumen de un espacio ocupado por fuentes sonoras. Cuando un sonido deja de vibrar aún se pueden oír durante cierto tiempo los ecos que van decayendo hasta el silencio.

Es necesario, por tanto, diseñar un reverberador para conseguir que los sonidos den la sensación de encontrarse dentro de una sala. Para la implementación de un reverberador se busca producir la suficiente densidad modal en función de la frecuencia y el tiempo, así se evitan las fluctuaciones. Mientras que la respuesta en frecuencia deber ser lo más plana posible para no tener coloraciones.

Un método analógico, para crear el efecto de cuasireverberación, consistía en hacer uso de los cabeceras de un cinta de *cassette*. Los cabezales son *erase*, *record* y *play* (por ese orden según el movimiento de la cinta). Entre el cabezal *record* y *play* hay una cierta distancia. Si se graba lo que se esta reproduciendo, teniendo en cuenta que físicamente hay una cierta distancia y por tanto un retardo temporal, se produce una sensación de eco a la que además se le podía variar la ganancia (modificando el volumen). El resultado era una serie de picos y valles con un retardo regular (periódico) y con baja densidad. Este tipo de respuesta es la de un *comb filter*. [11]

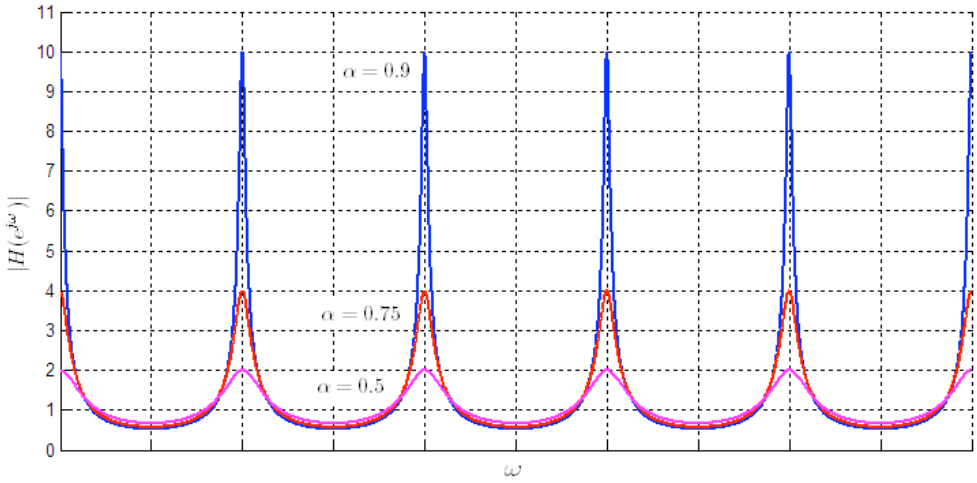


Figura 2.5: Respuesta frecuencial filtro comb (feedback)

El reverberador de Schroeder utiliza *comb filter*. Un *comb filter* simplemente con un bucle da una respuesta periódica, lo que provoca que suene muy artificioso o metálico. La densidad de ecos en un *comb filter* es contante. Y la densidad en frecuencia es inversamente proporcional a la densidad en tiempo, por tanto si están separados en tiempo estarán muy cerca en frecuencia.

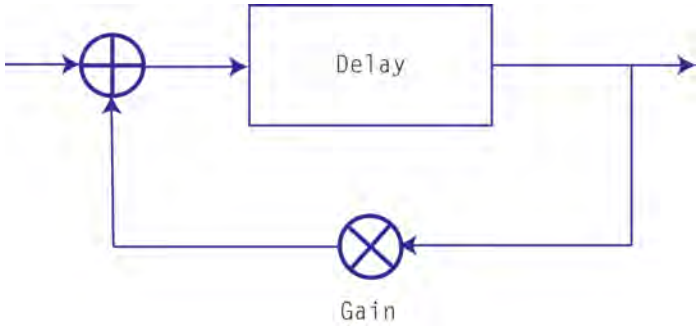


Figura 2.6: Diagrama de bloques comb filter (feedback)

La solución a este inconveniente esta en usar *allpass filter*. Estos filtros tan solo modifican la fase de la señal de salida, no su magnitud, es decir la respuesta en frecuencia es igual, lineal, pero la fase se ve modificada. El retardo de grupo de varios armónicos estará desplazado en el tiempo de forma análoga a como sucede cuando un sonido rebota en una superficie de la sala. Schroeder propone conectar varios *allpass filter* en serie. El resultado es una respuesta del eco densa y menos periódica.

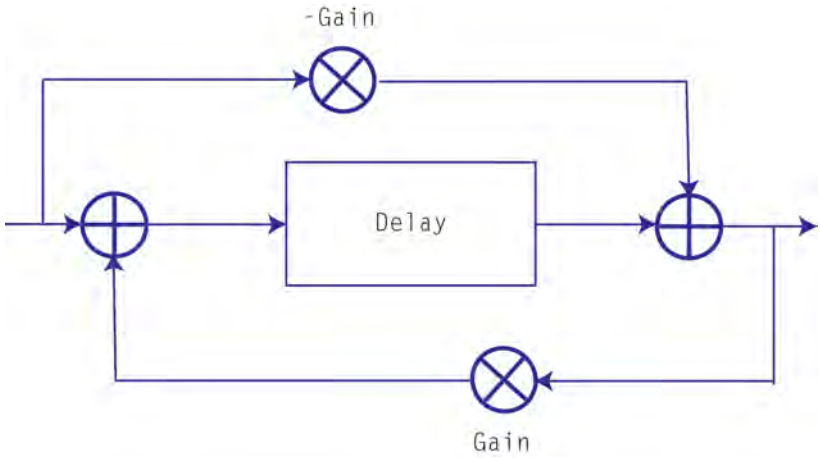


Figura 2.7: Diagrama de bloques allpass filter.

En definitiva, las limitaciones [11] del *comb filter* son:

- No es capaz de conseguir una densidades de ecos de sales reales. El máximo de ecos que puede generar en un segundo es igual a la frecuencia de muestreo.

- Hay un relación inversamente proporcional entre el espaciado de los modos en tiempo y frecuencia.

- No produce mayor densidad de ecos a medida que aumenta el tiempo.

El *allpass filter* por su parte, produce una respuesta en frecuencia plana y desacopla la relación frecuencia – tiempo. Sus principales limitaciones son:

- No consigue densidades de ecos de salas reales.

- No produce mayor densidad de ecos a medida que aumenta el tiempo.

Sin embargo en caso de ir creando sistemas con estos dos tipos de filtro podemos llegar a conseguir un reverberador bastante potente.

En caso de que coloquemos *filtros comb* en paralelo en frecuencia conseguimos mayor densidad de picos y en tiempo mayor densidad de ecos. Si unimos filtros *allpass* en serie se obtiene en frecuencia una ganancia unitaria y en tiempo la convolución de ecos.

Finalmente el diseño de un reverberador de Schroeder esta formado por un conjunto de filtros *comb* en paralelo y filtros *allpass* en serie, ambos filtros son IIR (Infinite Impulse Response). Esto hace que suene bien con niveles de reverberación moderados y que produzca un sonido metálico para tiempos de reverberación elevados. Obviamente sigue sin tener una densidad de ecos suficiente y no crece en el tiempo.

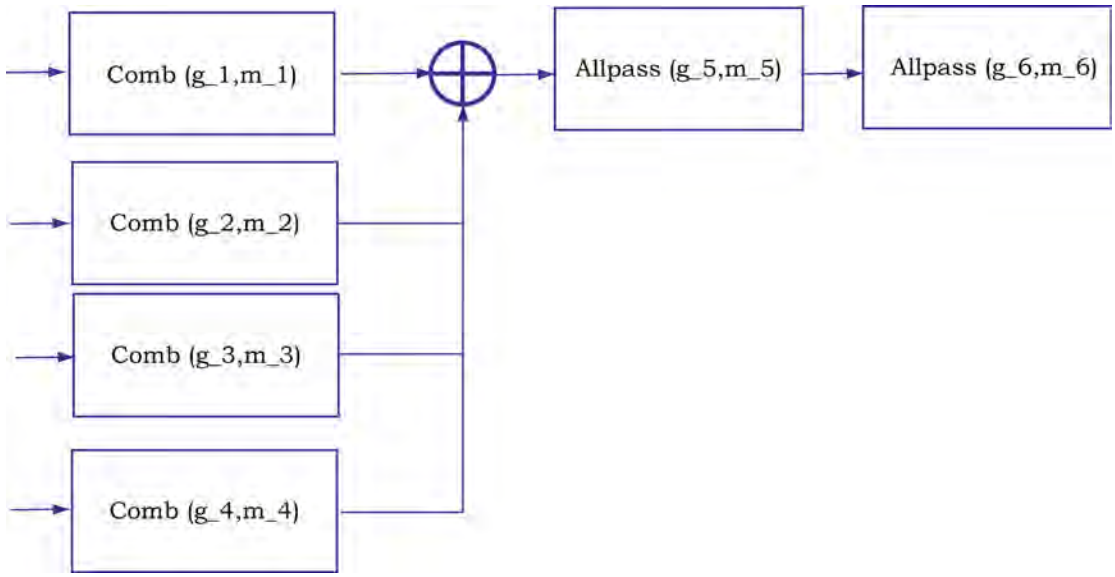


Figura 2.8: Diagrama de bloques reverberador de Schroeder

A este reverberador se le han introducido mejoras que minimizan sus carencias, son los reverberadotes de Moorer [25], Gardner [24] y Stautner and Puckette [26]. Así mismo existen gran diversidad de reverberadores [27] con mejoras específicas en comparación con el de Schroeder.

2.4. Modelado del receptor. [11]

El receptor debe localizar el sonido en un espacio acústico virtual. Es importante en este caso hablar de la auralización, que hace referencia a la generación del sonido tridimensional en un espacio.

2.4.1. Lateralización.

Para localizar un sonido es necesario conocer la posición angular de las fuentes, esto involucra la diferencia relativa entre las dos orejas en el plano horizontal. Las diferencias de tiempo y de nivel entre ellas dan idea de la posición en la que la fuente sonora se encuentra. La teoría dúplex de localización, enunciada por Rayleigh en 1907, estudia extensamente estos conceptos.

La lateralización es un caso especial de localización en el que la percepción espacial se produce a lo largo del eje interaural entre los oídos, esta percepción involucra las diferencias de tiempo y nivel que se producen entre ellas.

2.4.2. ITD (*Interaural Time Differences*)

Es la diferencia, en tiempo, de la llegada del sonido entre los dos oídos. Es especialmente importante a bajas frecuencias, el límite esta en la longitud de onda que equivale al diámetro de la cabeza (entre 1 y 1.5 KHz). Las frecuencias que están por encima son atenuadas porque la cabeza actúa como obstáculo, creando el efecto sombra al otro lado.

Hay algunos modelos para el cálculo de la ITD que hacen diferentes aproximaciones para altas frecuencias; Kuhn, Woodworth and Schlosberg. Los valores de las ITD también se pueden extraer de las HRTF.

2.4.3. ILD (Interaural level differences)

Es la diferencia, en nivel, de la llegada del sonido entre los dos oídos. Es especialmente importante a frecuencias altas, ya que las longitudes de onda son más pequeña y el efecto sombra que produce la cabeza es mayor. Por encima de 2 kHz, longitudes de onda más pequeñas que el diámetro de la cabeza, se hace importante. La influencia de este parámetro es diferente según las personas.

La percepción de la latelarización es curiosa, cuando un sonido mono (exactamente igual) es escuchado con unos cascos estereo, y por tanto en los dos canales suena exactamente lo mismo, la sensación es escuchar el sonido justo en el centro de la cabeza y no en cada oído, como uno podría pensar. Si se produce un cambio de volumen o retardo en alguno de los dos canales, parece que la fuente sonora cambia de posición sobre el eje interaural. La lateralización no es capaz de describir las posiciones delante/detrás de las fuentes, en ocasiones se usa como símil visual el cubo de Necker, *figura 2.9*.

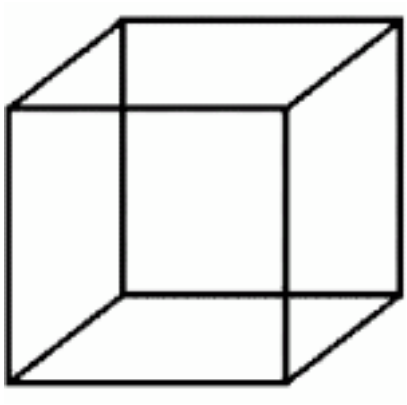


Figura 2.9: Cubo de Necker

2.4.4. Precedence Effect o Haas effect.

Haas examino la influencia de los retardos temporales en la inteligibilidad del habla. El *precedence effect* explica un mecanismo de inhibición del sistema auditivo sobre la localización de sonidos en presencia de reverberación.

Básicamente el efecto describe un aspecto de la percepción del sonido. Si varios sonidos independientes llegan al cerebro en un intervalo de tiempo inferior a 50ms, este los fusiona y los interpreta como uno solo. El cerebro deja de percibir la dirección e interpreta los otros sonidos como ecos o reverberación del primero.

Los intervalos de tiempo pueden ir variando según las intensidades del sonido… pero en general se puede decir que:

- Si el retardo es inferior a 5ms, el cerebro localiza un solo sonido en la misma ubicación que el primero.

- Si el retardo está entre 5 – 50 ms, se escucha un único sonido a de intensidad el doble y a medio camino entre ambos.

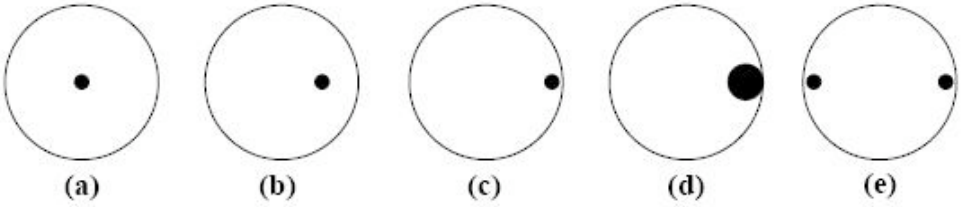


Figura 2.10: Efecto Haas

(a) La señal llega a ambos oídos de forma simultanea, sin retardo. (b) La señal llega al oído izquierdo 0.3ms después que al odio derecho, la fuente virtual se desplaza hacia la derecha. (c) La señal llega al odio izquierdo 0.6ms después que al oído derecho, la fuente deja de moverse. (d) El retardo es de 20ms, la fuente se oye el doble. (e) El retardo es mayor a 30ms, se distinguen las dos fuentes. [Los valores de temporales son aproximados y varían según la clase de sonido.]

La teoría dúplex como se ha ido comentando tiene ciertas limitaciones. Falla la distinción entre delante – detrás, y también se produce un cono de confusión. El cono de confusión esta formado por diferentes puntos del espacio donde las diferencias tanto de tiempo como de nivel son iguales.

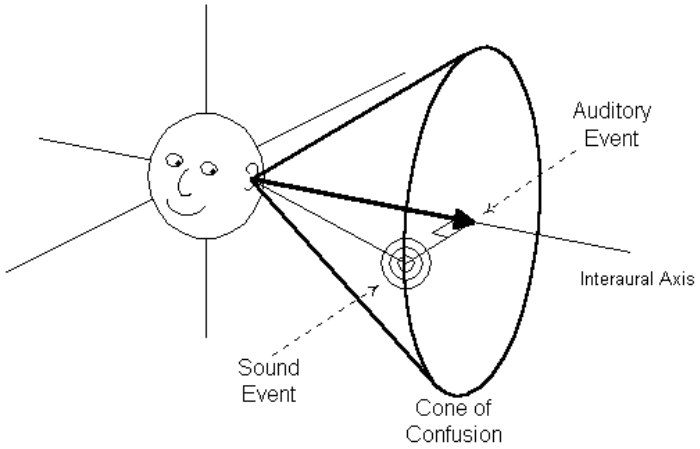


Figura 2.11: Cono de confusión

2.4.5. HRTF (Head related transfer function).

Definición: relación entre la presión sonora en la membrana timpánica y la presión sonora en un punto situado en el centro de la cabeza cuando el receptor esta ausente.

La HRTF binaural modela la modificación espectral que sufre el sonido debido al efecto de la cabeza, las reflexiones y resonancias en el pabellón auditivo, las reflexiones en la espalda y torso, y las diferentes atenuaciones debidas a la distancia.

La función de transferencia es binaural, y está en función del azimut y elevación. Cada persona tiene una caracterización diferente. Las HRTF llevan incluidas las ITD y ILD, pueden ser extraídas para utilizarse con la teoría dúplex y caracterizar los puntos débiles que esta pueda tener a nivel frecuencial.

La HRTF tiene elementos direccionales y no direccionales. Entre los direccionales y en función de la frecuencia encontramos:

- [0.1 – 2 kHz] Torso.
- [0.8 – 1.2 kHz] Reflexiones en la espalda.
- [0.5 – 1.6 kHz] Difracción y reflexión en la cabeza.
- [2 – 14 kHz] Pina y reflexiones en la cavidad coclear.

Mientras que como elementos no direccionales:

- [3 kHz] Frecuencia dominante de la cavidad coclear.
- [3 – 18 kHz] Canal auditivo e impedancia del tambor auditivo.

La HRIR (Head related impulse response) modela estos efectos como un sistema lineal e invariante en el tiempo, que puede ser expresado como un filtro FIR.

Para construir las HRTF se utilizan señales pseudoaleatorias en un entorno anecoico mediante la inserción de micrófonos en el odio de una persona o en un *dummy*.

En general las respuestas HRIR se expresan en fase mínima. La ventaja de utiliza representación en fase mínima radica en que permite insertar separadamente las ITD mediante línea de retardo, además es necesario si se desea interpolar. Un sistema de fase mínima tiene todos los ceros dentro del circulo unidad, así introduce menos retardo de grupo. Dada una determina respuesta en amplitud, los sistemas de fase mínima tienen el mínimo retardo de los sistemas realizables.

La HRTF a diferencia de la teoría dúplex permite caracterizar la elevación. Las bandas más significativa para la percepción de la elevación son:

- Según Blauert (1969)

  Frontal: 300 – 600 Hz , 3 – 6 kHz

  Encima de la cabeza: 8 kHz

  Detrás: 1.2 kHz , 12 kHz

- Según Henbrank and Wright (1974)

  Frontal: Notch con mínimo entre 4 – 8 kHz, pico en 13 kHz

  Encima de la cabeza: pico entre 7 – 9 kHz

  Detrás: pico alrededor de 12 kHz

  La elevación afecta a la frecuencia de corte del notch, entre 5 – 9 kHz

Las HRTF/HRIR[1] utilizadas en este trabajo son las que proporciona el IRCAM (Institut de Recherche et Coordination Acoustique/Musique).

Las HRIR están tomadas en cámara anecoica a un conjunto de 51 sujetos. No están calculadas partiendo de un Dummy, si no que son

[1] http://recherche.ircam.fr/equipes/salles/listen/index.html

diferentes personas con diferentes características físicas las que se han usado de modelo. Por tanto, en el momento de elegir la colección HRIR es importante consultar los datos fisiológicos que corresponden a cada una de ellas, en general son las principales medidas torso, cabeza y oído.

La información se puede encontrar en dos tipos de datos: RAW y COMPENSATED. Y a su vez en formato WAV o MAT

- RAW
    - Datos WAV

      Para cada sujeto se dispone de 187 archivos WAV estereo de 24 bits, uno por cada posición. La notación genérica es:

      IRC_<subject_ID>_<status>_R<radius>_T<azimuth>_P<elevation>

Nombre	Ejemplo
subject_ID	1000 a 1999
status	R si los datos son RAW
radius	radio en cm
azimut	0 a 179 si la fuente se encuentra a la izquierda y 180 a 359 si la fuente se encuentra a la derecha.
elevation	315 a 345 si la fuente se encuentra sobre la cabeza, 0 si se encuentra enfrente y 15 a 90 si la fuente esta posicionada sobre la cabeza.

Tabla 2.2: Nomenclatura archivo RAW-WAV

    - Datos MAT

      Para cada sujeto hay tan solo un archivo MAT con toda la estructura. La información que contiene es:

      elev_v : una columna con el valor de las elevaciones.
      azim_v : un columna con todas los valores de azimut
      content_m : la matriz de datos, una fila por posición.
      type_s : una palabra clave que indica el tipo de contenido.
      sampling_hz : frecuencia de muestreo.

El emparejamiento de elevación y azimut es el siguiente:

Elevación	azimut
-45	0
-45	15
…	…
-45	345
-30	0
-30	15
…	…
-30	345
…	…
…	…
…	…
75	345
90	0

Tabla 2.3: Elevación vr. azimut

- Compensated

  La respuesta impulsional esta ecualizada. Solo en magnitud, la fase no se tiene en cuenta. El formato de datos es el mismo para WAV y MAT.

3. Implementación audio 3D

Desglose del audio tridimensional que se utiliza para crear el espacio sonoro utilizando como base la teoría explicada en el capitulo anterior. Explicación de las diferentes clases realizadas en Java y la relación que existe entre ellas.

3.1. Introducción

Una vez realizada la aplicación software utilizando la librería JOAL (openAL) y vistas las limitaciones impuestas por el sistema, se opta por crear un software básico de audio 3D con el que interacciona la superficie tangible. Este software escrito en Java toma como base principal una serie de librerías realizadas bajo el mismo lenguaje por C. Vilella[1].

El funcionamiento genérico del software se puede dibujar como en la *figura 3.1*.

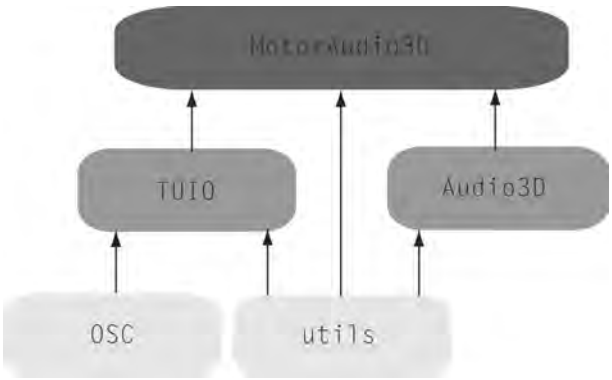


Figura 3.1: Diagrama de bloques algoritmo

MotorAudio3D controla los mensajes *TUIO* las librerías *Audio3D*. *TUIO* esta basado en los protocolos *OSC* y usa sus mensajes como base. Todos ellos utilizan las librerías *utils* en las que hay funciones matemáticas, vectores…

[1] Carles Vilella (carlesv@salle.url.edu). La Salle - URL.

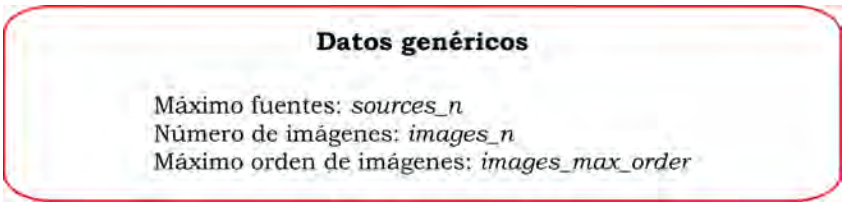
3.2. Motor Audio 3D

MotorAudio3D gobierna todo el sistema. Por un lado recibe datos de TUIO que son el enlace con la interfaz de usuario. Los datos de posicionamiento entran a formar parte del mecanismo del motor que realiza el procesamiento de audio tridimensional.

Se llevan a cabo las siguientes tareas:

1. Datos genéricos del programa

Se fija el número de fuentes máximo [sources_n], frecuencia de muestreo [fs], número y orden máximo de las imágenes [images_n, images_max_order].

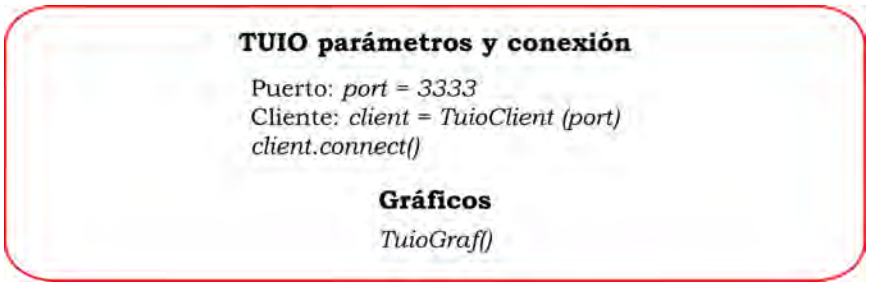


2. TUIO parámetros y conexión.

Se define, construyen y fijan los parámetros referentes a la conexión TUIO, que se inicia.

3. Gráficos

Se define y construye el objeto gráfico [TuioGraf] que representa los objetos TUIO en la interfaz.



4. Datos de componentes.

Se definen, construyen e inicializan los datos que utilizan las clases principales del programa.

- *Source.* Los datos necesarios para iniciar el uso de las fuentes: el nombre del fichero [sorce_filename], el tipo de fuente [source_from], volumen y posición.

- *Listener*. Para caracterizar el *listener* simplemente necesitamos saber la identificación [listener_id] del mismo (HRTF que lo describe), la posición y el tipo de reproducción (creación de un archivo o reproducción por altavoces) que se llevará a cabo.

- Sala. Para caracterizar la sala [rectRoom] es necesario conocer los vértices que la forman y el TR60. Es posible también caracterizarla con los datos de absorción, así el TR60 será calculado mediante la fórmula de Sabine.

5. Inicialización objetos.

Una vez conocemos todos los datos que forman los principales objetos estamos en condiciones de definirlos y construirlos. Principalmente son: HRTF, *Source*, *Listener*, *rectRoom*, *imageSources*, *reverbSchroeder* y *mixer*.

Se carga la HRTF del individuo que se ha seleccionado, lo mismo se hace con las *source*, inicializando en una determinada posición y orientación de salida desde donde esperará recibir ordenes para moverse. Lo mismo sucede con el *listener*. Tanto fuentes como receptor no serán activados hasta que se reciba la orden correspondiente.

El reverberador y el mixer están apunto para cuando se active el receptor y alguna fuente llevar a cabo el proceso que le compete a cada uno. Así mismo otras variables internas del programa deben

ser inicializadas ya que recibirán resultados desde otras clases, es el caso de las variables de sala (*itd, delay, out_hrtf_l/r*).

6. Bucle de ejecución

   - *Sources.* Para cada fuente que esté activa se realiza: la lectura de la fuente, se calcula la ITD, el retardo, la distancia y la HRTF de salida. Se hace exactamente lo mismo para las fuentes imagen y se calcula la reverberación.

     En este punto se realiza una primera mezcla que involucra la fuente, las fuentes imagen y la reverberación. A continuación una segunda mezcla, es la mezcla stereo del *listener*.

   - *Listener.* Se escribe el *listener.*

   - TUIO. Se cargan los datos TUIO de los objetos que se encuentran activos en la interfaz física. Para cada objeto activo, sea *source* o *listener*, se asigna la posición en la que se encuentra y la orientación.

     En caso de que los objetos físicos desaparezcan de la superficie estos desaparecen del entorno sonoro.

Bucle ejecución

Sources active true
source[n].read()
source[n]calculateParams()
source[n].getITD()
source[n].getDelay()
source[n].Filter_l() - hrtf_out_l -
source[n].Filter_r() - hrtf_out_r -

imageSources[n][i].calculateParams()
imageSources[n][i].getITD()
imageSources[n][i].getDelay()
imageSources[n][i].Filter_l() - hrtf_out_l -
imageSources[n][i].Filter_r() - hrtf_out_r -

reverb[n].reverb(source[n].getMono())

mixer[n].write(delay, itd, out_hrtf, out_rev)

listener.mixStereo()

Listener Stereo

listener.writeStereo()

TUIO

tuio.getObjActive()
tuio.getObjPosition()
tuio.getObjOrientation()

obj_active|| true
listener.nextRelativePosition (obj_position)
listener.setOrientation(obj_orientation)
source[].nextRelativePosition (obj_position)

source||.available false
source[].close()

3.3. *Source, Room, Listener*

Las relaciones que existen entre las clases se ilustran en el diagrama de bloques de la *figura 3.2.*

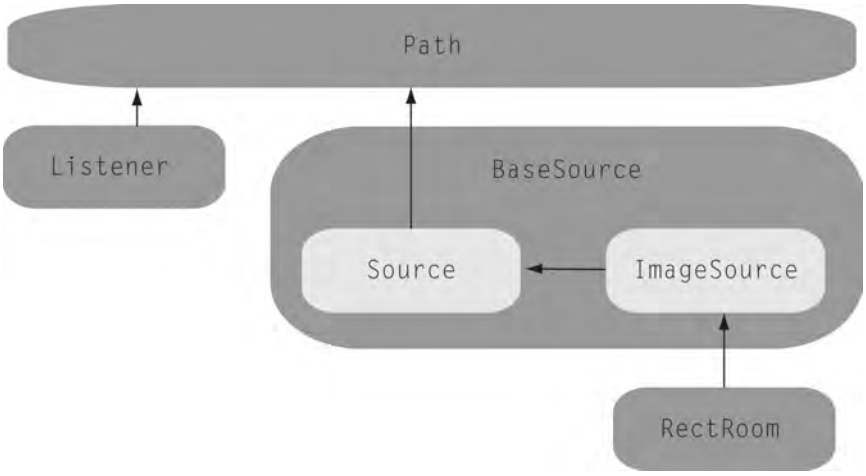


Figura 3.2: Diagrama de clases que relacionan *Listener, Source, Room.*

BaseSource.

Resuelva los procesos que son comunes a *Source* (fuentes) y *ImageSource* (fuentes imagen). Los principales parámetros son:

- Posición. [m_px, m_py, m_pz]
  Determina la posición en la que se encuentran las fuentes.

- Orientación. [m_ox, m_oy, m_oz]
  Determina la orientación en la que se encuentran las fuentes.

- Atenuación por distancia. [m_dst_gain, m_dst_filter]
  La atenuación por distancia se divide en baja y alta frecuencia. Para ello es necesario filtrar cada una de las bandas frecuenciales y calcular la ganancia.

- Filtro hrtf. [hrtfFilter_l, hrtfFilter_r]
  En este punto se encuentra el filtro hrtf que se utilizará para cada canal.

Source

Incorpora las funcionalidades propias exclusivamente de la fuente (*source*).

- Movimiento. [nextRelativePosition]
  Mediante la clase *Path* a la que se le envia directamente la posición donde se ha movido la fuente.

- Audio. [sndFromFile]
  Lee un audio monocanal desde un archivo origen [read]. Además nos dice si el audio que se esta leyendo esta disponible o ya se ha acabado [avaliable].

- Audio activo [active]
  Permite activa o desactivar el audio que se esta reproduciendo.

ImageSource

Incorpora las funcionalidades exclusivas de las fuentes imagen.

- Absorción [m_abs_filter, m_abs_coef]
  Filtro y coeficientes para aplicar la absorción que pueden provocar las paredes de la sala.

Listener

Es independiente de las fuentes, las características más importantes de su implementación son:

- Posición [m_px, m_py, m_pz]
  Determina la posición en la que se encuentra el receptor.

- Orientación [m_ox, m_oy, m_oz]
  Determina la orientación en la que se encuentra el receptor.

- Movimiento [nextRelativePosition]
  Se envia a la clase *Path* los puntos donde se ha movido el objeto que representa al receptor.

- Audio [sndToSpeaker, sndToFile]
  Se crea un audio stereo que se puede enviar a los altavoces o grabar en un archivo.

- Mezcla y escritura [MixStereo, WriteStereo]
  Mezcla y escribe el contenido stereo de todas las instancias de la clase *Mixer*. Es la mezcla final.

Path

Esta clase es la que da movimiento al sistema. A diferencia de otras aplicaciones, en este caso la variable tiempo es prescindible. Las posiciones de los objetos son enviadas en tiempo real mediante mensajes que el motor gestiona.

- Posicionamiento en tiempo real [nextRelPos]
  Tanto *Listener* como *Source* acaban entregando la información en tiempo real a esta clase que es la que gestiona la posición y si el objeto asociado esta activo o inactivo.

rectRoom

rectRoom crea una sala rectangular y tridimensional mediante dos vértices opuestos.

- Creación de fuentes imagen [getImageSources]
  Mediante la información de la geometria de la sala, el número [m_max_images] y orden [m_order], y obviamente la fuente [Source] crea las imágenes de estas últimas. Así también las va actualizando [updateImageSources]

3.4. HRTF

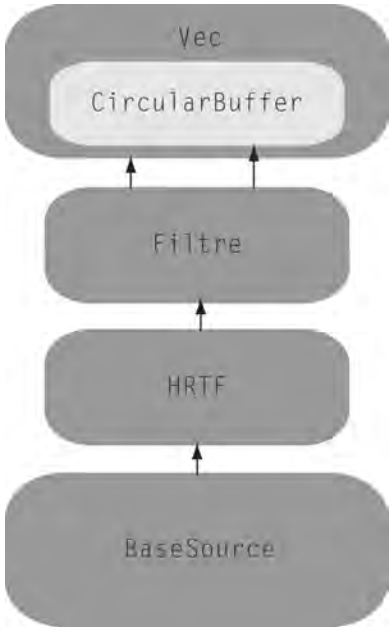


Figura 3.3: Diagrama de clases para HRTF

Para la implementación de este software se utilizan las HRTF del IRCAM. Antes de hacer uso de ellas es necesario modificarlas. Con la ayuda de un script en matlab, los archivos *wav* (stereo) serán pasados a fase mínima y se extraerán las ITDs. Es importante que el formato de los *wav* que contienen las HRIR sea el mismo que los audios a procesar.

Una vez cargada las HRTF y comprobado que el formato es correcto, se van pasando los ángulos [setAngles] que vienen de *BaseSource*. Si el ángulo no coincide se interpolan las HRIR de forma lineal.

BaseSource accede a los coeficientes [hrtfFilter_l, hrtfFilter_r] mientras que *Source* e *imageSource* heredan sus funciones [Filter_l, Filter_r].

3.5. Reverberador de Schroeder

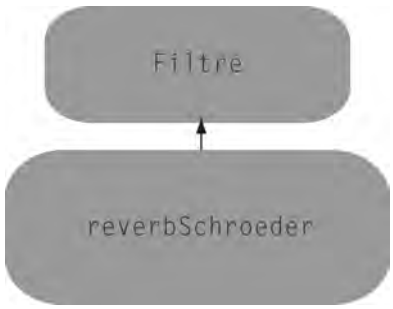


Figura 3.4: Diagrama de clases Reverberador de Schroeder

El reverberador de Schroeder implementa seis filtro, cuatro filtros *comb* y dos *allpass*. Los parámetros de los filtros se calculan a partir de la frecuencia de muestreo y el TR60. Básicamente hace uso de la clase *Filtre*.

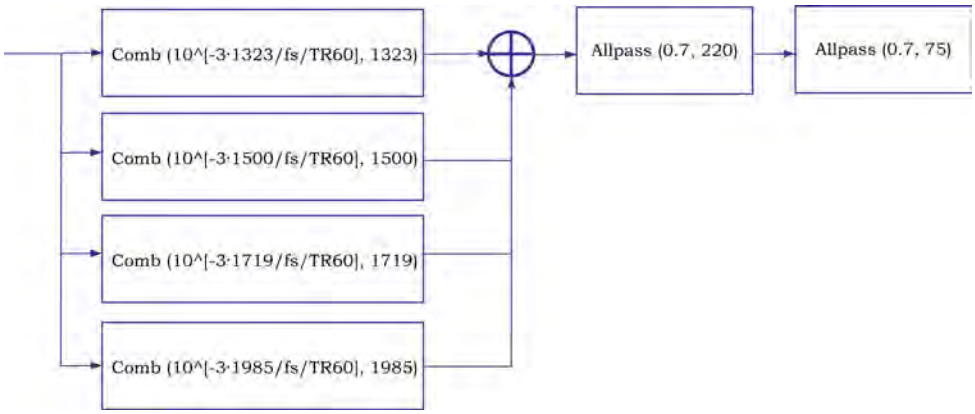


Figura 3.5: Reverberador de Schroeder

3.6. *Mixer*

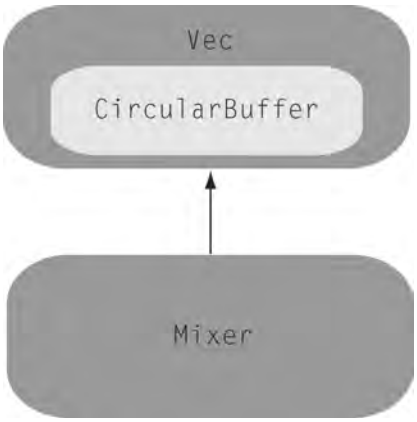


Figura 3.6: Diagrama de clases del Mixer

Se encarga de mezclar todo lo asociado a una instancia de fuente (a posteriori el *Listener* se encargará de mezclarlas todas). Una instancia de fuente debe mezclar tres informaciones relacionadas con la fuente.

- Camino directo: información directamente de la fuente una vez filtrada por la hrtf correspondiente.
- Imágenes: información de las N imágenes que se hayan creado una vez filtradas por sus hrtf correspondientes.
- Reverberación: reverberación de Schroeder derivada de la información de fuente.

Para el camino directo y las imágenes se utiliza la información de *delay* y las ITDs.

3.7. Entrada audio

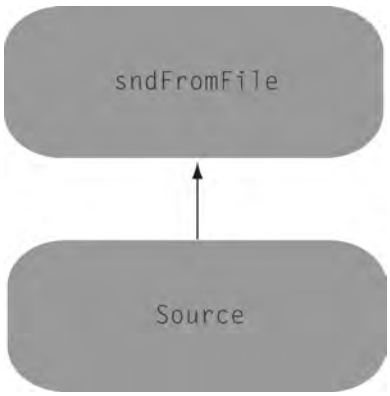


Figura 3.7: Diagrama de clases audio IN.

La entrada del audio se realiza mediante *sndFromFile*, en concreto con la función de lectura [.read] que se llama desde *Source*. Previamente es necesario inicializar [.init] y dar nombre al archivo que se quiere leer [.setFileName].

Se añade una sentencia [available] a la clase *sndFromFile* que nos indica si el archivo de audio que se lee a llegado a su fin, para en consecuencia tomar la decisión apropiada.

3.8. Salida audio

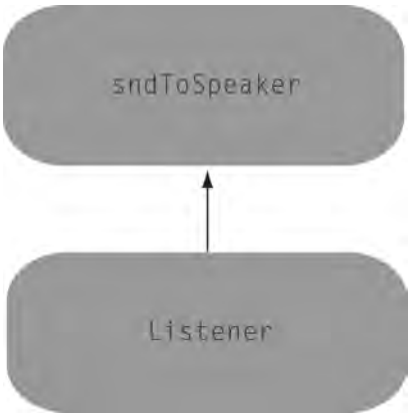


Figura 3.8: Diagrama de clases audio OUT.

El audio, en general, lo sacamos por los auriculares por lo que usamos la clase *sndToFile*. En este caso el que accede es el *listener*. Como se ha comentado anteriormente, es el *listener* el que lleva cabo el proceso de mezcla [.mixStereo] y escritura final [.writeStereo].

4. Resultados audio 3D

Se exponen los resultados objetivos y subjetivos que son consecuencia de la implementación del sistema de audio tridimensional. Se presta especial atención a los recursos computacionales que gasta y a la calidad del audio.

4.1. Introducción

En entorno de audio 3D en un principio se intentó implementar utilizando librerías libres en Java. Concretamente se las librerías utilizadas fueron JOAL (Java openAL), FMOD y JASS (Java audio síntesis system).

La implementación en JOAL acometía la auralización de manera muy correcta, se basa en ITD y ILD. Cabe tener en cuenta que las fuentes siempre se encuentran en el mismo plano XY. La inconveniencia en el uso de esta librería surge debido a la reverberación. El sonido sin reverberación suena en el interior de la cabeza, no da la sensación de encontrarse en un espacio determinado. La librería ofrece una reverberación, solo que para su funcionamiento necesita de un hardware específico. Este hardware ofrece otros muchos efectos y la posibilidad de añadir gran número de fuentes. Sin embargo, este hardware no esta disponible bajo Mac OS. Como consecuencia de todo ello se tiene que apostar por otra librería o por un software alternativo.

En referencia a FMOD y JASS sucede algo similar. La primera funciona muy bien en C++ pero para Java es procesado esta muy limitado. La segunda librería esta diseñado para procesar sonido en Java pero tiene algunos problemas a la hora de utilizar las HRTF.

Vistas las limitaciones que se presentan, se apuesta por librerías alternativas que realizar la implementación de audio binaural. Estas implementaciones tienen una serie de limitaciones, no funcionan con hardware dedicado y por tanto la eficiencia del desarrollo de los algoritmos es crucial. El número de fuentes y las reflexiones, que desde el punto de vista del software más fuentes, se ve limitado. En caso de añadir más fuentes, estas se corrompen y el sonido se entrecorta.

Se utiliza, por tanto, una serie de librerías que realizan todo el proceso de audio binaural necesario para implementar un entorno de audio tridimensional. El análisis de todo ese proceso se debe llevar a cabo desde dos puntos de vista, objetivo y subjetivo. En cuanto a la parte objetiva se analiza el comportamiento computacional del sistema, mientras que por lo que se refiere a la subjetividad, interesa sobretodo la percepción sonora de la sensación tridimensional.

4.2. Análisis subjetivo

- Espacio sonoro – Espacio tangible

En el espacio sonoro la ubicación de la fuente concuerda con el espacio tangible por regla general. En términos de localización la fuente es situada en el espacio tanto si se encuentra lejos o relativamente cerca de la cabeza del receptor. En principio, se podría pensar en algún salto si la fuente esta muy cerca del receptor o incluso una mala ubicación por el hecho de interpolar las HRTF. Los pequeños movimientos que se pueden dar a la fuente se corresponden con los del espacio tangible y lo mismo ocurre con los movimientos más amplios. Es decir, la localización del sonido es bastante buena incluso en los casos a priori más críticos.

- Sincronía en el cambio de posición

El funcionamiento en tiempo real de la aplicación es correcto, no se producen saltos discretos si no continuos, que se van realizando en tiempo real sin problemas, siempre y cuando no se sobrepase el límite de fuentes.

- Reverberación

Tanto el tiempo como la coloración de la reverberación proporciona la sensación de situarse en un espacio cerrado, permite que los sonidos tomen dimensión y suenen en el exterior de la cabeza.

En general la inmersión sonora es buena. Se tiene la sensación de estar en un espacio con ciertas fuentes que se desplazan alrededor del usuario. El sistema en reposo funciona de forma muy correcta, los filtros no provocan distorsiones, la reverberación es limpia y cuando se produce un movimiento el sistema funciona en tiempo real. Puede haber problemas si los movimientos de las fuentes son demasiado bruscos, en ese caso se puede llegar a percibir una perdida de muestras de audio que sonoramente se traduce en un sonido sucio. O bien una perdida de la localización que implica discontinuidad o salto en el espacio.

4.3. Análisis objetivo

En lo que se refiere a la aplicación que se ha realizado, el componente de procesado de audio tiene una dominio importante. El audio se encuentra interrelacionado con la interfaz mediante mensajes que parametrizan el comportamiento del entorno tridimensional. En este punto solo nos preocupan, en términos computacionales, el peso que tiene cada una de las clases, en definitiva el paquete completo dedicado exclusivamente al procesado de audio.

Mediante el plugin Profiler de Netbeans se calculan los recursos que gastan cada una de las clases, threads... de la aplicación. Centrados tan solo en lo que al audio 3D se refiere las clases que más CPU utilizan se respresentan el la *figura 4.1*.

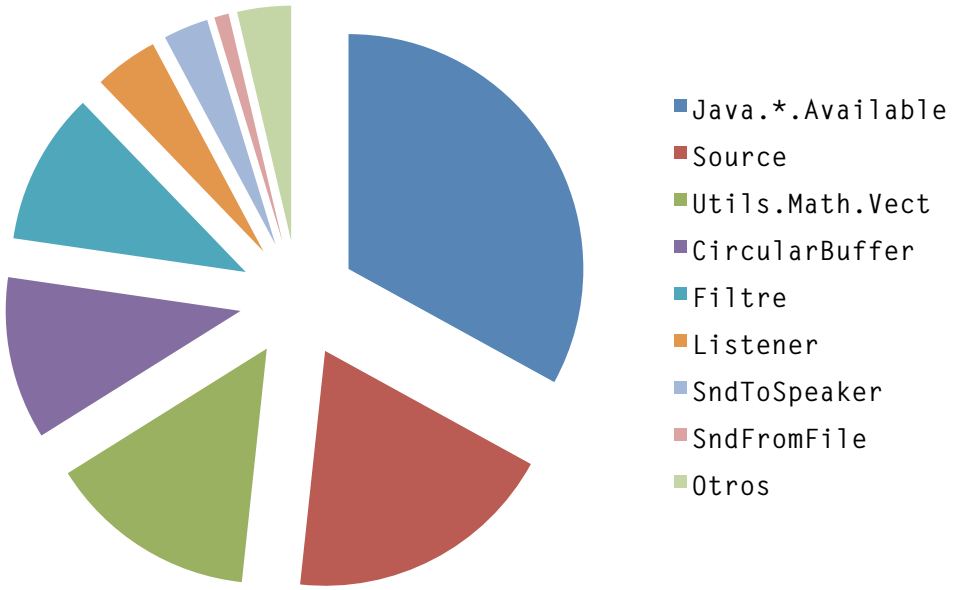


Figura 4.1: Uso de la CPU por clases Audio3D[1]

Llama la atención que la mayor parte del tiempo de CPU se este haciendo llamada a *javax.sound.sampled.audioinputstream.available*. Esto tal vez pueda limitar la eficiencia del algoritmo, ya que usa el 33%. En concreto al final del bucle de procesado se hace una llamada para conocer si se ha acabado de leer el archivo de audio y actuar en consecuencia. Este punto seguramente es mejorable y se puede plantear desde otra visión.

Por lo que respecta a las clases sucesivas es lógico los usos que se hacen de *Source, Vect, CircularBuffer*... Por source pasa todo lo referente a las fuentes que van modificando su posición de manera

[1] El cálculo es una media consecuencia de utilizar durante un minuto la aplicación de forma normal, es decir, añadiendo de forma gradual los eventos e ir eliminándolos de manera sucesiva.

eventual. *Vect* y *CircularBuffer* se utilizan constantemente como herramienta para filtrar, cargar datos… con lo que se usan prácticamente todo el tiempo.

Del total de CPU que usa el programa completo, incluyendo TUIO, audio 3D, gráficos… la mayor parte del coste computacional se lo lleva la parte de audio 3D. Por lo que es la parte más delicada en cuanto a eficiencia se refiere y por tanto la más susceptible a causar problemas en el procesado.

Así el máximo número de fuentes que permite el sotfware son dos. Ya sean fuentes reales o virtuales. En ese caso se producen saltos en el audio, se pierde calidad y verosimilitud.

5. Resultados

Se presentán los resultados objetivos y subjetivos del conjunto del sistema, tanto de la aplicación cliente de audio tridimensional, como de la NUI que se realiza en el trabajo adjunto y que funciona como un todo. En cuanto a los resultados objetivos se describen principalmente los costes computacionales a razón de uso de la CPU y consumo de memoria por parte de la aplicación, de modo global y de algunas clases concretas.

5.1. Introducción

El objetivo principal del proyecto era crear un entorno virtual de audio en 3D que respondiera a los eventos que suceden en un entorno físico, multiusario y en tiempo real. Para analizar todo el sistema al completo es imprescindible dividirlo en dos partes. En la primera se desarrolla la viabilidad y consistencia para la creación de un entorno que funcione en tiempo real, lo que implica un análisis del uso de recursos del sistema y sus limitaciones. En segundo lugar la calidad tanto de la aplicación de audio como de la interfaz, y sus posibles carencias.

5.2. Análisis subjetivo

La interfaz funciona en tiempo real sin problemas, el framework envia los mensajes a la aplicación y esta actúa con una latencia asumible y prácticamente impercentible en la mayor parte de casos. El problema o límite no se da tanto por el proceso de visión por computador que se lleva a cabo debido al reconocimiento de fiduciales en la interfaz, si no más bien por el volumén de muestras de audio que hay que procesar en un determinado instante. Es entonces cuando el sonido pierde calidad, se producen distorsiones…

A nivel de experiencia de usuario, la interfaz te permite interactuar de forma natural con objetos físicos, de forma intuitiva. Aún así un feedback visual haría la experiencia algo más agradable y probablemente más intuitiva. Las dimensiones de la superficie y la técnica utilizada, en la que el proyector debe estar dentro de la caja, hacen inviable su uso.

En general la experiencia tanto del entorno sonoro, como del entorno tangible permite un uso verosimil e intutivo respectivamente. Son las limitaciones computacionales las que merman de algún modo poder tener una experiencia más completa a nivel sonoro, y las carencias en las dimensiones de la interfaz las que impiden tener una experiencia a la vez que auditiva visual.

5.3. Análisis objetivo

La aplicación incluye: la comunicación y la recepción de mensajes (OSC, TUIO), el procesado del audio (Audio3D) y la presentación gráfica (TUIOGraf). Todo ello está programado utilizando Java y consume unos recursos del sistema que no son menospreciables, sobretodo cuando trabajamos en tiempo real.

Bajo estas condiciones se producen varias limitaciones. Para tener una sensación sonora de cierta calidad con la implementación realizada, que incluye todos los estadios para conseguir un sonido binaural correcto, el número de fuentes máximo se reduce a dos. Cuando se lee fuentes, se refiere tanto a fuentes propiamente sobre la superficie, como fuentes virtuales que se crean como reflexiones. Este hecho límita automáticamente poder tener un número indeterminado de fuentes sobre la superficie o bien poder crear suficientes reflexiones para tener una noción de sala más verosimil.

Los recursos que gasta la aplicación se describen a continuación. Han sido calculados mediante el plugin *Profiler* de *NetBeans* y el *Monitor de Actividad* de *MAC OS*.

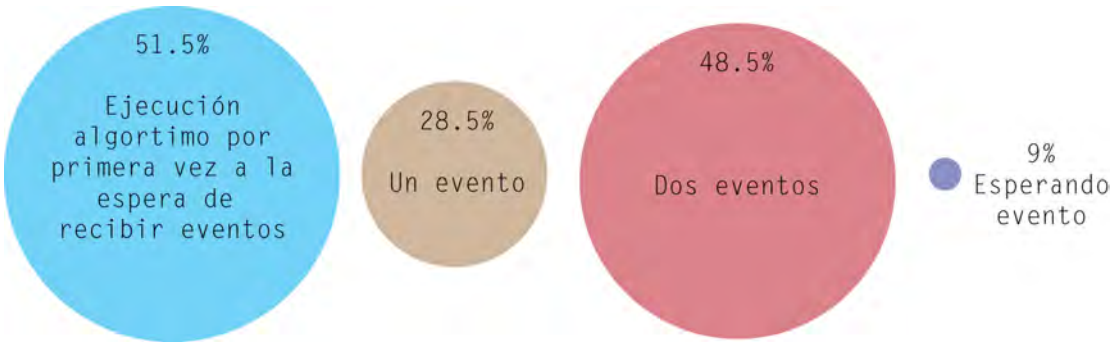


Figura 5.1: Gasto de CPU de la aplicación con respecto al total del sistema.

En la *figura 5.1* se puede observar el consumo que tiene la CPU cuando se ejecuta el programa con respecto al total del sistema. Se ha dividido en cuatro momentos importantes que se pasan analizar.

- Ejecución del algoritmo por primera vez a la espera de recibir eventos [51.5%]

  Se da lugar a esta situación nada más ejecutar la aplicación sin disponer encima de la superficie ningún objeto. Es decir, el sistema esta recien iniciado y a la espera de recibir eventos.

  En ese momento y de media, la aplicación consume un 51.5% de los recursos del sistema. Es el punto álgido en cuanto a consumo. Es importante saber que está pasando en la aplicación para que el consumo de recursos sea de estas proporciones.

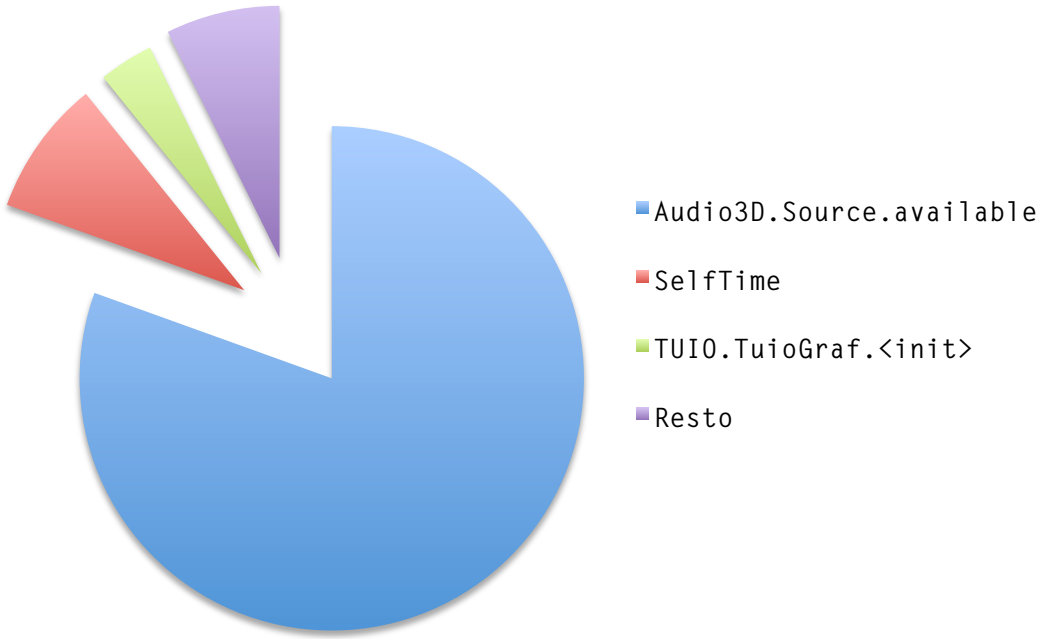


Figura 5.2: Consumo de CPU al iniciar la aplicación

Se puede observar como el 80,5% del tiempo la CPU es utilizada por el método *Audio3D.Source.available().* La función de este método es preguntar constatemente si hay muestras disponibles que leer, retornando un booleano. Es necesario para una vez a acabado el archivo de audio hacer un loop, en consecuencia poder volver a cargar y reproducir el audio.

- Un evento [28.5%]

  En esta situación se esta reproduciendo un solo audio. El consumo curiosamente disminuye con respecto al momento inicial donde se inicializan todos las clases y se espera evento.

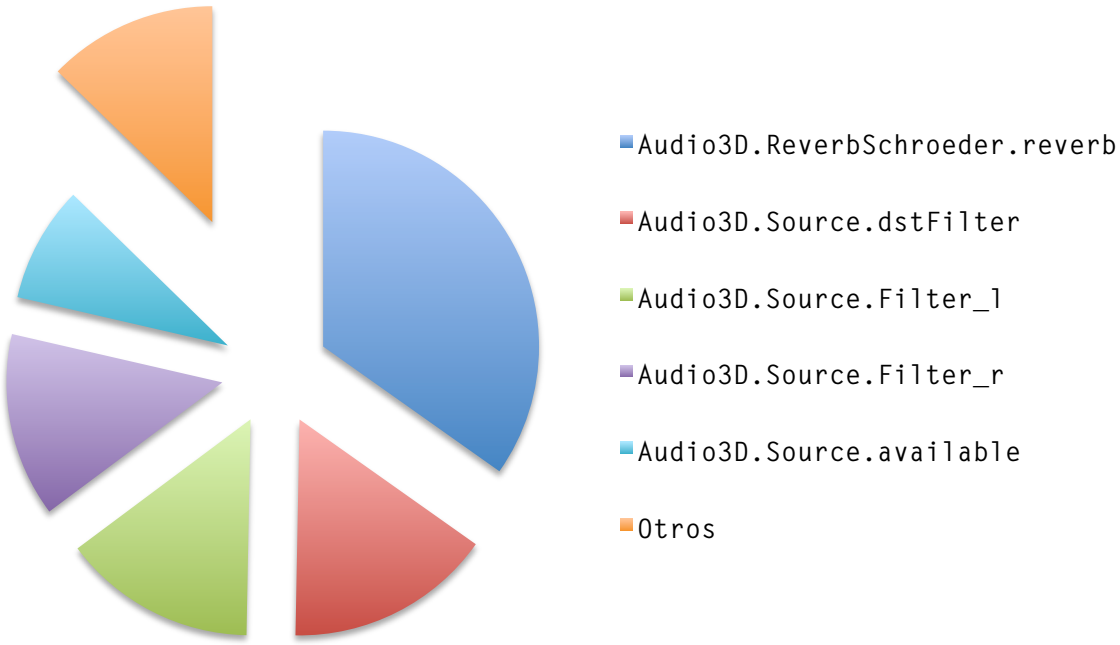


Figura 5.3: Consumo de CPU con una fuente

El mayor uso de recursos recae en la reverberación, en la cual se involucra el *BufferCircular* y la clase *utils.math.vec*. En este caso los resultados son previsibles. *Audio3D.Source.available* sigué estando presente con gran importancia.

- Dos eventos [48.5%]

  Se añade un audio, un evento. En este caso el consumo total de la aplicación es más importante, entorno al 48.5%.

  La disposición es muy similar que en el caso de una fuente como era de preveer, sin embargo en general el uso del total de CPU es prácticamente el doble por el hecho de anadir una fuente auxiliar. Aquí encontramos el límite para que el audio se procese y reproduzca con cierta calidad.

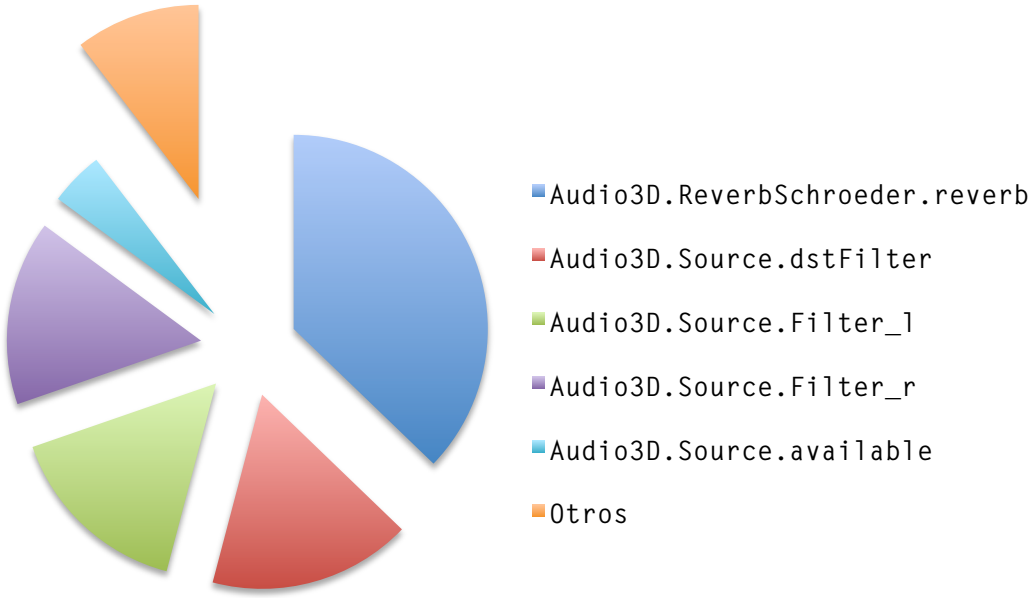


Figura 5.4: Consumo de CPU con dos fuente

- Esperando evento [9%]

  La sucesión lógica de eventos se sucede en primer lugar introduciendo las fuentes y a continuación se van eliminando y añadiendo a discreción. El último escenario a estudio es en el que las fuentes han sido eliminadas y el receptor está esperando detectar fuentes.

  El proceso de CPU genérico en este escenario es el más bajo de cuantos nos planteamos. Solo consume el 9% del total de CPU.

  En este caso vemos que el consumo se centra otra vez en solicitar si hay archivo de audio disponible. A priori se podría pensar que la función en este escenario y el primero de los comentados debería ser idéntico o al menos parecido. En realidad en ambos se esperan eventos, sin embargo parten de un punto distinto, en el primero se acabn de cargar los datos e inicializaciones, en este caso esta todo dispuesto para recibir una fuente cuando con anterioridad ya se han estado reproduciendo.

  Vemos que se ha añadido el Listener y la Reverb entre los principales métodos del primer movimiento. El Listener esta en el escenario esperando recibir fuentes, lo mismo sucede con el reverberador.

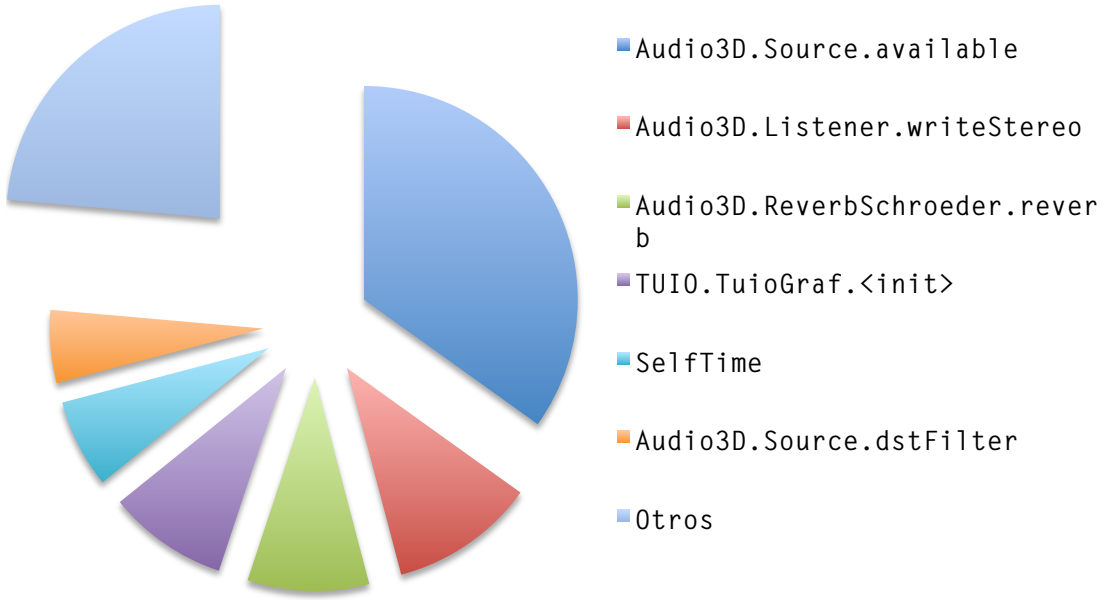


Figura 5.5: Consumo de CPU en espera de eventos

Los consumos de memoria también son cruciales en una aplicación de estas características. Para su análisis utilizando el mismo plugin que en el anterior caso y utilizando los mismos escenarios como resultado tenemos lo que se representa en las siguientes gráficas.

En cuanto a Heap Size y Used Heap, es la memoria total y la memoria usada por la aplicación, *figura 5.6*. En la *figura 5.7* se dibuja el tiempo que pasa entre la recoleccion de basura y el número de generaciones que sobreviven.

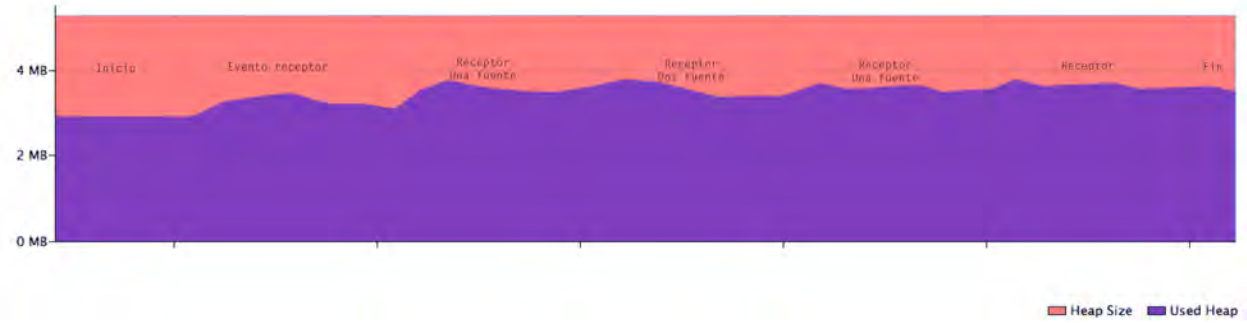


Figura 5.6: Uso de memoria heap

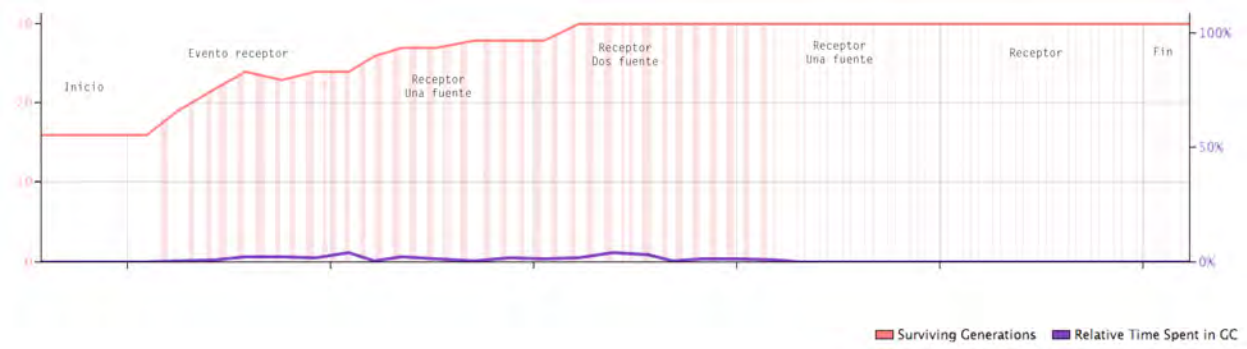


Figura 5.7: Uso de memoria recolección de basura

La recolección de basura es la eliminación de los procesos de memoria que se crean y se dejan de utilizar. Se miden a tres niveles:

1. Tiempos relativos: porcentaje del tiempo total de ejecución que la recolección de basura corre con todos los threads suspendidos.

2. Generaciones que sobreviven: las diferentes edades de todos los objetos localizados en memoria desde el inicio de sesión.

3. Edad de los objetos: es número de recolectores de basura del objeto que sobreviven.

6. Conclusiones

Análisis crítico de la consecución de objetivos planteados en el proyecto, poniendo cierto énfasis en las limitaciones y sus posibles mejoras. Se describen las conclusiones del proyecto en general, es decir, se incluyen las conclusiones del todo que forman la aplicación cliente de audio tridimensional y la NUI que se desarrolla en el trabajo adjunto.

Los objetivos principales del proyecto se han cumplido a cabalidad. Si bien, no con ciertas limitaciones que se han ido argumentando a lo largo del trabajo. A continuación se repasan los principales objetivos.

1. Interacción entre el entorno físico y virtual.

   Las metodologías para crear un entorno físico que mantenga una comunicación con un entorno virtual son diversas. En este caso se debía realizar un reconocimiento de fiduciales en objetos físicos y transmitir esta información con un protocolo rápido y eficiente. Era necesario hacerlo con un sistema estático fijo, y por esa razón se pensó en una table-top.

   La interacción de este tipo de sistemas naturales es intuitiva y sencilla en cuanto a la idea, una cámara un framework de visión por computador y un protocolo de comunicación son suficientes. Las técnicas sin embargo propiamente físicas para la detección de las formas encima de una superficie son diversas y tienen unas limitaciones que se superan dando al entorno lumínico unas ciertas condiciones. Esto influye de forma directa tanto en el tipo de objetos que podrá detectar la cámara como en la fluidez de la interacción. La técnicas que se ha utilizado para implementar dicha table-top permite de forma muy precisa detectar los fiduciales si estos tienen un tamaño y una iluminación correcta, esto último no siempre es posible.

   La idea era realizar esta interacción entre el mundo tangible e intangible, entre unos objetos que representan fuentes sonoras y un sala virtual donde estas se reproducen con la peculiaridad de poder localizarlas espacialmente. Esta idea se puede aplicar a otro tipo de interfaces que encontramos en plataformas propietarias.

La interacción se lleva a cabo con un protocolo que permite la comunicación de una manera rápida y eficiente, para esta aplicación no ha habido ningún tipo de restricción al respecto de la comunicación.

2. Medio fisico - NUI

   Las técnicas para construir natural user interface son muy diversas, como diverso es el concepto en si mismo. Estas interfaces pueden ser multitouch, reconocer objetos, gestos… y a su vez se pueden implementar utilizando técnicas IR, capacitivas, ópticas… Después de estudiar exactamente en que consisten cada una de esas técnicas y observar que la mayor parte de ellas usan alta tecnología propietaria es necesario tomar el camino que permite la experimentación con técnicas y software libre, del que se puede extraer conocimiento y armar instalaciones interesantes.

   Las técnicas ópticas permiten crear diferentes sistemas según sea la aplicación que uno quiere dar. En este caso el reconocimiento de fiduciales y la portabilidad de la mesa eran dos características importantes para su construcción. Esto por otro lado implica unas limitaciones, como consecuencia de estas la detección de blobs es dificultosa, sobretodo en las periferias de la superficie, y el feedback visual que aporta una interacción más natural e intuitiva no es posible realizar sobre la superficie por sus reducidas dimensiones.

3. Medio virtual – Audio 3D

   El sistema de audio 3D se puede implementar utilizando librerías en diferentes lenguajes, plugins para programas de procesado de audio o se puede construir una librería propia que siga la teoría binaural, holofónica…

   Las librerías más completas están disponibles en C++, su rendimiento es generalmente bueno y no utiliza tantos recursos de CPU o memoria. Sin embargo, en Java los recursos limitan sobremanera la cantidad de procesado que se lleva a cabo y esto implica directamente una perdida de atributos sonoros, ya que a menudo repercute sobre la reverberación o la teoría de imágenes, y por tanto en el entorno.

   Sin embargo a sido posible realizar la actualización de las posiciones y por tanto de las localizaciones que se reciben desde la interface en tiempo real. Con la limitación de fuentes y con un uso responsable de la interfaz.

4. Extrapolación del concepto

   La mínima esencia del concepto es tocar bits, es decir, modificar un espacio creado de manera virtual mediante objetos físicos. Visto desde dos puntos de vista alejados, nos encontramos con interfaces que modifican el sonido, estás nuevas interfaces pueden ser table-top ópticas, superficies multitouch, realidad aumentada, human brain computer… Si lo miramos desde el lado sonoro, en este caso se trata de un entorno tridimensional, pero las aplicaciones son amplias dependiendo de la creatividad de cada uno.

   En rigor el uso de estas interfaces permite una mayor interactividad, no solo con la máquina, si no también con la usuarios de esta. El acercamiento de las máquinas al comportamiento humano y no viceversa. En este caso utilizando con nexo un espacio de audio tridimensional.

7. Líneas de futuro

Se expone hacia donde avanzan las nuevas interfaces y la aplicaciones de audio tridimensional en diferentes medios. Con todo ello se explica los futuros trabajos que se pueden desarrollar a partir del que se presenta en este proyecto.

7.1. Introducción

Tanto las nuevas interfaces como las tecnologías 3D sufren un cambio de paradigma importante en este tiempo. Por una parte el cambio de tendencia a la hora de ver cine y televisión con las tecnologías de imagen en 3D hace que se expanda la implementación de dispositivos en otros campos, como por ejemplo los videojuegos. Por la misma razón y dado que el audio sigue siendo multicanal y no localiza exactamente el lugar donde se encuentran las fuentes de audio, se abren líneas de investigación para implementar audio en 3D en todas sus variantes e incorporarlo a este tipo de sistemas. Las interfaces naturales, por su lado, van sustituyendo progresivamente las GUI, de manera que es importante el volumen de penetración de la interfaz multitouch e incluso otras realizadas ad hoc para una aplicación.

Es interesante saber que está sucediendo con ambas tecnologías para entender como vamos a interaccionar con las máquinas y que beneficios se pueden obtener de ellas al implementar aplicaciones para estos soportes.

7.2. Nuevas interfaces

La interfaz que se ha implementando concretamente en este proyecto, es decir, la que usa técnicas ópticas (*rear ID*) es de primera generación. Este tipo de interfaces usan técnicas ópticas y de visión por computador. Es decir, usa elementos comunes, como cámaras, luz infrarroja, proyectores… para realizar la detección física de los elementos y un software de visión por computador para la detección digital y el posterior procesamiento.

La siguiente generación de este tipo de interfaz reduce considerablemente el volumen y la hace más manejable. Es el caso de la Surface de Microsoft, que usa una tecnología propia, PixelSense,

donde cada píxel es una pequeña cámara infrarroja. Esto reduce considerablemente el volumen de forma que se parece mucho más a una mesa convencional ya que no es una caja cerrada con un cámara en su interior.

La consultora de tendencias tecnológicas Gartner [13] en la ultima gráfica (2010) sobre tecnologías emergentes situaban entre las que tienen expectativas más altas en evolucionar positivamente las nuevas interfaces basadas principalmente en realidad aumentada y las *multitouch*. En periodo de crecimiento con largo recorrido, y un periodo de implantación a largo plazo se encuentra *human brain interface*.

Las interfaces *multitouch* las podemos encontrar en dispositivos móviles y tabletas, con los sistemas operativos de estas se pueden realizar aplicaciones de todo tipo. Por ejemplo el protocolo TUIO que se ha utilizado en este proyecto está implementado para ser utilizado con iPad, iPhone, Android…

7.2.1. El futuro de la interfaz

De un tiempo a esta parte se han ido desarrollado otras tecnologías e interfaces que sería interesante poder utilizar en un futuro para una aplicación similar a esta. Este trabajo conceptual puede ser extrapolado a una gran cantidad de interfaces, ya sean realizadas ad hoc (como este caso) o bien realizadas para una plataforma o dispositivo genérico.

A continuación se describen una serie de interfaces donde se podrían realizar aplicaciones o implementar aplicaciones como la que se describe en este proyecto:

- Interfaz gestual

Los magnetómetros compactos, acelerómetros y los giroscopios hace que sea posible detectar el movimiento de un dispositivo. Usando este tipo de sistemas funcionan tanto el iPhone como la Wii.

El siguiente paso es conseguir que los ordenadores reconozcan el movimiento de las manos y del cuerpo. El Eye de Sony permitió reconocer movimientos simples relativamente de forma fácil. Sin embargo cuando se trata de movimientos en 3D más sofisticados y con luz irregular es más complicado.

Oblong[1] (startup de Los Ángeles) a desarrollado un sistema operativo espacial que permite reconocer gestos mediante la utilización de unos guantes.

Las interfaces naturales, interfaces que interactúan con el entorno 3D de la misma manera como se interactúa con el entorno físico, permite a personas que no están familiarizadas con los ordenadores poder utilizarlos sin mayor problema. Microsoft muestra en unos videos conceptuales "Home, work and play" que presentó en la Conference of Human- computer Interaction (CHI) 2009 [14], la forma relacionarnos que creen para el futuro. En la actualidad ya disponen de una tecnología revolucionaria Kinect.

Kinect permite la interacción con una máquina (en principio la XBox) mediante una interfaz natural de usuario que reconoce gestos, vox, imágenes y objetos. El funcionamiento de la misma se basa en laser que trabaja en longitudes de onda de infrarrojos y que se emiten sobre ele espacio tridimensional, un sensor de imagen CMOS detecta los reflejos del haz de luz dentro de un patrón de segmentos y puntos, generando un mapa de intensidades de cada uno de ellos. Se obtiene resoluciones del orden de 1cm en el eje Z y de pocos milímetros en el plano XY. Mediante otro sensor CMOS se detectan los colores que se añaden a los datos recogidos por la otra cámara. Utilizando los datos capturados por las cámaras se reproducen en la pantalla. Los comandos de voz son capturados con cuatro micrófonos y cancelan ruido y ecos que se puedan producir en el espacio. Las aplicaciones que se pueden realizar con este sistema trascienden la idea inicial de Microsoft ya que el sistema se ha abierto debido a la perspicacia de algunos desarrolladores y creadores. IEEE Spectrum publicó un articulo mencionando los diez sistemas más interesantes creados a partir de este dispositivo.

Interfaz gestual colgable [15]

Se trata de una interfaz muy interesante, a parte de espectacular su construcción es relativamente simple y el software que la hace correr parece que va a ser *CreativeCommons* lo que va a permitir una gran experimentación y por tanto muchas posibilidades.

Es una interfaz que permite añadir al mundo físico información virtual mediante el uso de gestos naturales de las manos. El prototipo que explica como funciona esta interfaz es el SixthSense[2]. Se basa en la idea de añadir una entrada más al método de toma de decisión que usamos, es decir, proveer de información sobre los objetos que nos envuelven para poder tomar de decisiones (de ahí lo de sexto sentido).

---

[1] http://oblong.com/
[2] http://www.pranavmistry.com/projects/sixthsense/

La miniaturización de los dispositivos computacionales hace que llevemos en el bolsillo un aparato que esta constantemente conectado al mundo digital, pero no hay un enlace directo entre nuestras acciones y ese mundo virtual. Este dispositivo consigue crear esa relación entre el mundo tangible e intangible de manera que nos permite interactuar con la información mediante gestos naturales.

El prototipo consta de un pequeño proyector, un espejo y una cámara. Todos ellos están acoplados a un elemento que se lleva colgado del cuello. Cámara y proyector están conectados a un dispositivo que, a su vez, está conectado al ordenador. El proyector proyecta la información en objetos físicos, paredes y superficies diversas, mientras tanto la cámara reconoce y sigue los gestos realizados por las manos mediante técnicas de visión por ordenador. El software captura y analiza el *stream* de video y localiza los marcadores de colores de los dedos.

Las aplicaciones que se han realizado con este aparato demuestra la versatilidad y viabilidad de su uso. Permite tomar fotos simplemente enmarcando la escena con los dedos, mover textos e imágenes del papel al mundo virtual, ver información sobre productos, datos en tiempo real del tiempo y videos sobre noticias en los periódicos de papel…

- Realidad aumentada

  La realidad aumentada combina elementos reales y virtuales para aumentar nuestro compresión del mundo. Un *smartphone* con cámara, GPS y pantalla se convierte en una herramienta muy potente para la realidad aumentada. Hay una gran cantidad de aplicaciones que usan esta tecnología, algunas que actúan como interfaces, por ejemplo Popcode[3] o Twincle. Si además se añade un proyector, la imagen proyectada puede interaccionar con el mundo real.

  Otras tecnologías en periodo de investigación o bien por novedosas, o bien por la dificultad que implica implementarlas con éxito, puede que sean el marco en el que se encuentren las aplicaciones como la que se desarrolla en este proyecto.

- Interfaz de control cerebral

  Las interfaces para el control cerebral de una computadora llevan muchos años en investigación pero su puesta en marcha es dificultosa, no siempre responden adecuadamente al usuario y se necesita unos grandes tiempo de entrenamiento.

---

[3] http://www.popcode.info/

Sin embargo recientemente investigadores de la universidad de California han conseguido hacer una llamada a un teléfono móvil utilizando una interfaz de este tipo.

Tzyy-Ping Jung, investigador en el Centro Swartz de Neurociencia Computacional en la Universidad de California a creado una interfaz [16] que utiliza electrodos de encefalograma en la cabeza para analizar la actividad eléctrica del cerebro, se envía la información vía bluetooth a un teléfono que utiliza algoritmo para procesar la señal.

Los usuarios son entrenados mediante un novedoso sistema de feedback visual, que consiste en mostrar imágenes en un ordenador a diferentes velocidades y de manera casi imperceptible, estas oscilaciones son detectadas en la línea occipital del cerebro. En este caso concreto las imágenes eran los números del 0 al 9 parpadeando cada uno de ellos a diferentes frecuencias, estas frecuencias son las que mediante el EEG se pueden detectar y por tanto saber que número esta mirando el usuario.

No todas las personas son igual de sensibles a este tipo de interfaces. En este caso se llego a que siete de cada diez personas introdujeron el 100% de los números correctamente. Con lo que el sistema para este propósito funciona bastante bien.

Hay otras técnicas para realizar este tipo de interfaces, por ejemplo, una que usa una señal del cerebro conocida como P300. Es utilizada como señal de control para conocer si a una persona algo le ha llamado la atención. En este caso se necesita un entrenamiento mas largo.

La principal problemática que tiene la técnica del feedback visual es que es necesario un gran estimulo visual, con lo que en una pequeña pantalla de teléfono es difícil que este estimulo se pueda dar.

- Pantalla táctil de patrones (T-Pad)[17]

  Es una pantalla táctil que utiliza vibraciones de alta frecuencia que utiliza vibraciones de alta frecuencia para crear una capa delgada de aire entre el vidrio y el dedo del usuario. Variando las vibraciones a medida que el usuario mueve el dedo permite que las diferentes partes de la pantalla se sientan como suaves o adherentes. Se presento un prototipo en la *Conference on Human Factors in Computing Systems* de este mismo año. Se trata de una pantalla más física que permite de alguna manera sentir los botones. El objetivo de esta nueva interfaz, no es solo ofrecer señales de la interacción, si no ofrecer experiencias que se tienen cuando interactuamos con el mundo real.

Funciona básicamente con un piezoeléctrico que vibra a una frecuencia alrededor de los 24 kHz. Cuando un usuario pasa el dedo por encima de un botón las vibraciones se paran lo que da la sensación de que la pantalla es más rugosa. Cuando se mueve una barra de desplazamiento se sienten las marcas de graduación de la misma.

Aún así, se ha presentado el prototipo recientemente, le falta mucho camino para ser una realidad. El prototipo es muy grande y utiliza mucha energía. Solo produce las sensaciones cuando hay movimiento, el hecho de tocar la pantalla no produce ninguna sensación. No obstante es otra forma de crear un entorno más cercano a la realidad que al mundo virtual.

- Paredes sensibles al tacto [18]

  Microsoft, con Desney Tan a la cabeza, y la universidad de Washinton, ven la posibilidad de crear interfaces en paredes usando la radiación electromagnética. La radiación electromagnética ambiental creada por móviles, ordenadores y otros aparatos eléctricas dentro de las paredes se suele considerar ruido. Sin embargo se decide utilizar este ruido como centro de la interfaz.

  Cuando una persona toca una pared con radiación, el cuerpo actúa como antena y la señal eléctrica distinta, dependiendo de la posición del cuerpo, la distancia… Esta señal puede ser recogida e interpretada por un dispositivo que esta cerca del cuerpo. El cuerpo puede convertir una señal ruidosa en una señal útil para una interfaz gestual. El sistema de prueba consta de una banda de conexión a tierra (un brazalete que se utiliza para evitar la acumulación de electricidad estática en el cuerpo), esta banda esta conectada aun convertidor analógico/digital y este a su vez a un ordenador que recibe los datos. Según Shwetak Patel el próximo paso es conseguir procesar la información en tiempo real y utilizar un dispositivo más pequeño (smartphone).

  Desde el MIT se cree que tendrá casos de uso limitado ya que es importante que el usuario tenga una referencia visual sobre donde debe interactuar con la pared. Es una idea que necesita una maduración. Sin embargo la idea es rupturista y lo suficientemente innovadora como para llamar la atención.

  Por otro lado S. Patel a utilizado sistema de electricidad, agua y ventilación para localizar a personas en el interior de edificios. D. Tan a trabajado con sensores que funcionan con la actividad del cerebro y muscular para controlar componentes eléctricos de forma inalámbrica.

7.3. Audio 3D

La aplicación de audio tridimensional que se ha implementado en este trabajo funciona en tiempo real respondiendo a los mensajes que se reciben desde una mesa interactiva. Esta aplicación, previas modificaciones, debería poder funcionar sobre cualquier interfaz que enviará mensajes TUIO, ya que es el protocolo de comunicación que se utiliza. Esto permitiría su uso en smartphones y otros dispositivos portátiles capaces de comunicarse con la aplicación.

Esto puede ocurrir en diversos escenarios; trabajos conceptuales similares al presentado aquí, en que utilizando la interfaz propia del dispositivo se crean espacios virtuales en 3d, o bien haciendo uso de sistemas de detección de movimiento (acelerómetros, giroscopios…) modificar las fuentes de audio según sea el movimiento, o bien mediante sistemas de realidad aumentada conseguir escuchar los sonidos en el entorno virtual desde el lugar físico de donde provengan. Por tanto las posibilidades de aplicación son múltiples con la tecnología de la que hoy se dispone.

Si nos desvinculamos de la librería que se ha utilizado en este proyecto y nos centramos en librerías disponibles para otros dispositivos (Android, iPhone…) el conjunto de aplicaciones que se pueden implementar crece aún más.

Las tendencias tecnológicas a priori sitúan el audio 3D en el cine y la televisión. En los juegos y en la realidad virtual en ocasiones ya se utiliza, aunque probablemente en el futuro se extienda más o como mínimo se mejoren las prestaciones que se ofrecen.

A continuación se presentan algunas ideas de las aplicaciones que se le pueden dar al audio 3D en un futuro más o menos cercano.

- Audio georeferenciado

  Geoaudio[4] es una audio-guía-inteligente para su uso con dispositivos móviles. El audio se recibe desde el punto de vista del observador de manera que el realismo de la alocución es mucho más elevado. Se usa para el turismo, rutas urbanas, museos, exposiciones… Es el propio dispositivo móvil el que selecciona la información auditiva en función de la posición.

[4] http://mobility.grupodeimos.com/?q=es/node/82

- Chips para smartphones y tabletas.

  Los últimos chips que se presentan, por ejemplo el Tegra3[5] (presentado en el último Mobile World Congres de Barcelona) permite la descodificación de video H.264 de alta definición y soporte para audio 3D. Esto permitiría la decodificación de video con audio 3D, lo que de alguna manera indica por donde van a pasar los próximos estándares de video.

- Audio 3D en al industria del entretenimiento.

  El profesor Edgar Choueiri, investigador en la Universidad de Princeton, junto con la compañía Cambridge Mechatronics están diseñando sistemas de sonido en 3D para televisores con esta tecnología. Tanto Sony Pictures como ESPN se han interesado por este proyecto para unir el video 3D con el audio.

  Edgar Choueiri además habla de un futuro en el que no solo se oirá una mosca dar vueltas alrededor de la cabeza, si no que esta se verá e incluso si se posa encima de la nariz, se notará.[]

Por otra parte algunos juegos también van a aplicar el audio 3D, es el caso de un juego de puzzles con ambientación de terror para Nintendo DS3, en este juego será básico para su jugabilidad el audio 3D que se implementa utilizando la tecnología Otophonics.

7.4. Conclusiones finales

En general este trabajo puede coger diferentes caminos, por una parte perfeccionar las técnicas de audio 3D con hardware dedicado para crear un entorno más sofisticado que funcione en tiempo real. Para empezar se debería cambiar de lenguaje de programación, utilizar otras librerías… Desde el punto de vista de la interfaz, se puede utilizar la aplicación para otros dispositivos con otro tipo de interfaz, así con el hardware y APIs propio, en definitiva modificar el código para que sea más eficiente en el soporte en cuestión.

La interfaz que se implementa tiene como rasgo característico importante que es multiusuario, esto en un entorno en 3D sería bueno que permitiera la presencia de más de un receptor, de esta manera la interacción del audio también sería multiusuario. Para ello se debería disponer de más salidas hardware. La incorporación de un feedback visual sobre la superficie tangible otorgaría un impacto visual más interesante que obviamente el que se da en la pantalla del ordenador.

---

[5] http://www.nvidia.es/

Así mismo la interfaz que se desarrolla es un prototipo, una vez vista la posibilidad de funcionamiento sería necesario realizar estudios de usabilidad y análisis subjetivos de la experiencia de usuario. Estos estudios sería importante realizarlos no solo a nivel hardware si no también a nivel software. De la misma manera hacer test subjetivos de la inmersión sonora que produce el audio 3D. Por tanto, realizando una interfaz con un diseño centrado en el usuario a nivel software y hardware y con el respectivo estudio de usabilidad podría dar muchas más ideas de las aplicaciones que se pueden desarrollar no solo para el entretenimiento si no también para personas con capacidades reducidas.

Referencias bibliográficas

[1] André, C., Embrechts, J.J., Jacques, G.V. Adding. *3D sound to 3D cinema: Identification and evaluation of different reproduction techniques.* Audio Language and Image Processing (ICALIP), 2010 International Conference on Shangai. 2010.

[2] Edgar Choueiri. *Pure Sound*. 3D3A Lab Princeton University.

[3] http://smaf-yamaha.com/what/soundchip_ma7.html

[4] www.telefonica.com/en/descargas/mwc/ImmersiveConference.pdf

[5] OpenAL 1.1 Specification and referente

[6] Final IASIG 3D Audio Rendering and Evaluation Guidelines Level 1

[7] Final IASIG 3D Audio Rendering and Evaluation Guidelines Level 2

[8] 3Dxp: Direct Sound 3.0 Extension API

[9] Chun-geun Kim, Sang Chul Ahn, Ig-Jae Kim, and Hyoung-Gon Kim. *3-Dimensional Voice Communication System for Two User Groups* . Imaging Media Research Center, KIST.

[10] J. Huopaniemi. *Virtual acoustics and 3D sound in multimedia signal processing*. Helsinki University of Technology. 1999.

[11] Durand R. Begault. *3-D Sound for Virtual Reality and Multimedia*. Ames research center. 2000.

[12] Svensson, U.P., Andersson, R. and Vanderkooy, J. *A new interpretation of the Biot-Tolstoy Edge diffraction model.* Journal of the Acoustical Society of America. 1997.

[13] http://blogs.gartner.com/hypecyclebook/

[14] CHI EA '09 Proceedings of the 27th international conference extended abstracts on Human factors in computing Systems.

[15] P. Maes, P. Mistry. *Unveiling the "Sixth Sense," game-changing wearable tech.* TED 2009. Long Beach, CA, USA 2009.

[16] Duncan Graham-Rowe. *Marcar un teléfono con el pensamiento*. MIT Technology Review. 2011.

[17] V. Lévesque, L. Oram, K. MacLean, A. Cockburn, N. D. Marchuk, D. Johnson, J. E. Colgate, M. A. Peshkin. *Enhancing Physicality in Touch Interaction with Programmable Friction.* CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.

[18] K. Greene. *Hablando con la pared.* MIT Technology Review. 2011.

[19] M. Wells. *Musical sweet spot for 3D sound*. BBC News. 2011.

[20]http://alumni.media.mit.edu/~mkc/micArray/node15.html

[21]M. R. Schroeder and B. S. Atal. *Computer simulation of sound transmission in rooms*. IEEE Conv. Record, 7:150-155, 1963.

[22]Duane H. Cooper and Jerald L. Bauck. *Prospects for Transaural Recording*. J. Audio Eng. Soc., 37(1/2):3-19, 1989.

[23]Jeffrey Bodsh. *Extension of the image model to arbitrary polyhedra.* J. Acoust. Soc. Am. 75 (1827-1836). June 1984.

[24] Gardner, W. G. *A realtime multichannel room simulator*. Journal of the Acoustical Society of America, 92, 2395. 1992.

[25] Moorer, J. A. *About This Reverberation Business*. In C. Roads and J. Strawn (Eds.), Foundations of Computer Music. Cambridge, MA: MIT Press. 1985

[26] John Stautner and Miller Puckette. *Designing Multi-Channel Reverberators.* Computer Music Journal, Vol. 6, No. 1 , pp. 52-65. 1982.

[27] Mark Kahrs,Karlheinz Brandenburg. *Applications of digital signal processing to audio and acoustics*. Klumer academoc Publisher. pp. 85- 130. 1998.

Apéndice

A. Clases

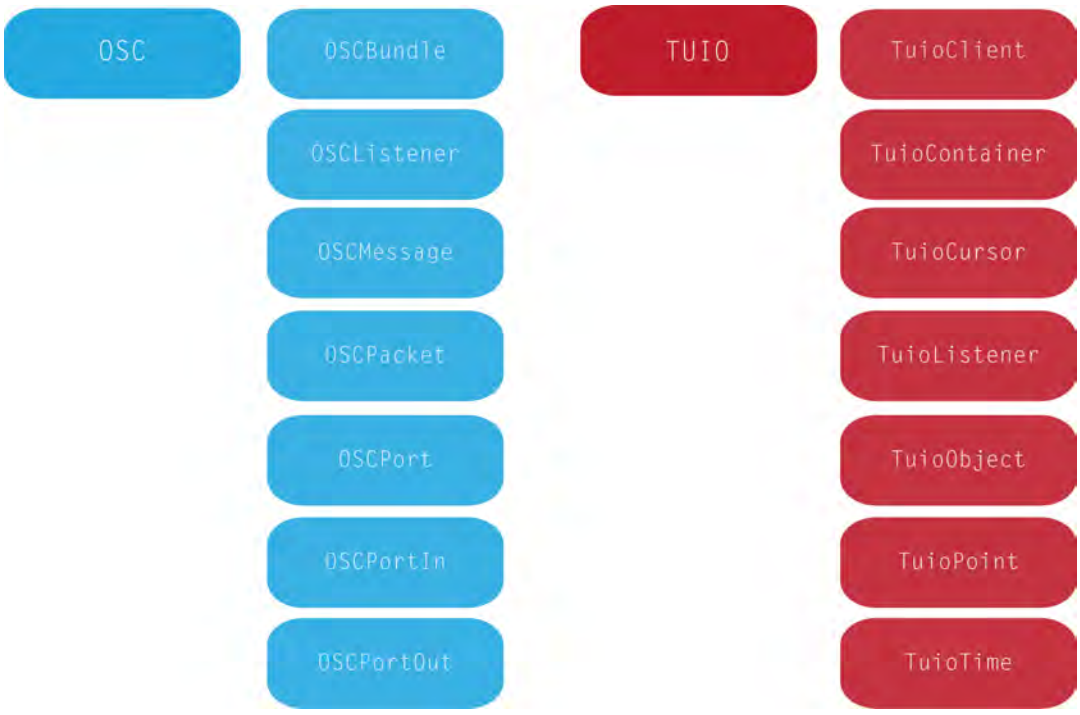


Figura A.1: Clases OSC y TUIO estándares

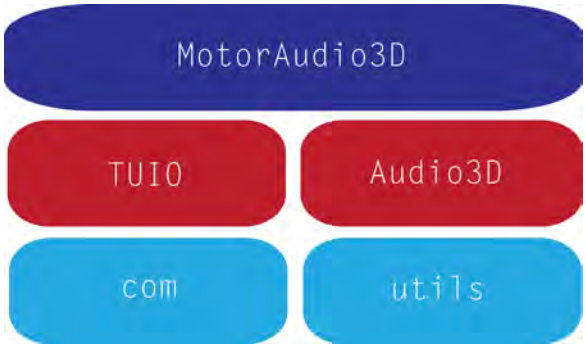


Figura A.2: Clases generales del proyecto audio3D

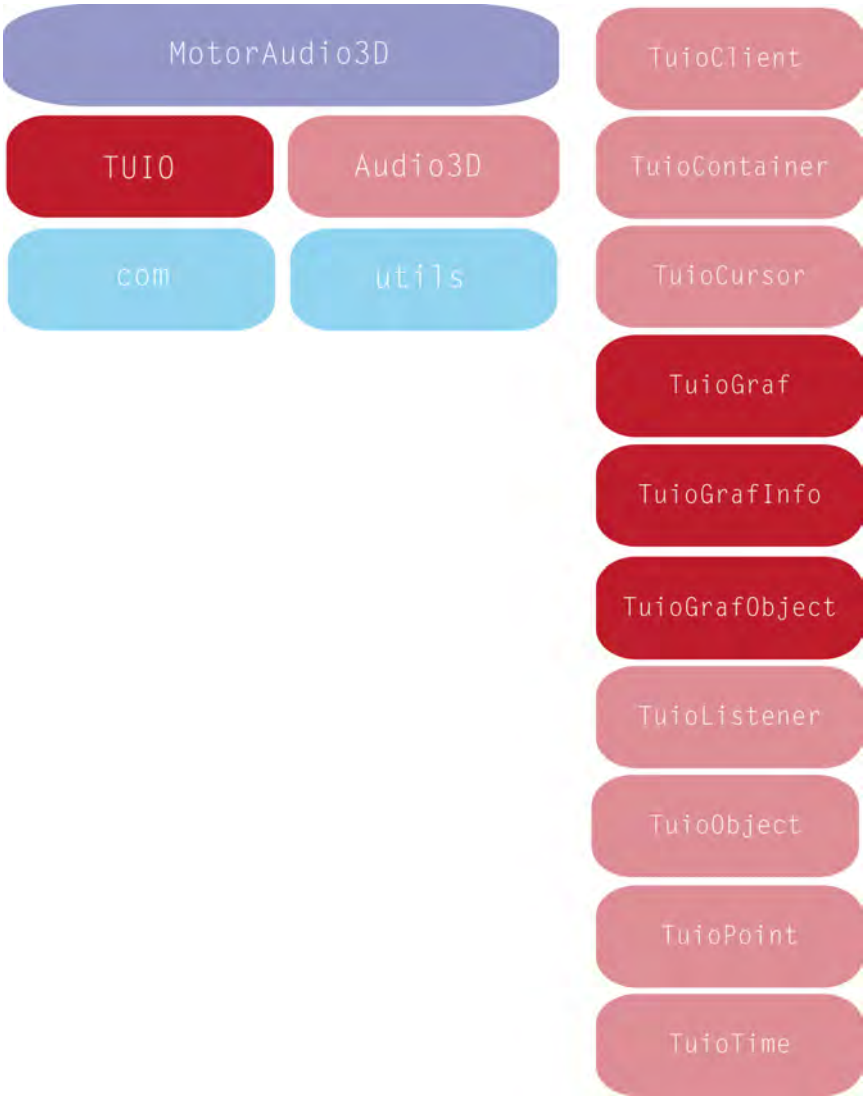


Figura A.3: Nuevas clases TUIO dentro del proyecto audio3D

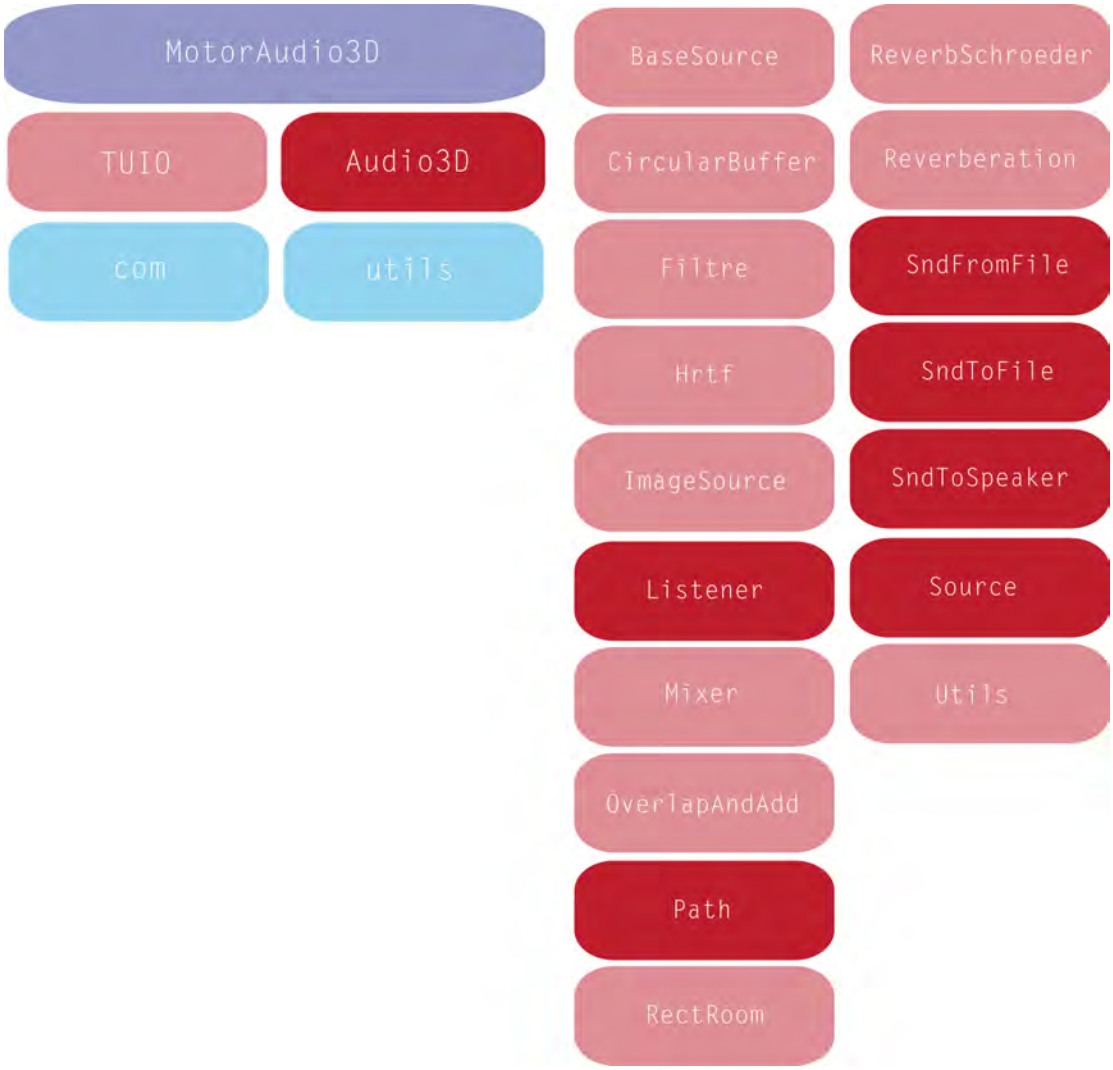


Figura A.4: Nuevas clases o modificaciones dentro de audio3D