UNIVERSIDAD
POLITECNICA
DE VALENCIA

Contribuciones a la implementación de sistemas de Wavefield Synthesis

TESIS DOCTORAL

Autor:
Sergio Bleda Pérez

Director:
José Javier López Monfort

Valencia,
Septiembre de 2009

Resumen

De entre los sistemas de reproducción de sonido 3D, *Wavefield Synthesis* (WFS) presenta una serie de ventajas sobre el resto, principalmente en lo que respecta al gran realismo y sensación de inmersión acústica que proporciona. Otra gran ventaja adicional, es que la zona útil de escucha es muy amplia, superando al resto de sistemas disponibles en la actualidad. La teoría de WFS fue propuesta a finales de los 80 y principios de los 90, no siendo hasta el siglo XXI cuando se han puesto en marcha los primeros prototipos de estos sistemas, aunque muchos aspectos no contemplados en la teoría inicial siguen siendo en la actualidad retos importantes. La presente tesis aborda el estudio de la implementación de los sistemas de WFS aportando soluciones prácticas a las limitaciones tecnológicas que presentan estos sistemas, así como otra serie de problemas de implementación y funcionamiento en tiempo real que, aunque en una primera instancia no se describen como limitaciones físicas, suponen un problema a superar cuando se busca un sistema que funcione eficientemente.

El objetivo final de esta tesis es aportar soluciones que contribuyan al desarrollo de un sistema de WFS totalmente funcional, por lo que durante su desarrollo ha sido necesario encontrar soluciones particulares y originales a multitud de problemas de diferente índole. Esta serie de problemas proviene por un lado de las limitaciones físicas de WFS y por otro de la implementación práctica del sistema. Por otro lado también se ha trabajo en los aspectos computacionales relacionados con la implementación en tiempo real de sistemas de WFS, los cuales necesitan una gran potencia de cálculo para dicho funcionamiento en tiempo real sin cortes ni grandes latencias. Este último se ha tratado de forma rigurosa dedicando un capítulo completo para su análisis y propuesta de soluciones eficientes y

efectivas en coste. Se ha conseguido aportar una solución que emplea un único ordenador personal de propósito general para gestionar un array de grandes dimensiones, por lo que ha sido necesario optimizar al máximo los diferentes aspectos de la arquitectura e ingeniería de programación.

Para finalizar la tesis, se plantea una estrategia para integrar de forma natural un sistema de síntesis WFS con las plataformas de producción musical actuales basadas en software y ordenadores, de forma que permita la creación de proyectos musicales de WFS de forma interactiva y eficiente.

Resum

Pel que fa als diversos sistemes de reproducció de so 3D, és necessari destacar que *WaveField Synthesis* (WFS) presenta una sèrie d'avantatges davant la resta de sistemes, sobretot en quant al gran realisme i sensació d'immersió acústica que proporciona. Un gran avantatge adicional, és l'amplitud de la zona útil d'escolta , que supera a la resta de sistemes disponibles en l'actualitat. En referència a l'origen, la teoria de WFS va ser proposada a finals dels 80 i principis dels 90, no va ser però fins al segle XXI que es van posar en marxa els primers prototips d'aquests sistemes, encara que molts aspectes no contemplats en la teoria inicial constitueixen en la actualitat reptes importants. La present tesi aborda l'estudi de la implementació dels sistemes de WFS, l'aportació de solucions pràctiques a les limitacions tecnològiques que presenten, així com altra sèrie de problemes d'implementació i funcionament en temps real que, encara que en una primera instància no es descriuen com limitacions físiques, sí que cal superar-les quan es tracta de buscar un veritable sistema que funcione de manera eficient.

L'objectiu final d'aquesta tesi és aportar solucions que contribueixen al desenvolupament d'un sistema de WFS totalment funcional, per la qual cosa durant la seua el·laboració ha estat necessari trobar solucions particulars i originals a multitud de qüestions d'una mena diferent. Aquesta sèrie de problemes prové, d'una banda de les limitacions físiques de WFS, i per un altra de la implementació pràctica del sistema. D'altra banda s'ha estat treballant en els aspectes computacionals relacionats amb la implementació en temps real de sistemes de WFS, els quals necessiten una gran potència de càlcul per al funcionament en temps real sense talls ni grans latències. Aquest últim s'ha tractat de forma rigorosa, dedicant un capítol

complet pel seu anàlisi així com una proposta de solucions eficients i efectives en cost. S'ha aconseguit aportar una solució que fa servir un únic ordinador personal de propòsit general per moure un array de grans dimensions, pel que ha estat necessari optimitzar els diferents aspectes tant de l'arquitectura com d'enginyeria de programació.

Com a finalització de la tesi, es planteja una estratègia per integrar de forma natural un sistema de síntesi WFS amb les plataformes de producció musical actuals basades en software i ordinadors, de manera que puga permetre la creació de projectes musicals de WFS de manera interactiva i eficient.

Abstract

Out of all the 3D sound reproduction systems, Wavefield Synthesis (WFS) presents a series of advantages, especially with regard to the realism and acoustic sense of immersion that provides. Moreover, an additional advantage is that the useful listening area is very wide, surpassing the rest of the systems available nowadays. The WFS theory was proposed between the end of the 80's and the beginning of the 90's, but until the XXI century there were no available prototypes of these systems, however, nowadays there are still several aspects not covered by the theory that mean significant challenges.

This thesis deals with the study of the WFS systems implementation providing practical solutions to the technological limitations that present these systems, as well as another set of implementation problems and real-time operation. In the first instance, this limitations are not described as physical constraints but they suppose a problem to overcome in the search for a efficient implementation.

The final aim of this thesis is to provide solutions that contribute to the development of a completely functional WFS system. Thus, during its development it has been necessary to overcome different types of problems proposing original solutions. These problems can be grouped in two major sets, on the one hand, the problems related with the physical limitations of WFS, on the other hand, those related with the practical implementation. Moreover, it has also been worked on computational aspects related to the real-time implementation of the WFS systems, which require a large calculation power for real-time operation without drop-outs and low latency. This last aspect has been treated rigorously devoting a full chapter for its analysis and the proposal of effective and efficient solutions. It has been

achieved a solution based on one and only personal computer even for arrays of great dimensions. For this reason, it has been necessary to optimize to the maximum the different aspects of the software architecture.

To finalize this thesis, it has been proposed a strategy to naturally integrate a WFS system between the current musical production platforms based on computer software, providing the creation of WFS musical projects in an efficient and interactive fashion.

Agradecimientos

En primer lugar, quisiera agradecer a mi director, el Dr. José Javier López Monfort, por todo el esfuerzo que ha invertido en ayudarme a realizar este trabajo y por llegar a ser un buen amigo.

Quisiera agradecer también a Jose y a Basilio, mis antiguos compañeros del ahora extinto Grupo de Audio de la Universidad de Alicante, por todos los buenos ratos que hemos pasado juntos tanto en los buenos como en los malos momentos. Durante todos estos años hemos discutido infinidad de aspectos de esta tesis y de otros trabajos que hemos hecho en común que han resultado muy valiosos.

También quisiera agradecer al personal del *i*TeAM de la Universidad Politécnica de Valencia por acogerme tan bien siempre que me he acercado por allí. En cuanto al desarrollo de esta tesis se refiere, han sido de especial interés los comentarios, charlas y discusiones realizadas con Germán, Laura, Emanuel y Máximo, muchas gracias a todos.

Otro grupo que no puede faltar aquí es el de Kiobus: Ramón, Luis, Angel (sí, sin el acento), Sadoc, Santiago, Jesús y, por supuesto, Jani. Con ellos he pasado los mejores momentos de mi vida antes de conocer a mi mujer, y también han sufrido mis innumerables charlas sobre los sistemas de WFS, aportando muchas veces ideas útiles al respecto.

De regreso a la Universidad de Alicante, quiero agradecer a todos mis compañeros del departamento de Física, Ingeniería de Sistemas y Teoría de la Señal, todo el apoyo que me han brindado desde que llegué al departamento hace ya nueve años.

Como no podría ser de otra forma, tengo que agradecer a mis padres y hermanos su apoyo y cariño incondicional, sin ellos no sería lo que soy. Tampoco debo olvidarme de mis suegros que me han acogido como si fuera

hijo suyo, y como no, de (todos) mis cuñados con los que es difícil aburrirse. También hay que incluir aquí a los recién llegados, mis sobrinos: Ignacio, Julia, Belén y otro que está en camino, solo con pensar en ellos se te alegra el corazón.

Para finalizar, solamente me falta agradecer a mi mujer María Llanos todo su cariño y comprensión y por darle sentido a mi vida, y a mi hijo Sergio por esa sonrisa tan maravillosa que pone cada vez que vuelvo a casa. Si esta tesis tiene un lado oscuro, es que no me ha permitido dedicarles todo el tiempo que se merecen.

Índice general

Índice de figuras

Capítulo 1

Introducción y objetivos

1.1. Introducción

La realidad virtual es un concepto moderno aunque proviene de la antigüedad. Desde tiempos remotos el ser humano ha pensado en la diferencia entre la realidad que percibimos y la realidad existente, y como consecuencia de esto, en la posibilidad de simular una realidad mediante la reproducción de las sensaciones que afectan a los cinco sentidos. Un claro ejemplo de este pensamiento es la denominada Alegoría de la Caverna [Platon, 395 AC], seguramente el texto más antiguo que nos habla acerca de la diferencia entre realidad y realidad virtual.

Con la llegada de las tecnologías electrónicas en el siglo XX, fue posible empezar a pensar en la posibilidad de reproducir sensaciones en el ser humano a partir de estímulos de sus sentidos, de forma que se consiguiera una sensación de inmersión en un entorno artificial lo más realista posible. La vista y el oído siempre han sido los sentidos más explotados, siendo la ingeniería de las Telecomunicaciones la disciplina que ha llevado el peso de la maduración de las mismas desde el descubrimiento de la válvula de vacío

y los semiconductores hasta nuestros días.

La industria de las Telecomunicaciones se ha focalizado en los aspectos de transmisión de la información, es decir, recoger la información, procesarla y almacenarla o transmitirla a través del espacio y del tiempo. Sin embargo, la realidad virtual va más allá, y es claramente una ciencia interdisciplinar que exige conocimientos en muchos campos del saber. Además, esto ocurre para cualquiera de los cinco sentidos que se quieren tratar. Más adelante volveremos a comentar las consecuencias de esta interdisciplinariedad.

Dentro de los cinco sentidos, esta tesis se enmarca dentro de la realidad virtual aplicada al sentido del oído, es decir, en las técnicas que consiguen que una persona perciba en cierto lugar y en cierto instante de tiempo una sensación acústica que se originó en otro lugar y en otro instante de tiempo distintos, esto es, la auralización [Kleiner et al., 1991]. Podemos ir más allá y considerar que esta sensación acústica pudo haber sido percibida por otra persona, o incluso más, que esta sensación nunca ha existido sino que es producto de la síntesis. Sin embargo, al final de la cadena tendremos siempre lo mismo, el sistema auditivo de un oyente, sobre el que hay que reproducir un conjunto de presiones acústicas que constituyen la sensación acústica.

Los sistemas para reproducir esa presión acústica en los oídos del oyente son los que se denominan sistemas de reproducción de sonido espacial. Desde que en 1881 el ingeniero francés Clement Ader realizó la primera transmisión telefónica en estéreo, la evolución experimentada en este campo ha sido lenta pero constante. En 1930 Alan Blumlein patentó la grabación estéreo y el celuloide estéreo. La evolución de los sistemas de sonido espacial ha ido muy de la mano de la industria del cine. El primer largometraje en estéreo del mundo del cine fue Fantasía en 1940, pero ya

en los años 50 se empezaron a realizar las primeras producciones en sonido multicanal. En el mundo de la música también se introdujo el estéreo en 1958, aunque la introducción del sonido multicanal siempre ha ido mucho más lenta por la forma en que las personas tendemos a escuchar la música, pocas veces sentados en un lugar fijo prestando atención a la misma, sino más bien como complemento de otras actividades, cosa que no ocurre en el cine.

La cuadrafonía [Bauer et al., 1973], fue uno de los primeros sistemas de sonido multicanal que se introdujo a nivel de consumidor. Estaba orientado curiosamente al mundo de la música en vez de al cine, y fue un fracaso por la dificultad de almacenar las grabaciones en un soporte de bajo coste [Fellget, 1977]. Sin embargo, de la mano de la industria del cine, se introdujeron los sistemas de sonido envolvente, que constituyen hoy en día un conjunto de estándares de sonido multicanal con la particularidad de estar enfocados a la industria del cine, disponiendo de un canal de diálogos central y otros traseros destinados a los efectos de sonido. La evolución de los mismos ha consistido en ir incrementando progresivamente el número de canales para intentar mejorar la espacialidad del sonido y la zona útil de escucha, como se describe en el Capítulo 2 de esta tesis.

Todos estos sistemas de sonido envolvente fueron concebidos fundamentalmente para mejorar el disfrute de las películas en las salas de cine y para esta tarea tan limitada (efectos especiales, explosiones, crear ambiente) podemos decir que cumplen su función. Aunque su área útil de escucha es muy limitada (tan sólo los oyentes situados en el centro del círculo de altavoces tienen una sensación espacial aceptable) suponen en todo caso un atractivo superior para el espectador que el sistema estéreo convencional. Sin embargo, la verdadera sensación de encontrarse en un entorno acústico

determinado, de percibir los sonidos en las distintas posiciones del espacio y con los matices precisos en una amplia área de escucha y sin limitación de nuestros movimientos no se logra con estos sistemas.

Para reproducir un verdadero ambiente acústico en una amplia zona de escucha, hay que sintetizar el campo sonoro de manera fiel, tal como existiría en el lugar y la situación que queremos reproducir. Este concepto se denomina holografía acústica y, al igual que en imagen, se basa en la reproducción del campo (sonoro o visual) en un volumen, independientemente de la posición o punto de vista del sujeto sobre el que se va a realizar la reproducción. La idea de la holografía acústica se describe por primera vez bajo el concepto de cortina acústica [Snow, 1953] y consistía en transportar la señal de un muro de micrófonos a sus correspondientes altavoces en la sala de destino.

Esta primera idea constituye la base de lo que será posteriormente formalizado a finales de los 80 y principios de los 90 bajo el concepto de *Wavefield Synthesis* (WFS)[1] en [Berkhout, 1988; Berkhout et al., 1993]. WFS es una simplificación del concepto de cortina acústica a un array lineal. Mientras que en el primer caso se requiere un muro de altavoces, en la segunda es suficiente un array lineal en una sola dimensión. Esta simplificación, aunque tiene sus desventajas porque anula la elevación de las fuentes, reduce enormemente el número de altavoces necesario para sintetizar el campo acústico. En el Capítulo 2 de esta tesis, se hará un repaso a los conceptos teóricos que soportan esta tecnología y se describirán otras de sus ventajas y limitaciones.

[1] La traducción de *Wavefield Synthesis* al Español es Síntesis del Campo de Ondas. Por estar la denominación en Inglés tan reconocida, la utilizaremos en todo este texto en vez de su traducción al Español.

1.2. Motivación

A principios de los años 90, WFS era una teoría muy prometedora, pero tecnológicamente resultaba compleja. Para construir un sistema WFS, es necesario construir un array lineal de altavoces separados una distancia pequeña. Esta distancia debe ser más pequeña cuanto mayor sea la frecuencia de trabajo a la que queramos operar el sistema sin producir distorsión en el campo sintetizado (*aliasing* espacial, consultar el Capítulo 2). Por tanto, en un sistema típico para una sala mediana, suelen ser necesarios cientos de altavoces. Además, para cada uno de los altavoces hay que sintetizar una señal diferente y específica para ese altavoz, que debe ser calculada mediante técnicas de procesado digital de señales. Esto implica altos costes de cálculo y altos costes de equipos, ya que para cada canal son necesarios, además del altavoz, un tiempo de cálculo en un DSP (*Digital Signal Processor*) o RISC/CISC (*Reduced/Complex Instruction Set Computer*), un conversor analógico-digital, un amplificador de potencia y un sistema de cableado que conecte todos los elementos hasta llegar al altavoz.

No será hasta principios de siglo XXI, cuando se aborda la construcción de un verdadero sistema WFS completo y plenamente funcional por medio de un proyecto de investigación de la Unión Europea denominado CARROUSO[2] [Sporer et al., 2001]. En este proyecto, además de la Universidad Técnica de Delf, inventora del sistema, participaban institutos de investigación europeos de gran prestigio como el Instituto Fraunhofer IDMT o empresas como France Telecom.

En paralelo a este proyecto, surge en la Universidad Politécnica de Valencia una línea de trabajo en estos sistemas. Inicialmente de la mano

[2] *Creating, assessing and rendering in real-time of high-quality audio-visual environments in MPEG-4 context.*

de un proyecto interno financiado por la Universidad en 2002, "Integración de Sonido 3D junto a Imagen 3D en un sistema de realidad virtual", posteriormente por un proyecto financiado por la Generalitat Valenciana y finalmente por un proyecto del Plan Nacional de I+D, se ha ido desarrollando un sistema completo de WFS con tecnología totalmente propia y con aportaciones originales.

Para poder llegar a un sistema plenamente funcional, era necesario construir todo el software de procesado digital de señal que sea capaz de gestionar la compleja maquinaria de cálculo y manejo de dispositivos que constituyen estos sistemas.

En el año 2002, se decide por el equipo de trabajo, que se quiere construir todo este sistema de procesado fundamentándose en la potencia de los procesadores de los ordenadores personales. Esta decisión, en un principio arriesgada frente a la oferta de utilizar la potencia de los DSP externos, resultó finalmente acertada, porque como hemos visto en estos últimos años la tendencia de crecimiento en la potencia de los procesadores de uso general (Intel/AMD) ha seguido un crecimiento espectacular sin salirse de la ley de Moore, proporcionando cada año mayores recursos de cálculo hasta llegar al proceso paralelo de la mano de los procesadores *multi-core*. Sin embargo, la tendencia en DSP ha sido orientarse al bajo consumo y a los dispositivos portátiles, no siendo en la actualidad una alternativa adecuada.

Fruto de esta acertada decisión, nace la necesidad de construir un eficiente software que funcione específicamente sobre procesadores de ordenadores personales y que sea el motor de procesado mencionado, a la vez que solucione infinidad de aspectos de implementación que no se contemplan en la teoría WFS, pero que suponen un gran reto a la hora de implementar un sistema real de estas características.

La complejidad de este reto es enorme. No sólo hay que abordar aspectos relacionados con la acústica, sino que al mismo tiempo hay que tener conocimientos de procesado de señal y por supuesto fundamentos muy vastos de ingeniería de computadores. Por tanto, este reto es claramente multidisciplinar. Hay que dominar tres disciplinas, que aunque próximas, es difícil encontrar ingenieros o equipos de ingenieros que dominen todas ellas y que estén en disposición de desarrollar un sistema de este tipo.

De este reto, surge esta tesis, que a lo largo de bastantes años y con el apoyo de un equipo de trabajo, ha dado como resultado un gran sistema WFS totalmente funcional, muy eficiente computacionalmente, muy robusto y fiable y a un coste económico tan competitivo que se podría decir que lo sitúa en un producto pre-comercial.

1.3. Objetivos

Después de introducir la tesis y describir la motivación que lleva a su desarrollo dentro de los sistemas WFS, concretamos cuál es su objetivo principal:

Desarrollar un sistema de síntesis WFS plenamente funcional, que opere en tiempo real sobre procesadores de propósito general, que sea escalable, y que además de implementar la ecuación de síntesis WFS, proporcione soluciones a todos los problemas de implementación intrínsecos y extrínsecos que implica la WFS. Para ello, se desarrollará con una arquitectura e interface de usuario que permita la producción fácil de contenidos mediante la integración con las plataformas de producción musical.

De este objetivo principal, podemos extraer los objetivos particulares siguientes:

- Diseño de una arquitectura eficiente.

  Requiere un estudio previo de las necesidades del sistema. Un análisis del funcionamiento del *hardware*, principalmente de las capacidades de las arquitecturas de computadores de propósito general como de las restricciones impuestas por el *hardware* de sonido.

- Implementación eficiente de la ecuación de síntesis WFS.

  Se analizará la ecuación para programarla de la forma más estructurada posible, investigando sus posibilidades de paralelización y la optimización del procesado que conlleva.

- Implementación de auralización.

  Se añadirá auralización a la escena mediante fuentes virtuales de ondas planas, las cuales se distribuirán alrededor de la sala. Este punto es potencialmente peligroso en cuanto al coste computacional requerido, por lo que debe asegurarse una implementación eficiente, que además debe funcionar en tiempo real.

- Optimización de la activación de altavoces en topologías complejas.

  No todos los sistemas WFS emplean un único array lineal de altavoces, en general, se adaptan a la forma de la sala y se convierten en poliedros. Se pretende buscar una técnica que permita determinar de forma sencilla y eficiente los altavoces que deben ser activados para simular una fuentes sonora.

- Búsqueda de una solución para las fuentes próximas al array.

Aunque este punto ha sido resuelto por otros autores, no se ha publicado solución alguna, por lo que es necesario resolverlo por nuestra cuenta para implementar el sistema de WFS.

- Optimización de la síntesis de fuentes focalizadas.

  Las fuentes focalizadas, o fuentes interiores al array, suponen un reto importante ya que WFS es el único sistema de síntesis del campo sonoro que permite recrearlas. El tratamiento que se les da actualmente no permite una localización adecuada, por lo que es necesario mejorarlo.

- Implementación eficiente de la síntesis de fuentes en movimiento.

  Las herramientas matemáticas que permiten describir la WFS no tienen en cuenta la posibilidad del movimiento de las fuentes, por lo que es necesario aportar una solución que permita moverlas de forma sencilla y eficiente. Debe poder moverse las fuentes con la aplicación del efecto *doppler* y sin él.

- Reducción de los efectos producidos por el *aliasing* espacial.

  Estudiar el problema del *aliasing* espacial y proponer un método que permita paliar en la medida de lo posible sus efectos perniciosos en el campo sonoro sintetizado.

- Integración con plataforma de producción musical.

  Proponer una arquitectura que permita integrar la creación de proyectos de WFS en las plataformas actuales de producción multipista.

- Codificación de escenas e intercambio de producciones.

Realizar un estudio de los distintos esquemas de codificación de las escenas que hay disponibles y seleccionar el estándar que mejor se adecúe a WFS.

1.4. Organización de la tesis

El resto de capítulos de esta tesis describen todo el proceso de la misma y se estructuran en los siguientes capítulos:

En el Capítulo 2, *Reproducción espacial de sonido*, se realiza un repaso de la evolución de los sistemas de reproducción de sonido desde el estéreo hasta nuestros días, para lo cual se realizará una clasificación de los sistemas según la técnica que emplean para reproducir el sonido. De entre todos los sistemas se hace especial hincapié en *Wavefield Synthesis*, del cual se describirá en profundidad la teoría en la que se fundamenta, así como sus bondades y limitaciones físicas.

En el Capítulo 3, *Soluciones prácticas a aspectos críticos en la implementación de sistemas WFS*, se pretende, por un lado, aportar soluciones prácticas a las limitaciones físicas de WFS, y, por otro lado, abordar una serie de problemas relacionados directamente con la implementación, los cuales suponen un verdadero problema a superar cuando se construye un sistema de WFS.

En el Capítulo 4, *Procesado de WFS en tiempo real: arquitectura e implementación*, se describe en detalle la arquitectura interna de un software de WFS en tiempo real. En primer lugar, se describen las posibles estrategias de implementación, de entre las cuales se selecciona la implementación en un único ordenador de propósito general. A partir de ahí, se realiza un estudio de las herramientas que proporcionan los procesadores de estos ordenadores y del *hardware* de audio, para, a continuación, describir

cómo puede el software sacar el máximo rendimiento en este escenario. Por último se describe la arquitectura interna del software desarrollado y los prototipos que se han construido.

En el Capítulo 5, *Arquitectura software para la producción de sonido espacial en WFS*, se aborda el último punto de esta tesis doctoral, la integración de los sistemas de WFS en la cadena de producción musical existente en la actualidad. De nuevo se describirán las posibles estrategias de integración, de la cual se seleccionará la híbrida, para posteriormente dar paso a la completa descripción del sistema implementado.

Finalmente en el Capítulo 6, *Conclusiones y líneas abiertas*, se puntualizan todas las conclusiones y aportaciones científicas del presente trabajo así como sus implicaciones prácticas, describiendo también las posibles líneas de investigación que quedan abiertas.

Por último, en los apéndices se describe, en primer lugar, la nomenclatura empleada en el desarrollo de la tesis. En segundo lugar, el protocolo de comunicación que permite manipular al motor de WFS en tiempo real de forma dinámica y, por último, la implementación del software de simulación numérica desarrollado para comprobar diversos aspectos de WFS.

Capítulo 2

Reproducción espacial de sonido

2.1. Introducción

El objetivo de todo sistema de reproducción espacial de sonido es conseguir reconstruir un ambiente sonoro alrededor del oyente, de tal forma que este no sea capaz de distinguir el ambiente ficticio de uno real.

Para conseguir esto, ha sido necesario estudiar el funcionamiento del oído humano como si se tratara de un sistema de tratamiento de señales [Blauert, 1997], lo que ha permitido ir mejorando progresivamente la sensación de inmersión sonora.

A lo largo de los años han ido surgiendo multitud de sistemas de reproducción de sonido, los cuales aseguraban conseguir una inmersión del oyente en el ambiente sonoro cada vez más real. Sin embargo, aun existiendo multitud de sistemas, todos ellos se basan únicamente en tres técnicas diferentes o, a lo sumo, en una combinación de estas [Theile and Wittek,

2004].

Las técnicas en cuestión son:

- Sistemas basados en el efecto *phantom*.

- Reconstrucción binaural de las señales en los oídos.

- Síntesis del campo sonoro alrededor del oyente.

En el presente capítulo repasaremos los fundamentos de estas técnicas a la par que mostramos los diferentes sistemas que las emplean.

2.2. Sistemas basados en el efecto *phantom*

Históricamente los sistemas de reproducción de sonido espacial se han basado únicamente en el efecto *phantom* para intentar simular un ambiente sonoro alrededor del oyente. En esta sección se realizará un repaso de los sistemas más importantes de esta categoría, realizando previamente un pequeño repaso sobre el funcionamiento del efecto *phantom* en el que se basan.

2.2.1. El efecto *phantom*

Según la teoría dúplex, establecida por Lord Rayleigh a principio del siglo XX [Rayleigh, 1907], el cerebro usa las características de diferencia de tiempo interaural (ITD, *Inter-aural Time Difference*) y diferencia de nivel interaural (ILD, *Inter-aural Level Difference* o IID, *Inter-aural Intensity Difference*) para localizar una fuente sonora en el plano horizontal.

Cuando dos o más fuentes sonoras son coherentes el cerebro interpreta que el sonido proviene de una única fuente por lo que únicamente detecta una posición que viene a ser una media de las posiciones de cada fuente

ponderada en función del nivel sonoro de cada una de ellas. A esta «suma de localizaciones» es a lo que se conoce en la literatura como el efecto *phantom* ya que proporciona una posición ficticia desde la que aparentemente proviene el sonido.

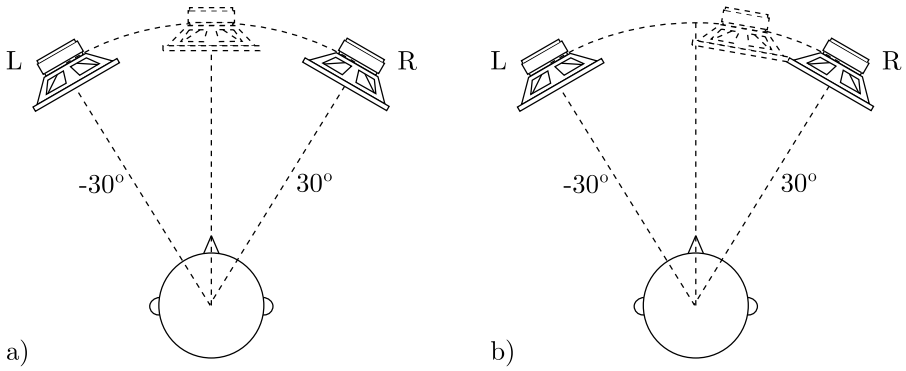


Figura 2.1. Imagen *phantom*: a) Centrada, b) Escorada hacia la derecha.

En la Figura 2.1 se puede ver un ejemplo del funcionamiento del efecto phantom. En el caso a) los dos altavoces mantienen el mismo nivel de señal por lo que el ILD es cero lo que el cerebro interpreta como que el sonido proviene del centro. En el caso b) el altavoz de la derecha tiene un nivel mayor por lo que el ILD de decanta por el lado derecho, lo que indica al cerebro que la fuente debe estar hacia esta dirección.

En la actualidad, la característica más empleada para generar el efecto *phantom* es el ILD [Blauert, 1997]. En la siguiente sección correspondiente al sistema estéreo, se describirán con mayor rigurosidad las leyes que determinan el funcionamiento del efecto *phantom*.

2.2.2. Sistema estéreo

En un sistema estéreo convencional se emplean dos canales, izquierdo y derecho, L y R, lo que permite crear imágenes *phantom* entre las posiciones de los dos altavoces. La separación entre los altavoces no debe ser superior a 60º para poder mantener una imagen estable [Theile, 1990]. La forma de situar la imagen sonora es mediante un *panning* de amplitud, esto es, una distribución del nivel de la señal de forma ponderada entre los dos canales. Dependiendo de la ley que siga el *panning* se conseguirá una posición de la imagen más o menos exacta.

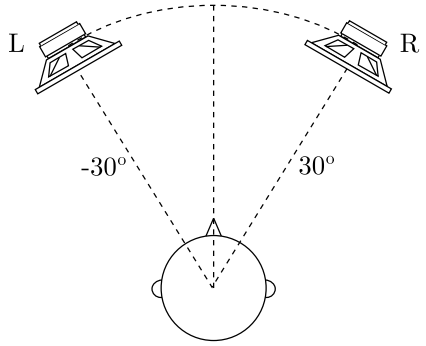


Figura 2.2. Diagrama del sistema estéreo.

La opción más simple es realizar un *panning* lineal siguiendo las ecuaciones $g_L = \alpha$ y $g_R = 1 - \alpha$, siendo α un valor comprendido entre 0 y 1. Sin embargo, esta opción tiene el inconveniente de que al mover la imagen no se mantiene la percepción del volumen de la fuente sonora.

Una segunda aproximación desarrollada en [Bauer, 1961] es la denominada «ley del seno» la cual mantiene el volumen percibido aunque los cálculos se realizan sobre un modelo simple que no incluye la interferencia de la cabeza en la transmisión del sonido, tal y como puede verse en la Figura 2.3 a) el sonido atraviesa la cabeza para llegar al oído contrario.

$$\frac{\operatorname{sen}\theta_p}{\operatorname{sen}\theta_o} = \frac{g_L - g_R}{g_L + g_R} \tag{2.1}$$

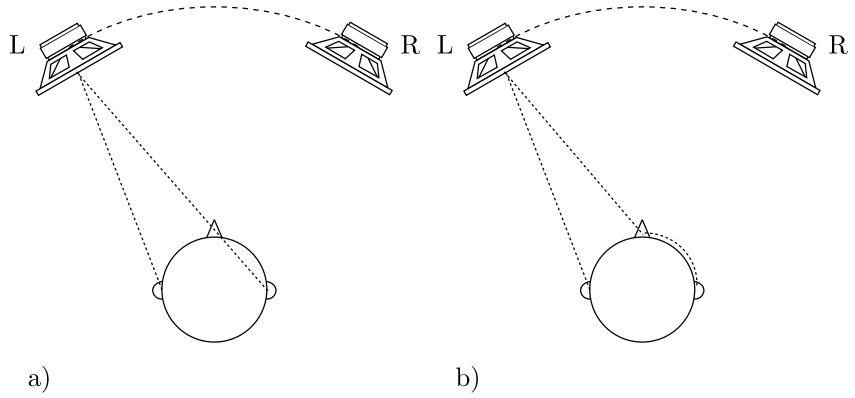


Figura 2.3. Geometría empleada en las leyes de panning. a) Ley del seno, b) Ley de la tangente.

Como mejora a la ley del seno en [Bennett et al., 1985] se formuló la «ley de la tangente» para que incluyera la curvatura de la cabeza en los cálculos (véase la Figura 2.3 b).

$$\frac{\tan\theta_p}{\tan\theta_o} = \frac{g_L - g_R}{g_L + g_R} \tag{2.2}$$

Ambas leyes son idénticas en la localización de fuentes sonoras en los extremos (posiciones de los dos altavoces) o en el punto central, pero la ley de la tangente mejora la localización percibida en posiciones intermedias.

Como puede apreciarse las ecuaciones (2.1) y (2.2) únicamente proporcionan una relación entre las ganancias de las señales de los dos altavoces pero no permiten calcular su valor, para ello es necesario incluir otra ecuación que nos permita mantener constante el volumen aparente de la fuente sonora

$$\sqrt[p]{\sum_{n=1}^{n=N} g_n^p} = 1. \tag{2.3}$$

En esta ecuación, p es un parámetro que se puede elegir dependiendo de la acústica de la sala de escucha, y el efecto que produce su variación es un cambio en el volumen aparente de la fuente sonora. En general este parámetro suele ser 1 en salas anecoicas y 2 en salas con cierta reverberación [Pulkki, 2001a].

Si al par de altavoces estéreo le añadimos más altavoces podremos aumentar el abanico de posibilidades para situar la imagen *phantom* de la fuente sonora, realizando un *panning* entre cada par de altavoces. Por tanto, aunque se empleen más altavoces, para situar una fuente sonora únicamente se usarán dos, aquellos entre los que se encuentre la posición a simular.

2.2.3. Sistemas Surround 5.1, 6.1, 7.1, 10.2 y 22.2

A lo largo de los años se han ido añadiendo progresivamente más canales al par estéreo con el fin de mejorar la sensación espacial de las escenas sonoras aunque no siempre este aumento de canales ha resultado en un acierto pues han habido grandes fracasos en la industria, valga como ejemplo los sistemas de cuadrafonía descritos en [Gerzon, 1970a,b] y cuestionados por el propio autor en [Gerzon, 1974]. Después de algunos intentos fallidos, se llegó a un estándar que representaba un compromiso entre las necesidades de obtención de una sensación espacial óptima y de una implementación práctica que fuera compatible con el sistema estéreo [Rumsey et al., 2001]. Este estándar, denominado 3/2 por la colocación de los altavoces (3 delanteros y 2 traseros), es conocido generalmente por la cifra 5.1 la cual, indica que emplea cinco canales completos más uno opcional cuyo

ancho de banda está reducido a una décima parte (de ahí el «0.1») el cual se emplea para efectos de baja frecuencia (LFE, *Low Frequency Extension*).

En este sistema, al par de altavoces estéreo L y R localizados a $\pm 30^{o}$ al frente, se añade un altavoz central C detrás de la pantalla y dos altavoces traseros LS y LR cuyas posiciones no son tan críticas (se permite un pequeño margen entre 100^{o} y 120^{o}). El canal central viene definido históricamente por el empleo de los sistemas de sonido en las salas cinematográficas, las cuales al emplear pantallas de grandes dimensiones necesitan incluir este canal detrás de la pantalla para estabilizar la posición del sonido en los diálogos.

Idealmente todos los altavoces deben residir en un círculo centrado en la posición del oyente, tal y como se describe en la Figura 2.4.

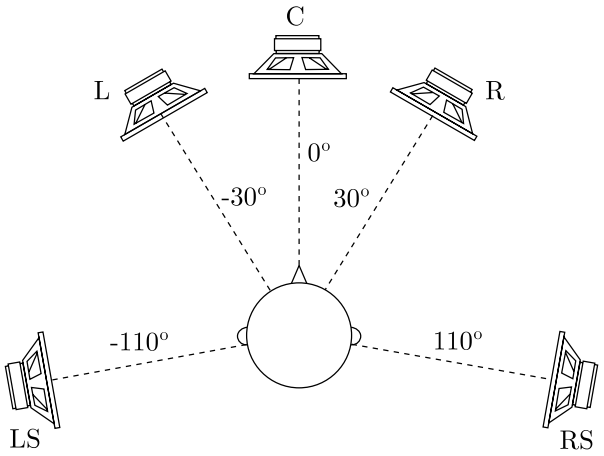


Figura 2.4. Diagrama del sistema 5.1.

En principio, el sistema 5.1 no puede hacer más que un sistema estéreo convencional en cuanto a localización espacial se refiere [Theile, 1990], esto es, únicamente puede crear una imagen sonora estable entre los altavoces

frontales con una extensión angular limitada a $\pm 30^{o}$. Los altavoces traseros se encuentran separados más de 60^{o} lo que imposibilita que sean usados para localizar fuentes sonoras estables. Así, los altavoces traseros se emplean principalmente para radiar reflexiones y sonido difuso como la reverberación. De esta forma, el oyente se siente envuelto por el sonido, y cree estar en otra sala distinta a la que se encuentra. Además, al poder generar reflexiones laterales, la percepción de la distancia mejora notablemente, dando como resultado que la escena sonora frontal aparente mayor profundidad que en el caso estéreo.

En contra de los beneficios mencionados, el sistema 5.1 conlleva las siguientes limitaciones:

- Aunque la simulación de un ambiente mejora notablemente todavía no es una verdadera imagen tridimensional, la cual requeriría un mayor número de altavoces.

- La reproducción se basa en la imagen estereofónica transmitida por los canales L y R. Por tanto, el área de escucha óptima *no es sustancialmente mayor* que en el caso estéreo.

- En muchos casos no es posible colocar los altavoces en las posiciones adecuadas y la extensión del formato a un mayor número de altavoces agrava aún más este problema.

Sistemas 6.1 y 7.1

Al sistema 5.1 se le han ido añadiendo progresivamente canales para mejorar la sensación de inmersión y la localización de las fuentes sonoras. Estas mejoras se conocen en el mercado como los sistemas 6.1 y 7.1, al añadir uno y dos canales respectivamente. Las únicas mejoras que intro-

ducen estos sistemas en una mayor resolución en la situación de fuentes sonoras traseras, pero seguimos sin tener fuentes estables fuera del arco definido por los canales L y R.

La distribución de los altavoces en estos sistemas se puede ver en la Figura 2.5. Los ángulos empleados para estos sistemas pueden variar ligeramente, el ejemplo mostrado en la figura se ha obtenido de la página web de Dolby. Nótese que en el sistema 7.1 sí es posible localizar una fuente sonora estable entre los canales LS y RLS por un lado y RS y RRS por otro, al haber una separación inferior a 60º entre ellos, aunque esta posibilidad no suele emplearse y siguen dedicándose a recrear efectos de ambiente.

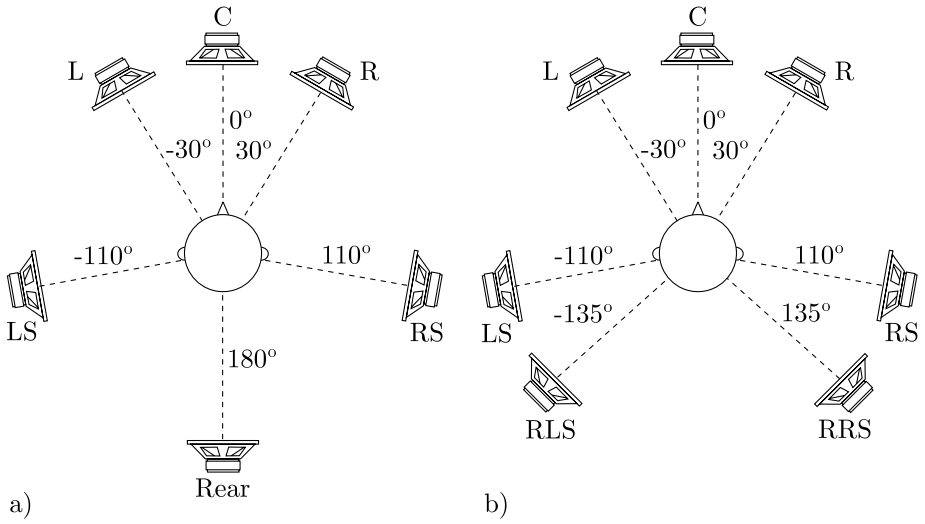


Figura 2.5. Diagrama de los sistemas a) 6.1 y b) 7.1.

Sistema 10.2

Como una evolución a todas estas configuraciones ha surgido recientemente el sistema 10.2, el cual trata de mejorar la percepción sonora

empleando tanto consideraciones acústicas como psicoacústicas [DellaSala, 2006]. Este sistema duplica el conjunto de altavoces del 5.1, lo que le da pie a emplear como lema "*Twice as good as 5.1*".

Los canales añadidos son LW, RW, BS, LH, RH más un canal LFE adicional y su distribución está centrada principalmente en la parte delantera de la escena (véase la Figura 2.6), al contrario que en los sistemas 6.1 y 7.1 en los que únicamente se añaden canales en la parte trasera.

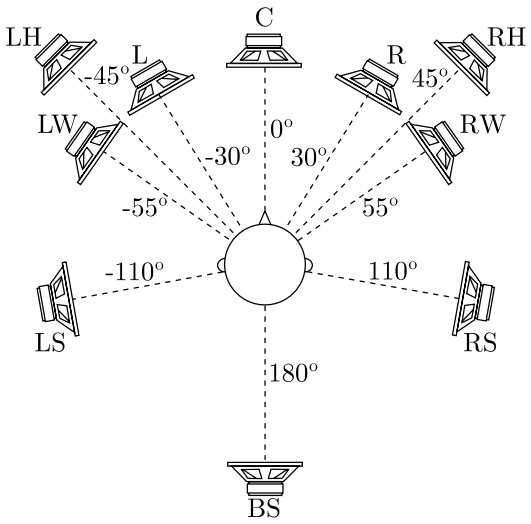


Figura 2.6. Diagrama del sistema 10.2.

Los canales LW y RW (*Left* y *Right Wide* respectivamente) están situados a $\pm 55^{o}$ y, además de permitir una mayor resolución en la colocación de fuentes sonoras delanteras, se emplean para introducir reflexiones laterales, lo que incrementa notablemente la sensación espacial [DellaSala, 2006]. Al igual que el sistema 6.1, se añade un canal trasero BS (*Back Surround*) a 180^{o} el cual se emplea para rellenar el hueco entre los canales LS y RS mejorando la localización en la parte trasera. Los otros dos canales LH y

RH (*Left* y *Right Height* respectivamente) están situados a ±45° y tienen la peculiaridad de que están situados en altura (también a 45°), lo que permite ofrecer efectos de elevación aunque únicamente en la parte delantera de la escena.

Por último, se añade otro canal LFE independiente del primero y se define la posición que deben emplear ambos *subwoofers*, al contrario que en el sistema 5.1 en el cual no se establece su posición. Estos canales deben situarse a ±90°, es decir a la izquierda y a la derecha de la escena.

Sistema 22.2

De entre los sistemas estereofónicos se encuentra como máximo exponente el sistema 22.2 desarrollado por Kimio Hamasaki como complemento para el sistema de televisión de ultra alta definición *Super Hi-Vision* de la televisión pública japonesa [NHK, 2002].

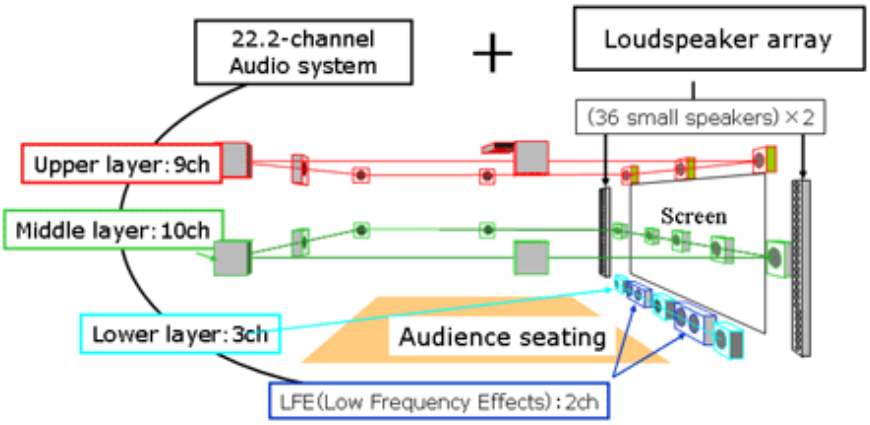


Figura 2.7. Diagrama del sistema 22.2, extraído de [NHK, 2002].

En este sistema se distribuye un conjunto de 22+2 altavoces en tres capas situadas a distintas alturas (véase la Figura 2.7), lo que permite simular la elevación de las fuentes sonoras. En la capa inferior únicamente se emplean tres altavoces delanteros y ambos *subwoofers*, todos ellos situados debajo de la pantalla. En la capa intermedia, situada a la altura del oído, se encuentran diez altavoces: cinco delanteros, tres traseros y dos laterales y por último en la capa superior se sitúan otros nueve canales: tres delanteros, tres traseros, dos laterales y uno en el centro del techo.

2.2.4. *Vector base amplitude panning* (VBAP)

El método VBAP fue desarrollado por V. Pulkki en [Pulkki, 2001a] y es un método de reconstrucción de fuentes virtuales que emplea únicamente el *panning* en amplitud, igual que el sistema estéreo, de hecho, en el caso de escenas en dos dimensiones el VBAP no es más que una reformulación del sistema estéreo [Pulkki and Karjalainen, 2001]. La principal novedad que aporta el VBAP con respecto al estéreo es que extiende el *panning* en amplitud a escenas en tres dimensiones [Pulkki, 2001b].

El funcionamiento del VBAP es bastante sencillo y efectivo, si nos centramos en el caso de tres dimensiones debemos rodear el escenario con una superficie ficticia la cual deberemos triangular y colocar un altavoz en cada vértice. La síntesis de una imagen *phantom* se consigue en dos fases, en primer lugar es necesario determinar en qué triángulo se encuentra situada la fuente para posteriormente calcular la ganancia que deberá aplicarse a cada uno de los tres altavoces del triángulo. La forma de determinar la ganancia de cada altavoz se consigue expresando la posición de la imagen *phantom* como una combinación lineal de las posiciones de los tres altavoces, esto es, las posiciones de los altavoces forman una base.

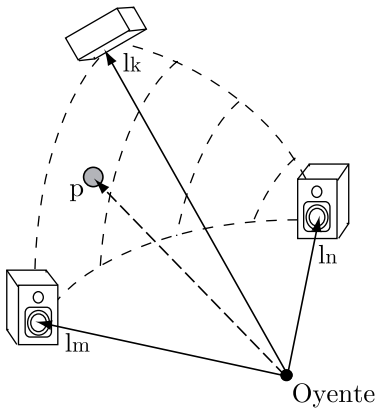


Figura 2.8. Descripción geométrica del VBAP.

En la Figura 2.8 se expresa la geometría del problema. Si tomamos como origen de coordenadas la posición del oyente y definimos l_m, l_n y l_k como las posiciones de los altavoces, la posición de la imagen *phantom* p se puede expresar como:

$$p = g_m l_m + g_n l_n + g_k l_k \tag{2.4}$$

donde g_m, g_n y g_k son las ganancias que se deben aplicar a cada altavoz. Expresando todos los términos de forma vectorial, las ganancias se pueden calcular resolviendo el sistema:

$$g = p^T L^{-1}. \tag{2.5}$$

En [Pulkki, 2001a] se demuestra que la matriz L es invertible siempre que el oyente (el origen de coordenadas) no se encuentre dentro del plano que contiene al triángulo en cuyos vertices están los altavoces.

Por último, falta indicar que antes de poder emplear los factores de ganancia es necesario normalizarlos, para lo cual se suele emplear como normalización $|g| = 1$ [Pulkki, 2001b].

2.3. Reconstrucción binaural

Los sistemas de reconstrucción binaural tratan el problema de la reproducción de sonido desde un punto de vista distinto. Aunque la escena sonora sea muy compleja el oyente únicamente va a captar de ella aquello que llega a sus oídos. Por tanto, parece lógico diseñar un sistema que reconstruya exacta y únicamente eso: la señal que llega a los oídos del oyente, esto es, un sistema capaz de proporcionar de forma exacta los valores ITD e ILD.

La ITD y la ILD son complementarias, pues mientras la ITD funciona bien por debajo de 1,5 kHz y la ILD lo hace por encima. La elección de este umbral de frecuencia se debe a que su longitud de onda se corresponde con el diámetro promedio de una cabeza humana, aunque este no es un valor muy preciso pues según determinados estudios más que un valor debería ser un rango de frecuencias comprendido entre los 1,5 y los 3 kHz [Blauert, 1997].

El uso de los valores ITD e ILD por si solos no nos permite localizar una fuente sonora en elevación, dado que dicha elevación no puede ser captada por un tercer oído fuera del plan horizontal. Fue necesario esperar hasta los años 60 [Batteau, 1967] para que se definiera la «Respuesta al Impulso Relativa a la Cabeza» (*Head Related Impulse Response*, HRIR), o su versión en frecuencia la «Función de Transferencia Relativa a la Cabeza» (*Head Related Transfer Function*, HRTF). La HRTF comprende todos los aspectos físicos de la localización del sonido para una posición concreta de la cabeza (diferencia de tiempos interaural, sombra de la cabeza, difracciones en los pliegues de la oreja, canal auditivo, ...). Una vez se conocen las HRTF para ambos oídos y la posición de una fuente virtual en el espacio, se pueden sintetizar las señales binaurales correspondientes a los dos oídos

para un sonido emitido por dicho fuente.

Para generar sonido binaurales se pueden seguir dos métodos principales e igualmente válidos:

- Registrar la escena sonora mediante micrófonos/maniquíes especiales.

- Sintetizar la escena a partir de la HRTF.

El primero de los métodos es el más sencilla y con la que primero se experimentó [Damaske and Wagener, 1969; Kürer et al., 1969] el cual consiste en registrar la señal de una escena sonora colocando dos micrófonos en los oídos de un oyente mientras se reproduce la escena, aunque también es posible realizar la grabación sustituyendo a la persona por un maniquí acústico que simula el torso y cabeza humanos (véase la Figura 2.9). Al realizar la grabación de esta forma, la señal registrada tiene como punto de referencia la cabeza del oyente, por lo que la reproducción permitirá oír lo que este estaba escuchando. Como se puede apreciar, las necesidades de almacenamiento de un sistema binaural son las mismas que las de un sistema estéreo, sin embargo, el realismo de la escena es muy superior [Wilkens, 1972].

Por otro lado, las grabaciones binaurales también pueden generarse de forma artificial al filtrar la señal de una fuente sonora grabada en condiciones anecoicas con las HRIR para la dirección de llegada del sonido [Møller, 1992]. Muchos sistemas emplean datos medidos experimentalmente, por ejemplo las medidas realizadas en [Gardner and Martin, 1994], de esta forma es posible obtener las señales para cada uno de los oídos al filtrar con las HRIR medidas. En general, este proceso se realiza empleando la convolución en frecuencia, filtrando la señal con la HRTF ya que el filtrado es menos costoso de realizar que en el tiempo.

Figura 2.9. Ejemplo de maniquí acústico.

Al tomar como punto de referencia al oyente obtenemos grandes ventajas pero también varios inconvenientes. El primero de ellos es que la señal debe ser reproducida mediante auriculares para que llegue tal cual se registró a los oídos. El uso de auriculares produce sobre el oyente la sensación de que el sonido proviene de dentro de la cabeza, efecto que se conoce en la literatura como «*Inside the head effect*», lo que resta realismo a estos sistemas [Begault, 1994]. En segundo lugar, si el oyente mueve la cabeza durante la reproducción está cambiando el punto de referencia y por tanto está moviendo toda la escena con él/ella, esto último, aparte de ser antinatural, crea confusiones en la localización delante/detrás [Begault, 1994]. Por último, también nos encontramos con el problema de que cada individuo tiene sus propias HRTF, por lo que, para que la reproducción sea lo más exacta posible es necesario calcular y almacenar las HRTF de cada oyente que desee emplear el sistema. El uso de las HRTF de un oyente distinto produce una degradación de la localización, además de crear confusiones delante-detrás y cierta reducción de la externalización de los sonidos [Begault, 1992].

Para que la escena no cambie con el movimiento de la cabeza del oyente es necesario realizar un seguimiento (*tracking*) de este movimiento para compensarlo cambiando y/o interpolando las HRTF dinámicamente [Algazi et al., 2004]. Esto implica que un sistema binaural debe realizar un procesado intensivo y de baja latencia durante la reproducción.

2.3.1. Sistemas transaurales

Los sistemas transaurales tienen como objetivo reproducir señales binaurales utilizando altavoces convencionales en vez de auriculares. Con esto se consigue una reproducción más cómoda para el oyente que utilizando auriculares, al mismo tiempo que se elimina por completo el efecto de interiorización.

La primera propuesta para presentar una grabación binaural utilizando altavoces fue sugerida en [Bauer, 1961], aunque no se formaliza una solución concreta hasta que en [Atal and Schroeder, 1966] se propone un cancelador de *cross-talk*.

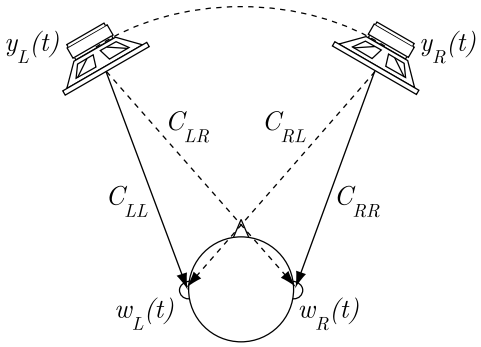


Figura 2.10. Cruce de caminos al emitir con altavoces.

Al reproducir la señal con altavoces en vez de auriculares se producen ciertos efectos indeseados. Tomando como referencia la Figura 2.10, se

puede ver que la señal emitida por el altavoz izquierdo, la cual debería ser recibida únicamente por el oído izquierdo, alcanza ambos oídos, sucediendo lo mismo para el altavoz derecho. Este efecto es lo que se conoce como «cruce de caminos» (o *cross-talk*), el cual desvirtúa en gran medida el efecto conseguido con la grabación binaural, perdiéndose los aspectos relativos a la localización contenidos en la grabación [Atal and Schroeder, 1966]. En esta figura, los términos $C_{xy}(z)$ que aparecen representan los canales acústicos por los que viajan las señales desde cada altavoz a cada oído.

Aparte del *cross-talk*, si la reproducción se realiza fuera de una sala anecoica, las reflexiones que se producen en la sala también alcanzan a los oídos, de tal forma que la señal recibida incluye reflexiones que no pertenecen a la sala original por lo que falsean el entorno acústico que se pretende conseguir [López, 1999]. Así, se entiende que los términos $C_{xy}(z)$ descritos en la Figura 2.10 son las HRIR medidas entre cada par altavoz-oído.

La forma de minimizar todos estos efectos es a través de un cancelador de *cross-talk*, cuyo diagrama de bloques se representa en la Figura 2.11 [López et al., 1999].

La señal que debe emitir cada altavoz se puede determinar invirtiendo el sistema de ecuaciones (2.6), el cual representa el cancelador de *crosstalk*, donde $y_L(t)$ y $y_R(t)$ son las señales que emitirán los altavoces, i. e. las que se pretende determinar, $w_L(t)$ y $w_R(t)$ son las señales que alcanzan los oídos del oyente, y por tanto se desea conseguir que sean idénticas a $x_L(t)$ y $x_R(t)$, las señales binaurales originales. Además, es necesario incluir los canales acústicos $C_{xy}(z)$ ya que deben compensarse.

$$\begin{bmatrix} w_L(t) \\ w_R(t) \end{bmatrix} = \begin{bmatrix} C_{LL}(z) & C_{LR}(z) \\ C_{RL}(z) & C_{RR}(z) \end{bmatrix} \begin{bmatrix} y_L(t) \\ y_R(t) \end{bmatrix} \tag{2.6}$$

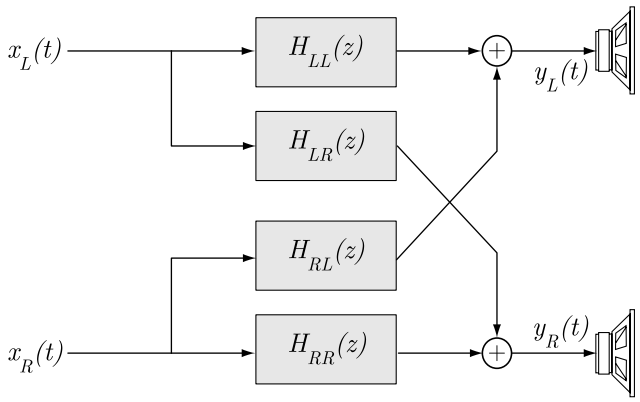


Figura 2.11. Diagrama de bloques del procesador de *cross-talk*.

Al igual que sucede con los sistemas binaurales, si el oyente mueve la cabeza la escena se mueve con él/ella, por lo que sigue siendo necesario realizar un seguimiento del movimiento para actualizar las HRTF dinámicamente.

2.4. Síntesis del campo sonoro

En contraposición a todos los sistemas vistos hasta el momento, en la síntesis del campo sonoro se pretende recomponer el campo sonoro por completo en toda la sala de escucha, no únicamente en la posición en la que se encuentra el oyente. Por tanto, la principal ventaja de estos sistemas es que consiguen erradicar el concepto de *sweet spot*, lo que permite que el oyente pueda situarse libremente en la sala e incluso moverse por ella sin que ello modifique la calidad de la escena sonora. Además, al no depender los cálculos ni de la posición ni de la orientación de la cabeza del oyente, tampoco es necesario realizar un seguimiento de su cabeza como sucedía en los sistemas binaurales.

En la presente sección repasaremos brevemente las dos aproximaciones actuales a la síntesis del campo sonoro: Ambisonics y *Wavefield Synthesis*, para, en el capítulo siguiente, profundizar en la segunda técnica en la cual se centra esta tesis.

2.4.1. Ambisonics

Ambisonics es una técnica de reproducción de sonido espacial [Gerzon, 1973, 1975] la cual se fundamenta en las técnicas de grabación microfónica para reconstruir el campo sonoro. Para registrar una escena sonora tridimensional emplea cuatro micrófonos, uno omnidireccional y tres bidireccionales, el primero registra la presión (W) y los otros tres el gradiente de presión en cada eje (X, Y, Z). Si almacenamos estos cuatro valores cada uno en una pista obtenemos el formato ambisonics de primer orden comúnmente denominado Ambisonics-B o B-*format*. Al contrario que los sistemas basados en el estéreo, la reproducción no es una simple asignación canal-altavoz. En este caso el equipo reproductor debe conocer la posición de los altavoces y determinar que debe reproducir cada uno de ellos para que, en el punto donde se encuentra el oyente, la presión y los tres gradientes de velocidad coincidan con los registrados, esto es, el equipo reproductor necesita decodificar la señal codificada con ambisonics al reproducir la escena (véase la Figura 2.12).

La mayor ventaja de ambisonics frente a los sistemas estéreo es la flexibilidad que ofrece al poder definir la cantidad y posiciones de los altavoces al reproducir, sin embargo, estas posiciones deben formar una figura regular para que los cálculos puedan ser llevados a cabo de forma sencilla, pues el empleo de una figura irregular puede hacer que los cálculos sean extremadamente complicados o incluso imposibles de realizar [Daniel et al.,

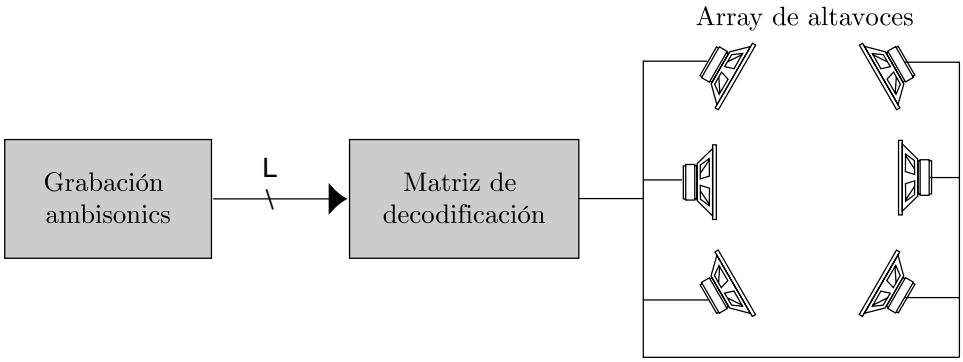


Figura 2.12. Decodificación de una escena sonora ambisonics.

1998].

También es posible emplear ambisonics para registrar una escena sonora virtual y, así, sintetizar una escena sin necesidad de crear físicamente el montaje para almacenarlo con el conjunto de micrófonos. En este caso, lo que debemos hacer es simular el funcionamiento de los micrófonos empleando sus diagramas de directividad. Si definimos los ángulos de azimut (θ) y elevación (ϕ) de la fuente virtual a simular (siempre desde el punto de vista del oyente), la codificación con el formato B se obtendría empleando el conjunto de ecuaciones (2.7), así una grabación en este formato tiene cuatro pistas. Ya que la codificación ambisonics es lineal [Daniel, 2001], la creación de una escena sonora más compleja con múltiples fuentes sonoras simultáneas se puede crear por combinación lineal.

$$\begin{aligned} W(t) &= s(t) \\ X(t) &= s(t)\cos\theta\cos\phi \\ Y(t) &= s(t)\,\mathrm{sen}\,\theta\cos\phi \\ Z(t) &= s(t)\,\mathrm{sen}\,\theta \end{aligned} \tag{2.7}$$

Tal y como acabamos de describir el funcionamiento de ambisonics, el

sistema solamente es capaz de reproducir fielmente el campo sonoro en un punto concreto, la posición del oyente, es decir, al igual que los sistemas anteriores sólo funciona en el *sweet spot*. Para ampliar el área de escucha útil fue necesario desarrollar un estudio más profundo de su funcionamiento lo que dio lugar al denominado «Ambisonics de Orden Superior» (*Higher Order Ambisonics*, HOA).

HOA recrea el campo sonoro empleando la descomposición en armónicos esféricos de la presión sonora [Daniel, 2001]. Sirva como ejemplo el conjunto de ecuaciones 2.7 del Ambisonics-B, en él se puede ver la presión sonora W y la descomposición de ésta en sus tres armónicos esféricos de primer orden X, Y y Z. Aumentando el orden, es decir, aumentando el número de armónicos esféricos empleados, conseguiremos una reproducción más fiel y a la vez aumentaremos el área de escucha progresivamente [Daniel et al., 2003]. La relación entre el orden (m) y el número de armónicos se obtiene a partir de la ecuación $(m+1)^2$ [Daniel, 2001], por tanto, si para orden 1 tenemos cuatro pistas, para orden 3 tendremos dieciséis.

En la Tabla 2.1 se muestran las expresiones de los armónicos esféricos hasta el orden tercero, la tabla en cuestión se ha extraído íntegramente de [Daniel, 2001]. El cálculo de una pista codificada en ambisonics se obtiene multiplicando la señal de presión por la función $Y_n(\theta, \phi)$ correspondiente de la tabla.

Como resumen cabe destacar que ambisonics reconstruye fielmente el campo sonoro en un único punto del espacio y conforme aumentamos el orden y, por tanto la complejidad, ese punto va creciendo hasta en el límite ocupar toda la sala de escucha. Así se puede deducir que ambisonics en el límite realiza la misma tarea que hará *Wavefield Synthesis*, es decir, ambisonics es un caso particular de *Wavefield Synthesis* [Nicol and Emerit,

Orden	Canal	$Y_n(\theta, \phi)$
0	W	1
1	X	$\cos(\theta)\cos(\phi)$
	Y	$\operatorname{sen}(\theta)\cos(\phi)$
	Z	$\cos(\phi)$
2	R	$\sqrt{3}/2\cos(2\theta)\cos^2(\phi)$
	S	$\sqrt{3}/2\operatorname{sen}(2\theta)\cos^2(\phi)$
	T	$\sqrt{3}/2\cos(2\theta)\operatorname{sen}^2(\phi)$
	U	$\sqrt{3}/2\operatorname{sen}(2\theta)\operatorname{sen}^2(\phi)$
	V	$(3\operatorname{sen}^2(\phi)-1)/2$
3	K	$\sqrt{5/8}\cos(3\theta)\cos^3(\phi)$
	L	$\sqrt{5/8}\operatorname{sen}(3\theta)\cos^3(\phi)$
	M	$\sqrt{15}/2\cos(2\theta)\operatorname{sen}(\phi)\cos^2(\phi)$
	N	$\sqrt{15}/2\operatorname{sen}(2\theta)\operatorname{sen}(\phi)\cos^2(\phi)$
	O	$\sqrt{3/8}\cos(\theta)\cos(\phi)(5\operatorname{sen}^2(\phi)-1)$
	P	$\sqrt{3/8}\operatorname{sen}(\theta)\cos(\phi)(5\operatorname{sen}^2(\phi)-1)$
	Q	$\operatorname{sen}(\phi)(5\operatorname{sen}^2(\phi)-3)/2$

Tabla 2.1. Expresiones ambisonics de la expansión en armónicos esféricos hasta el orden 3.

1998]. Sin embargo, el mayor inconveniente de ambisonics es su necesidad de un array de altavoces regular (circular en 2D) [Daniel et al., 2003].

2.4.2. *Wavefield Synthesis*

Wavefield Synthesis (WFS) es una técnica de reconstrucción del campo sonoro que se fundamenta en el principio de Huygens la cual, con determinadas limitaciones, consigue reconstruir el campo sonoro en toda la superficie de la sala de escucha. Así, uno puede situarse en cualquier posición de la sala e incluso moverse por ella sin que eso altere la imagen sonora reconstruida. Por tanto, WFS consigue erradicar el concepto de *sweet spot*, además, al no tomar como punto de referencia para los cálculos la posición y orientación del oyente tampoco sufre los problemas de los sistemas binaurales, es decir, no necesita realizar un seguimiento de la posición de la cabeza del oyente. Ya que es en este sistema en el que se centra esta tesis, dedicaremos siguiente sección para describirlo en profundidad.

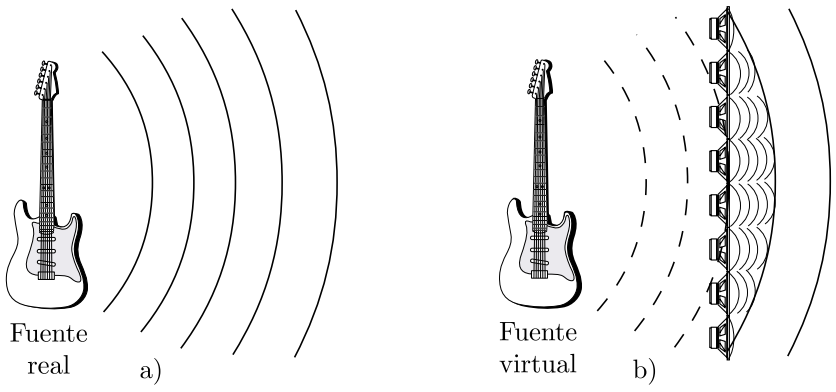


Figura 2.13. Aplicación del principio de Huygens, a) campo sonoro emitido por una fuente real, b) campo sonoro sintetizado mediante WFS.

2.5. *Wavefield Synthesis*

Wavefield Synthesis es una técnica de reproducción sonora análoga a la holografía en la óptica, y es la técnica de reconstrucción del campo sonoro en la que se fundamenta esta tesis. En el presente capítulo se describirán los principios en los que se fundamenta con el desarrollo matemático necesario así como las limitaciones que se encuentran al llevar esta técnica a la práctica.

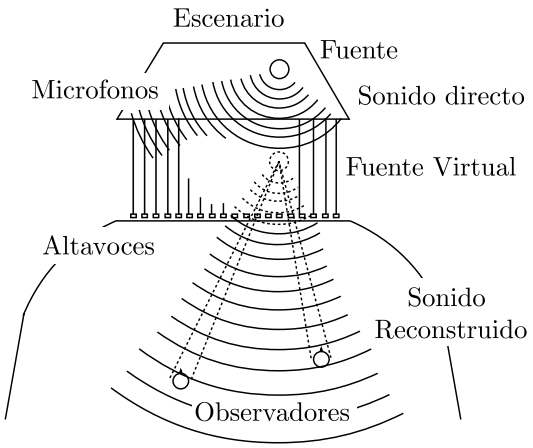


Figura 2.14. Cortina acústica.

El concepto que más intuitivamente describe el funcionamiento de WFS es el de la «cortina acústica» [Snow, 1953], véase la Figura 2.14. El funcionamiento de la cortina acústica es el siguiente: si delante de un escenario colocamos una cortina a modo de telón sobre la cual hubieran infinidad de micrófonos, seríamos capaz de registrar el sonido que viaja desde el escenario hasta el patio de butacas. Si ahora colocamos otra cortina sobre la cual hay dispuestos tantos altavoces como micrófonos teníamos en la cortina anterior, podríamos enviar el sonido registrado por cada micrófono a su altavoz correspondiente con lo cual al otro lado de la segunda cortina

estaríamos reconstruyendo la escena sonora original. Debido a las restricciones tecnológicas de la época, Snow redujo el infinito número de pares de micrófonos-altavoces a únicamente tres limitando en gran medida el efecto producido.

Este concepto intuitivo fue descrito formalmente por Berkhout mediante la teoría de ondas a finales de la década de los 80 [Berkhout, 1987, 1988], comenzando a emplearse el término *Wavefield Synthesis* a principios de la década de los 90 [Berkhout et al., 1992, 1993].

2.5.1. Fundamentos teóricos clásicos

En esta sección se describirán los fundamentos teóricos clásicos de WFS, que son los que se ha fundamentado el desarrollo de esta tesis. Recientemente, en [Spors et al., 2008] se ha revisado todo el desarrollo teórico extendiéndolo para el caso tridimensional, pero también generalizando el bidimensional, esto último se verá más adelante en la Sección 2.5.2.

WFS se fundamenta en el principio de Huygens [Huygens, 1678]. Este principio establece que el frente de ondas radiado por una fuente sonora se comporta a su vez como una conjunto infinito de fuentes puntuales distribuidas a lo largo del frente de ondas, a las cuales se las denomina fuentes secundarias. Por tanto, es posible describir la radiación del campo sonoro mediante este conjunto de fuentes secundarias. La forma directa de recrearlas es mediante altavoces, los cuales se deben situar en la misma posición de las fuentes. Ya que la colocación de los altavoces formando la curva del frente de ondas, la cual además varía al moverse la fuente, no es práctica, se los coloca formando una línea recta y se compensa su desplazamiento mediante la aplicación de una ganancia y retardo, tal y como se muestra en el apartado b) de la Figura 2.13.

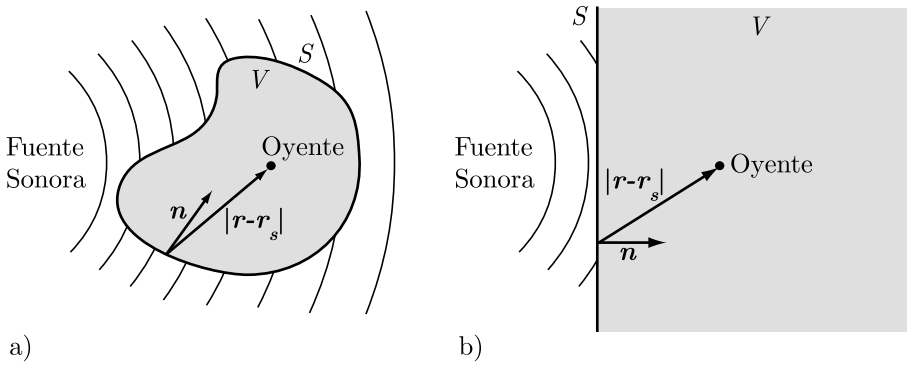


Figura 2.15. Parámetros de la integral de Kirchhoff-Helmholtz: a) Superficie arbitraria, b) Superficie degenerada.

A partir de la teoría de la acústica lineal se conoce que el campo acústico encerrado en un volumen V, el cual se ha generado únicamente por fuentes sonoras externas a la superficie S que delimita al volumen V, se puede calcular reemplazando las fuentes sonoras externas por una distribución de fuentes de tipo monopolo sobre la superficie S (véase el caso a) de la Figura 2.15). Por tanto, es posible recrear artificialmente el campo sonoro en el interior de V mediante un conjunto de fuentes sonoras situadas sobre la superficie S. Esta última afirmación se puede cuantificar mediante la integral de Kirchhoff-Helmholtz [Boone, 2001]:

$$P(\boldsymbol{r}, w) = \frac{1}{4\pi} \oint_S \left[P(\boldsymbol{r}_s, w) \frac{\partial}{\partial n} \left(\frac{e^{-jk|\boldsymbol{r}-\boldsymbol{r}_s|}}{|\boldsymbol{r}-\boldsymbol{r}_s|} \right) - \frac{\partial P(\boldsymbol{r}_s, w)}{\partial n} \frac{e^{-jk|\boldsymbol{r}-\boldsymbol{r}_s|}}{|\boldsymbol{r}-\boldsymbol{r}_s|} \right] dS, \tag{2.8}$$

en la cual $P(\boldsymbol{r}, w)$ es la transformada de Fourier de la presión sonora $p(\boldsymbol{r}, t)$, $\boldsymbol{r}$ es el vector de coordenadas de un punto de observación en el espacio, $\boldsymbol{r}_s$ es el vector de coordenadas de los operandos de la integral sobre la superficie S, k es el número de onda y $\boldsymbol{n}$ es el vector normal a la superficie S. La primera parte de la integral representa una distribución de

dipolos cuya amplitud es proporcional a la presión en la superficie S. La segunda parte representa una distribución de monopolos cuya amplitud es proporcional a la componente normal de la velocidad de las partículas, la cual es a su vez proporcional a la variación de la presión ($\partial P/\partial n$).

En la práctica, la integral (2.8) permite sintetizar el campo sonoro en el interior del volumen V, libre de fuentes sonoras, si se conoce la presión del campo sonoro en la superficie S, en concreto, este hecho es el que emplea WFS para reconstruir el campo sonoro.

Si la superficie S degenera en un plano que separa el área de reproducción del área en la que se encuentran las fuentes sonoras (véase la Figura 2.15 b), la ecuación (2.8) se puede reducir a dos ecuaciones distintas, las denominadas Integrales de Rayleigh I [Boone and Verheijen, 1993] y II [Berkhout et al., 1993]:

$$P(\boldsymbol{r}, w) = \rho c \frac{jk}{2\pi} \int \int_S \left[V_n(\boldsymbol{r}_s, w) \frac{e^{-jk|\boldsymbol{r}-\boldsymbol{r}_s|}}{|\boldsymbol{r}-\boldsymbol{r}_s|} \right] dxdy \tag{2.9}$$

$$P(\boldsymbol{r}, w) = \frac{jk}{2\pi} \int \int_S \left[P(\boldsymbol{r}_s, w) \frac{1+jk|\boldsymbol{r}-\boldsymbol{r}_s|}{jk|\boldsymbol{r}-\boldsymbol{r}_s|} \cos\phi \frac{e^{-jk|\boldsymbol{r}-\boldsymbol{r}_s|}}{|\boldsymbol{r}-\boldsymbol{r}_s|} \right] dxdy \tag{2.10}$$

Discretizando ambas ecuaciones, se obtienen las siguientes expresiones:

$$P(\boldsymbol{r}, w) = \rho c \frac{jk}{2\pi} \sum_{n=1}^{N} V_n(\boldsymbol{r}_n, w) \frac{e^{-jk|\boldsymbol{r}-\boldsymbol{r}_n|}}{|\boldsymbol{r}-\boldsymbol{r}_n|} \Delta x \Delta y \tag{2.11}$$

$$P(\boldsymbol{r}, w) = \frac{jk}{2\pi} \sum_{n=1}^{N} P_n(\boldsymbol{r}_n, w) \frac{1+jk|\boldsymbol{r}-\boldsymbol{r}_n|}{jk|\boldsymbol{r}-\boldsymbol{r}_n|} \cos\phi_n \frac{e^{-jk|\boldsymbol{r}-\boldsymbol{r}_n|}}{|\boldsymbol{r}-\boldsymbol{r}_n|} \Delta x \Delta y \tag{2.12}$$

La integral de Rayleigh I (2.11) indica que la presión en un punto del interior del volumen se puede obtener como la combinación lineal de

diversas fuentes sonoras actuando como monopolos, empleando para ello únicamente la velocidad de las partículas en el plano S, mientras que la integral de Rayleigh II (2.12) permite su cálculo empleando altavoces actuando como dipolos empleando para ello únicamente la presión en el plano S.

Aplicando la aproximación de campo lejano $k|\boldsymbol{r} - \boldsymbol{r_n}| \gg 1$, es posible simplificar la ecuación (2.12), siempre y cuando el punto de escucha esté lo suficientemente alejado del plano de reproducción, quedando así reducida la ecuación a:

$$P(\boldsymbol{r}, w) = \frac{jk}{2\pi} \sum_{n=1}^{N} P_n(\boldsymbol{r}_n, w) \cos\phi_n \frac{e^{-jk|\boldsymbol{r}-\boldsymbol{r}_n|}}{|\boldsymbol{r} - \boldsymbol{r}_n|} \Delta x \Delta y \tag{2.13}$$

Por motivos prácticos, el siguiente paso a realizar es reducir el plano de reproducción a una única línea de fuentes secundarias, las repercusiones que conlleva esta simplificación se tratarán en la Sección 2.5.4, con lo que la ecuación queda de la siguiente forma:

$$P(\boldsymbol{r}, w) = \frac{jk}{2\pi} \sum_{n=1}^{N} P_n(\boldsymbol{r}_n, w) \cos\phi_n \frac{e^{-jk|\boldsymbol{r}-\boldsymbol{r}_n|}}{|\boldsymbol{r} - \boldsymbol{r}_n|} \Delta x \tag{2.14}$$

Derivación de las funciones *driving*

En la sección anterior se ha demostrado como es posible sintetizar el campo sonoro en el interior de un volumen a partir del conocimiento de la presión en una superficie que encierra a ese volumen. En este punto, lo único que nos falta para sintetizar un campo sonoro es, cómo determinar la señal que debe alimentar a cada altavoz. En la literatura, a estas señales se las conoce por *driving functions*. Para poder realizar su cálculo, es necesario primero introducir la Figura 2.16 en la cual se representa la

geometría del problema. El desarrollo aquí mostrado está basado en el descrito en [de Vries, 1996], pero, por simplicidad, se ha tomado como origen de coordenadas la posición de la fuente virtual.

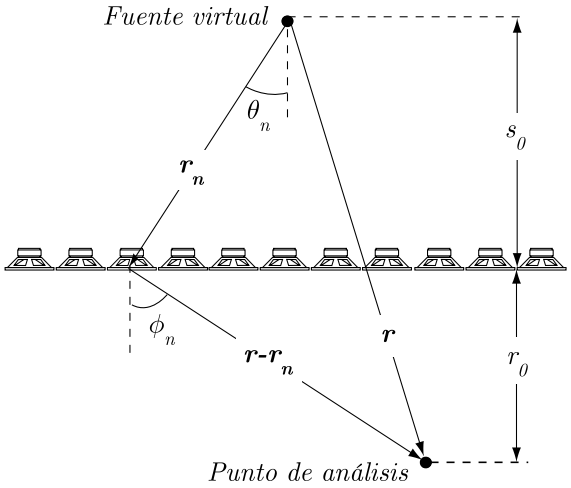


Figura 2.16. Descripción geométrica de WFS.

En primer lugar, es posible determinar el campo sonoro que generaría una fuente virtual (situada en el origen de coordenadas) en un punto de cualquiera de análisis $\boldsymbol{r}$ si tenemos en cuenta que al ser puntual produce un campo sonoro esférico, por lo que

$$P(\boldsymbol{r}, w) = S(w)\frac{e^{-jkr}}{r}, \tag{2.15}$$

donde $S(w)$ es el espectro de la señal emitida por la fuente virtual y la fracción que la acompaña no es más que el retardo y la atenuación producidos por la distancia desde la fuente hasta ese punto.

Tal y como se deduce de la integral de Rayleigh II (2.13), las funciones *driving* que alimentarán a los altavoces son proporcionales a la presión del campo sonoro en la superficie S, por tanto

$$Q(\boldsymbol{r_n}, w) = A_n P(\boldsymbol{r_n}, w), \tag{2.16}$$

donde A_n es la función que realiza la proporción descrita en la integral de Rayleigh II, la cual depende de la posición del altavoz y de la frecuencia. Si ahora combinamos las ecuaciones (2.15) y (2.16) tenemos que:

$$Q(\boldsymbol{r_n}, w) = A_n S(w) \frac{e^{-jkr_n}}{r_n}, \tag{2.17}$$

Por otro lado, el campo sonoro sintetizado en un determinado punto de análisis se puede obtener a partir de la integral de Rayleigh:

$$P(\boldsymbol{r}, w) = \sum_{n=1}^{N} \left[Q(\boldsymbol{r}_n, w) G(\phi_n, w) \frac{e^{-jk|\boldsymbol{r}-\boldsymbol{r}_n|}}{|\boldsymbol{r}-\boldsymbol{r}_n|} \right] \Delta x, \tag{2.18}$$

donde N es el número de altavoces empleado, $Q(\boldsymbol{r_n}, w)$ es la señal de excitación del enésimo altavoz (*driving signal*), ϕ_n es el ángulo entre el eje principal del enésimo altavoz y la línea que lo une con el punto de análisis, $G(\phi_n, w)$ es el índice de directividad del altavoz, $|\boldsymbol{r} - \boldsymbol{r}_n|$ es la distancia entre el altavoz y el punto de análisis y Δx es la separación entre altavoces. Repasando la ecuación, se puede comprobar que el único término desconocido son las funciones *driving*.

Si ahora se sustituye la ecuación (2.17) en (2.18), y se aplica una aproximación de fase estacionaria [Bleistein, 1984], la cual implica que los altavoces más cercanos a la fuente serán aquellos que aportarán más información al campo sonoro a sintetizar, es posible obtener el siguiente resultado:

$$Q(\boldsymbol{r_n}, w) = S(w) \frac{cos(\theta_n)}{G(\theta_n, w)} \sqrt{\frac{jk}{2\pi}} \sqrt{\frac{r_0}{s_0 + r_0}} \frac{e^{-jkr_n}}{r_n} \tag{2.19}$$

Si en la ecuación (2.19) agrupamos todos los términos de amplitud junto con el filtro de +3 dB/oct ($\sqrt{jk}$) y tenemos en cuenta que la directividad

de los altavoces no se compensa, por lo que $G(\theta_n, w) = 1$, obtenemos una ecuación más manejable que es la que se empleará en el resto de la tesis para referirnos a las funciones *driving*:

$$Q(\boldsymbol{r_n}, w) = S(w)A(\boldsymbol{r}_n, w)e^{-jkr_n}. \tag{2.20}$$

Al estudiar la ecuación (2.20) se puede apreciar que las señales que se envían a los altavoces se obtienen modificando la amplitud de la señal original y aplicándole un retardo.

2.5.2. Fundamentos teóricos actualizados

Recientemente, en [Spors et al., 2008] se ha realizado una revisión de los desarrollos matemáticos en los que se fundamenta WFS extendiendo las ecuaciones para el caso tridimensional y generalizando el caso bidimensional. En la presente sección se hará un breve resumen de estos desarrollos, aunque, debe tenerse en cuenta que todo el trabajo realizado en esta tesis se ha basado en el desarrollo clásico.

Antes de comenzar con los desarrollos es conveniente indicar que, por claridad del texto, todas las afirmaciones contenidas en las siguientes secciones que no se referencian explícitamente se han extraído de [Spors et al., 2008].

La integral de Kirchhoff-Helmholtz

Un sistema de altavoces que rodea a un oyente puede tratarse como una condición de contorno no homogénea para la ecuación de onda. La solución de la ecuación de onda homogénea para una región V con respecto a condiciones de contorno no homogéneas viene dada por la integral de Kirchhoff-Helmholtz [Williams, 1999]

$$P(\boldsymbol{x}, w) = -\oint_{\partial V} \left(G(\boldsymbol{x}|\boldsymbol{x}_0, w) \frac{\partial}{\partial \boldsymbol{n}} P(\boldsymbol{x}_0, w) - P(\boldsymbol{x}_0, w) \frac{\partial}{\partial \boldsymbol{n}} G(\boldsymbol{x}|\boldsymbol{x}_0, w) \right) dS_0, \tag{2.21}$$

donde $P(\boldsymbol{x}, w)$ representa el campo de presiones dentro del volumen V encerrado por el contorno ∂V, $G(\boldsymbol{x}|\boldsymbol{x}_0, w)$ es una función de Green adecuada, $P(\boldsymbol{x}_0, w)$ es la presión en el contorno ∂V y $\boldsymbol{n}$ es la normal a la superficie ∂V que apunta al interior de la superficie. La abreviatura $\frac{\partial}{\partial \boldsymbol{n}}$ representa el gradiente direccional en la dirección del vector normal $\boldsymbol{n}$. El campo de presiones en el exterior de V es nulo y V no contiene ninguna fuente sonora en su interior. La figura 2.17 muestra la geometría descrita.

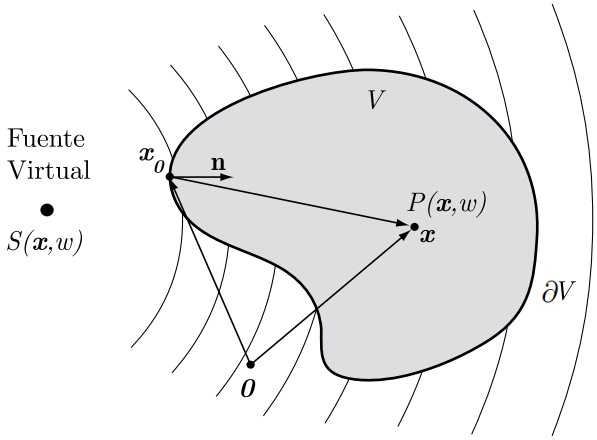


Figura 2.17. Geometría empleada para la integral de Kirchhoff-Helmholtz.

La función de Green $G(\boldsymbol{x}|\boldsymbol{x}_0, w)$ representa la solución de la ecuación de onda no homogénea para una excitación de tipo pulso de Dirac espacio-temporal en la posición $\boldsymbol{x}_0$ y, debe cumplir las condiciones de contorno homogéneas impuestas en ∂V. Para la reproducción de sonido, se suele asumir la propagación en campo libre dentro del volumen V, esto implica que en V no hay ningún objeto que pueda obstruir el paso del sonido y que

∂V tampoco restringe la propagación. En estas condiciones la función de Green genérica pasa a ser la función de Green de campo libre $G_0(\boldsymbol{x}|\boldsymbol{x}_0, w)$, la cual se puede interpretar como la función de transferencia espacio-temporal de un monopolo situado en el punto x_0 y su gradiente direccional como la función de transferencia espacio-temporal de un dipolo situado en el mismo punto, cuyo eje principal apunta en la dirección de $\boldsymbol{n}$.

La Ecuación (2.21) implica que el campo de presiones $P(\boldsymbol{x}, w)$ dentro de V se puede determinar a partir de la presión y su gradiente direccional en la superficie ∂V. Si la función de Green se implementa mediante una distribución continua de monopolos y dipolos situados sobre la superficie ∂V y son alimentados convenientemente, el campo contenido en V se determina al completo por estas fuentes. Como se verá a continuación, este último principio se puede emplear para la reproducción de sonido. En este contexto, la distribución de fuentes (monopolos y dipolos) se denomina fuentes secundarias, de acuerdo con el principio de Huygens.

Para conseguir un sistema de reproducción de sonido, se debe conseguir que el campo sonoro $S(\boldsymbol{x}, w)$ emitido por una fuente virtual se reproduzca de la forma más exacta posible en el interior del volumen V (la sala de audición). Esto se puede conseguir si la distribución de fuentes secundarias se alimenta con la presión (monopolos) y el gradiente direccional (dipolos) de la fuente $S(\boldsymbol{x}, w)$. Así, se puede afirmar que si en la Ecuación (2.21) $P(\boldsymbol{x}_0, w)$ se corresponde con los valores de la fuente $S(\boldsymbol{x}, w)$ en ∂V, el campo $P(\boldsymbol{x}, w)$ en el interior de V será igual al producido por la fuente virtual $S(\boldsymbol{x}, w)$.

Para la implementación práctica de un sistema de reproducción de sonido, es conveniente desechar uno de los dos tipos de fuentes secundarias que emplea la integral de Kirchhoff-Helmholtz. En general, se suelen

desechar las fuentes secundarias de tipo dipolar, ya que los monopolos se pueden asimilar razonablemente por altavoces montados en un *baffle*. Para derivar una función que solamente contenga monopolos se pueden emplear dos técnicas diferentes [Williams, 1999]:

- Aproximación mediante una fuente simple.

- Modificación de la función de Green.

La aproximación mediante una fuente simple da como resultado la ecuación en la que se basa HOA, mientras que la modificación de la función de Green es la base de WFS. En la siguiente sección se verá esta última.

Modificación de la función de Green

Es posible desechar el segundo término de la integral de Kirchhoff-Helmholtz (2.21), el cual representa las fuentes secundarias de tipo dipolo, realizando una modificación de la función de Green que emplea la integral [Williams, 1999]. Para eliminar los dipolos, la función de Green modificada $G_N(\boldsymbol{x}|\boldsymbol{x}_0, w)$ debe cumplir la siguiente condición

$$\frac{\partial}{\partial \boldsymbol{n}} G_N(\boldsymbol{x}|\boldsymbol{x}_0, w)\Big|_{\boldsymbol{x}_0 \in \partial V} = 0. \tag{2.22}$$

La Ecuación (2.22) impone una condición de contorno homogénea de Neumann sobre ∂V, por ello se suele denominar a la función de Green modificada como la función de Green Neumann. Como consecuencia de esta condición, la superficie ∂V se modela como una superficie acústicamente rígida para las fuentes secundarias. La función de Green Neumann deseada $G_N(\boldsymbol{x}|\boldsymbol{x}_0, w)$ se puede derivar añadiendo una solución homogénea adecuada (con respecto de la región V) a la función de Green de campo libre $G_0(\boldsymbol{x}|\boldsymbol{x}_0, w)$, aunque la forma explícita de la función de Green Neumann

depende de la geometría del contorno ∂V. La solución para figuras cerradas solamente se puede encontrar en geometrías relativamente simples como esferas o contornos planos. Otra restricción que debe imponerse es que toda derivación de las funciones de Green Neumann debe realizarse con fuentes secundarias realizables físicamente, ya que, dependiendo de la forma explícita de las funciones, las fuentes secundarias requeridas pueden llegar a ser difícilmente realizables.

En el contexto de WFS las soluciones empleadas se han basado principalmente en contornos lineales. Una función de Green Neumann adecuada para un contorno ∂V lineal o planar viene dada por [Williams, 1999]

$$G_N(\boldsymbol{x}|\boldsymbol{x}_0, w) = G_0(\boldsymbol{x}|\boldsymbol{x}_0, w) + G_0(\boldsymbol{x}_m(\boldsymbol{x})|\boldsymbol{x}_0, w). \tag{2.23}$$

Una solución que cumple la Condición (2.22) se puede obtener eligiendo un punto receptor $\boldsymbol{x}_m(\boldsymbol{x})$ como el punto $\boldsymbol{x}$ reflejado en la superficie plana ∂V en la posición $\boldsymbol{x}_0$. Debido a que se ha forzado una superficie plana $|\boldsymbol{x} - \boldsymbol{x}_0| = |\boldsymbol{x} - \boldsymbol{x}_m|$ y por tanto

$$G_N(\boldsymbol{x}|\boldsymbol{x}_0, w) = 2G_0(\boldsymbol{x}|\boldsymbol{x}_0, w). \tag{2.24}$$

Así, para este caso, $G_N(\boldsymbol{x}|\boldsymbol{x}_0, w)$ es equivalente a una fuente puntual de nivel doble. Al introducir la función de Green Neumann sobre la integral de Kirchhoff-Helmholtz para una geometría plana, se obtiene la primera integral de Rayleigh, la cual es la base de la teoría clásica de WFS [Berkhout, 1988].

Extensión para superficies arbitrarias

En el desarrollo siguiente, se asumirá que la Ecuación (2.23) es una aproximación válida para geometrías distintas del plano. En este caso, el

punto receptor $\boldsymbol{x}_m$ se elige como el punto $\boldsymbol{x}$ reflejado en la tangente de la superficie ∂V en la posición $\boldsymbol{x}_0$. La eliminación de las fuentes secundarias dipolares en el caso de un contorno arbitrario tiene dos consecuencias directas:

- El campo sonoro fuera de la superficie ∂V no es nulo.
- El campo sonoro reproducido en el interior de V no es exacto.

La primera consecuencia implica que la superficie ∂V debe ser convexa, para que las contribuciones del campo exterior no se propaguen de nuevo al interior. La segunda es una consecuencia de la aproximación realizada al asumir que la Ecuación (2.23) se puede emplear con geometrías arbitrarias. De entre los problemas que conlleva esta aproximación, el más prominente es la aparición de reflexiones en el interior de V, al haber impuesto que la superficie ∂V sea acústicamente rígida. Sin embargo, estos inconvenientes pueden reducirse modificando la función *driving* que alimentará a las fuentes secundarias.

La mayor parte de estas reflexiones involuntarias proviene de las fuentes secundarias en las que la dirección de propagación local del campo sonoro virtual no coincide con el vector normal $\boldsymbol{n}$ de la fuente secundaria. Por tanto, es posible reducir drásticamente las reflexiones silenciando estas fuentes secundarias. Siguiendo este concepto, el campo sonoro reproducido queda de la siguiente forma

$$P(\boldsymbol{x}, w) = -\oint_{\partial V} 2a(\boldsymbol{x}_0)\frac{\partial}{\partial \boldsymbol{n}} S(\boldsymbol{x}_0, w) G_0(\boldsymbol{x}|\boldsymbol{x}_0, w) dS_0, \tag{2.25}$$

donde $a(\boldsymbol{x}_0)$ representa una ventana que silencia aquellas fuentes que emiten la mayor parte de las reflexiones. Esta ventana se definió en [Spors, 2007a,b], y se tratará detalladamente en la Sección 3.3.3.

La función de Green de la ecuación (2.25) caracteriza el campo de las fuentes secundarias, determinando el resto de términos su fuerza. Esta fuerza a la que se denominará en adelante como la función *driving* de las fuentes secundarias, es decir, la señal que alimentará a los altavoces, viene determinada por

$$D(\boldsymbol{x}_0, w) = 2a(\boldsymbol{x}_0)\frac{\partial}{\partial \boldsymbol{n}}S(\boldsymbol{x}_0, w). \tag{2.26}$$

Sustituyendo (2.26) en (2.25) obtenemos:

$$P(\boldsymbol{x}, w) = -\oint_{\partial V} D(\boldsymbol{x}_0, w)G_0(\boldsymbol{x}|\boldsymbol{x}_0, w)dS_0, \tag{2.27}$$

por lo que es posible concluir que el campo de presiones en el interior de V se puede determinar (de forma aproximada) mediante las funciones *driving* de las fuentes secundarias y las funciones de Green de los monopolos en ∂V.

Para poder continuar con el desarrollo es necesario introducir los campos sonoros que producen los modelos típicos de fuentes virtuales. En WFS se emplean tradicionalmente dos modelos de fuentes: fuentes puntuales y de ondas planas. El campo sonoro producido por una fuente puntual es:

$$S_{SW}(\boldsymbol{x}, w) = \hat{S}_{SW}(w)\frac{e^{-jk|\boldsymbol{x}-\boldsymbol{x}_s|}}{|\boldsymbol{x}-\boldsymbol{x}_s|}, \tag{2.28}$$

donde $\boldsymbol{x}_s$ representa la posición de la fuente puntual, el centro del campo sonoro, y $\hat{S}_{SW}(w)$ es el espectro en la dirección radial.

El campo sonoro producido por una fuente de ondas planas es:

$$S_{PW}(\boldsymbol{x}, w) = \hat{S}_{PW}(w)e^{-jk\boldsymbol{n}_{PW}^T\boldsymbol{x}}. \tag{2.29}$$

Elección de la función de Green

Llegados a este punto en el desarrollo, para poder reconstruir el campo sonoro dentro del volumen V faltan por indicar dos puntos, por un lado la expresión de la función de Green que se va a emplear y por otro la forma de la superficie ∂V. A continuación se mostrarán las distintas funciones de Green que pueden emplearse, dependiendo de si el sistema de reproducción va a permitir sintetizar un campo sonoro tridimensional o solamente bidimensional.

La función de Green de campo libre en tres dimensiones viene dada por

$$G_{3D}(\boldsymbol{x}|\boldsymbol{x}_0, w) = \frac{1}{4\pi}\frac{e^{jk|\boldsymbol{x}-\boldsymbol{x}_0|}}{|\boldsymbol{x}-\boldsymbol{x}_0|}, \tag{2.30}$$

y puede interpretar como el campo generado por una fuente puntual monopolo situada en $\boldsymbol{x}_0$.

La función de Green de campo libre para dos dimensiones viene dada por [Williams, 1999]

$$G_{2D}(\boldsymbol{x}|\boldsymbol{x}_0, w) = \frac{j}{4}H_0^{(2)}(k|\boldsymbol{x}-\boldsymbol{x}_0|), \tag{2.31}$$

donde $H_0^{(2)}(\cdot)$ representa la función de Hankel de orden cero y segunda especie [Abramowitz and Stegun, 1972]. En este contexto, la Ecuación (2.31) representa el campo generado por una fuente lineal, esto es, una fuente que genera un campo cilíndrico. Esta fuente se encuentra situada en paralelo al eje z e intersecta con la reproducción en $\boldsymbol{x}_0$.

Aunque este tipo de fuentes sonoras es el adecuado para sintetizar el campo sonoro en 2D, son fuentes difícilmente realizables en la práctica por lo que no se suelen emplear. En su lugar se utiliza la aproximación de la

Ecuación (2.30) a dos dimensiones, ya que los altavoces encerrados en un *baffle* se comportan de forma bastante aproximada.

La aproximación se puede representar mediante

$$G_{2D}(\boldsymbol{x}|\boldsymbol{x}_0, w) \approx \sqrt{\frac{2\pi|\boldsymbol{x}-\boldsymbol{x}_0|}{jk}} \underbrace{\frac{1}{4\pi}\frac{e^{jk|\boldsymbol{x}-\boldsymbol{x}_0|}}{|\boldsymbol{x}-\boldsymbol{x}_0|}}_{G_{3D}(\boldsymbol{x}|\boldsymbol{x}_0.w)}, \tag{2.32}$$

Como puede comprobarse, la aproximación de la función de Green de 2D solamente aplica una corrección espectral y de amplitud sobre la original de 3D. Debe hacerse notar, que esta aproximación, denominada aproximación de fase estacionaria, supone que la fuente está en campo lejano, para posiciones de la fuente virtual cercanas a la fuentes secundarias puede llegar a no ver válida, al igual que para frecuencias muy bajas.

Determinación de la función *driving* para 2D

La función *driving* ya se ha descrito anteriormente a través de la Ecuación (2.26), sin embargo, es necesario describir el comportamiento de la Ecuación (2.28) al sustituir en ella la aproximación de la ecuación de Green del campo libre en 2D (2.32). Al hacer la sustitución obtenemos

$$P(\boldsymbol{x}, w) = -\oint_{\partial V} \sqrt{\frac{2\pi|\boldsymbol{x}-\boldsymbol{x}_0|}{jk}} D_{2D}(\boldsymbol{x}_0, w) G_{3D}(\boldsymbol{x}|\boldsymbol{x}_0, w) dS_0. \tag{2.33}$$

El término de compensación que introduce la aproximación se puede integrar dentro de la función *driving*, pudiendo emplearse de esta forma la función de Green de 3D (2.30) en la reproducción en 2D. Como puede desprenderse de la ecuación (2.33) la corrección espectral no depende de la posición dentro del volumen (la sala), sin embargo, la corrección en amplitud si. Por tanto, la corrección en amplitud se podrá realizar únicamente

para un punto de la sala manteniendo el resto un error de amplitud. Si al punto para el cual se realizará la corrección se le denomina $\boldsymbol{x}_{ref}$, la función *driving* corregida queda de la siguiente forma:

$$D_{2,5D}(\boldsymbol{x_0}, w) = \sqrt{\frac{1}{jk}}\sqrt{2\pi}|\boldsymbol{x}_{ref} - \boldsymbol{x}_0|D_{2D}(\boldsymbol{x}_0, w). \tag{2.34}$$

La notación 2,5D hace referencia a la mezcla de la síntesis en 2D con la aproximación de la función de Green en 3D.

Si a continuación volvemos a sustituir (2.34) sobre (2.28) obtenemos la ecuación del campo generado por una fuente virtual cuando las fuentes secundarias son esféricas

$$P(\boldsymbol{x}, w) = -\oint_{\partial V} D_{2,5D}(\boldsymbol{x}_0, w)G_{3D}(\boldsymbol{x}|\boldsymbol{x}_0, w)dS_0. \tag{2.35}$$

Si obligamos a que la distribución de fuentes secundarias formen una línea recta, a la cual situamos sobre el eje x, obtenemos la denominada integral de Rayleigh de dimensión 2,5 [Start, 1997; Verheijen, 1997]

$$P(\boldsymbol{x}, w) = -\int_{-\infty}^{\infty} D_{2,5D}(\boldsymbol{x}_0, w)G_{3D}(\boldsymbol{x}|\boldsymbol{x}_0, w)dx_0. \tag{2.36}$$

2.5.3. Comparación

En [Spors et al., 2008] el propio autor hace una comparación de los fundamentos teóricos clásicos con la nueva formulación. La teoría clásica únicamente contempla la recreación de escenas sonoras en 2D mediante arrays de altavoces lineales, mientras que la nueva formulación permite recrear escenas sonoras tanto en 3D como en 2D empleando una distribución de los altavoces arbitraria. Aunque en un primer momento esto puede parecer una ventaja, en la práctica es muy aconsejable emplear superficies

planas para reproducir escenas 3D y arrays lineales para las escenas en 2D [Spors et al., 2008].

En el caso 2D, para arrays lineales de altavoces, la formulación tradicional y la nueva proporcionan el mismo resultado siempre que la fuente virtual esté lo suficientemente alejada de las fuentes secundarias, esto es, siempre que la aproximación de campo lejano se cumple [Spors et al., 2008].

2.5.4. Limitaciones

En el desarrollo teórico de WFS se han realizado una serie de simplificaciones hasta conseguir limitar la situación de las fuentes secundarias sobre una única línea de reproducción. En la práctica es necesario ir más allá y añadir otras limitaciones.

En primer lugar, una línea es infinita en longitud por lo que necesitaríamos infinitos altavoces para cubrirla en su totalidad. Así, será necesario reducirla a un segmento. En segundo lugar los altavoces deberán estar separados entre sí una distancia mínima. Esta distancia deberá ser al menos la debida a su propio tamaño. Por último, los altavoces no se comportan como una fuente sonora omnidireccional a todas las frecuencias sino que tienen cierta directividad, sobre todo a altas frecuencias.

En las siguientes secciones vamos a repasar las diversas consecuencias que conllevan todas las simplificaciones realizadas.

Restricción a un array lineal

Aunque la teoría en la que se basa WFS contempla a la perfección la generación de campos sonoros tridimensionales en la práctica hay muy pocos argumentos convincentes para instalar un array de estas características. Ya en su momento se realizaron experimentos en esta dirección [Ono and

Komiyama, 1997] (véase la Figura 2.18), pero el coste material y la potencia de calculo necesaria resultaron excesivos. En [Spors et al., 2008] se ha realizado recientemente una revisión de la teoría en la que se fundamenta WFS extendiéndola para el caso tridimensional, sin embargo, su aplicación sigue distando bastante de ser práctica debido principalmente a la ingente potencia de cálculo necesaria al aumentar el número de canales.

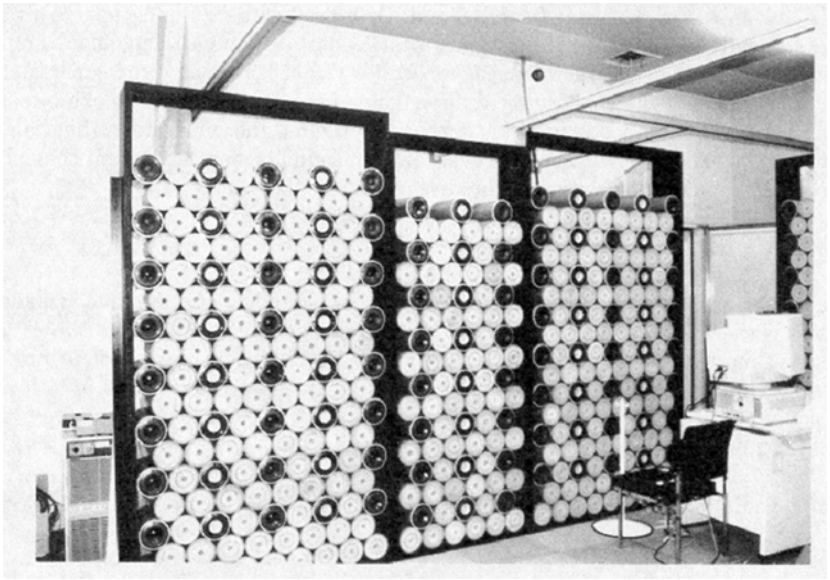


Figura 2.18. 'Muro de altavoces', imagen extraída de [Ono and Komiyama, 1997].

Así, para simplificar enormemente la implementación de sistemas de WFS se reduce el campo sonoro sintetizado a dos dimensiones. En general para ello se emplean arrays lineales de altavoces en vez de superficies de altavoces. Las consecuencias de emplear arrays lineales de altavoces son, en primer lugar y debido a que el campo sintetizado correctamente se reduce al plano horizontal en el que se encuentra situado el array [Wittek, 2003]:

- Aquellos oyentes que se encuentren fuera del plano horizontal tendrán

errores de localización de las fuentes sonoras.

- Únicamente se pueden sintetizar las fuentes sonoras situadas en ese mismo plano horizontal. Para solucionar este problema se pueden emplear diversas técnicas, pero en general la solución consiste en colocar varios altavoces por encima y por debajo del plano horizontal, apoyándonos en la poca sensibilidad que tiene el oído para distinguir los efectos de elevación en comparación con los de panorámica.

En segundo lugar, hay que tener en cuenta que bajo determinadas circunstancias un array lineal no puede sintetizar ondas esféricas, en su lugar genera ondas cilíndricas. Un ejemplo claro son las ondas planas, idealmente una onda plana es una onda esférica cuyo origen es lejano (fuente sonora alejada del array), al intentar sintetizarla el array genera una onda cilíndrica y por tanto la atenuación con la distancia es distinta (véase Fig. 2.19). Así, la mayor diferencia entre una onda plana real y una sintetizada por un array lineal es una atenuación de 3 dB cada vez que se duplica la distancia.

Esto no es así cuando se sintetiza una fuente cercana al array, en el caso extremo en el que la fuente se sitúa sobre uno de los altavoces del array el frente de onda sintetizado será esférico (suponiendo que el altavoz es ideal). En los casos intermedios el error de atenuación con la distancia será intermedio. Por tanto, este error es dependiente de la posición en la que se encuentre la fuente con respecto al array.

Difracción

En teoría, la síntesis del campo sonoro se consigue al sumar la contribución de infinitas fuentes secundarias, en nuestro caso estas fuentes son altavoces. En la práctica, como esto no es posible, el array de altavoces empleado siempre tiene una longitud finita. Este array finito puede verse

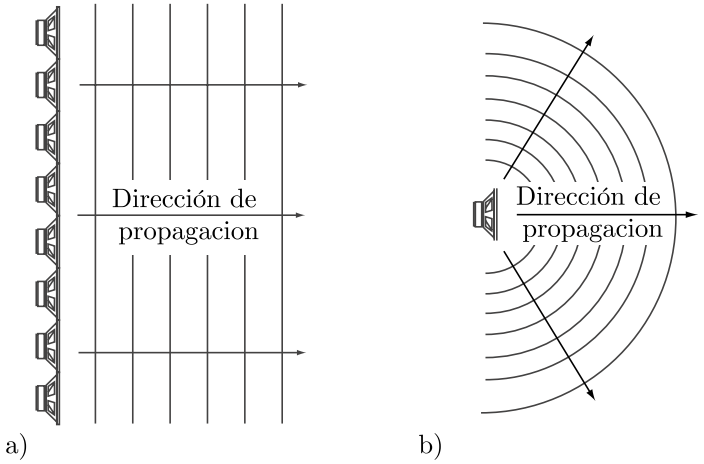


Figura 2.19. Sección horizontal a) y vertical b) de un array lineal reproduciendo una onda plana.

como una ventana, a través de la cual la fuente sonora virtual es visible, o no, para el oyente. Así, existe un área «iluminada» por la fuente virtual junto con otro área de «sombra» [Sonke, 2000] (véase la Figura 2.20). Empleando esta analogía se puede ver fácilmente que en los extremos del array se producirá difracción, impidiendo de esta forma generar un campo sonoro correcto. Estas contribuciones erróneas al campo sonoro se verán como ecos, o pre-ecos para fuentes focalizadas (internas a la sala de escucha), que dependiendo de su amplitud y desfase al llegar al oyente pueden producir coloraciones. Para reducir el efecto de la difracción, también conocido como efecto de «truncado del array», se aplica una ventana (*tapering window*) a las señales de excitación del array. Esto es, se aplica un peso menor a los altavoces cercanos al extremo del array. De esta forma la difracción disminuye drásticamente pero a costa de reducir la zona efectiva de escucha [Sonke, 2000; Boone et al., 1995].

Una solución alternativa a los efectos de la difracción consiste en medir

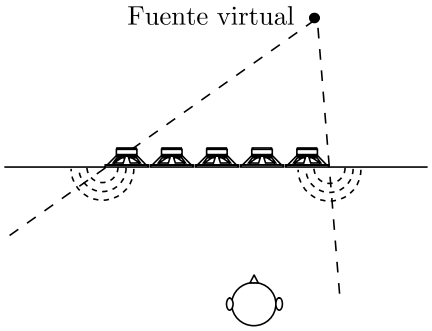


Figura 2.20. Difracción en los extremos del array.

la radiación obtenida por difracción en un punto de referencia y considerarla como una versión escalada y con un patrón de directividad específico de la difracción producida por los extremos del array. Una vez conocida esta radiación, se puede compensar con la señal emitida por el array, aunque hay que tener en cuenta que esta compensación solo se produce correctamente en el punto de referencia tomado y que la coloración puede ser incluso mayor fuera del área de escucha [de Vries et al., 1994].

Aliasing espacial

La separación inherente entre los altavoces de un array produce *aliasing* espacial en el campo sonoro sintetizado a partir de una determinada frecuencia. La solución intuitiva consiste en poner más altavoces y más juntos, pero esto, en la práctica no es viable. En primer lugar, al aumentar el número de canales estamos aumentando las necesidades de cálculo, en segundo lugar, no es posible acercar más los altavoces debido a su tamaño y reducir el tamaño tampoco es una solución, puesto que cuanto menor es el diámetro del diafragma peor se emiten las bajas frecuencias [Pueo and Romá, 2003].

La frecuencia de *aliasing* o, frecuencia límite a partir de la cual se produce el *aliasing* espacial, depende principalmente de la separación entre altavoces y se puede obtener mediante la siguiente ecuación [de Vries, 1996]:

$$f_a = \frac{c}{2\Delta x \cos \Theta_{\text{máx}}} \tag{2.37}$$

Siendo Δx la separación entre altavoces y $\Theta_{\text{máx}}$ el ángulo de incidencia máximo.

El *aliasing* conlleva los siguientes efectos adversos:

- Se produce una alteración de la forma del frente de onda, lo que degrada la localización de la fuente, sobre todo para fuentes con un fuerte contenido en altas frecuencias.

- A partir de la frecuencia de *aliasing* se produce una coloración debido a los efectos de filtrado peine.

- Los oyentes que se encuentran en movimiento experimentan un molesto efecto de modulación en las altas frecuencias.

En la Figura 2.21 se puede ver un ejemplo de un frente de onda bien formado y el mismo frente de onda con *aliasing* espacial. En el ejemplo mostrado, se emite en los dos casos un tono puro de 1 kHz con dos separaciones de altavoces distintas.

Hasta el momento se ha tratado el problema desde varios enfoques:

- Ignorar el problema.

- Incrementar el número de *tweeters* en el array.

- Emplear altavoces de modos distribuidos (DML, *Distributed Mode Loudspeakers*.

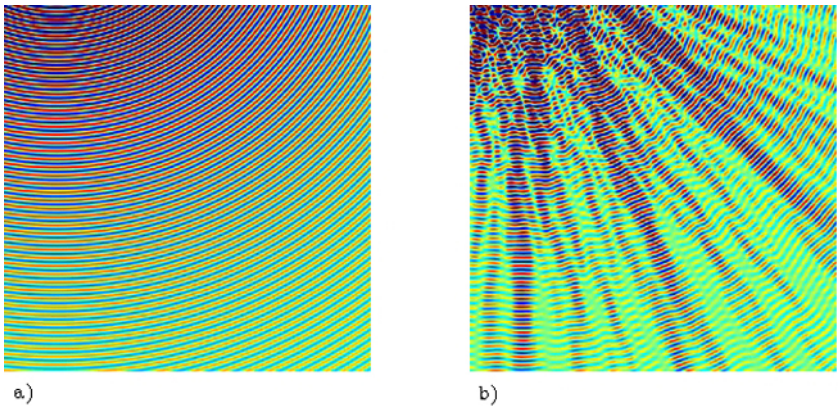


Figura 2.21. Simulación numérica del frente de onda de un tono puro a) sin y b) con aliasing espacial.

- Reducir el ancho de banda espacial.

- Emplear la técnica OPSI (*Optimised Phantom Source Imaging*).

En primer lugar, ignorar el problema no es una solución, aunque el *aliasing* se ha considerado como un problema menor, ya que perceptualmente pasa bastante desapercibido como se ha comprobado en diferentes trabajos previos [Boone et al., 1995].

En segundo lugar, incrementar el número de *tweeters*, esto es, emplear más *tweeters* que *woofers* [Theile et al., 2003], reduce el *aliasing*, pero a cambio incrementa notablemente la potencia de cálculo necesaria y de todas formas no lo resuelve de manera definitiva, simplemente aumenta la frecuencia de *aliasing* aunque esta permanece en un valor muy bajo.

En tercer lugar, el empleo de altavoces DML actuando como paneles multi-actuador (MAP, *Multiactuator Panel*) no mejora la frecuencia de *aliasing*, pero si reduce en cierta manera su percepción por un oyente al radiar estos altavoces un campo difuso [Boone and de Bruijn, 2000][Pueo

et al., 2007]. Por contra, las fuentes sonoras con un fuerte contenido en altas frecuencias aparentan ser mucho mayores de lo que son en realidad lo que dificulta su localización. Un estudio riguroso sobre el uso de paneles DML en la implementación de sistemas de WFS se encuentra en [Pueo, 2008].

El cuarto lugar, la técnica de la reducción del ancho de banda espacial [Start et al., 1995] es más una curiosidad teórica que una técnica en sí. La idea de este método es realizar un filtrado paso bajo de la señal que llega al array desde la fuente virtual, minimizando así su contribución sobre el *aliasing* espacial. El problema de este método reside en que el filtrado se hace sobre en el espectro de la señal en el dominio espacial, no en el temporal. Este filtrado requiere tal potencia de cálculo que no es viable para ser realizado en tiempo real.

En último lugar, la técnica OPSI emplea los métodos clásicos de la imagen *phantom* estéreo [Wittek, 2002]. Para ello divide la señal de cada fuente en dos bandas, la banda de baja frecuencia se procesa mediante WFS, ya que su reconstrucción es correcta, y la banda de alta frecuencia, que es la que tiene *aliasing*, se reconstruye mediante una imagen *phantom*.

Directividad de los altavoces

Idealmente, los altavoces que se van a emplear en sustitución de las fuentes secundarias del principio de Huygens deben actuar como fuentes sonoras omnidireccionales. En la práctica ningún altavoz se comporta como una fuente sonora omnidireccional, o al menos no lo hacen para todo el margen de frecuencias. En baja frecuencia si es posible conseguir un altavoz omnidireccional, pero conforme aumenta la frecuencia se vuelven más directivos [Pueo and Romá, 2003].

En [de Vries, 1996] se realiza un estudio sobre la posibilidad de adaptar

la señal *driving* que se envía a los altavoces para compensar en la medida de lo posible su directividad. Aunque de este estudio se concluye que esto es posible siempre y cuando todos los altavoces del array sean iguales, en la práctica no se implementa la compensación puesto que conlleva un alto coste computacional y además no es muy difícil encontrar unos altavoces que se comporten de forma omnidireccional por debajo de la frecuencia de *aliasing*, frecuencia a partir de la cual WFS deja de funcionar.

Capítulo 3

Soluciones prácticas sobre aspectos críticos en la implementación de sistemas WFS

En el capítulo anterior se presentó la teoría que soporta el sistema de reproducción de sonido *Wavefield Synthesis*. De la teoría desarrollada se obtiene la ecuación básica que proporciona las señales de excitación para cada uno de los altavoces que componen un array WFS. Así mismo, se establecieron las limitaciones físicas de este sistema, que deben ser tenidas en cuenta en la implementación de los mismos.

En este capítulo, se pretende, por un lado, aportar soluciones prácticas a las limitaciones físicas ya comentadas en el capítulo anterior. Además, por otro lado, aparecen otra serie de problemas relacionados con la implementación, que aunque no se describen como limitaciones físicas en una

primera instancia, si que suponen un problema a superar cuando se busca un verdadero sistema que funcione eficientemente en tiempo real.

Entre las limitaciones del primer tipo, se tratará el aspecto del *aliasing*, así como el efecto de difracción producido por el truncamiento de array. También se darán posibles soluciones eficientes para la corrección de la respuesta de los altavoces.

Sobre las limitaciones del segundo tipo (mucho menos comentadas en la bibliografía), se proporcionarán soluciones propias e innovadoras que suponen una aportación importante de esta tesis doctoral. Entre ellas se encuentran las relacionadas con fuentes en movimiento, fuentes próximas al array, fuentes focalizadas (internas al array) y algoritmos eficientes para la selección de los subarrays óptimos que deben ser excitados, para cada posición espacial de las fuentes virtuales.

Todas estas soluciones, convenientemente programadas y funcionando en tiempo real, darán como resultado un software de alta eficiencia que se encuentra a la altura o incluso supera a algunas aplicaciones comerciales surgidas recientemente.

3.1. Consideraciones previas

Antes de entrar de lleno en los problemas y limitaciones que hemos tenido que superar conviene tener en cuenta unas consideraciones previas que permitirán por un lado centrarse en el trabajo desarrollado y por otro entender más adelante algunos aspectos sutiles de la implementación.

Ecuación objetivo

Para comenzar a trabajar en la implementación de un sistema de WFS es necesario tener claro cuál es el objetivo de estos sistemas.

Un sistema de WFS intenta sintetizar el campo sonoro que radiaría una fuente virtual situada detrás de los altavoces. Para ello, lo que el sistema hace es determinar qué debe emitir cada uno de los altavoces de forma que al combinar el sonido de todos ellos se forme el campo sonoro deseado. Esto es, el trabajo desarrollado por WFS es aplicar una transformación al sonido de la fuente virtual para obtener el sonido que deben emitir los altavoces. Esta transformación está descrita en la teoría de WFS por medio de las funciones *driving* las cuales están descritas en la Sección 2.5.1 y que por comodidad se reproduce de nuevo aquí:

$$Q(\boldsymbol{r_n}, w) = S(w)A(\boldsymbol{r_n}, w)e^{-jkr_n}. \tag{3.1}$$

Así, se puede comprobar que todo el trabajo que va a realizar el sistema de WFS es determinar la función *driving* adecuada a cada par fuente virtual/altavoz y aplicarla sobre la señal de cada fuente virtual.

Procesado por bloques

Siempre que se procesa sonido en tiempo real con un ordenador personal es necesario procesar las muestras por bloques (para más detalles ver la Sección 4.3.1). Esto viene a indicar que la mayor parte de los parámetros con los que se trabaja únicamente podrán cambiar su valor una vez por bloque, por lo que no será posible modificarlos de forma continua. Esto debe tenerse en cuenta en algunas de las soluciones previstas en el presente capítulo.

Definición del centro de sala para casos particulares

Un sistema de reproducción de WFS está compuesto por un array de altavoces el cual está compuesto a su vez por varias tiras (segmentos) de

arrays lineales de altavoces, los cuales idealmente se encuentran situados en las parades de la sala de reproducción rodeándola por completo. Así cuando uno habla del centro de la sala está hablando también del centro del array, es normal emplearlos como sinónimos. En la práctica, esto no siempre es así, es decir, el array no abarca toda la sala o incluso está compuesto por un único segmento lineal por lo que no rodea ningún área y su centro está en la mitad del segmento.

Según el concepto teórico de WFS, el campo sonoro va a ser reconstruido en todo un área por completo, eliminando el *sweet spot*. Como más adelante se verá esto no siempre es posible y habrá una zona preferente cuya reconstrucción del campo sonoro será más fiel que la del resto del área. Esto es, para poder llevar a la práctica la implementación de un sistema de WFS es necesario en más de una ocasión apoyarse en la situación de un punto en el espacio, cuya posición es arbitraria, al que, por decirlo de una forma simple, se va a tratar mejor que al resto. Para este punto arbitrario suele escogerse el centro del array y si este no es viable por la geometría que tiene el array (por ejemplo si es un único segmento), se elige el centro de la sala o cualquier otro punto que haga esta función.

Por tanto, cuando en el resto del texto hablemos del centro de la sala, del centro del array o simplemente del centro, nos estaremos refiriendo a este punto arbitrario.

3.2. Síntesis de una fuente virtual

La síntesis de una fuente virtual mediante WFS es bastante sencilla. Lo único que hay que hacer es aplicar la transformación descrita por la teoría sobre la señal sonora de la fuente para cada uno de los altavoces, obeniendose así la señal *driving* de cada altavoz, tal y como se ha recordado

en la Sección 3.1.

La transformación a aplicar se puede dividir en dos términos, uno de amplitud y otro de fase. Para aplicar el término de amplitud se multiplica la señal de la fuente virtual por el valor obtenido y, para introducir el término de fase es necesario aplicar un retardo.

La forma más sencilla y eficiente de aplicar un retardo relativamente elevado sobre una señal es mediante una línea de retardo. La cual se suele implementar en forma de un *buffer* circular con dos punteros, uno de lectura y otro de escritura. Manteniendo una distancia equivalente al retardo (en muestras) entre los dos punteros se consigue aplicar el retardo sin prácticamente ningún esfuerzo.

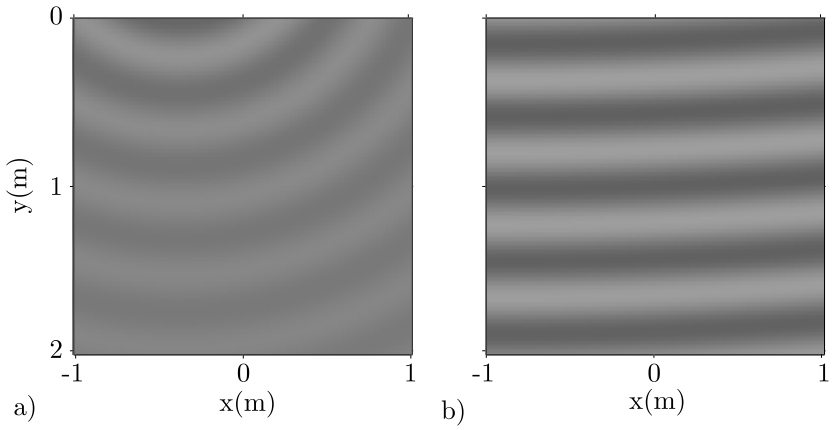


Figura 3.1. Representación del campo de presión generado por cada tipo de fuente sonora. a) Puntual. b) De ondas planas.

Los tipos de fuentes que podemos encontrar en WFS son: puntuales y de ondas planas. Aunque en realidad son iguales, con la salvedad de que una fuente de ondas planas es una fuente puntual muy lejana a la cual se le ha aumentado la ganancia para compensar la atenuación que produce la distancia. En la Figura 3.1 se puede ver la simulación del campo sonoro

generado por ambos tipos de fuentes.

3.2.1. Retardo fraccionario

Siempre que se trabaja en un sistema digital la unidad mínima de tiempo es el periodo T, por lo que solamente se puede aplicar un retardo a una señal un número entero de muestras. Así todo retardo que se desee aplicar deberá ser redondeado al entero más cercano, introduciendo un error por exceso o por defecto, el cual está confinado en el intervalo $[-T/2, T/2]$.

Haciendo un sencillo cálculo se puede demostrar que introducir un error en el retardo de hasta una muestra en el cálculo de la señal *driving* de WFS equivale a tener desplazado el altavoz casi 8 milímetros de su posición, siendo fs=44,1 kHz. Así, al redondear los retardos estamos produciendo el mismo efecto que si desplazamos ligeramente los altavoces en un radio de 4 milímetros, impidiendo que los frentes de onda generados encajen perfectamente para formar el campo sonoro sintético. En [Strauß and Munderloh, 2007] se estudia el problema del mal posicionamiento de los altavoces y la conclusión a la que se llega es que afecta más a las fuentes focalizadas. Este problema se acentúa con el movimiento de la fuente pues el desplazamiento se vuelve prácticamente aleatorio, creándose un patrón de interferencia aleatorio en el campo sonoro sintético. Obviamente, la molestia que introduce el patrón de interferencia aumenta con la frecuencia, de ahí que no siempre se preste atención a este problema, pues como ya se explicó en el capítulo anterior, a partir de la frecuencia de *aliasing* el campo sonoro sintético no es correcto.

En [Franck et al., 2007] se recomienda encarecidamente el uso de retardos fraccionarios, haciendo especial énfasis en la buena elección del método de interpolación ya que algunos pueden introducir defectos audibles sobre

la señal, sin embargo, no se indica qué métodos son adecuados por lo que es necesario realizar un estudio por nuestra cuenta.

Para conseguir un retardo intermedio es necesario interpolar el valor de las muestras mediante filtros específicos de retardo fraccionario [Laakso et al., 1996], sin embargo, a cambio estamos elevando el coste del cálculo de las muestras de forma considerable. Los requisitos que se imponen a todo interpolador son tres [Rocchesso, 2000]:

- Respuesta en frecuencia plana.

- Fase lineal.

- Repuesta libre de transitorios al variar el retardo.

Además, para que el interpolador sea utilizable en WFS es necesario añadir un cuarto requisito:

- Mínimo coste de interpolación.

Este último requisito va a ser primordial en la elección, pues WFS requiere mucha potencia de cálculo y el uso del interpolador va a aumentar considerablemente esta necesidad.

Uno de los interpoladores más sencillos y más empleados , que se acerca bastante al cumplimiento de todos los requerimientos para señales de tipo paso bajo, en general la señal de audio suele cumplir sobradamente esta suposición, es el filtro de interpolación lineal [Smith, 2008]. Este filtro se puede construir de forma eficiente con un FIR de primer orden, siendo la ecuación que lo describe la siguiente:

$$H(z) = \tau - 1 + \tau z^{-1} \tag{3.2}$$

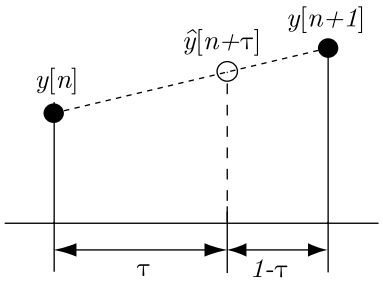


Figura 3.2. Representación de la interpolación aplicada por el filtro FIR de primer orden.

En la cual $0 < \tau < 1$ representa el retardo fraccionario a aplicar, tal y como se muestra en la Figura 3.2.

Si se representa el filtro por su ecuación en diferencias esta queda de la siguiente forma:

$$\hat{y}(n-\tau) = (\tau - 1)y(n) + \tau y(n-1) \tag{3.3}$$

Aunque es posible minimizar el coste de su cálculo reformulando la ecuación al sacar factor común τ:

$$\hat{y}(n-\tau) = y(n) + \tau(y(n-1) - y(n)) \tag{3.4}$$

De esta forma el cálculo solamente requiere dos sumas y una multiplicación por cada muestra interpolada, por lo que es esta última expresión la que se va a emplear en esta tesis para aplicar el retardo fraccionario.

Es necesario hacer constar que el filtro únicamente es capar de introducir retardos de menos de una muestra, por lo que para conseguir un retardo mayor se deberán realizar dos pasos:

1. Mediante una línea de retardo se introduce la parte entera del retardo.

2. Con el filtro interpolador se introduce la parte fraccionaria faltante.

El diagrama de bloques que describe este funcionamiento está representado en la Figura 3.3.

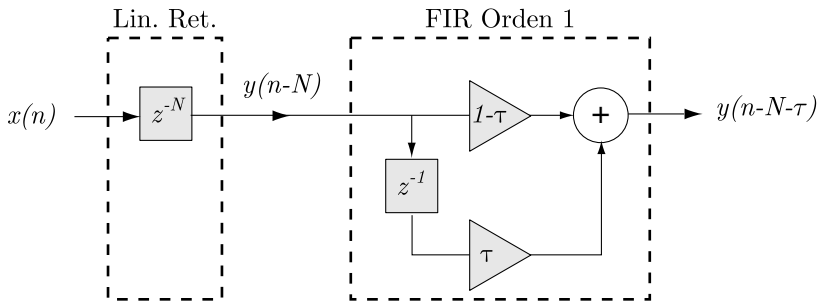


Figura 3.3. Diagrama de bloques del retardo fraccionario completo.

En cuanto a los requerimientos mencionados anteriormente, el filtro no los cumple a la perfección, pero su funcionamiento ha demostrado ser notable. El coste del cálculo de la interpolación es mínimo comparado con otros interpoladores, la respuesta en frecuencia es plana en casi todo el espectro de la señal de audio, siendo perfecta en baja frecuencia, la fase no es lineal pero no introduce ninguna distorsión audible, por lo menos ninguna distorsión perceptible por el autor, y, por último, no introduce ningún tipo de transitorios al variar el retardo de la señal, siendo este el argumento principal esgrimido en [Franck et al., 2007] para realizar una elección cuidadosa.

3.3. Selección de sub-arrays para topologías complejas

Cuando se implementa un sistema WFS normalmente es necesario emplear más de un array lineal de altavoces para cubrir todo el área de la sala.

Generalmente, el montaje se construye distribuyendo varios arrays lineales alrededor de la sala exceptuando una pequeña zona en la que se encuentra la puerta de entrada. A la hora de sintetizar el campo sonoro generado por una fuente no siempre es conveniente activar simultáneamente todos los altavoces del array, en general solamente habrá que activar aquellos altavoces que emitan sonido en la dirección en la que debe viajar el frente de onda.

A continuación se mostrará la técnica de activación de altavoces que están empleando otros sistemas en la actualidad para a continuación mostrar la técnica propuesta por el autor.

3.3.1. Técnica de la línea de referencia

Una solución por la que optan algunas de las implementaciones de sistemas WFS [Spors et al., 2002] es trazar una línea de referencia, perpendicular al segmento que une la fuente sonora y el centro del array (véase la fig.3.4). Todo altavoz que se encuentre en el mismo semiplano que la fuente debe ser activado.

Aunque este tratamiento es bastante simple e intuitivo, en determinadas circunstancias no siempre es la mejor solución. Con esta técnica es posible encontrar muchas situaciones en las que el campo sonoro reconstruido no es el más adecuado al existir altavoces que radian sonido en sentido contrario al avance del frente de onda, generalmente estos altavoces se encontrarán cerca de la línea de referencia, como en el ejemplo de la Figura 3.4 en la que hay tres altavoces que radiarán en sentido contrario, los dos altavoces más bajos y el altavoz superior que está junto a la línea de referencia.

Como se puede apreciar esta técnica hace uso del «centro» como punto de apoyo para determinar que altavoces activar. Así debemos tener en cuen-

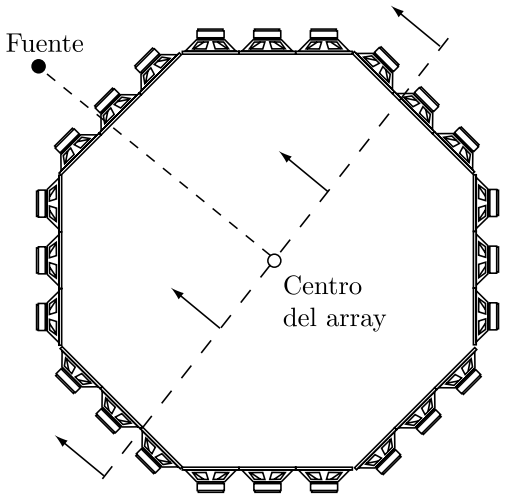


Figura 3.4. Selección de los altavoces por medio de la línea de referencia.

ta los posibles inconvenientes que esto pueda ocasionar, como por ejemplo donde ubicar este centro. En el ejemplo de la Figura 3.4 es sencillo, pero no todas las configuraciones de arrays son tan simples, para más detalles repasar la Sección 3.1.

3.3.2. Técnica de la visibilidad

La contribución realizada por el autor difiere sensiblemente de la solución previa, ya que es un método bastante más elaborado. Puesto que para que el frente de onda reconstruido sea natural únicamente deben emitir aquellos altavoces cuya radiación viaje en la misma dirección y sentido que el frente de onda, es necesario localizar estos altavoces para cada fuente. Como el frente de onda arranca desde la posición de la fuente y viaja expendiéndose a su alrededor, todos aquellos altavoces que al avanzar desde la fuente estén de espaldas a ella emitirán sonido en la dirección del frente de onda, y aquellos que estén de cara harán lo contrario. Así, aquellos alta-

voces que tengan la fuente detrás deberán emitir sonido y aquellos que la tengan delante no. Siguiendo este método, en el ejemplo de la Figura 3.4 de la selección de altavoces anterior se eliminarían los tres que emitían en sentido contrario.

De cara a la implementación, este método se puede optimizar si tenemos en cuenta que el array está construido con segmentos de arrays lineales, ya que basta con determinar si la fuente está detrás o delante de un solo altavoz de cada segmento, el resultado para este altavoz será el mismo para el resto de altavoces del segmento. Así podemos reducir el cálculo de la comprobación del número de altavoces al número de segmentos que generalmente es mucho menor. Sirva de ejemplo el prototipo del laboratorio, en el cual se reduce el número de cálculos a una doceava parte ya que se reduce el cálculo de 96 altavoces a 8 segmentos. Un caso degenerado sería el de un array circular, en el que cada segmento solo tiene un altavoz, por lo que no se conseguiría ninguna optimización en el cálculo.

Una forma más intuitiva de entender el problema es asimilándolo a un problema de visibilidad: una fuente solamente empleará aquellos altavoces «visibles» desde su posición. En la Figura 3.5 se muestra un array en configuración en U abierta en la que se seleccionan aquellos altavoces visibles para distintas posiciones de la fuente sonora.

Por último, será necesario hacer una excepción al realizar la selección de sub-arrays en el caso en que la fuente sonora se encuentre dentro de la sala, lo que en la literatura se denomina una fuente focalizada, este caso especial se tratará con detenimiento en la Sección 3.5.

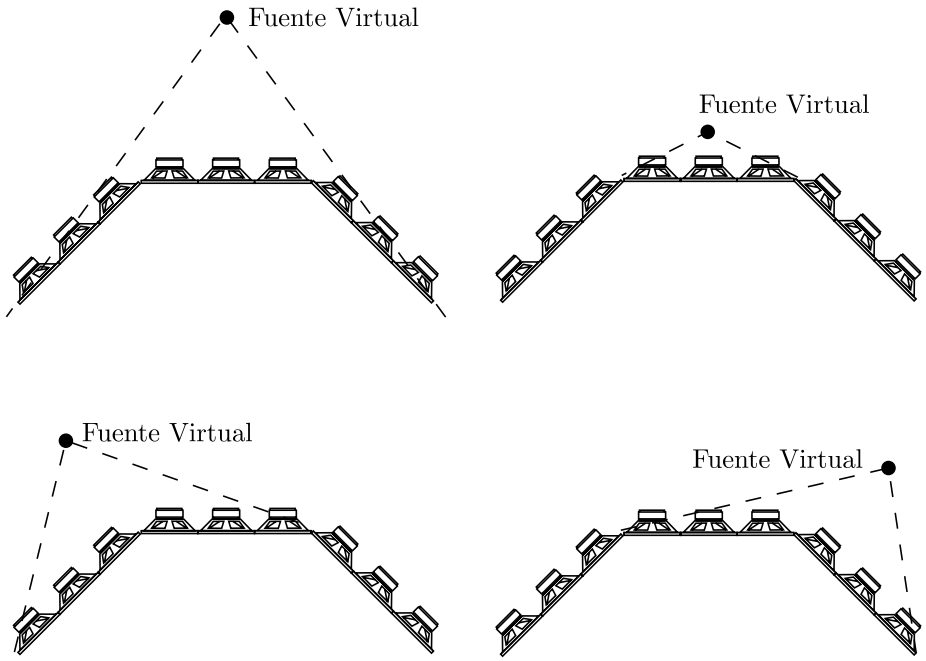


Figura 3.5. Selección de los sub-arrays según la posición de la fuente virtual.

3.3.3. Técnica de la intensidad

Recientemente, en [Spors, 2007a] se realizó un estudio sobre las técnicas de selección de los altavoces y se propuso una técnica analítica de selección basada en la intensidad acústica.

El objetivo de esta técnica es derivar un criterio analítico de selección de las fuentes secundarias basado en el uso del vector de intensidad acústica $\boldsymbol{i}(\boldsymbol{x}, t) = p(\boldsymbol{x}, t)\boldsymbol{v}(\boldsymbol{x}, t)$. El vector de intensidad acústica representa la cantidad de energía por unidad de tiempo y área con respecto a la dirección. La dirección del vector intensidad apunta en la dirección en la que se incrementa la densidad de energía. Por tanto, se puede emplear la intensidad para caracterizar la dirección de propagación de las ondas acústicas [Spors,

2007a]. Sin embargo, es recomendable promediar el vector de intensidad sobre un periodo de la señal. Este promediado temporal se puede realizar de forma conveniente en el dominio de la frecuencia, y se puede expresar mediante [Williams, 1999]:

$$\overline{\boldsymbol{I}}_S(\boldsymbol{x}, w) = \frac{1}{2}\Re\{S(\boldsymbol{x}, w)\boldsymbol{V}_S(\boldsymbol{x}, w)^*\}, \tag{3.5}$$

en la cual $\Re\{\cdot\}$ representa la parte real del argumento y el superíndice * representa la función de conjugación compleja. La ventana de selección de altavoces se pude expresar empleando la intensidad promediada:

$$a(\boldsymbol{x}_0) = \begin{cases} 1 & Si\ \langle \overline{\boldsymbol{I}}_S(\boldsymbol{x}, w), \boldsymbol{n}(\boldsymbol{x}_0)\rangle > 0 \\ 0 & En\ otro\ caso \end{cases} \tag{3.6}$$

donde $\langle \cdot, \cdot \rangle$ representa el producto escalar de dos vectores.

Definida la ventana, en [Spors, 2007a] se continúa comprobando los resultados que se obtienen para fuentes puntuales y de ondas planas (véase la Figura 3.6). En ambos casos el resultado es el mismo, todos aquellos altavoces cuyo ángulo formado entre el vector intensidad y la normal sea agudo son activados.

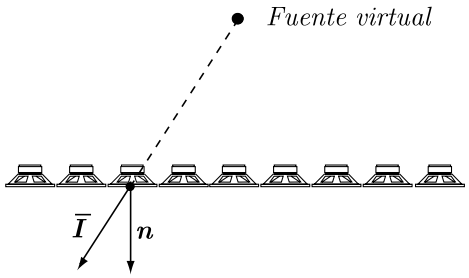


Figura 3.6. Aplicación del criterio de selección para fuentes virtuales.

3.3.4. Comparación y discusión

Una vez presentados los métodos de selección de altavoces, se va a proceder comparar los resultados obtenidos con cada uno de ellos.

Es interesante señalar que la técnica de la intensidad acústica selecciona los mismos altavoces que el método de la visibilidad, la cual se propuso en [Bleda et al., 2003a,b], por lo que el resultado que se obtiene al generar el campo sonoro es el mismo. Con la técnica de la intensidad se seleccionan aquellos altavoces cuyo ángulo formado por el vector de intensidad acústica y la normal del altavoz es agudo, por otro lado, la técnica de la visibilidad selecciona aquellos altavoces visibles, esto es, aquellos que tienen la fuente virtual detrás. No es difícil comprobar que en esta situación, es decir, con la fuente virtual detrás, el ángulo formado entre el vector intensidad acústica y la normal siempre es agudo. En [Spors, 2007a], el propio autor reconoce que la técnica de la intensidad selecciona los mismos altavoces que los criterios empleados en las implementaciones prácticas, aunque no indica cuales son estos criterios. Así, a continuación solamente se compararán las técnicas de la línea de referencia y la visibilidad, ya que la técnica de la intensidad proporciona los mismos resultados que la segunda.

Para realizar la comparación se ha diseñado un software de simulación numérica en Matlab. El funcionamiento de este software de simulación está descrito en el Apéndice C. Este software recrea tanto el campo sonoro original (presión y velocidad de las partículas) que generaría una fuente sonora puntual en una sala dada, sin tener en cuenta sus efectos, como el campo generado por esa misma fuente sintetizado mediante WFS. Comparando ambos campos es posible determinar el error cometido.

Para comenzar con la discusión, se muestra la configuración de la escena utilizada en la Figura 3.7, en la cual se simula una fuente sonora puntual

que genera un tono puro de 800 Hz situada en la posición (0,75; -0,5), dentro de una sala de 1,6 metros de ancho por 1,6 metros de largo. El array de altavoces está compuesto por 48 altavoces dispuestos en configuración de U cerrada, con un espaciado de 10 cm. Por razones de espacio no se han representado todos los altavoces en la figura.

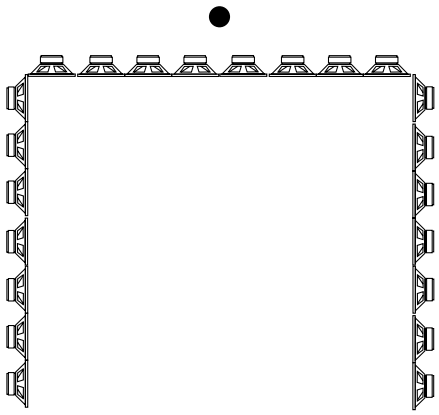


Figura 3.7. Configuración de la escena sonora simulada.

En la Figura 3.8 se puede ver el campo sonoro generado, tanto las presiones como las velocidades de las partículas en los dos ejes, para una fuente real y para fuentes virtuales sintetizadas mediante WFS con las técnicas de selección de los altavoces de la línea de referencia y visibilidad. A simple vista se puede apreciar como las velocidades de las partículas en el eje X, no se están reconstruyendo de forma adecuada al emplear la técnica de la línea de referencia (ver el caso e) de la Figura 3.8), ya que los altavoces situados en los dos laterales generan un frente que no avanza en la misma dirección que el frente de la fuente real.

Antes de comenzar la discusión es necesario hacer constar que el error cometido cerca de los altavoces del array es siempre muy elevado debido a los efectos de campo cercano, aunque no es junto a los altavoces donde

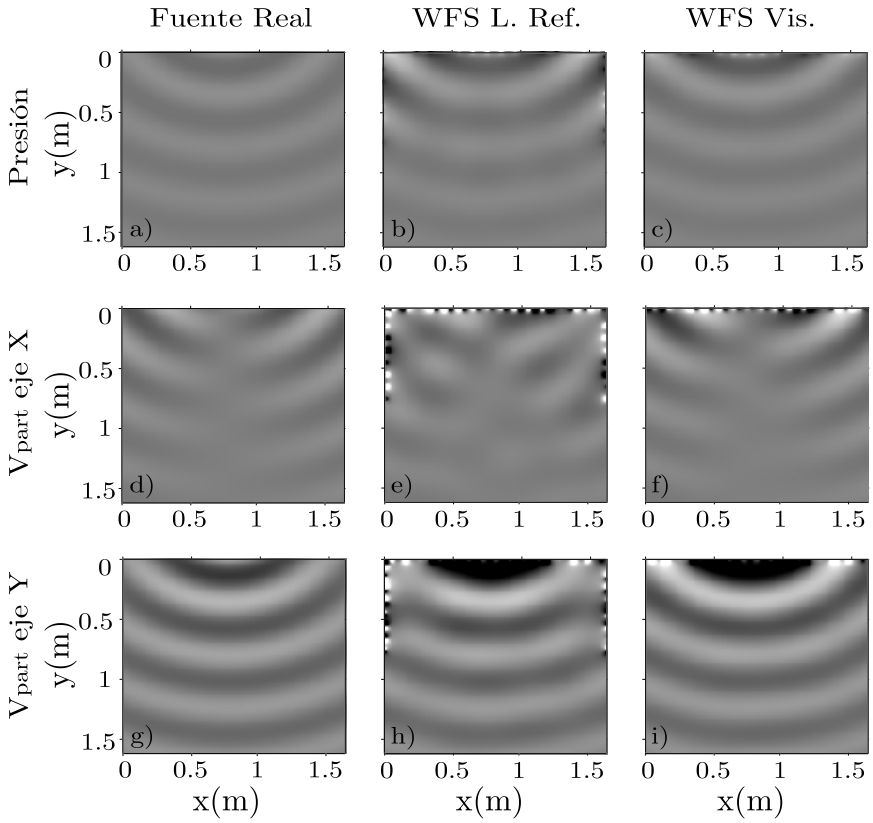


Figura 3.8. Simulación numérica del campo sonoro generado por una fuente puntual. a) Presión, fuente real. b) V_{part} el eje X, fuente real. c) V_{part} eje Y, fuente real. d) Presión WFS, línea de referencia. e) V_{part} eje X WFS, línea de referencia. f) V_{part} eje Y WFS, línea de referencia. g) Presión WFS, visibilidad. h) V_{part} eje X WFS, visibilidad. i) V_{part} eje Y WFS, visibilidad. Todas las distancias están en metros.

interesa que se genere el campo de forma correcta.

En la Figura 3.9 se simula una fuente sonora puntual que genera un tono puro de 800 Hz situada en la posición (0,5; -0,5), dentro de una sala de 1,6 metros de ancho por 1,6 metros de largo. El array de altavoces

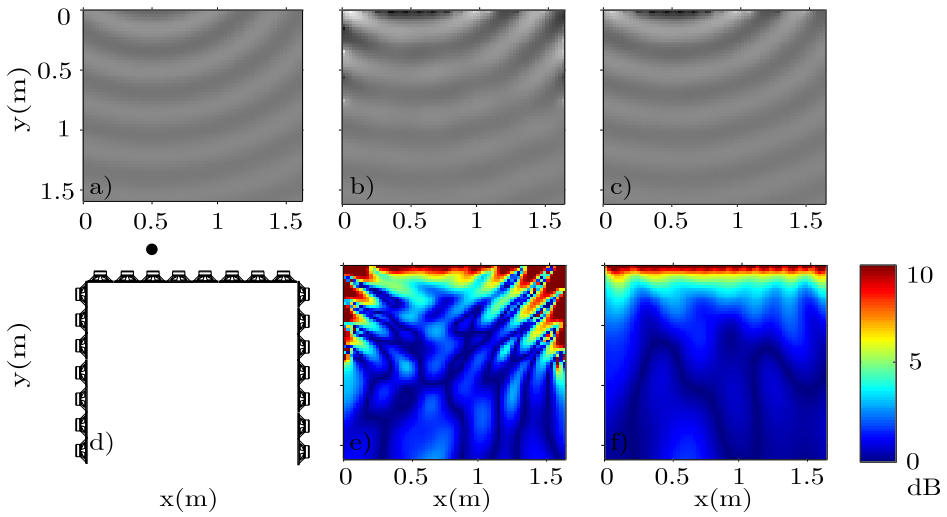


Figura 3.9. Simulación numérica del error cometido al sintetizar una fuente puntual. a) Presión generada por una fuente real. b) Presión generada mediante WFS, línea de referencia, c) Presión generada mediante WFS, visibilidad. d) Configuración del array y posición de la fuente. e) Error cometido, línea de referencia. f) Error cometido, visibilidad.

está compuesto por 48 altavoces[1] dispuestos en configuración de U cerrada, con un espaciado de 10 cm.

Aparentemente, ambos métodos sintetizan el campo sonoro de forma parecida, tal y como se desprende de las imágenes b) y c) de la figura. Sin embargo al comparar los campos sintéticos con el generado por una fuente real se puede comprobar que el error cometido por la línea de referencia es muy superior.

En la Figura 3.10 se realiza la comparación para un array de 32 altavoces en configuración en U abierta con un espaciado de 10 cm, la fuente puntual se encuentra en la posición (2; -0,5) y emite un tono puro de 800

[1]El diagrama d) de la Figura 3.9 representa menos altavoces por razones de espacio.

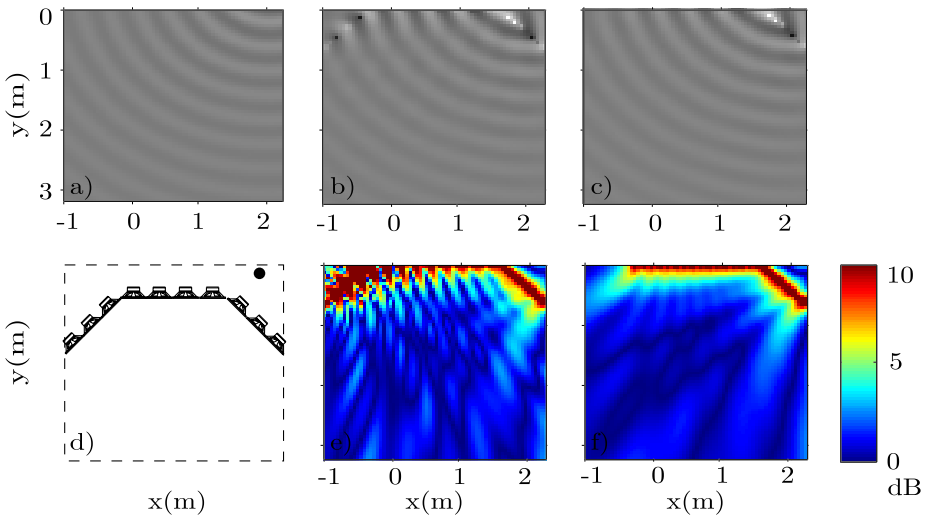


Figura 3.10. Simulación numérica del error cometido al sintetizar una fuente puntual. a) Presión generada por una fuente real. b) Presión generada mediante WFS, línea de referencia, c) Presión generada mediante WFS, visibilidad. d) Configuración del array y posición de la fuente. e) Error cometido, línea de referencia. f) Error cometido, visibilidad.

Hz. El array está compuesto por 16 altavoces en el segmento central y 8 altavoces en cada uno de los segmentos laterales, el ángulo de apertura usado es de 45º. En este caso también se puede apreciar cómo el error cometido al no emplear los altavoces que radian en sentido contrario es mucho menor.

En las situaciones en la que los altavoces empleados por una u otra técnica coinciden, el campo sintetizado es idéntico por lo que el error cometido también lo es. Por todo ello se desprende que el uso de la técnica propuesta es beneficioso pues en el peor de los casos se va a comportar igual que la otra.

3.3.5. Comentarios sobre la difracción

Un aspecto distinto pero dependiente de la selección de altavoces realizada es la difracción que se produce en los bordes del array. Este tema ya se trató en la Sección 2.5.4 aunque debemos hacer aquí una matización.

Cuando se radia sonido empleando un array finito la difracción que se produce en los bordes se mitiga empleando una ventana espacial, la denominada *tapering window*. En [Spors, 2007b] se indica que la ventana de *tapering* debe ser dinámica, ya que si al moverse la fuente virtual cambian los altavoces que se van a emplear, es necesario modificar la ventana de *tapering* de forma acorde. En algunas implementaciones de sistemas WFS se usa este enventanado de forma estática ya que define una ventana para cada segmento del array [Baalman, 2004], y esto no es correcto a menos que los segmentos estén separados entre sí. En la Figura 3.11 se muestra un ejemplo del ajuste dinámico de la ventana de *tapering*.

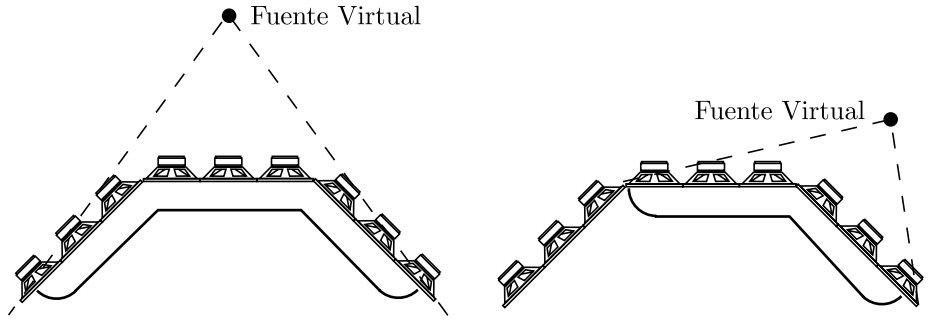


Figura 3.11. Ajuste dinámico de la ventana de *tapering*.

3.4. Fuentes próximas al array

Otro de los problemas que nos encontramos al implementar un sistema de WFS sucede cuando una fuente virtual se acerca demasiado al array de altavoces. Este problema está relacionado con el factor de amplitud de las señales *driving* $A(\boldsymbol{r}_n, w)$, debido a su dependencia del vector $\boldsymbol{r}_n$.

Por un lado, al acercarnos a un altavoz r_n, el módulo del vector, disminuye progresivamente pudiendo llegar a anularse si la fuente se sitúa justo encima de un altavoz. Por otro lado, θ_n, ángulo formado entre el vector y la normal del altavoz, va creciendo hasta llegar a los 90^o justo al cruzar la línea que contiene a los altavoces del segmento. En la Figura 3.12 se muestra la evolución de estos parámetros cuando una fuente se encuentra cerca de un altavoz del array.

El efecto que produce la variación de estos dos parámetros sobre el factor de amplitud es contrario, por un lado, la disminución de r_n provoca un aumento en la amplitud pudiendo llegar a $+\infty$ al anularse r_n, por otro lado, el aumento de θ_n produce una disminución de la amplitud llegando a anularse cuando el ángulo alcanza los 90^o, ya que la amplitud depende del coseno del ángulo. Esta variación contraria de los parámetros produce un descontrol enorme en la amplitud de la señal de salida en la ganancia final de la fuente cerca de los altavoces. En los diferentes casos de la Figura 3.12 se puede ver cómo evoluciona la ganancia proporcionada por los parámetros θ_n y r_n (casos b) y c) respectivamente) conforme se mueve la fuente alrededor del altavoz. El caso d) muestra la ganancia resultante al combinar las dos previas, siendo este factor el que proporcionará la ganancia de la fuente.

Debemos recordar que este problema solamente afecta a los dos altavoces más cercanos a la fuente, para el resto, aún estando la fuente cerca del array, permanece lo suficientemente alejada como para no tener que apli-

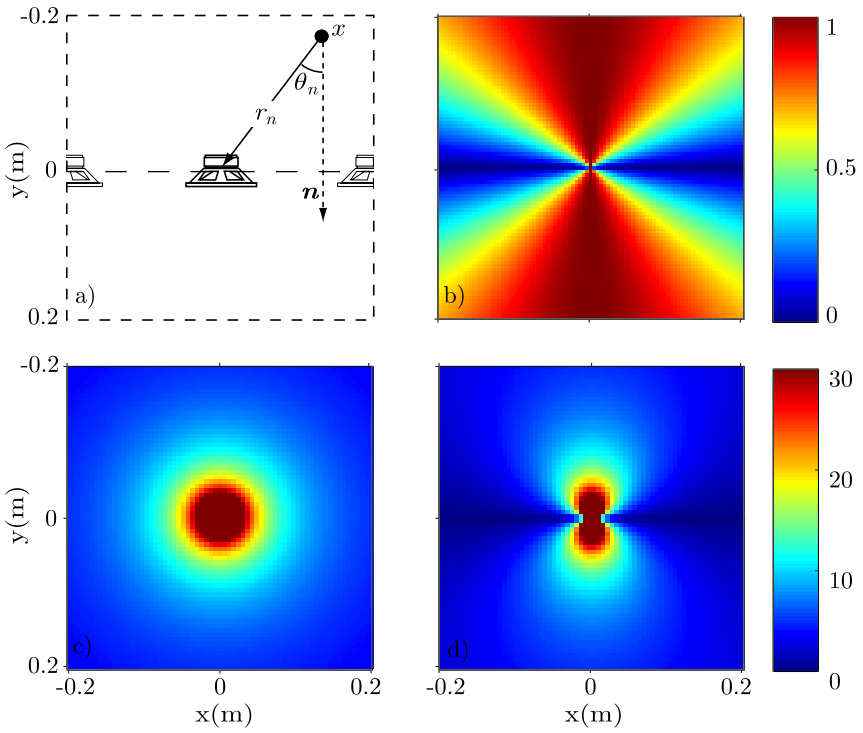


Figura 3.12. Variación de los parámetros que modulan la amplitud de las fuentes cercanas a los altavoces. a) Diagrama descriptivo de las variables implicadas. b) Factor de ganancia proporcionado por $\cos(\theta_n)$. c) Factor de ganancia proporcionado por $1/r_n$. d) Combinación de ambos factores. Los factores de ganancia están en escala lineal.

car ningún tratamiento especial, de hecho, el resto de altavoces enmudecen progresivamente por la acción directa del incremento de θ_n.

En cuanto a la solución a este problema es necesario hacer constar que en la literatura se comenta que debe solucionarse pero no se indica cómo. Esto tiene su lógica pues, por un lado los sistemas que incluyen una solución son comerciales y por tanto no les interesa hacer público el algoritmo. Por otro lado, los prototipos suelen emplearse para investigación y por tanto no

incluyen una solución.

A continuación se mostrará la solución general aportada para este fenómeno la cual trata por separado los dos parámetros problemáticos y comienza con la localización de los altavoces más cercanos a la fuente.

3.4.1. Tratamiento del módulo de r_n

Comenzando con el tratamiento del módulo, la única solución posible es su limitación. Es necesario imponer un umbral a partir del cual no se permitirá su disminución.

Como siempre que se introduce un umbral, es necesario proponer un valor coherente con el problema a tratar. Ya que la situación problemática comienza cuando la fuente está a una distancia similar a la separación entre altavoces Δx, debe recordarse que solamente afecta a los dos altavoces más cercanos pues los siguientes ya está a distancia prudencial, parece lógico imponer esta distancia como umbral. La situación del umbral está descrita en la Figura 3.13 y tal y como se puede apreciar es un conjunto de circunferencias centradas en cada altavoz, no es una línea paralela al segmento que contiene a los altavoces como generalmente se piensa.

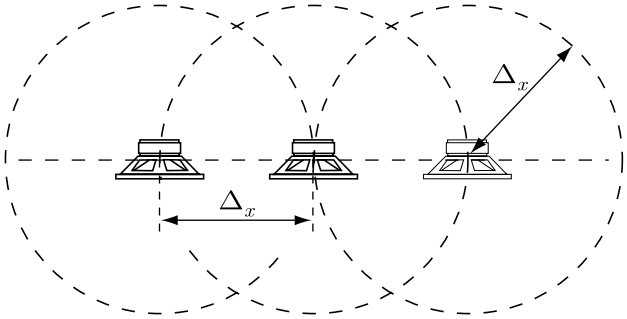


Figura 3.13. Umbral de limitación del módulo.

De esta forma, a medida que la fuente se acerca al array se produce un

aumento progresivo de la amplitud, el cual se limita al llegar a posiciones muy cercanas.

Huelga comentar que esta limitación sólo se tendrá en cuenta en el cálculo de la ganancia, no siendo así para el cálculo de los retardos.

3.4.2. Tratamiento del ángulo θ_n

La solución para el problema derivado de θ_n no es tan sencilla por lo que necesita de un tratamiento más elaborado. Una vez limitada la amplitud debida a la reducción del módulo, nos encontramos con que al acercarse la fuente al array todos los altavoces se silencian progresivamente ya que θ_n se aproxima a los 90º para todos ellos a la vez, así, se da la paradoja de que estando la fuente muy cerca del array no se oye absolutamente nada. Por lo tanto, es necesario asegurar que al menos uno de los altavoces continúe emitiendo sonido, en la práctica, en vez de un único altavoz vamos a emplear dos para evitar el efecto de escalonado que se produce al moverse la fuente y cambiar de forma brusca de un altavoz a otro.

Para evitar este efecto adverso es necesario sustituir el término θ_n de la señal *driving* por un término de ganancia g, el cual será distinto y complementario para los dos altavoces, por lo que, en realidad, estaremos aplicando un *panning* de amplitud. El problema reside ahora en determinar qué ley debe regir este *panning* pues las leyes típicas no sirven como se verá a continuación.

En un *panning* convencional la ganancia aplicada a cada altavoz debe ser proporcional a las distancias horizontales d_L y d_R desde la fuente a cada uno de ellos (véase la Figura 3.14), según las ecuaciones:

$$g_L = \frac{d_R}{\Delta x}, \quad g_R = \frac{d_L}{\Delta x} \tag{3.7}$$

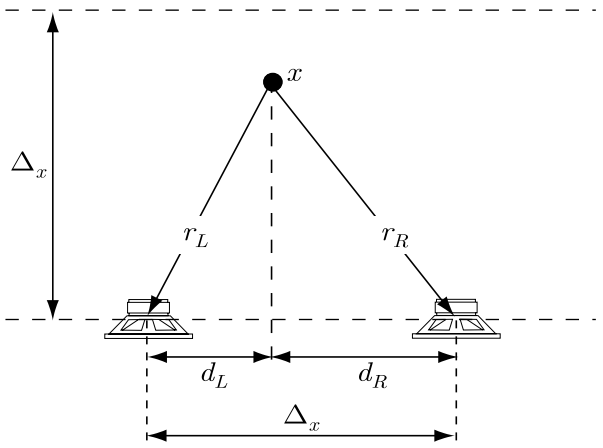


Figura 3.14. Parámetros para el cálculo del *panning* en fuentes cercanas.

Si aplicamos este método en determinadas situaciones se produce un sonido inadecuado, por ejemplo, al acercarse una fuente de forma perpendicular al segmento de altavoces justo por la posición del altavoz izquierdo, en cuanto se supera el umbral el altavoz derecho se silencia bruscamente ya que d_L es cero. Por tanto, el *panning* no solo debe tener en cuenta la distancia horizontal sino también la vertical, esto se consigue empleando como valores de ponderación las distancias r_L y r_R, por lo que la ley que regirá el *panning* queda de la siguiente manera:

$$g_L = \frac{r_R}{r_L + r_R}, \quad g_R = \frac{r_L}{r_L + r_R} \tag{3.8}$$

Como se verá en la siguiente sección, aplicando esta ley se consigue una transición progresiva sea cual sea la trayectoria de acercamiento de la fuente al array.

3.4.3. Discusión

Una vez descrito el algoritmo propuesto para el tratamiento de las fuentes cercanas al array es necesario demostrar su funcionamiento, para lo cual se muestran los valores de la amplitud corregidos en la Figura 3.15. El caso b) muestra como queda el factor que sustituye al $cos(\theta_n)$ mientras el c) muestra el factor correspondiente al módulo. El resultado final se puede apreciar en el caso d).

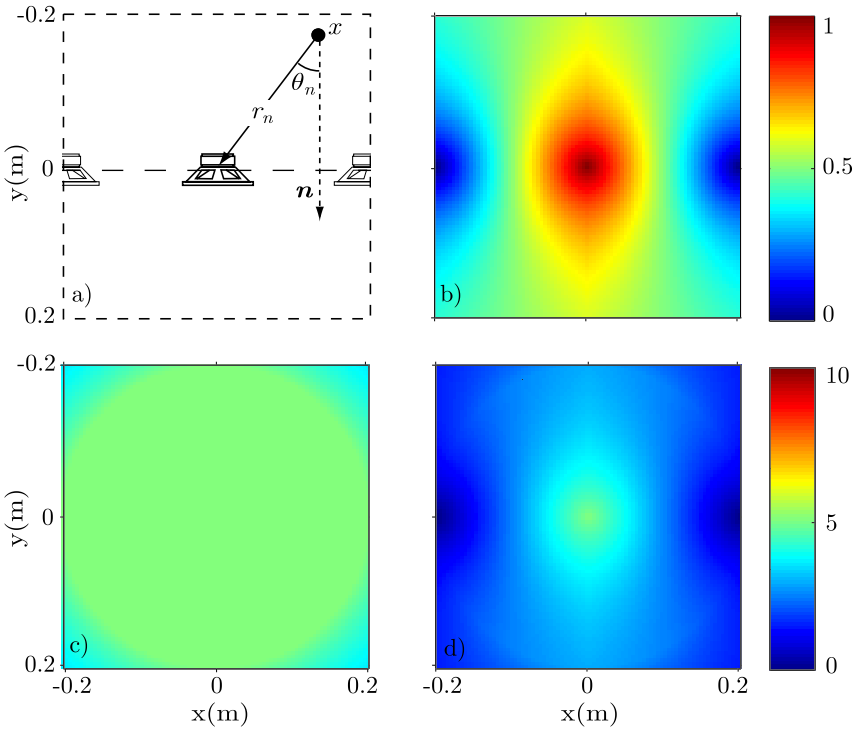


Figura 3.15. Variación de los parámetros que modulan la amplitud de las fuentes cercanas a los altavoces. a) Diagrama descriptivo de las variables implicadas. b) Factor de ganancia sustitutivo del ángulo. c) Factor de ganancia sustitutivo del módulo. d) Combinación resultante de ambos factores. Los factores de ganancia están en escala lineal.

Antes de pasar a su descripción es necesario comentar que las distintas gráficas representan únicamente la ganancia del altavoz que se encuentra en el centro de la superficie, los altavoces laterales tienen sus propios factores, los cuales son idénticos pero desplazados en el espacio.

Comenzando por la caso c) se puede comprobar la limitación aplicada sobre el factor de amplitud, quedando marcado un círculo alrededor del punto central que determina la posición del altavoz. El caso b) es algo más complejo de visualizar, pero si se tiene en cuenta que al estar tan cerca se va a realizar un *panning* entre los dos altavoces, es normal que el factor de ganancia crezca en el centro y tienda a anularse en los extremos, los cuales representan la posición de altavoces laterales. Al combinar ambos factores obtenemos el resultado que muestra el caso d), en el cual se comprueba cómo las fluctuaciones de la ganancia han quedado muy reducidas. Es necesario notar que, la ganancia no debe ser constante en toda la superficie, pues tiene que poder notarse el movimiento de la fuente al acercarse a uno u otro altavoz.

El correcto funcionamiento (además en tiempo real) del método propuesto ha quedado demostrado mediante diversas pruebas realizadas con los distintos prototipos en las que se han ido desplazando fuentes próximas al array sin que aparezcan artefactos sonoros audibles, clicks u otras degradaciones apreciables.

3.5. Fuentes focalizadas

Una de las principales aportaciones que introduce WFS es la posibilidad de crear objetos sonoros delante de los altavoces, es decir, dentro de la sala. Por el momento, WFS es el único sistema existente capaz de sintetizar fuentes en el interior de la sala. En [Ahrens and Spors, 2008a] se presen-

ta una propuesta que evalúa la posibilidad de utilizar fuentes focalizadas en *Higher Order Ambisonics*, aunque resulta muy complejo y no existen pruebas concluyentes de su funcionamiento.

Como ya se ha comentado anteriormente, a este tipo de fuentes sonoras que se encuentran entre el oyente y los altavoces se las denomina fuentes focalizadas (*focused sources*).

3.5.1. Fundamentos

Las fuentes focalizadas no están previstas por la teoría de WFS, ya que según esta, todas las fuentes sonoras deben residir fuera de la superficie que hace las funciones de sala. Sin embargo, es posible simularlas fácilmente con solo invertir los retardos aplicados a las señales.

El problema que introducen este tipo de fuentes es que son anticausales. Si para simular que una fuente está detrás del array es necesario aplicar un retardo en función de la distancia, para simular que está delante el retardo será negativo por lo que se convierte en un adelanto, es decir, necesitamos predecir el futuro. Como esto no es posible es necesario aplicar un retardo global a toda la escena para tener un pequeño margen de maniobra. Este retardo dependerá del tamaño del array, en concreto, de la distancia existente entre los dos altavoces más alejados entre sí.

La denominación de «focalizadas» que se le aplica a estas fuentes proviene de su propio funcionamiento. El sonido se emite desde cada uno de los altavoces del array y se dirige hasta la posición de la fuente, posición en la cual convergen todos los frentes de onda creando un foco (véase el caso a) de la Figura 3.16) a partir del cual surge el campo sonoro correcto (caso b) de la misma figura). Por tanto, en la zona comprendida entre el array y la fuente no se consigue sintetizar el campo sonoro de forma correcta, así,

cuanto más adentrada esté la fuente en la sala menor será el área correcta en la reproducción.

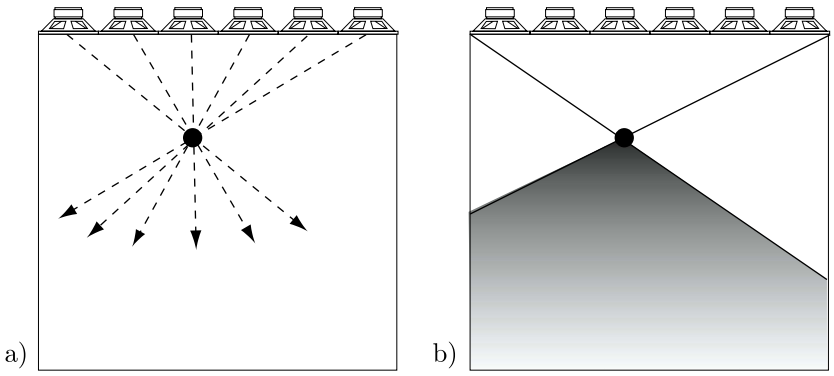


Figura 3.16. Fuentes focalizadas: a) Focalización, b) Área de reproducción correcta.

Otro aspecto a tener en cuenta es la variación de la amplitud para este tipo de fuentes. Es lógico que una fuente situada dentro de la sala tenga mayor nivel que una situada fuera, pues, en principio está mas cerca de los oyentes. El problema radica aquí en que hay que poner un límite a esta ganancia, pues, dependiendo de la configuración del array, puede crecer desmesuradamente. En general, la solución aplicada es imponer una ganancia tope, la cual se alcanza al llegar al centro de la sala.

Acabamos de repasar los fundamentos que permiten sintetizar fuentes focalizadas en WFS. Sin embargo, esta aproximación teórica, no incluye aspectos tan críticos como la activación de los diferentes segmentos en arrays complejos, así como aspectos relacionados con la percepción humana.

A continuación se describirá el tratamiento que dan otros programas de WFS a este tipo de fuentes para seguidamente proponer una técnica mejorada.

3.5.2. Síntesis de fuentes focalizadas

En la literatura existente se han hecho diversos estudios sobre la síntesis de fuentes focalizadas [Caulkins et al., 2003; Spors and Ahrens, 2009], algunos de ellos muy recientes, sin embargo, el único estudio público en el que se indica la conveniencia de seleccionar unos u otros altavoces para ser activados es [Spors, 2007b], por lo que el único método que se ha venido empleando para la síntesis de fuentes focalizadas es el de la línea de referencia, el mismo que se empleaba para fuentes no focalizadas.

Tal y como ya sucedía para las fuentes no focalizadas, no siempre es conveniente activar todos los altavoces que indica la línea de referencia ya que algunos de los altavoces que deben emplearse generan un campo sonoro que viaja en dirección contraria al que generaría la fuente virtual. Por ello se ha llevado a cabo un estudio y se ha propuesto un método alternativo que se verá a continuación.

Un inconveniente adicional del uso de la línea de referencia para sintetizar fuentes focalizadas aparece cuando la fuente se mueve cerca del centro de la sala. En este caso, la línea puede girar de forma descontrolada creando un efecto completamente molesto, pues se encienden y apagan los altavoces casi de forma aleatoria. Es de suponer que los sistemas que implementen este método tendrán un caso especial para tratar las fuentes se acerquen demasiado al centro de la sala.

3.5.3. Técnica del cono de apertura

A continuación se propone un método empírico que mejora la percepción de las fuentes focalizadas, en concreto, el aspecto que debemos aclarar es qué altavoces es necesario activar para realizar una buena simulación, eliminando aquellos altavoces que emiten un frente de onda contrario. La

técnica de la visibilidad no sirve para solucionar el problema, pues al entrar dentro de la sala, ningún altavoz está «visible», de ahí que en la Sección 3.3 se hiciera un inciso.

Una primera aproximación a la solución pasa por activar todo el array simultáneamente, con ello se consigue una simulación correcta pero se produce un cambio muy brusco al pasar la fuente del exterior al interior de la sala y viceversa, ya que estando fuera se emplean muy pocos altavoces y al entrar se encienden todos a la vez, además de que existen multitud de altavoces que emiten sonido en dirección contraria a la deseada.

Por tanto se ha optado por un encendido progresivo de los altavoces conforme la fuente se adentra en la sala, el método de cálculo de qué altavoces emplear es el siguiente: alrededor de la fuente se creará un cono, el cual parte del centro de la sala, con una apertura comprendida entre una apertura mínima α y la total (360^o), los altavoces abarcados por el cono serán los altavoces empleados (véase la Figura 3.17).

La apertura mínima es aquella que abarca dos altavoces, ya que únicamente se emplean dos altavoces al atravesar una fuente el array. Esta apertura angular $|\alpha|$ no es una constante pues depende del tamaño del array y de la separación entre altavoces. Cuando una fuente se encuentre muy cercana al array se empleara un ancho de cono de α^o ($\alpha/2$ a cada lado de la fuente), empleándose así únicamente dos altavoces para generar su sonido, exactamente los mismos que se emplean al atravesar la fuente el array. Conforme la fuente se adentre en la sala el ancho del cono irá aumentando progresivamente hasta llegar a englobar a todos los altavoces.

Para que el efecto producido por el empleo de más o menos altavoces no sea muy acusado, sobre todo en arrays de grandes dimensiones y cuando los oyentes no están centrados en la sala, la apertura completa se debe

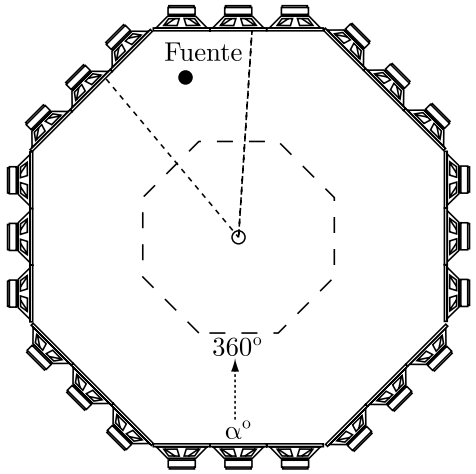


Figura 3.17. Fuentes focalizadas, zonas para el cono de apertura.

conseguir mucho antes de llegar al centro de la sala. Es decir, en vez de ser un único punto el que tenga una apertura de 360^o será toda una zona la que emplee esta apertura.

El cálculo de esta zona es dependiente de la forma del array, en realidad será una versión escalada y centrada de la figura formada por los segmentos del array. Por ejemplo, en la Figura 3.17, es un octógono centrado idéntico al array pero de la mitad de tamaño.

El valor de la apertura variará linealmente con la profundidad de la fuente en la sala. Como la geometría del array puede ser arbitraria, el cálculo de la profundidad puede parecer complicado, pero se puede determinar fácilmente mediante el procedimiento que se describe a continuación (véase la Figura 3.18).

En primer lugar es necesario determinar delante de qué segmento nos encontramos. La forma de averiguarlo es la siguiente: trazaremos líneas desde el centro de la sala a los dos extremos de cada segmento, aquel segmento

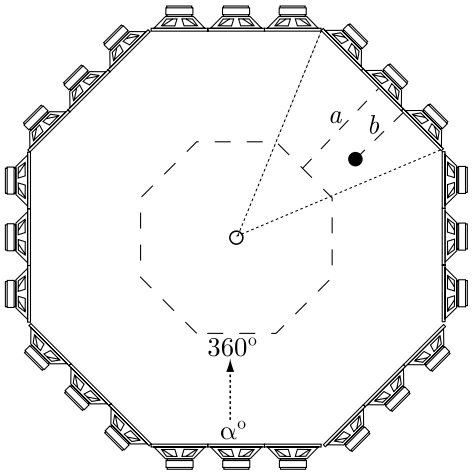


Figura 3.18. Cálculo de la apertura del cono.

en el que la fuente virtual se encuentre a la izquierda de una de las líneas y a la derecha de la otra, es el segmento elegido. En la Figura 3.18 el segmento es el delimitado por las dos líneas de puntos que parten del centro de la sala.

Una vez localizado el segmento debemos calcular dos distancias a y b, las cuales están descritas gráficamente en la Figura 3.18 y se definen a continuación:

- a es la longitud del segmento perpendicular al segmento del array que parte desde la zona de 360^o (el punto de origen es indistinto pues los segmentos son paralelos).

- b es la longitud del segmento perpendicular al segmento del array que parte desde la fuente.

El ángulo de apertura se puede obtener fácilmente haciendo una proporción entre las distancias a y b. Cuando b es nula el ángulo de apertura será α^o y cuando alcanza el valor de a la apertura será de 360^o.

El efecto que se consigue con esta forma de activar los altavoces es una reducción del *aliasing* espacial inherente a las fuentes focalizadas, lo que permite mejorar la localización de estas fuentes, como se verá en el siguiente apartado.

3.5.4. Técnica de la intensidad

Recientemente, en [Spors, 2007b] se amplió la técnica de la intensidad acústica [Spors, 2007a] para que tuviera en cuenta el caso de las fuentes focalizadas.

El funcionamiento de la técnica de la intensidad acústica se ha mostrado en la Sección 3.3.3, y su funcionamiento para fuentes focalizadas es prácticamente idéntico. En este caso se determina el vector de intensidad acústica que induce cada fuente secundaria sobre la posición de la fuente focalizada, no sobre las fuentes secundarias (altavoces). El problema radica ahora en como definir el vector normal, que antes quedaba definido directamente por la superficie de reproducción. Para este caso, Spors propone trazar una línea recta que pase por la posición de la fuente focalizada y emplear la normal a esta recta. La inclinación de esta recta es arbitraria, y variará conforme se mueva la fuente, pero, en general, debería procurar que su normal se dirija hacia algún punto concreto del espacio, preferiblemente el centro de la sala. Esto se describe en la Figura 3.19.

La ventana de selección de altavoces se pude expresar empleando la intensidad (recordemos que debe ser promediada):

$$a(\boldsymbol{x}_0) = \begin{cases} 1 & Si\ \langle \overline{\boldsymbol{I}}_0(\boldsymbol{x}_{fs}, w), \boldsymbol{n}_{fs} \rangle > 0 \\ 0 & En\ otro\ caso \end{cases} \tag{3.9}$$

donde $\langle \cdot, \cdot \rangle$ representa el producto escalar de dos vectores.

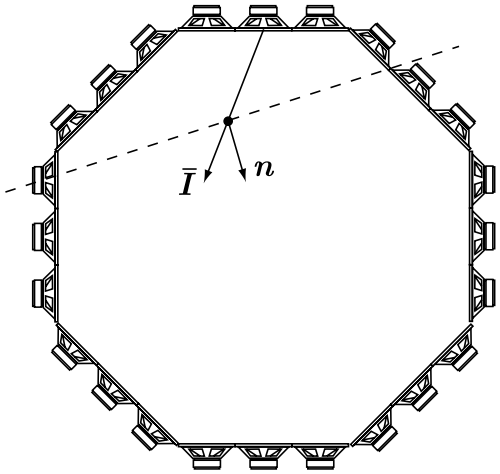


Figura 3.19. Selección de los altavoces activos por el método de la intensidad.

Como en el caso anterior, si el ángulo formado entre el vector intensidad acústica y la normal es agudo, se selecciona el altavoz. En la práctica, esto implica que todos los altavoces que se encuentren detrás de la línea trazada deben ser activados, esto es, viene a ser lo mismo que la línea de referencia, pero en este caso la línea no pasa por el centro de la sala sino por la posición de la fuente focalizada.

La técnica de la intensidad permite modificar de forma dinámica la inclinación de la línea de referencia que emplea en la selección, esto en principio permite que se pueda ajustar cada fuente virtual por separado con el fin de que su reproducción sea más correcta en un área u otra de la sala [Spors, 2007b].

3.5.5. Comparación y discusión

Una vez descrito el funcionamiento de la técnica propuesta para sintetizar las fuentes focalizadas se verá su comparación con la técnica de la

línea de referencia y la intensidad.

En la síntesis de las fuentes focalizadas, cada altavoz empleado genera un frente de ondas que se fundirá con el resto en la posición de la fuente virtual, a partir de la cual se recompone el campo deseado. El problema que nos encontramos al sintetizar estas fuentes es que, debido a la inversión de los retardos temporales, los frentes de onda de los altavoces lejanos comienzan a atravesar la sala antes de que la fuente converja, produciendo *aliasing* espacial el cual repercute en mayores errores en la localización. Este efecto se puede apreciar en el caso a) de la Figura 3.20, en la cual se representa una instantánea del frente de ondas de una señal tipo pulso, la cual debe focalizar en mitad de la sala, el *aliasing* espacial se puede apreciar representado por las líneas amarillas y anaranjadas que viajan por delante del frente de ondas, el cual se puede ver en rojo.

En los diferentes casos de la Figura 3.20 se aprecia como se reduce progresivamente el *aliasing* espacial al emplear cada vez menos altavoces en la síntesis, de 32 se pasa a 16, luego a 8 y por último a 4. En contraposición a la reducción del *aliasing* espacial, conforme se reduce el número de altavoces, el frente de ondas original va perdiendo curvatura, lo que implica una reducción progresiva del área de escucha correcta [Spors, 2007b], efecto que se acentúa conforme la fuente se adentra en la sala. De ahí, que la técnica del cono de apertura disminuya la selección de altavoces cerca del array y la aumente al adentrarse en la sala, manteniendo de esta forma la forma del frente de ondas, y por tanto el área de reproducción correcta, y reduciendo en la medida de lo posible el *aliasing* espacial.

En la Figura 3.21 se muestra un ejemplo del error cometido en la síntesis del campo de presiones. Como se puede apreciar fácilmente, el campo de presiones generado por el cono de apertura es mucho más uniforme que el

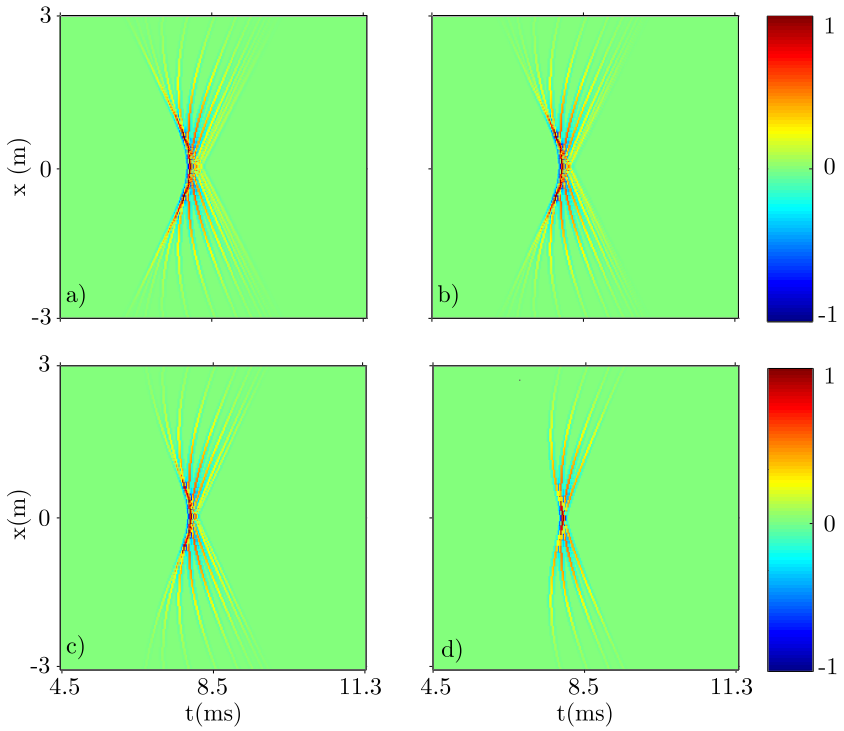


Figura 3.20. Simulación del campo sonoro emitido por un array lineal al simular una fuente focalizada. a) Empleando 32 altavoces. b) Empleando 16 altavoces. c) Empleando 8 altavoces. d) Empleando 4 altavoces. El array está compuesto por 32 altavoces, $\Delta x = 0{,}18m$, y $x_0 = [0, -1]m$.

generado por el algoritmo de la línea de referencia. El algoritmo propuesto minimiza el conjunto de altavoces que radian en sentido contrario para la mayor parte de la sala, reduciendo drásticamente el *aliasing* espacial. La técnica de la línea de referencia no tiene en cuenta esto y por eso se produce un patrón de interferencias acusado a lo largo de toda la sala.

La comparación de la técnica del cono de apertura con la técnica de la intensidad no es tan trivial, pues ambos métodos pueden dar tanto resul-

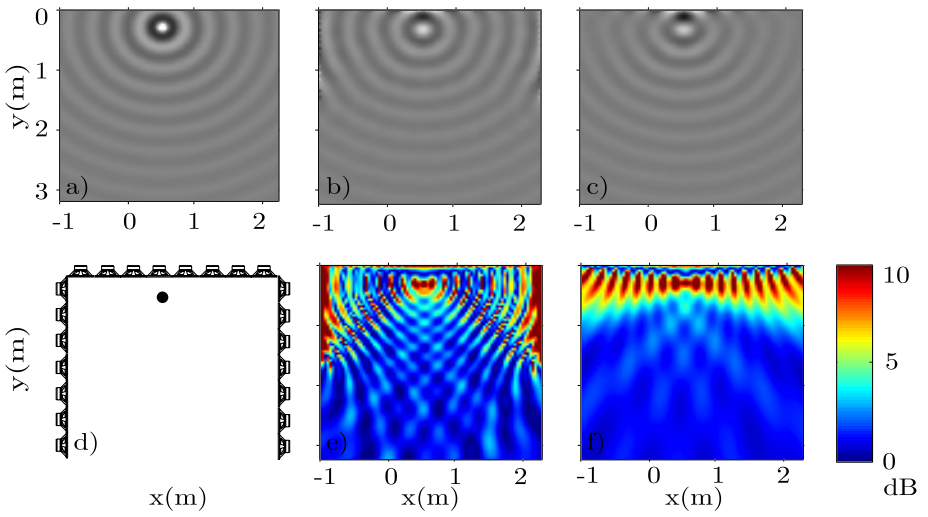


Figura 3.21. Simulación numérica del error cometido al sintetizar fuentes focalizadas. a) Campo de presión de una fuente en el interior de la sala. b) Campo de presión sintetizado por WFS, línea de referencia. c) Campo de presión sintetizado por WFS, algoritmo mejorado. d) Diagrama con el formato del array. e) Error cometido, línea de referencia. f) Error cometido, cono de apertura.

tados parecidos como totalmente dispares, al poder cambiar la inclinación de la línea de referencia a voluntad.

En general, si se mantiene la normal de la línea de referencia en dirección al centro, ambos algoritmos se comportan de forma muy similar cuando la fuente focalizada no se encuentra muy adentrada en la sala, aunque el comportamiento depende en gran medida de la geometría del array. Sin embargo, al adentrarse la fuente en la sala, el cono de apertura selecciona rápidamente más altavoces abarcándolos todos rápidamente. En el caso hipotético en que la fuente focalizada se mueva por el centro de la sala y la normal apunte al centro, la técnica de la intensidad puede proporcio-

nar resultados totalmente descontrolados al girar bruscamente la línea de referencia, y seleccionar los altavoces de forma prácticamente aleatoria.

En todos los casos es necesario aplicar la ventana de *tapering* para reducir la difracción en los extremos de la selección de altavoces aunque con ello se reduzca el área de reproducción correcta [Spors, 2007b].

Según los test realizados con el prototipo, el cono de apertura permite determinar con mayor precisión la posición de las fuentes focalizadas cuando se encuentran relativamente cerca del array, gracias a la notable reducción del aliasing espacial. En el peor de los casos, cuando la fuente está muy adentrada en la sala se comporta de igual forma que al encender todos los altavoces.

3.6. Fuentes en movimiento

El desarrollo teórico visto en la Sección 2.5 nos permite determinar la señal que debe emitir cada altavoz (*driving signal*), sin embargo, este desarrollo mantiene una visión estática y las escenas sonoras son de naturaleza dinámica. En general, las escenas deben evolucionar con el tiempo, las fuentes sonoras cambiarán tanto de contenido como de posición, por tanto, si deseamos recrear una escena real, es necesario incluir todos estos matices. Recientemente en [Ahrens and Spors, 2008b] se ha formalizado la definición de las fuentes sonoras en movimiento, sin embargo, el trabajo aquí mostrado es anterior [Bleda et al., 2005] y no tiene en cuenta esta definición.

Para recrear una escena mediante WFS el sonido emitido por una fuente sonora es modificado y emitido por cada una de las fuentes secundarias (altavoces) de acuerdo con la ecuación (2.20). Sin embargo, esta ecuación no tiene en cuenta el posible movimiento de la fuente por lo que, para si-

mularlo, la única opción es cambiar la posición de la fuente a intervalos regulares, obteniendo de esta forma un conjunto de funciones *driving* para cada punto de la trayectoria. El procedimiento es análogo al que se aplica en el cine para conseguir un vídeo en movimiento a partir de muchas instantáneas. Aún así, nos encontramos con una serie de problemas que deberemos solucionar y para ello es necesario relegar a un segundo plano la teoría de WFS ya que no los tiene en cuenta.

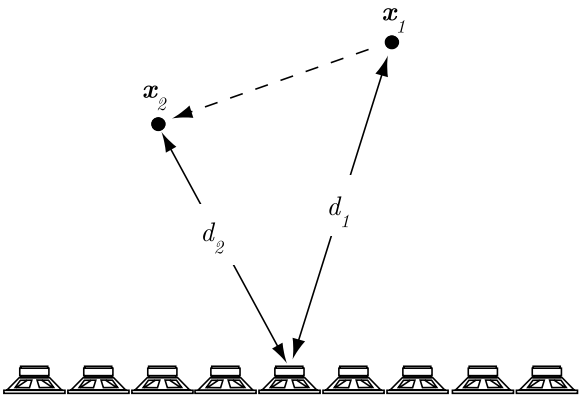


Figura 3.22. Movimiento de una fuente y distancias implicadas.

Siguiendo el esquema presentado en la Figura 3.22, podemos ver que para el altavoz indicado, tendremos dos funciones *driving*, una para la posición inicial $\boldsymbol{x}_1$ y otra para la final $\boldsymbol{x}_2$, las cuales serán distintas al haber cambiado, entre otras cosas, la distancia fuente-altavoz. Al reproducir dos bloques de sonido consecutivos, cada uno con la función *driving* correspondiente, nos encontramos con una serie de problemas que pasamos a detallar:

- Se produce un sonido entrecortado al no coincidir los retardos introducidos por cada función *driving*, los cuales son dependientes de la distancia.

- Se produce una modulación en amplitud, ya que la ganancia también se ve afectada por el cambio en la distancia y el ángulo relativo fuente/altavoz. Esta modulación produce una distorsión que puede llegar a ser perfectamente audible dependiendo del sonido.

- No estamos aplicando la modulación en frecuencia que debería producir el efecto *doppler*, el cual se debe tener en cuenta siempre que hay movimiento.

El sonido entrecortado que se produce al variar los retardos introducidos en la función *driving* es una consecuencia de obviar el efecto *doppler* en el tratamiento del sonido, por tanto, el primer y el último de los problemas descritos están enlazados. Así, es necesario realizar un estudio de estos problemas y dar una solución eficiente a cada uno de ellos.

3.6.1. El efecto *doppler*

Cuando una fuente sonora se acerca o aleja de un oyente, el frente de ondas emitido se comprime o expande respectivamente, produciendo una variación en la frecuencia de la señal percibida por el oyente, a todos estos efectos es a lo que se conoce como el efecto *doppler* [Doppler, 1842]. La expresión matemática de esta variación se puede encontrar en cualquier libro de fundamentos de acústica, por ejemplo en [Pierce, 1991], y está descrita por la ecuación

$$f_o = f_f \left(\frac{1 + \frac{v_o}{c}}{1 + \frac{v_f}{c}} \right). \tag{3.10}$$

En la cual, f_f es la frecuencia de la señal emitida por la fuente, f_o la frecuencia percibida por el oyente, c es la velocidad del sonido en el aire (340 m/s), v_o es la velocidad relativa del oyente en la dirección de la fuente

y v_f es la velocidad relativa de la fuente en la dirección del oyente.

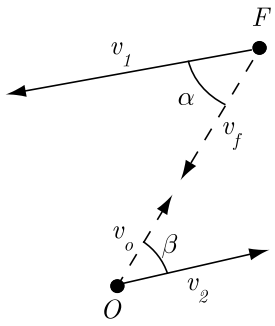


Figura 3.23. Velocidades relativas en el efecto *doppler*.

Para determinar las velocidades relativas es necesario realizar el cálculo descrito en la Figura 3.23. Si la fuente F se desplaza con velocidad v_1 y el oyente O con velocidad v_2, es posible obtener las velocidades relativas mediante un sencillo cálculo trigonométrico:

$$\begin{aligned} v_f &= v_1 \cos|\alpha| \\ v_o &= v_2 \cos|\beta| \end{aligned} \tag{3.11}$$

Debe tenerse en cuenta que aunque la fuente y el oyente mantengan una velocidad constante, la variación en la frecuencia no tiene por qué serlo ya que depende de los ángulos α y β, los cuales variarán con el tiempo.

Como podemos ver, este efecto se produce tanto si se mueve la fuente como el oyente, pero para el problema que estamos tratando el oyente está representado por cada una de las fuentes secundarias (altavoces), las cuales siempre van a permanecer estáticas por lo que la ecuación se puede reducir a:

$$f_o = f_f \left(\frac{1}{1 + \frac{v_f}{c}} \right) \tag{3.12}$$

Así, si tratamos cada fuente virtual por separado, tendremos una única fuente en movimiento y múltiples oyentes estáticos. Para el resto del texto siempre se entenderá que el único elemento que se va a mover es la fuente, permaneciendo estático el oyente.

3.6.2. Variación del retardo: modulación en frecuencia

Una forma sencilla de simular el efecto *doppler* se consigue empleando una línea de retardo variable [Smith et al., 2002]. Recuérdese que ya se está empleando una para simular los retardos de las señales *driving* para fuentes estáticas (véase la Sección 3.2). Si la actualización de los punteros de lectura y escritura se realizan muestra a muestra, únicamente se consigue un retardo, pero si la actualización de los punteros se realiza con valores superiores o inferiores a una muestra (interpolando las muestras necesarias), se consigue una modulación en frecuencia de la señal que es equivalente a la producida por el efecto *doppler* si el incremento de los punteros coincide con las velocidad relativas de la fuente.

Según el desarrollo realizado en [Smith et al., 2002], al modificar la velocidad del puntero de escritura se consigue simular la variación producida por el movimiento de la fuente, mientras que la variación de la velocidad de lectura simula el movimiento del oyente. Para simular varias fuentes es necesario emplear varios punteros de escritura y para simular varios oyentes, varios punteros de lectura, aunque la simulación de ambos casos no es posible de forma simultánea con una única línea de retardo.

A continuación se muestra la solución elegida por el autor. En la Figura 3.22 se puede apreciar que al cambiar una fuente de posición la distancia que hay entre ésta y cada fuente secundaria varía. Calculando el tiempo que tarda el sonido en recorrer las distancias d_1 y d_2 obtenemos los retardos t_1

y t_2 que deberemos aplicar en las funciones *driving*.

En la implementación realizada utilizamos una línea de retardo por fuente, y para cada línea un puntero de lectura por fuente secundaria, esto es, un puntero de escritura y varios de lectura. Al contrario de lo propuesto en [Smith et al., 2002], se emplean los punteros de lectura para simular el movimiento de la fuente, no de los oyentes.

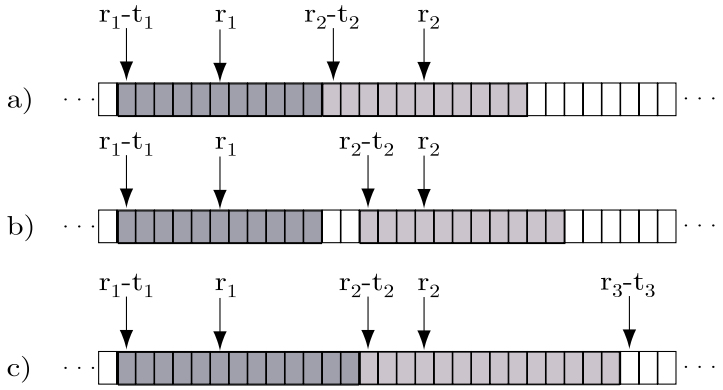


Figura 3.24. Linea de retardo variable: a) Fuente en reposo, b) Fuente en movimiento sin aplicar *doppler*, c) Fuente en movimiento aplicando *doppler*.

Centrándose en el funcionamiento de la línea de retardo se puede ver que el cálculo de la variación de la velocidad de lectura se puede realizar de forma muy eficiente. En la Figura 3.24 está representado el funcionamiento de la línea de retardo para diferentes casos, mostrándose el cálculo de las muestras leídas para dos bloques de datos consecutivos. Si el tamaño de bloque empleado es N y para el primer bloque el puntero de lectura se encuentra en r_1, para el segundo bloque estará en r_2, indicando el comienzo del siguiente bloque, N muestras más allá. Veamos cada caso por separado:

a) Fuente en reposo: el primer bloque de datos se obtiene a partir de

las t_1 muestras previas a la posición del puntero de lectura. Para el segundo bloque, el puntero de lectura ha avanzado hasta r_2, y como la fuente no se ha movido, t_2 coincide con t_1 por lo que el sonido obtenido es continuo.

b) Fuente en movimiento sin *doppler*: la lectura de los dos bloques es idéntica al caso en reposo, pero ahora t_2 no coincide con t_1 por lo que se produce un salto de varias muestras en el sonido, o incluso puede que se repitan muestras ya reproducidas.

c) Fuente en movimiento con *doppler*: para evitar la discontinuidad en el sonido es necesario leer todas las muestras comprendidas entre $r_1 - t_1$ y $r_2 - t_2 - 1$, pero debemos entregar exactamente N muestras y si las leemos de una en una obtendremos muestras de más (o de menos) por lo que debemos incrementar el puntero de lectura de forma fraccionaria según la ecuación:

$$\triangle r = \frac{(r_2 - t_2) - (r_1 - t_1)}{N}, \tag{3.13}$$

siendo necesario interpolar las muestras durante la lectura. Esta ecuación se puede simplificar si tenemos en cuenta que la distancia entre r_1 y r_2 siempre es N, por lo que queda de la siguiente forma:

$$\triangle r = 1 - \frac{t_1 - t_2}{N}. \tag{3.14}$$

Como se puede ver, hemos reducido el cálculo del factor de velocidad a una sencilla operación algebraica, dos sumas y una multiplicación, ya que se precalcula el valor de 1/N, empleando para ello los retardos de dos bloques consecutivos sin necesidad de realizar ningún cálculo trigonométrico. La

interpolación de las muestras se realiza con un interpolador lineal, de forma análoga al empleado para el retardo fraccionario (véase la Sección 3.2.1).

3.6.3. Variación de la ganancia: modulación en amplitud.

Otro de los efectos adversos con los que tenemos que tratar al mover una fuente es la variación de la ganancia que se produce. Continuando con el ejemplo de la Figura 3.22 y teniendo en cuenta que se trabaja por bloques, la señal *driving* de la posición inicial tendrá un factor de amplitud distinto al de la posición final. Este cambio de amplitud se produce al comienzo de un bloque y se mantiene constante hasta el comienzo del siguiente, por lo que claramente se está introduciendo una modulación de la amplitud de la señal emitida.

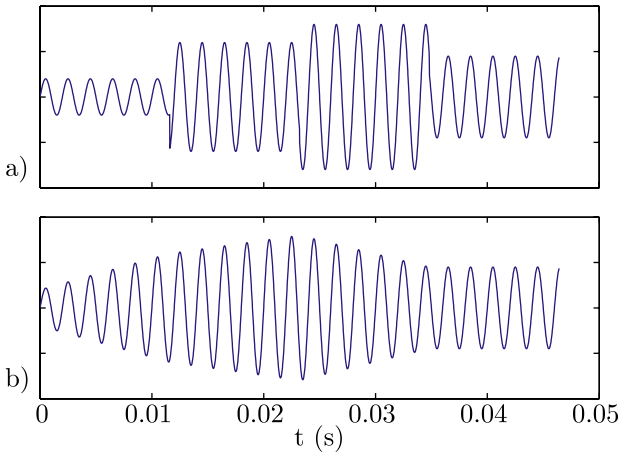


Figura 3.25. Modulación en amplitud producida por el movimiento. a) Escalonada. b) Progresiva.

Ya que la modulación se produce de forma escalonada, puede llegar a ser perfectamente audible y producir un efecto bastante molesto. Esto puede apreciarse en el ejemplo de la Figura 3.25, en el caso a) se distingue

claramente como al cambiar del primer al segundo bloque la variación de la amplitud distorsiona la forma de onda de la señal de forma ostensible y por tanto audible. Una solución sencilla pasa por realizar el cambio de la amplitud de forma progresiva, tal y como se muestra en el caso b).

La implementación realizada modifica la amplitud de forma progresiva interpolando los valores de la ganancia intermedios de forma lineal. Así se consigue proporcionar la modulación adecuada con un mínimo de distorsión de la señal.

3.6.4. Movimiento sin efecto *doppler*

Para terminar con el tema del movimiento entre fuentes, es necesario prever que en determinadas situaciones puede ser beneficioso no aplicar el efecto *doppler* al mover una fuente. Tengamos en cuenta, por ejemplo, las siguientes situaciones:

- Una fuente se mueve lentamente, por ejemplo una persona andando. En este caso el efecto *doppler* es inapreciable por lo que podríamos ahorrar el procesado.

- Una motocicleta grabada con un micrófono en reposo. Esta grabación ya incluye el efecto *doppler*, por tanto, si lo volvemos a aplicar sólo conseguiremos distorsionar el sonido.

- Una fuente que se teletransporta. Ya que no se mueve no debe haber *doppler*, además, si se aplica y las dos posiciones son lejanas puede producirse distorsión por el exceso de velocidad aparente.

- Un cambio en el punto de vista en la escena. Por ejemplo en una película, al cambiar el plano de un personaje a otro, todo cambia aparentemente de posición sin embargo nada se ha movido.

Como vemos, puede ser útil mover una fuente sonora sin aplicar el efecto *doppler*, pero como ya hemos visto (caso b) de la Figura 3.24), esto produce un sonido entrecortado que debemos evitar.

La solución que proponemos para evitar este problema es bastante sencilla y efectiva, y consiste en duplicar la fuente sonora. Ya que la fuente cambia de posición calculamos el sonido que se produce desde cada una de ellas y realizamos un *crossfade*, correspondiendo la parte del *fade-out* al sonido de la posición inicial y la del *fade-in* a la de la posición final.

De esta forma se consigue mover una fuente de forma «estática».

3.6.5. Discusión

La simulación del efecto *doppler* supone un reto importante dentro de un sistema de WFS, pues para conseguir que sea realista debe implementarse de forma cuidadosa y, en general, una buena implementación requiere de un procesado intensivo, el cual no es posible en este marco de trabajo. Debe tenerse en cuenta que este efecto se aplica de forma independiente para cada par fuente-altavoz, lo que eleva de forma considerable la carga del procesador. Por ejemplo, con 10 fuentes y 100 altavoces, puede llegar a tener que calcularse 1000 *dopplers* distintos.

Por otro lado, es necesario hacer constar que si una fuente se moviera a una velocidad excesiva es posible que se produjera *aliasing*, pues no se ha previsto el filtrado *antialiasing* para no aumentar la carga computacional.

En la presente sección se ha mostrado una forma sencilla y sobre todo eficiente de simular el movimiento de las fuentes sonoras, con y sin efecto *doppler*. Por un lado, las diferentes pruebas de escucha realizadas con los prototipos validan el buen funcionamiento de la implementación, por otro, se ha comprobado que un procesador estándar es capaz de simular el movi-

miento de una gran cantidad de fuentes incluso si el array está compuesto de un centenar de altavoces.

3.7. Mejora del *aliasing* espacial

La discretización del frente de ondas secundario previsto por el principio de Huygens da como resultado uno de los problemas de mayor dificultad de solución: el *aliasing* espacial.

El campo sonoro sintetizado se reconstruye al superponerse los campos sonoros generados por cada uno de los altavoces del array. Esta superposición deja de ser correcta a partir de una determinada frecuencia, pues, conforme aumenta la frecuencia disminuye la longitud de onda y, en apariencia, los altavoces están cada vez más separados, de tal forma que el campo recompuesto comienza a tener coloraciones cada vez mayores debidas a efectos de filtrado peine.

Este efecto se acentúa con el movimiento de las fuentes sonoras ya que se modifica el patrón de interferencias generado. Donde más se agudiza es con el movimiento del oyente ya que al moverse va recorriendo el patrón de interferencias.

En la Sección 2.5.4, dentro de los fundamentos teóricos de WFS, se realizó un repaso de las distintas técnicas que existen en la actualidad para intentar paliar el problema del *aliasing* espacial. Ninguna de las técnicas propuestas consigue erradicar el problema y de todas ellas únicamente la técnica OPSI [Wittek, 2002] se puede llevar a la práctica con resultados de mejora sustancial.

En la presente sección se mostrará el funcionamiento de la técnica OPSI para seguidamente introducir una técnica alternativa, propuesta por el autor, que puede rivalizar con la anterior, y en determinadas situaciones

mejorar los resultados.

3.7.1. Técnica OPSI

La técnica OPSI emplea las técnicas clásicas de la imagen *phantom* estéreo para minimizar el efecto del aliasing espacial [Wittek, 2002].

Ya que el espectro de la señal sintetizado por WFS es correcto hasta la frecuencia de *aliasing*, a partir de la cual el espectro se degrada progresivamente, parece lógico dividir este espectro en dos bandas, empleando como límite de división esta frecuencia.

Así nos encontramos con dos bandas de frecuencia, la banda paso bajo se procesa mediante WFS, mientras que la banda paso alto emplea una imagen *phantom* para sintetizar el contenido en alta frecuencia, posteriormente se mezclan ambas bandas y se envían al array. En la Figura 3.26 se muestra un ejemplo del funcionamiento de la técnica OPSI.

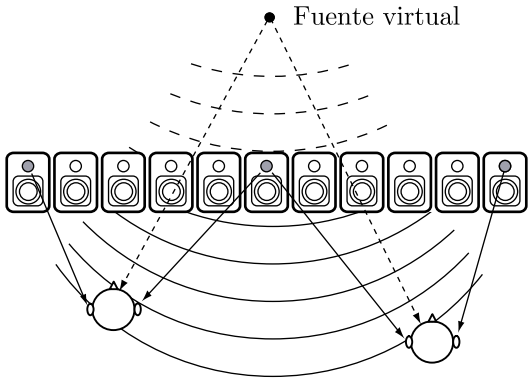


Figura 3.26. Ejemplo de funcionamiento de la técnica OPSI.

En cuanto a localización se refiere, al oído se le entregan dos posiciones distintas. La componente de baja frecuencia del espectro apunta directamente hacia la posición de la fuente virtual, sin embargo, la componente

de alta frecuencia indica otra posición que, dependiendo de la situación del oyente, puede o no coincidir con la de la fuente virtual.

La idea subyacente bajo la técnica OPSI es la siguiente: cuantos más altavoces se empleen en la reproducción de la banda de alta frecuencia mayor será el *aliasing* espacial, puesto que cuantos más altavoces empleemos se creará un patrón de interferencias cada vez más complicado. Por eso la técnica reduce el número de *tweeters* a emplear a un mínimo, con el fin de que el patron de interferencias sea lo más sencillo posible. El campo sintetizado ya no intenta ser fiel al original por encima de la frecuencia de *aliasing*, aunque es necesario tener en cuenta que tampoco era fiel al aplicar WFS con *aliasing*.

Así, la técnica OPSI se limita a emplear tres *tweeters* para fuentes puntuales cercanas al array, mientras que para fuentes de ondas planas emplea un *tweeter* de cada cinco, permitiendo de esta forma mantener la apariencia de onda plana.

3.7.2. Aproximación por subbandas

La aproximación descrita aquí se publicó inicialmente en [López et al., 2005]. Ya que el espectro de la señal sintetizado por WFS es correcto hasta la frecuencia de *aliasing*, a partir de la cual el espectro se degrada progresivamente, parece lógico dividir este espectro en dos bandas con el límite de división esta frecuencia.

La banda paso bajo estará correctamente reconstruida, por lo que no será necesario tratarla, y así dedicaremos todo el esfuerzo sobre la banda paso alto, con problemas de *aliasing*. Como se puede apreciar, esta forma de trabajar ya la utiliza la técnica OPSI, sin embargo el tratamiento que vamos a dar a la banda de alta frecuencia va a ser completamente distinto.

OPSI propone emplear la técnica de la imagen *phantom*, tal y como hace el sistema estéreo, mientras que la aproximación por subbandas propuesta se va a centrar en el *panning* de amplitud, empleando un único altavoz por fuente.

En la Figura 3.28 se puede ver en detalle el diagrama de bloques del procesado a realizar sobre cada fuente sonora. En primer lugar la señal se divide en dos bandas empleando para ello dos filtros de cuarto orden en configuración Linkwitz-Riley [Linkwitz, 1976], el uso de estos filtros asegura una reconstrucción perfecta de la señal en amplitud, y casi perfecta en fase en la zona de transición (véase la Figura 3.27).

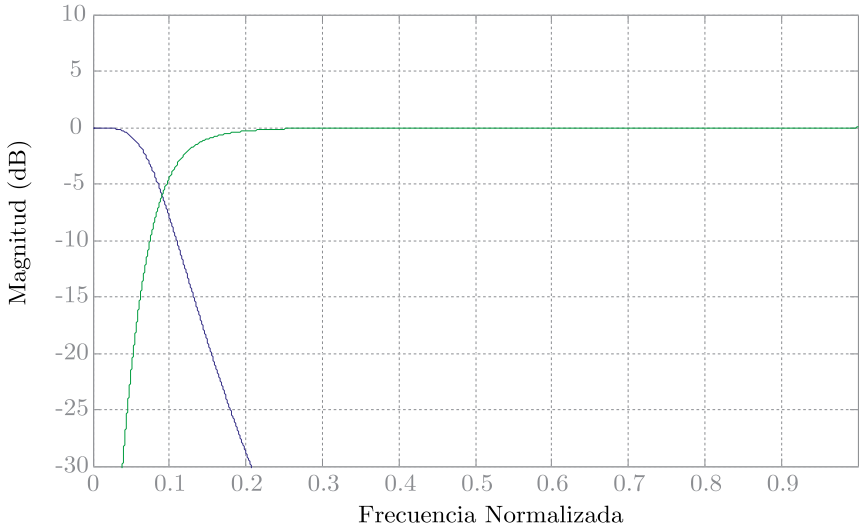


Figura 3.27. Respuesta en frecuencia normalizada de los filtros Linkwitz-Riley.

Después del filtro paso bajo, la señal se procesa mediante el algoritmo de WFS obteniéndose la parte de baja frecuencia de la señal de excitación de cada altavoz del array. Por otra parte, la señal de alta frecuencia se envía a un algoritmo clásico de *panning* ajustando la ganancia para simular

la distancia y compensar el uso de un único altavoz. Posteriormente, en aquellos altavoces seleccionados en el *panning* se mezclan las dos señales obtenidas para cada subbanda y para finalizar el resultado es enviado a cada altavoz, por tanto, el número de salidas del algoritmo será igual al número de altavoces empleados en el array.

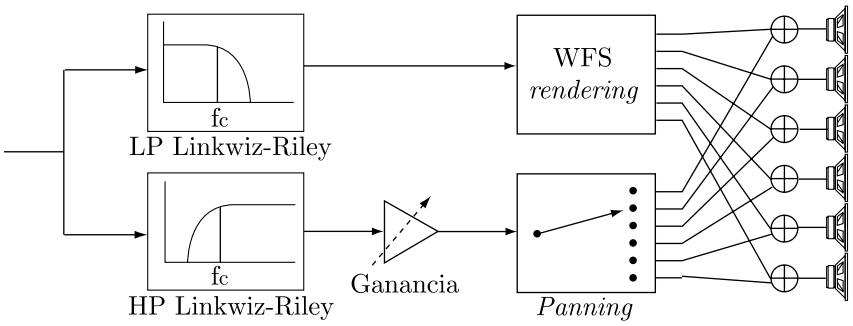


Figura 3.28. Diagrama de bloques de la aproximación por subbandas.

Para que la señal reconstruida permita una correcta localización espacial de la fuente sonora es necesario implementar cuidadosamente el algoritmo de *panning*. En nuestra implementación se ha optado por emplear únicamente un altavoz o, a lo sumo dos si la fuente está en movimiento.

La selección de los altavoces a emplear para reproducir el contenido de alta frecuencia se realiza trazando una línea recta desde el centro de la sala hasta la posición de la fuente virtual. El punto de intersección entre esta línea y el array de altavoces determina los altavoces a emplear, este procedimiento está detallado en la Figura 3.29 a).

En un primer momento se pretendió emplear un único altavoz, el más cercano al punto de intersección, eliminando de esta forma todo efecto de filtrado peine en la parte de altas frecuencias del espectro. Sin embargo, al moverse la fuente se produce un efecto de escalonado muy desagradable

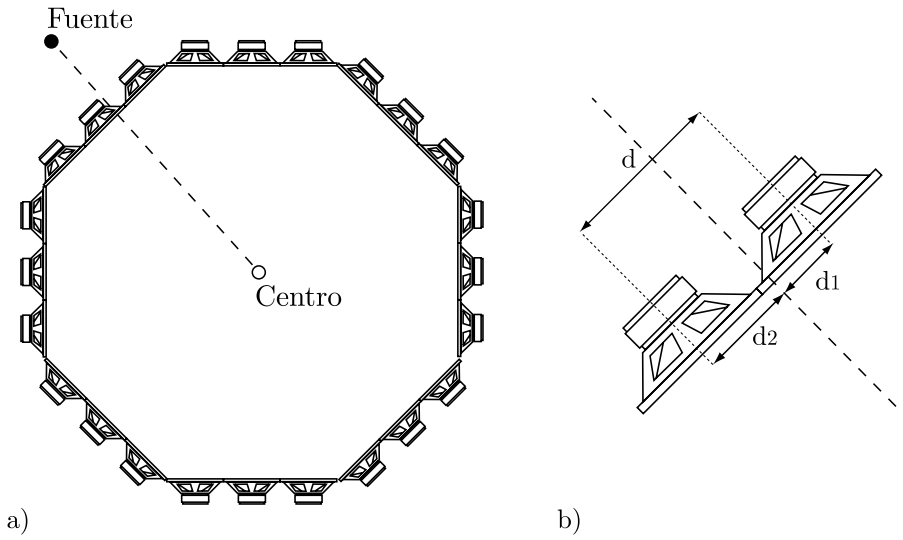


Figura 3.29. Algoritmo de *panning*: a) Elección del altavoz. b) Distancias implicadas en el cálculo de las ganancias.

cuando cambia de un altavoz a otro. Así, para permitir un movimiento suave se decidió emplear dos altavoces. Las ganancias a emplear para cada altavoz se determinan en función de la distancia del punto de intersección al centro de estos, véase el caso b) de la Figura 3.29. Mediante estas distancias se pueden calcular las ganancias de forma sencilla empleando las ecuaciones (3.15) y (3.16).

$$g_1 = \frac{d_2}{d} \tag{3.15}$$

$$g_2 = \frac{d_1}{d} \tag{3.16}$$

Siguiendo esta estrategia el contenido en baja frecuencia determina una posición exacta de la fuente mientras que el contenido en alta frecuencia indica otra posición, la del altavoz empleado en el algoritmo del *panning*. Tal y como se expresa en la Figura 3.30 existe cierto error angular en la

componente de altas frecuencias del espectro, este error está representado por los ángulos e en la figura. Este error es nulo si el oyente se encuentra justo en el centro de la sala, pero conforme nos alejamos de este punto y nos acercamos al array el error se puede acentuar, dependiendo del par de posiciones fuente/oyente.

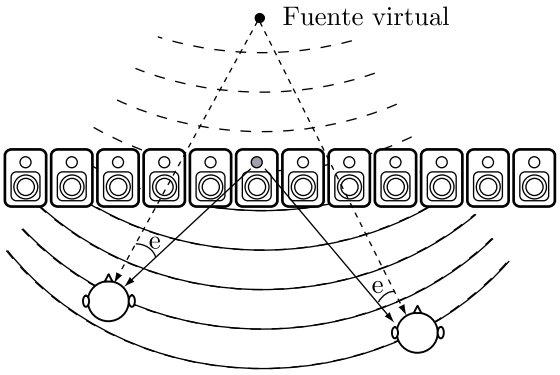


Figura 3.30. Ejemplo de aplicación de la aproximación por subbandas.

En diversos informales test realizados con el prototipo, en general el error angular cometido no suele ser apreciable salvo si el oyente se encuentra situado muy cerca del array. Es necesario hacer notar que estas posiciones no son recomendables ni siquiera para las bajas frecuencias pues se produce un efecto de proximidad que distorsiona la percepción de la escena por completo. También debe tenerse en cuenta que cuando la fuente entra dentro de la sala el error de localización puede hacer impracticable el uso de esta técnica. Mientras la fuente permanezca cerca del array, aunque esté dentro de la sala, el funcionamiento será el esperado, pero conforme se adentre en la sala el error crecerá, llegando a ser total para aquellos oyentes que se encuentren entre la posición de la fuente y el array. Por tanto, se desaconseja el uso de esta técnica para fuentes focalizadas. Así, para este

tipo de fuentes se propone hacer una transición suave entre la aproximación por subbandas y el tratamiento estándar de WFS.

Como contrapartida al error angular de localización obtenemos ciertos beneficios. Siempre en base a los test realizados con el prototipo, podemos afirmar que el efecto de filtrado peine desaparece incluso si el oyente se mueve por la sala, situación en la que más se acentúa este efecto adverso. Otro efecto beneficioso es que las fuentes sonoras con un contenido espectral elevado en altas frecuencias no crecen de tamaño ya que están focalizadas sobre los dos altavoces empleados por el *panning*.

3.7.3. Discusión

El efecto negativo del *aliasing* espacial es quizás uno de los temas más difíciles de solucionar en un sistema de WFS ya que proviene de una limitación física relacionada con la separación de altavoces. Aunque este problema se menciona constantemente en los artículos científicos existen escasas referencias que aporten una solución al problema, que es considerado de menor importancia por muchos autores [Boone et al., 1995].

Una de las técnicas más interesantes para minimizar los efectos del aliasing es el método OPSI. Esta técnica intenta reducir el aliasing especial mediante la combinación de la ecuación de WFS a bajas frecuencias y el método de la imagen *phantom* para altas frecuencias. El inconveniente fundamental de esta técnica radica en que al basarse en la imagen *phantom* depende directamente de la posición del oyente, funcionando mejor en unas posiciones del oyente que en otras.

La técnica propuesta por el autor, la aproximación por subbandas, tampoco soluciona el *aliasing*, únicamente reduce sus efectos nocivos. Al igual que sucede con la técnica OPSI, el resultado depende de la posición

del oyente y de la posición de la fuente virtual a simular. Sin embargo, hay situaciones en las que su uso es muy beneficioso, por ejemplo cuando los oyentes se encuentran relativamente centrados en la sala, o cuando las fuentes virtuales se sitúan cerca del array. En estas situaciones, el uso de la aproximación por subbandas permite localizar las fuentes de forma mucho más precisa a la vez que consigue erradicar el efecto del filtrado peine que se puede apreciar con el movimiento del oyente por la sala. Un punto en su contra es que no puede ser empleada para la simulación de fuentes de ondas planas, ya que concentra las altas frecuencias en un punto del array.

El método propuesto ha sido implementado en tiempo real y testeado en el prototipo construido en el laboratorio. Así mismo, se ha implementado el método OPSI y se ha comparado in-situ con el propuesto.

Se han realizado test informales por paneles de expertos (básicamente todo el equipo de personas del grupo de investigación de audio) comparando de manera no formal un sistema WFS puro, uno con OPSI y otro con la aproximación por subbandas, tanto para fuentes estáticas como en movimiento. Las opiniones informales vertidas por los sujetos no son del todo claras. Con estos experimentos se intuyen algunos efectos de los anticipados por la teoría, pero no permiten afirmar que un método sea superior, ni en la mayoría de situaciones ni en una en particular.

Sería necesario planificar una batería de test relativamente complejos y que se prevén de bastante duración y realizarlos con un jurado extenso y compuesto de individuos de toda índole, no solo expertos en audio. Este trabajo queda abierto en esta tesis y podría ser una línea futura de trabajo.

En cualquier caso, se puede concluir que, hasta el momento, no hay ninguna técnica definitiva para solucionar el problema del *aliasing* espacial. Por lo que resulta conveniente que el sistema de reproducción sea capaz de

usar varias técnicas y emplear la más conveniente en cada situación.

3.8. Conclusiones

Durante el desarrollo del presente capítulo se han ido repasando diversos temas pendientes o sobre los que no existe información de dominio público para la implementación práctica de sistemas WFS. Sobre cada punto se ha ido indicando la problemática encontrada y las soluciones particulares aportadas.

En cuanto a la necesidad de aplicar retardos fraccionarios a la señal se ha previsto el uso de un filtro interpolador FIR de primer orden, ya que permite realizar la interpolación de forma muy sencilla y eficiente, siendo esta última característica primordial para implementar un sistema de WFS.

En segundo lugar se ha descrito la problemática que existe a la hora de realizar la selección altavoces que deben ser empleados para simular una fuente sonora. Como alternativa a la técnica de la línea de referencia se ha propuesto la técnica de la visibilidad, la cual se ha comprobado que realiza una selección más adecuada ya que evita que haya altavoces que radian sonido en sentido contrario al avance del frente de ondas de la fuente virtual. También se ha demostrado que en el peor de los casos ambas técnicas se comportan igual, por lo que siempre es preferible emplear la técnica de la visibilidad. Recientemente se ha propuesto una nueva técnica basada en el vector de intensidad sonora [Spors, 2007a], pero, como se ha comprobado, proporciona los mismos resultados que la técnica de la visibilidad, ya que en realidad lo que ha hecho ha sido describir la técnica formalmente.

El tercer problema tratado ha sido el de las fuentes cercanas al array. Como se ha comprobado ciertos parámetros de las señales *driving* se comportan de forma indeseada en las cercanías, por lo que ha sido necesario

sustituirlos o limitarlos por otros factores que eviten los efectos indeseados. En este caso se ha comprobado el buen funcionamiento de los parámetros propuestos, aunque no se ha realizado comparación alguna con otras técnicas pues hasta el momento no se ha hecho pública ninguna, lo cual sólo implica que las técnicas que existen se están explotando a nivel comercial por lo que no interesa hacerlas públicas.

En cuarto lugar se ha estudiado el problema de la síntesis de las fuentes focalizadas. Al igual que sucedía con las fuentes puntuales, para sintetizar las fuentes focalizadas se emplea la línea de referencia, técnica que si bien consigue su propósito, no es óptima en cuanto a la pureza del campo sintético conseguido. En contraposición se ha propuesto otra técnica que mejora notablemente la síntesis de las fuentes focalizadas. Aquellas fuentes que no se adentran mucho en la sala se sintetizan mucho mejor, sin embargo, conforme se alejan de la periferia se consigue una síntesis similar a la de la línea de referencia, esto es totalmente normal y es lo que cabe esperar, pues al adentrarse en la sala, el área de reproducción correcta disminuye drásticamente.

En quinto lugar se ha tratado el problema de generar fuentes en movimiento. Para ello se ha propuesto una implementación muy eficiente que consigue simular el movimiento aplicando el efecto *doppler* a voluntad. La implementación de este punto es crucial es un sistema de síntesis WFS, pues si no es posible mover las fuentes sonoras no se pueden generar las escenas. Otro punto fuerte de la implementación realizada es la eficiencia conseguida, ya que, como se verá en el capítulo siguiente, es posible recrear escenas sonoras complejas con multitud de fuentes virtuales y un centenar de canales, todo ello con un único ordenador personal y sin por ello sacrificar la calidad del sonido.

Por último se ha tratado el problema del *aliasing* espacial. En este caso se ha propuesto una técnica que, aunque no mejora el *aliasing*, consigue reducir drásticamente sus efectos en determinadas situaciones. Haciendo una comparación con la técnica OPSI, las mejoras conseguidas son similares, pero no en las mismas situaciones. Por ejemplo, OPSI permite simular ondas planas reduciendo el efecto del aliasing, mientras que la aproximación por subbandas esto es imposible. Sin embargo, en la síntesis de fuentes que no se alejan mucho del array la aproximación por subbandas consigue erradicar por completo el efecto del filtrado peine que se observa al moverse el oyente en la sala, a la vez que permite localizar de forma más precisa la posición de la fuente, aunque con un error angular que dependerá de la posición del oyente en la sala. La técnica OPSI también introduce un error en la posición ya que emplea el efecto *phantom* en la localización, pero este se acentúa si el oyente se desplaza por la sala. Por tanto, debemos concluir que de momento no existe ninguna técnica definitiva para tratar el *aliasing* espacial.

Capítulo 4

Procesado de WFS en tiempo real: arquitectura e implementación

4.1. Introducción

En el capítulo anterior se desarrollaron y propusieron una serie de técnicas que permiten solventar ciertos problemas de índole acústica y de procesado digital de señal cuando se pretenden implementar sistemas de WFS en la práctica. Sin embargo, una vez se dispone de todas estas herramientas algorítmicas y matemáticas, es necesario encontrar una arquitectura informática y un *hardware* para hacerlas funcionar. En esta tesis, no sólo se pretende proporcionar contribuciones en materia algorítmica o de procesado de señal a la WFS, sino que se pretende llegar más allá y diseñar también las estructuras informáticas y de cálculo que permitan implementar estos sistemas de manera *fiable*, *efectiva* y *realista*, hasta alcanzar un

prototipo plenamente funcional e incluso pre-comercial.

La implementación de la WFS tiene diferentes inconvenientes ya comentados en capítulos anteriores. Sin duda, el más importante está relacionado con el gran número de canales independientes de sonido que hay que sintetizar, equivalentes al número de altavoces de la sala y que por tanto crecen en relación al perímetro de la misma. Este gran número de canales, además de complicar todos los dispositivos de entrada/salida (I/O) del sistema de computación, repercute también en un alto coste computacional. La importancia, por tanto, de la arquitectura de procesado elegida resulta crítica y su elección y diseño requiere de un profundo estudio que ha ocupado gran parte de esta tesis doctoral hasta llegar a la solución óptima que se presenta en este capítulo y que ha sido reconocida por expertos en este campo.

Como hemos comentado, un sistema de WFS requiere una potencia de cálculo muy importante. La consecución de la potencia de cálculo necesaria se puede llevar a cabo de diferentes maneras que dan lugar a distintas arquitecturas. En concreto, se han utilizado tres estrategias principales para llegar a cabo estos objetivos:

- El uso masivo de DSPs.

- El uso de una red sincronizada de ordenadores personales.

- El uso de un único ordenador personal de última generación.

En los últimos años han aparecido tanto prototipos desarrollados por centros de investigación y universidades, como sistemas comerciales, es interesante señalar que a principios del desarrollo de esta tesis, cuando ya se disponía de un prototipo propio funcional, no existía en el mercado ningún sistema comercial. Un ejemplo de sistema comercial que emplea DSPs es el

desarrollado por la empresa IOSONO©, una *spin-off* del instituto Fraunhofer IDMT, y sin duda uno de los más extendidos actualmente. Por otro lado la empresa *Sonic Emotions* dispone de un sistema de WFS basado en una red sincronizada. Como ejemplo de la última categoría existen varias implementaciones las cuales suelen ser meramente prototipos como la descrita en [Spors et al., 2002], también existe una versión de software libre [Baalman, 2004] que no es un prototipo aunque todavía está en desarrollo[1]. En general, los sistemas que emplean un único ordenador personal suelen estar bastante limitados en cuanto al tamaño de la escena y número de fuentes sonoras simultáneas que pueden generar.

En el presente capítulo se describirá el funcionamiento de la arquitectura propuesta por el autor para implementar un sistema de WFS en tiempo real. La opción elegida será la tercera, el uso de un único ordenador de propósito general, y, como quedará demostrado, esta arquitectura va a permitir exprimir al máximo los procesadores de varios núcleos de última generación lo que faculta al software para generar escenas sonoras complejas con multitud de fuentes simultáneas en movimiento a un coste tanto computacional como económico muy efectivo.

En las Secciones 4.2 y 4.3 de este capítulo, se realizará un repaso del estado del arte de la arquitectura de los computadores de propósito general y del funcionamiento de las comunicaciones con el *hardware* de sonido, con el fin de justificar las contribuciones, desarrollos y decisiones técnicas adoptadas en las Secciones 4.4 y 4.5 en los que se comenta por un lado como obtener el máximo rendimiento posible del *hardware* al programar y, por otro lado, la arquitectura interna del software de WFS.

[1]En el momento de redactar estas líneas este software sigue estando disponible en internet, aunque la página web de su desarrolladora ya no está disponible.

4.2. Arquitectura de computadores

La elección de la estrategia basada en un único ordenador limita en gran medida la potencia de cálculo disponible en comparación con las otras estrategias, sin embargo, a cambio de esta limitación se consigue reducir el coste económico de la instalación de forma considerable. Además, cuenta a favor de este sistema que la potencia de los ordenadores personales está en continuo crecimiento, como demuestra el histórico de años anteriores. Adicionalmente, este aumento en la potencia de cálculo se consigue sin un aumento de precio de los nuevos procesadores. Por otro lado, la tendencia actual es la de añadir más núcleos (*cores*) a los procesadores, lo que también contribuye de forma notable al aumento de su potencia.

Puesto que se ha optado por emplear un único ordenador de propósito general, es conveniente realizar una revisión de la arquitectura de estos ordenadores con el fin de que la implementación posterior del software sea capaz de aprovechar todas las herramientas de que disponen y maximizar la eficiencia en los cálculos.

En la carrera por el aumento de la potencia de cálculo la arquitectura de los computadores ha ido evolucionando en diferentes aspectos, los cuales se pueden dividir en dos grupos de estrategias principales [Ortega et al., 2005]: tecnológicas y organizativas.

4.2.1. Estrategias tecnológicas

En este grupo entran todos los elementos que tienen que ver principalmente con la mejora de la electrónica del ordenador.

Entre otros elementos, en este grupo se encuentran:

- La disminución de la escala de integración

- El aumento de la frecuencia de reloj.

También es posible considerar en este grupo aquellas estrategias que supongan la repetición de elementos electrónicos de proceso, aunque generalmente se suelen incluir este tipo de estrategias en las organizativas.

En general las estrategias tecnológicas son transparentes desde el punto de vista del software que las va a emplear, por lo que no serán estas estrategias las que haya que estudiar en profundidad para optimizar el software.

Disminución de la escala de integración

La disminución de la escala de integración ha permitido incluir cada vez más transistores en el mismo espacio, lo que ha su vez a redundado en un aumento directo de la potencia de cálculo. La ley de Moore [Moore, 1965] establece que cada año y medio o dos años se duplica la capacidad de integración de los circuitos integrados, esta ley está constantemente en debate pues no se cree posible que se pueda mantener este ritmo, ya que están apareciendo nuevas restricciones en el diseño, relacionadas con el consumo y disipación de la potencia y las comunicaciones locales en los propios circuitos [Ortega et al., 2005].

Aumento de la frecuencia de reloj

El aumento de la frecuencia de reloj ha sido un arma básica con la que se ha conseguido aumentar la potencia de cálculo de los procesadores de forma directa. Dadas las limitaciones de la tecnología electrónica actual, se ha llegado a un punto en el que el aumento de la frecuencia de reloj supone más inconvenientes que ventajas. Por lo que, recientemente se ha cambiado la mentalidad y ya no se considera una estrategia que proporcione buenos resultados. Al igual que la anterior, esta estrategia es totalmente

transparente desde el punto de vista del software.

4.2.2. Estrategias organizativas

Dentro de este grupo se encuentran aquellas estrategias que tienen que ver con el diseño del flujo de los datos y las instrucciones dentro de los ordenadores. Se apoyan en la tecnología pero las estrategias propuestas no se basan en ella.

Dentro de este tipo de estrategias se encuentran varias opciones las cuales conseguirán aumentar la potencia de cálculo siempre que el software que las emplee esté optimizado para ello, por tanto, es conveniente estudiar el funcionamiento de estas estrategias para programar un software eficiente.

La segmentación

La segmentación del cauce de procesamiento (*pipeline*) es una forma de diseño del proceso de las instrucciones y los datos, la cual consiste en dividir el cálculo de la operación en una serie de etapas las cuales forman lo que se denomina el cauce de procesado. El término *pipeline* se usó por primera vez en [Bucholtz, 1962], aunque el primer ordenador en hacer uso de esta técnica fue el IMB709 en 1958 [Ortega et al., 2005].

Esta forma de trabajar permite que una vez se haya terminado el proceso de la primera etapa de una instrucción y se pase a realizar el proceso en la segunda etapa, se pueda comenzar a la vez el proceso de la primera etapa de la siguiente instrucción. El proceso completo de una instrucción es más lento que si se realizara en un único bloque, pero a cambio, el proceso de varias instrucciones es mucho más rápido, es un símil electrónico de las cadenas de montaje de la industria.

La segmentación introduce el paralelismo entre instrucciones [Bucholtz,

1962], es decir, permite ejecutar múltiples instrucciones de forma simultánea aunque terminan de ejecutarse secuencialmente conforme salen del cauce de procesamiento.

Escalabilidad

La escalabilidad consiste en la repetición, por duplicado, triplicado, etc... de algunos elementos de la arquitectura de proceso. Se pueden repetir elementos completos o etapas dentro de los elementos. Esta repetición permite disponer de más recursos durante el proceso de las instrucciones y los datos, lo que permite acelerarlo. Aquellas arquitecturas que emplean la escalabilidad, en la actualidad son prácticamente todas, son conocidas por el nombre arquitecturas superescalares [Ortega et al., 2005].

Arquitecturas vectoriales

Las arquitecturas vectoriales están especialmente diseñadas para optimizar las operaciones con estructuras vectoriales. En la actualidad es difícil encontrar arquitecturas vectoriales puras, en su lugar se suele emplear un procesador escalar con un coprocesador vectorial, un ejemplo de ello es el coprocesador de las tarjetas gráficas actuales.

El procesador vectorial emplea como unidad de trabajo el vector, el cual es un conjunto de datos escalares del mismo tipo almacenado en memoria. Sobre los vectores permite realizar tanto operaciones vectoriales como procesado vectorial. Las operaciones vectoriales permiten repetir la misma operación sobre un conjunto de datos, mientras que el procesado vectorial permite aplicar operaciones lógicas o aritméticas de forma reiterada sobre vectores.

Las ventajas del procesado vectorial se pueden resumir en tres [Ortega

et al., 2005]:

- Aprovechamiento del paralelismo de datos: algunos bucles pueden sustituirse directamente por una instrucción.

- Reducción del ancho de banda de instrucciones: una única instrucción especifica mucho trabajo.

- Optimización de los accesos a memoria: al emplear vectores los accesos a memoria son predecibles.

Como ya se ha comentado, las arquitecturas de propósito general que encontramos hoy en día son escalares, no vectoriales, sin embargo, para aprovechar ciertas características de los procesadores vectoriales se crearon los juegos de instrucciones SIMD (*Single Instruction Multiple Data*). Estos juegos de instrucciones facultan al procesador escalar para realizar procesado vectorial aunque a menor escala, ya que los tamaños máximos permitidos para los vectores son menores.

Los juegos de instrucciones SIMD comenzaron a emplearse de forma generalizada en los ordenadores personales a partir del año 1996 cuando Intel sacó al mercado el conjunto de instrucciones MMX [Mittal et al., 1997]. Con el tiempo han ido surgiendo versiones mejoradas entre las que se encuentran: SSE, SSE2, SSE3 y SSE4 por parte de Intel y 3DNow!, 3DNow!2 y SSE5 por parte de AMD, para más información sobre cada versión visitar las páginas web de Intel y AMD respectivamente. Inicialmente, este tipo de instrucciones operaban en aritmética entera pero en la actualidad se trabaja en coma flotante pues se obtiene un rendimiento muy superior.

Para maximizar el rendimiento en cálculos intensivos siempre que sea posible deben emplearse estos juegos de instrucciones. Sin embargo, su uso conlleva ciertos inconvenientes, los cuales podemos resumir en dos:

- El hecho de que haya tantos conjuntos de instrucciones diferentes obliga a programar para todas las versiones si se desea que el software sea portable.

- Su uso requiere de un elevado conocimiento del *hardware*, pues son juegos de instrucciones máquina.

Debido a estos inconvenientes, los juegos de instrucciones SIMD solamente suelen emplearse en aplicaciones multimedia [Ortega et al., 2005].

Multiprocesamiento

El paralelismo en los ordenadores se consigue duplicando algunos elementos de proceso, de esta forma es posible realizar tantas operaciones simultáneas como elementos duplicados haya.

El paralelismo contribuye enormemente al aumento de la potencia de cálculo del ordenador. Los mayores rendimientos se consiguen al duplicar el procesador o añadir varios núcleos a un único procesador. A lo largo de los años han surgido diferentes tecnologías menores que permiten cierto grado de paralelismo como la *HiperThreading* [Intel, 2009b] aunque la mejora de rendimiento obtenida dista bastante de la otras dos.

La principal diferencia entre un multi-procesador y un multi-núcleo radica en el hecho de compartir o no la caché del procesador y los accesos a memoria. En un multi-procesador, cada procesador tiene su propia caché y bus de acceso a la memoria, mientras que en un multi-núcleo tanto la memoria cache como el bus de acceso a memoria es compartido. Actualmente, los procesadores multi-núcleo de Intel proporcionan una caché L1 para cada núcleo, mientras que la caché L2 es compartida por todos [Intel, 2009a]. Así, por un lado cabe esperar mayor rendimiento del multi-procesador que del multi-núcleo, pero, por otro, la sincronización es más sencilla en el multi-

núcleo ya que la memoria es común para todos los núcleos y en el otro caso la memoria se divide en zonas preferentes para cada procesador. Además, como se verá en la Sección 4.4.3 el funcionamiento de uno y otro no es precisamente transparente de cara al software.

La tendencia actual en la fabricación de los ordenadores de propósito general se ha decantado por el modelo multi-núcleo, pues, al emplear un único bus la fabricación de las placas base es más sencilla y barata que en el caso del multi-procesador.

4.2.3. El sistema operativo multitarea

Una vez vistas las distintas estrategias de que dispone un procesador para conseguir elevar su potencia de cálculo, es necesario describir el uso que hace el sistema operativo de él.

El planificador de tareas es el mecanismo del sistema operativo encargado de distribuir el tiempo de procesador entre las distintas tareas a realizar. Este mecanismo es el que hace posible que se ejecuten varios programas de forma simultánea, y su uso es el que da el apelativo de «multitarea» al sistema operativo.

El planificador de tareas obliga a dividir el tiempo de uso del procesador en *quantums* los cuales se asignan a cada tarea en función de su necesidad de proceso. El cambio de contexto, es decir, la asignación del procesador a las diferentes tareas, lo realizan a su vez dos procesos del sistema operativo: el *scheduler* y el *dispatcher*. El *scheduler* es el encargado de asignar el procesador y el *dispatcher* de retirar la tarea a la cual se le ha agotado el tiempo asignado, está pendiente de una operación de I/O o simplemente está preparada otra tarea con mayor prioridad [Tanenbaum and Woodhull, 1998].

El tamaño de los *quantums* varía dependiendo del sistema operativo y suele rondar los 20 ms [Microsoft, 2004], aunque en aplicaciones en tiempo real (por ejemplo videojuegos) se suele reducir temporalmente a 5 ms o incluso a 1 ms para conseguir un tiempo de respuesta más ágil. En general no se emplea directamente un tiempo menor porque al disminuir los *quantums* se gasta proporcionalmente más tiempo en los cambios de contexto que en el procesado de las tareas en sí.

Como podemos ver, al trabajar en un ordenador ninguna tarea (o programa) dispone del procesador durante todo el tiempo, sino que tiene que ir turnándose con el resto para emplearlo. La división de los programas en distintas tareas maximiza el rendimiento pues el tiempo de ocupación del procesador es muy superior. Más adelante se verá que desde el punto de vista del software, las tareas son procesos y/o hilos. La realización de un programa con una única tarea es muy ineficiente pues en cuanto se necesita realizar una operación de I/O, las cuales suelen durar una eternidad en tiempos relativos al procesador, el *dispatcher* requisa el uso del procesador a la tarea la cual queda suspendida en espera hasta que se complete le operación de I/O, por lo que el procesador queda ocioso hasta que se complete la operación.

4.3. Modelo de acceso al *hardware* de I/O

La elección de un ordenador como sistema de procesado del sonido en vez del uso de DSPs, introduce ciertas ventajas, a la vez que limitaciones, sobre el procesado que se puede realizar. Por tanto, antes de comenzar a explicar el funcionamiento de la arquitectura propuesta es necesario tener en cuenta todas las limitaciones impuestas y la forma en la que se verá afectado el desarrollo de la implementación.

4.3.1. Procesado por bloques

La primera consecuencia con la que se encuentra cualquier aplicación de procesado de señal al trabajar con un ordenador que accede a un *hardware* de sonido[2] es que tendrá que procesar las muestras por bloques. Estos bloques suelen tener un número de muestras potencia de dos, siendo valores típicos 64, 128, ..., 1024 y 2048 muestras. Aunque pueda parecer que el uso de DSPs nos evita este problema, esto no resulta cierto hoy en día, donde los DSPs más modernos también necesitan comunicarse por bloques de muestras con los dispositivos A/D-D/A para ser eficientes (generalmente mediante DMA). Por tanto, el uso de PCs frente a DSPs no supone un inconveniente en este sentido.

4.3.2. *Hardware* de sonido

Como se acaba de comentar en la sección anterior, la comunicación con el procesador se realiza por bloques de muestras. Dependiendo de la marca y del modelo de la tarjeta existe un límite inferior en el tamaño de bloque que se puede emplear. El tamaño de dicho bloque definirá la latencia mínima que se podrá conseguir con dicho *hardware*.

El funcionamiento de la comunicación con el procesador, viene gobernado por lo que se conoce como técnica del «doble *buffer*». Para cada canal de I/O el *hardware* creará un *buffer* de memoria de tamaño doble al tamaño de bloque elegido, en el cual se almacenarán las muestras a reproducir (si

[2]Como *hardware* de sonido, se entiende un dispositivo de conversión A/D y D/A conectado al ordenador o bien un *hardware* que dispone de entradas y salidas digitales de sonido como SPDIF, AES/EBU, ADAT, MADI, etc. Popularmente se les conoce como tarjetas de sonido, aunque los modelos más profesionales son módulos externos que no se insertan en forma de tarjeta en las placas madre de los PC, sino que se conectan mediante interfaces *Firewire* o USB 2.0.

es un canal de salida), o las muestras grabadas (si es un canal de entrada). Este *buffer* de tamaño doble tendrá asignados dos punteros, el primero apuntará al comienzo del *buffer* y el segundo a la mitad del *buffer*, por lo que en realidad lo que tendremos serán dos *buffers* consecutivos del tamaño de un bloque. La información aquí mostrada se puede obtener del manual de cualquier librería de reproducción de sonido, por ejemplo en [Steinberg, 2006].

Veamos su funcionamiento con un ejemplo. Si se va a reproducir un fichero con un único canal de sonido, lo primero que hay que hacer es rellenar los dos *buffers* con las primeras muestras de sonido y comenzar la reproducción. Cuando el *hardware* de sonido detecta que ya ha reproducido todo el contenido del primer *buffer* avisa a la aplicación de que ese *buffer* está disponible para ser rellenado. Este aviso se realiza llamando a una función *callback* la cual se suele ejecutar en un hilo independiente, esto último depende en gran medida de la librería empleada en la comunicación con el *hardware* de sonido. En la función *callback* se obtienen las muestras de sonido siguientes y se copian en el *buffer* que acaba de quedar vacío. Todo este trabajo se realiza en paralelo a la reproducción por parte del *hardware* del segundo *buffer*. Cuando el segundo *buffer* también se vacía, el *hardware* de sonido continúa la reproducción por el primero y avisa de nuevo de que hay un *buffer* libre. Este mecanismo se repite de forma continua hasta que la aplicación o el usuario detiene la reproducción.

Todo el trabajo realizado depende del tamaño de bloque elegido, cuanto mayor sea el tamaño del bloque se dispone de más tiempo para su proceso, pero, a cambio, introducimos una latencia equivalente a la duración del bloque, por lo que la elección del tamaño siempre es una elección de compromiso.

El sistema de reproducción de WFS desarrollado permite trabajar con cualquier tamaño de bloque que proporcione el *hardware* de sonido, aunque se ha elegido como tamaño preferente 512 muestras. Este valor es equivalente a unos 10 u 11 ms con una frecuencia de muestreo fs de 44,1 o 48kHz. Un tamaño menor puede llegar a producir interrupciones de sonido (*glitches*) en ordenadores de inferior potencia, debido principalmente a fallos de sincronización con el planificador de tareas del sistema (véase la Sección 4.2.3). Tamaños de bloques superiores a 512 muestras funcionan correctamente aunque la latencia introducida en el sistema es superior.

Otro de los factores que determina la elección del *hardware* de sonido es el formato de las muestras con el que es capaz de trabajar. En este caso el *hardware* no suele ser muy flexible y generalmente sólo acepta un formato. El *hardware* empleado en el desarrollo de esta tesis, el *hardware* en cuestión se describe en detalle en la Sección 4.6). permite el uso de muestras de tipo entero de 24 bits las cuales se empaquetan en enteros de 32 bits alineados en los bits más significativos para simplificar su tratamiento por el software.

4.3.3. La librería ASIO

En la sección anterior se ha descrito el funcionamiento general del *hardware* de sonido basado en doble *buffering*. Sin embargo queda por definir la forma en que el software se comunica con dicho *hardware*.

En general cuando una aplicación va a trabajar con un *hardware* de sonido debe emplear una librería que hará las veces de intérprete entre la aplicación y el *driver* del *hardware* de sonido. El uso de la librería evita tener que enseñar a la aplicación como comunicarse con todos los posibles *drivers* existentes, los cuales no suelen ser compatibles entre sí. De esta manera, la aplicación puede funcionar con cualquier *hardware* de sonido,

siempre y cuando este sea compatible con la librería.

En la actualidad existen diversas librerías que permiten la comunicación con el *hardware* de sonido, entre las cuales podemos destacar las siguientes:

- Windows Multimedia Extensions (MME)
- DirectSound
- ASIO (*Audio Stream Input/Output*)
- ALSA (*Advanced Linux Sound Architecture*)
- CoreAudio

Las dos primeras librerías son específicas del sistema operativo Windows. La librería ASIO puede funcionar tanto bajo Windows como Apple OSX, mientras que la librería ALSA funciona bajo Linux y finalmente CoreAudio solamente sobre Apple OSX.

La elección de la plataforma correcta es esencial pues la eficiencia de la implementación depende de ello.

La primera opción se descartó rápidamente pues no permite una implementación eficiente, al trabajar mediante capas software intermedias en lugar de trabajar directamente sobre el *hardware* de sonido. La opción de la librería DirectSound, subconjunto de la librería DirectX dedicado al sonido, se descartó pues aunque sí permite sacar el máximo rendimiento del *hardware* de sonido (esto ha dejado de ser así a partir del sistema operativo Windows Vista) su programación es altamente complicada en comparación con otras opciones.

La librería ALSA está dedicada íntegramente a Linux [ALSA, 2009], sin embargo el problema de este sistema operativo es que multitud de marcas

de *hardware* de sonido profesional no proporcionan *drivers* ni soporte para este sistema operativo.

La librería ASIO [Steinberg, 2006] es una de las más eficientes y utilizadas en el campo del audio profesional y proporciona tiempos de latencia muy bajos. Esta librería la diseño la empresa *Steinberg* para proporcionar un marco de trabajo sencillo y eficiente en el tratamiento del audio a nivel profesional. Por todo ello, ha sido la librería elegida para realizar el trabajo.

4.4. Diseño del software para rentabilizar el *hardware*

El software que se pretende desarrollar debe realizar todas las tareas necesarias de la síntesis del campo sonoro mediante WFS. Debido a que todas estas tareas requieren de un cálculo intensivo, y que además se ha optado por la estrategia de emplear un único ordenador de propósito general, es necesario optimizar el software para aprovechar al máximo las capacidades de cálculo del *hardware*.

En las dos secciones anteriores se ha descrito la arquitectura de los ordenadores de propósito general y el funcionamiento del *hardware* de sonido, lo que permite delimitar un marco de trabajo sobre el cual plantear el diseño del software optimizado.

En las siguientes secciones se mostrará la forma en la que un software puede aprovechar las características del *hardware* sobre el que se ejecuta, lo que permitirá en muchos casos maximizar la eficiencia del procesado que se realiza.

Antes de continuar con la descripción del diseño del software, es necesario indicar el lenguaje de programación con el que se va a programar,

en concreto el C++. La elección de C++ como lenguaje de programación fue hecha principalmente por comodidad, pues es con este lenguaje con el que el autor tiene más experiencia. Además, permite programar tanto con niveles de abstracción elevados como a muy bajo nivel, siendo esta última característica la que permite obtener el máximo rendimiento del *hardware*.

4.4.1. Optimización del código

La optimización del código es uno de los procesos de la creación de un software que más hay que cuidar. Debe realizarse siempre al final del trabajo, pues, de otro modo se desperdiciará mucho trabajo en optimizar código que finalmente no será empleado al ser sustituido por otro durante las continuas versiones.

La optimización manual del código es un proceso lento en el cual se debe tener cuidado, pues no siempre es recomendable. Por un lado, gran parte de las optimizaciones que se pueden realizar no suponen una verdadera mejora que justifique el esfuerzo invertido y, por otro lado, el código optimizado suele ser mucho menos legible, por lo que, es muy fácil introducir fallos nuevos y más difícil localizarlos. Así, únicamente deben optimizarse aquellas partes del código que de verdad supongan una mejora sustancial en el procesado .

Un aspecto de suma importancia en la optimización del software es el conocimiento del *hardware* sobre el que va a funcionar. En nuestro caso, el *hardware* elegido son los procesadores de propósito general. Con este marco de trabajo se ha delimitado la arquitectura, pero todavía existe una gran variedad de procesadores y versiones de procesadores distintos, por lo que la optimización del código en lenguaje ensamblador no es una opción muy adecuada. Un código óptimo para un procesador puede no serlo para

otra versión de ese mismo procesador aunque mantenga el mismo juego de instrucciones, por ejemplo, el simple hecho de tener una memoria caché de tamaño menor puede cambiar radicalmente su ejecución al no caber todos los datos necesarios en ella.

No es intención del autor hacer en este apartado un análisis exhaustivo de las técnicas de optimización, solamente proporcionar unos consejos útiles que permiten obtener programas más eficientes siguiendo una guía general de buenas conductas al programar. Como ya se ha comentado anteriormente, la optimización a muy bajo nivel no tiene sentido salvo que se identifique de forma inequívoca el *hardware*. Más adelante, en las Secciones 4.4.2 y 4.4.3 se comentará como, a parte de las recomendaciones realizadas, se consigue maximizar el rendimiento sin necesidad de programar en lenguaje ensamblador.

El uso de las siguientes recomendaciones permite mejorar la eficiencia del código de dos maneras distintas, por un lado evita que se realicen trabajos encubiertos innecesarios, y por otro ayuda al optimizador de código del compilador a detectar mejor las situaciones en las que puede ser útil.

En internet es posible encontrar multitud de guías de buenas prácticas de programación, por ejemplo [EventHelix, 2009] y una especialmente exhaustiva [Fog, 2009], sin embargo, la relación que se muestra a continuación es bastante sencilla de seguir y se ha ido formando a lo largo de muchos años de experiencia en programación en C/C++, aunque, sinceramente, me es imposible recordar de donde ha surgido cada punto. Sin ánimo de ser exhaustivos, entre los ejemplos de buenas prácticas de programación podemos encontrar:

- Emplear `int` en las llamadas a funciones en lugar de `char` o `short` para tipos enteros cortos. Puesto que el compilador de C++ siempre

direcciona los datos en enteros de 32 bits, el uso de tamaños menores implica la conversión forzosa del tipo a entero para su procesado y su posterior reconversión al tipo original.

- Evitar encadenar las operaciones matemáticas, si es necesario emplear paréntesis para romper el encadenado. De esta forma es posible realizar varias operaciones en paralelo, en vez de tener que esperar a que termine la primera para comenzar con la siguiente.

- Siempre que sea posible realizar los bucles con cuentas descendentes. Las operaciones de decremento y comparación suelen ser más eficientes de procesar que las equivalentes con incremento.

- Posponer al máximo la declaración de variables. Al hacer esto, se maximiza la probabilidad de que no se llegue a crear la variable al ramificarse el código.

- Evitar la reserva y/o liberación de memoria dinámica en las operaciones. Siempre que sea posible realizar la reserva y liberación antes y después del procesado en tiempo real. Si fuera necesario, reservar previamente una zona de memoria (*pool*) para ser usada posteriormente.

- Precalcular las operaciones costosas que vayan a ser utilizadas reiteradamente. Es mucho más eficiente crear previamente una tabla con los valores más comunes (*lookup table*) para posteriormente acceder a ella.

- Simplificar las tareas a realizar en los constructores de los objetos. La gran mayoría de las operaciones con objetos suelen construir objetos

temporales de forma inadvertida, por lo que un constructor pesado debe evitarse siempre que sea posible.

- Emplear el calificador *inline* para funciones sencillas de uso continuado de los objetos. Al hacer esto, se sustituyen las llamadas a la función por el código de la función, lo que evita la sobrecarga de las llamadas, a cambio claro está de aumentar el tamaño del código.

- Inicializar los objetos al declararlos en vez de asignarles el valor de forma posterior. En el primer caso se llama al constructor pertinente, en el segundo al constructor por defecto en la declaración y posteriormente al constructor de copia en la asignación.

- Emplear las operaciones prefijas en vez de las postfijas al usar objetos. El uso de una operación postfija obliga a crear una copia temporal del objeto inicial para poder devolver su valor.

- Pasar siempre los objetos por referencia a las funciones. Si se pasan por valor se obliga a crear una copia. Esta recomendación es esencial al diseñar los operadores básicos de un objeto.

Antes de continuar con el siguiente aspecto de la optimización del software, se matizarán dos puntos concretos que, desde el punto de vista del autor, es conveniente tener en cuenta al programar, sobre todo si el código va dirigido al procesado de señales.

Uso de punteros en lugar de arrays

Ya que las señales no son más que una colección de muestras, la forma natural de representarlas en un programa es mediante los arrays, sin embargo, esta forma de representación no es óptima pues, como se verá, produce

un exceso de trabajo al ser usada. Una de las optimizaciones más útiles para el procesado de señal es el uso de punteros en sustitución de los arrays. El funcionamiento de ambos métodos se puede encontrar en cualquier libro de programación en C, por ejemplo en [Gottfried, 2005].

La mejor forma de explicarse es mediante un ejemplo, por lo que veamos uno concreto: la implementación del filtrado con un filtro FIR.

El filtrado FIR se puede programar fácilmente mediante el siguiente código en C, el cual es bastante autoexplicativo:

```
y[n]=0.0;
for (int i=0; i<N; i++)
    y[n] = y[n] + b[i] * x[n-i];
```

El código anterior se puede optimizar simplemente empleando punteros en lugar de arrays.

```
float *yp, *bp, *xp; % Declaración de los punteros
yp = &y[n];          % Asignación de los punteros a los vectores
bp = &b[0];
xp = &x[n];

*yp=0.0;             % Equivalente a la instrucción y[n]=0.0;
for (int i=0; i<N; i++)
    *yp = *yp + *bp++ * *xp--;
```

Como se puede apreciar, el código no optimizado es mucho más legible que el optimizado, pero este último tiene ciertas bondades, siendo la principal de ellas la reducción de los accesos a memoria que deben realizarse en cada iteración del bucle.

Si nos centramos únicamente en el código `x[n-i]` podemos ver que para determinar su valor es necesario calcular dos sumas y hacer cuatro accesos a memoria:

- Un acceso a memoria para determinar el valor de `i`.
- Un acceso a memoria para determinar el valor de `n`.
- Una suma (resta) para calcular `n-i`.
- Un acceso a memoria para localizar el principio del array `x`, que se corresponde con `x[0]`.
- Una suma para determinar el desplazamiento de `x[n-i]`.
- Un último acceso a memoria para determinar el valor de `x[n-i]`.

Veamos ahora las operaciones que deben realizarse, aparte de la inicialización de los punteros que es una única vez, para ejecutar el código `*xp--` que sustituye al anterior:

- Un acceso a memoria para determinar la dirección a la que apunta `xp`, que es donde se encuentra `x[n]`.
- Un acceso a memoria para obtener el valor de `*xp`, esto es, el valor de `x[n]`.
- Una resta, que al ser post-decremento sobre un puntero, desplaza el puntero a la posición anterior, por lo que al terminar `xp` apunta a `x[n-1]`, listo para la siguiente iteración.

Es fácil deducir de este ejemplo que el uso de punteros en lugar de los arrays es una buena práctica, sobre todo en los bucles cuya ejecución sea crítica en el rendimiento de la aplicación.

Sentencias condicionales

Tal y como se comentó en la Sección 4.2 las arquitecturas actuales emplean la segmentación en su diseño, la cual introduce el paralelismo a nivel de instrucción, es decir, permite ejecutar varias instrucciones simultáneamente.

Este tipo de paralelismo tiene el inconveniente de que en un programa no siempre se conoce a ciencia cierta cuál será la siguiente instrucción a ejecutar, por lo que, la ejecución de instrucciones en paralelo se ve comprometida. Un ejemplo claro de esta situación son las sentencias condicionales, por ejemplo los `if`, hasta que no se ha evaluado la condición no se sabe que rama debe ejecutarse, si la correspondiente al `if` o la correspondiente al `else`. Una buena práctica de programación consiste en colocar siempre el caso más probable en la rama correspondiente al `if` y, por tanto, el menos probable en la rama del `else`, ahora se verá porqué.

Cuando se llega a una instrucción condicional, para no desaprovechar el cauce esperando a que se averigüe el valor de la condición, lo que se hace es predecir qué rama se va a ejecutar y comenzar su ejecución inmediata en paralelo a la evaluación de la condición, así, si se ha acertado en la predicción se consigue ganar tiempo. Sin embargo, si no se ha acertado, es necesario deshacer todo el trabajo hecho, vaciar el cauce de ejecución y comenzar la ejecución de la otra rama, con la consiguiente merma en el rendimiento.

La predicción que decide la rama a elegir la realiza el propio *hardware* y es dinámica [Ortega et al., 2005]. Inicialmente se elige una rama cualquiera y después, dependiendo de la estadística se va aprendiendo que es mejor hacer en cada caso. Esto no ha sido siempre así, inicialmente era común elegir la primera rama siempre o hacer caso a las instrucciones del compilador,

cosa no muy recomendable pues en tiempo de compilación no se conocen los valores de las variables incluidas en la condición. De hecho, las arquitecturas que relegan sus predicciones en el compilador no han tenido mucho éxito debido principalmente a la dificultad de su programación [Ortega et al., 2005].

Así, al colocar siempre el caso más probable en la primera rama, estamos ayudando a aprender rápidamente qué rama elegir. Por lo que, en caso necesario, es más rentable reformular la condición para que la rama más probable sea la primera, que dejar la sentencia tal cual.

Una de las prácticas de programación más desaconsejadas es la inclusión de sentencias condicionales dentro de bucles cuando no se conoce la probabilidad de ejecución de cada rama, ya que se provoca una parada y vaciado de los cauces de ejecución de forma reiterada con la consiguiente pérdida de rendimiento. Esto no quiere decir que no se deban incluir sentencias condicionales en los bucles, de hecho todo bucle incluye una sentencia condicional implícita, que es la que determina cuando debe acabar, sino que deben estudiarse cuidadosamente, ya que un `if` mal colocado en un bucle cuyo procesado es crítico puede resultar nefasto en el rendimiento. En esta situación, siempre que sea posible, es preferible sacar la sentencia condicional fuera y crear un bucle distinto para cada rama de ejecución.

4.4.2. Procesado vectorial

En esta sección se describe como es posible aprovechar la capacidad de procesado vectorial de que disponen los procesadores de propósito general actuales.

Para aprovechar las herramientas de procesado vectorial sin tener que bajar al nivel de las instrucciones máquina, se puede confiar en el optimiza-

dor de código del compilador, el cual sustituye el código generado por una versión que emplea entre otras cosas los juegos de instrucciones SIMD del procesador, sin embargo, aunque estas optimizaciones mejoran bastante la eficiencia del código suelen ser mejorables a su vez, por lo que si se desea un código verdaderamente eficiente no queda más remedio que realizar la optimización a mano, con el consiguiente trabajo extra para el programador.

Una opción intermedia es la de emplear una librería específica de procesado que ya implemente las optimizaciones. De esta forma solo hay que preparar los datos para ser aceptados por las funciones de la librería. Esta última opción fue la tomada por el autor, el cual ha empleado en el desarrollo del software la librería de procesado de señal [Intel, 2000].

Esta librería tiene la ventaja adicional de que se adapta al procesador en el que se está ejecutando la aplicación. Su distribución incluye un conjunto de DLLs (*Dynamic Link Library*) de las cuales solamente se carga la correspondiente a la versión del procesador de que dispone la máquina. De todas las herramientas proporcionadas por la librería la parte más empleada ha sido el cálculo vectorial, la mayor parte de las operaciones de cálculo se han preparado para ser realizadas de esta forma. Así, en vez de tener que ejecutar un bucle que realice una operación elemento a elemento, se llama a una función la cual ejecuta en su interior unas pocas instrucciones SIMD. No es posible asegurar este punto pues no se conoce la programación interna de las funciones, pero el rendimiento que alcanzan hace suponer que así es.

Otras implementaciones de sistemas de WFS [Baalman, 2004; Spors et al., 2002] emplean la librería de procesado *BruteFIR* [Torger, 2001], la cual, aunque no tiene la misma filosofía que la librería de Intel, pues

está enfocada al filtrado con filtros FIR, si consigue una optimización muy elevada. Sin embargo, el propio autor de *BruteFIR* afirma en [Torger and Farina, 2001] de que la librería de Intel es más eficiente. Un punto en contra de esta librería es que no está optimizada para los procesadores de AMD.

Antes de acabar la sección es necesario hacer un inciso. Ya que todo el procesado se va a realizar en coma flotante, pues el rendimiento se incrementa de forma abrumadora, será necesario que siempre que se envíen datos al *hardware* de sonido realizar una conversión de coma flotante a enteros, ya que este trabaja únicamente con muestras enteras.

4.4.3. Paralelismo

Del funcionamiento del planificador de tareas de los procesadores, se desprende fácilmente que el diseño multi-tarea no es una opción sino una obligación. A este tipo de programación que emplea múltiples tareas es a lo que se conoce en la informática por el nombre de «programación concurrente» [Andrews, 1991]. La principal ventaja de programar de forma concurrente es que se minimizan los tiempos muertos del procesador, tiempos en los que no está haciendo nada.

El inconveniente que introduce la programación concurrente es el aumento de la complejidad del software [Andrews, 1991]. El trabajo a realizar es el mismo pero ahora no hay un única tarea sino un grupo. Todo aquel que en algún momento haya tenido que trabajar en equipo entiende rápidamente los problemas que conlleva, no solamente es necesario hacer una buena división del trabajo sino que además hay que coordinar a los trabajadores, pues generalmente el trabajo de uno depende del trabajo del resto. En un programa pasa exactamente lo mismo, si no se planifica todo bien desde el principio el programa será un caos. En el mejor de los casos, una

mala planificación provocará que se desaproveche el procesador, en el caso peor (el más normal) el programa no funcionará.

Ventajas de los procesadores de varios núcleos

El uso de procesadores con varios núcleos permite aumentar de forma considerable la potencia de cálculo del ordenador. Un procesador con dos núcleos tiene una potencia de cálculo similar a uno de un núcleo que emplea una frecuencia de reloj doble. Este aumento de la potencia no es lineal con el número de núcleos pues parte de la potencia se debe gastar en la sincronización de los núcleos, y esta aumenta rápidamente con su número.

A la pregunta de si es más rentable un procesador rápido o uno más lento con varios núcleos, la respuesta es «depende», aunque si se diseña bien el software, y el problema a resolver es paralelizable, entonces la respuesta es casi siempre que es mejor tener varios núcleos.

La principal ventaja de tener varios núcleos es que se pueden ejecutar varias tareas en paralelo, pero para que esto sea posible el software debe dividir su trabajo en varias tareas. El planificador de tareas del sistema operativo puede asignar cada tarea a un núcleo distinto, esta asignación es transparente al usuario y depende principalmente del porcentaje de ocupación de cada núcleo. Si el software está compuesto por una única tarea, esta será asignada a un núcleo, siendo imposible para el planificador de tareas asignarle otros núcleos de forma simultanea aunque estos estén ociosos.

Cuando en año 2003 comenzó el desarrollo de esta tesis, se apostó por emplear la estrategia de emplear un único ordenador de propósito general. La tecnología no estaba tan avanzada pero se podía intuir por donde iba a continuar la evolución de los procesadores y, como se puede ver, no fue una elección desacertada. En el momento de la redacción de este documento,

los procesadores actuales incorporan 2 o 4 núcleos aunque se pueden encontrar versiones de servidor que incorporan hasta 6. Además, el precio de estos procesadores no se ha incrementado con las prestaciones, sino que en muchos casos se ha reducido.

Procesos e hilos

Si en arquitectura de computadores se habla de tareas, en programación este término se convierte en dos: procesos (en inglés se puede traducir indistintamente por *tasks* o *process*) e hilos (o hebras, en inglés *threads*). Así que la primera pregunta que se debe responder es: ¿qué se debe emplear, procesos o hilos?.

Para poder responder a esta pregunta antes es necesario conocer la diferencia entre ambos. Todo programa tiene un proceso con el cual el sistema operativo lo ejecuta, al que se le suele denominar proceso padre, a partir del cual el programa puede crear más procesos o hilos hijo.

Si el proceso padre crea un proceso hijo, el padre y el hijo tendrán contextos distintos, lo que en la práctica permite que sean tratados como si fueran dos programas independientes, de hecho, el proceso padre no es más que un proceso hijo de otro proceso del sistema operativo. Así cada uno tiene sus propias variables, descriptores de fichero, etc..., este aislamiento implica que si quieren comunicarse entre sí deben establecer algún canal de comunicación explícitamente: ficheros, *sockets*, *pipes*, una zona memoria compartida, etc.

Si por el contrario el proceso padre crea un hilo hijo, el padre y el hijo comparten el mismo contexto. Es decir, ambos comparten prácticamente todo, las variables, los descriptores de fichero, los *sockets*, etc, aunque mantienen cierta información que permite al sistema operativo diferenciarlos. Si

uno de ellos modifica el valor de una variable, está modificando la variable para los dos, por lo que debe tener mucho cuidado al modificar cualquier elemento que el otro pueda emplear. Así, debe asegurarse de que el valor modificado es coherente con lo esperado por el otro, pero también debe tener cuidado con el instante en que realiza la modificación. A cambio no es necesario ningún mecanismo especial de comunicación entre ellos puesto que lo comparten todo [Márquez, 2004].

Visto esto se entiende que a los procesos se les denomine «hilos pesados» y a su vez a los hilos se les denomine «procesos ligeros». La programación bajo el sistema operativo Windows suele emplear hilos, mientras que en Linux es más común emplear procesos. De todas formas se pueden emplear los dos métodos en ambos sistemas operativos.

Históricamente los procesadores de propósito general disponían únicamente de un procesador con un único núcleo. En ese contexto, se puede asegurar que la programación con hilos es casi siempre más eficiente, pero, una vez se da el salto a la arquitectura con varios núcleos o procesadores, esto ya no es así. Cuando se dispone de más de un procesador, el planificador de tareas asigna cada proceso al procesador más adecuado, en el cual residirá hasta que finalice su ejecución, es decir, un proceso no puede cambiar de procesador, ya que la memoria de cada procesador es privada y el resto de procesadores no pueden acceder a ella, o por lo menos no tan libremente como su dueño, por lo que si se permitiera cambiar al proceso de procesador habría que trasladar todo su contexto a la zona de memoria del otro procesador. En cuanto a los hilos, si tenemos en cuenta que se crean a partir del proceso padre, están confinados con él en el mismo procesador. Así, para aprovechar varios procesadores es necesario emplear varios procesos, una aplicación multi-hilo sacará el máximo rendimiento de un único

procesador, el resto no podrán ser usados.

Recientemente se ha generalizado el uso de procesadores de varios núcleos, aquí, la cuestión es una mezcla de las anteriores, y además es dependiente de la arquitectura interna del procesador. Al residir los núcleos dentro del mismo chip, es normal que compartan la memoria ya que acceden a ella por un bus compartido, pero también es posible que se diseñe de forma que haya que dividirla. En las arquitecturas cuya memoria sea compartida tanto los procesos como los hilos pueden ser asignados a uno u otro núcleo, los procesadores de varios núcleos actuales entran dentro de esta categoría [Intel, 2009a]. En las arquitecturas con zonas de memoria independientes, los procesos son asignados a un único núcleo sin posibilidad de cambiar de núcleo, y con ellos todos sus hilos.

Cuando comenzó el desarrollo del software, el autor se decantó por la programación multi-hilo pues las diferentes tareas en las que se divide el procesado de WFS requieren de un intercambio de datos considerable, además de que su gestión por el sistema operativo es más eficiente, ya que no es necesario crear contextos nuevos y copiar su contenido, a cambio ha sido necesario prestar especial atención a la sincronización y la protección de los datos compartidos. Puesto que se ha elegido los hilos como técnica de programación, en el resto del texto solamente se hablará de ellos.

4.4.4. Problemas de sincronización

Como ya se ha comentado en la sección anterior, uno de los mayores problemas que conlleva la programación concurrente es la buena sincronización de todos los hilos.

Los dos problemas típicos que deben evitarse en la programación concurrente son:

- La condición de carrera (*Race condition*): Sucede cuando dos o más hilos emplean un mismo recurso compartido y el resultado de las operaciones realizadas depende del orden en que se haya usado el recurso.

- El interbloqueo o abrazo mortal (*Deadlock*): Sucede cuando un hilo reserva parte de los recursos compartidos que va a necesitar y queda bloqueado a la espera de la liberación de otros recursos que están en uso por otro hilo. Si este segundo hilo intenta a su vez reservar los que ya tiene reservados el primero, se bloquea también a la espera de que sean liberados. Así ambos hilos están bloqueados esperando a que el otro libere los recursos y ninguno de los dos puede continuar su ejecución.

Para solucionar los problemas de las condiciones de carrera se crearon las *secciones críticas*, las cuales al ser usadas impiden que más de un hilo opere a la vez en su interior, por lo que serializan el trabajo, esto es, los hilos deben acceder a su interior de uno en uno [Microsoft, 2006].

A cambio de eliminar las condiciones de carrera las secciones críticas introdujeron los interbloqueos, los cuales solamente se pueden evitar creando un protocolo de trabajo adecuado sobre la zona de recursos compartidos, siendo común en la literatura del tema reducir los distintos tipos de problemas a varias plantillas de protocolos entre las que se encuentran: lectores/escritores, productores/consumidores, el barbero, los cinco filósofos, etc [Andrews, 1991].

Como una evolución de las secciones críticas surgieron entre otros mecanismos los semáforos y el paso de mensajes [Márquez, 2004]. El uso de los semáforos está más extendido en los sistemas Linux, mientras que en Windows el mecanismo general es el paso de mensajes. Todo problema de

sincronización que se pueda resolver con un mecanismo se puede resolver también con los otros, invirtiendo más o menos trabajo por parte del programador. Así, es preferible emplear la técnica que resulte más sencilla en cada momento. También debe tenerse en cuenta la destreza del programador con la técnica pues los algoritmos de sincronización suelen ser extremadamente complejos de depurar cuando no se tiene experiencia, aunque, incluso teniendo experiencia la depuración de estos algoritmos es bastante complicada.

En la programación del software desarrollado en esta tesis Doctoral se ha utilizado como mecanismo de sincronización principal el semáforo, debido principalmente a que es el mecanismo más afín al autor, aunque en determinadas situaciones se ha empleado el paso de mensajes pues se simplificaba mucho la sincronización.

4.5. Arquitectura del software de WFS

Plantear la creación de un software viene a ser como planificar un proyecto. En primer lugar es necesario efectuar un estudio del trabajo que va a realizar el software con el fin de delimitar claramente las diferentes tareas que es necesario llevar a cabo. Si, además, la aplicación debe funcionar en tiempo real, es necesario indicar un orden estricto de realización de las tareas así como la prioridad que va a tener cada una de ellas, para, en la medida de lo posible, intentar hacer el mayor trabajo en paralelo, lo que permite tener cierta holgura en los tiempos de ejecución de determinadas tareas.

En la siguiente sección se describirá la arquitectura interna del software de WFS desarrollado. Para poder acometer la descripción de un software de esta envergadura, se ha procedido a dividir su estructura desde tres pun-

tos de vista distintos. En primer lugar se describirán los objetos en los que se han compartimentado tanto las estructuras de datos como las entidades físicas que representan. A continuación se describirá la paralelización realizada del algoritmo, esto es, la división de las tareas en los diferentes hilos que podrán trabajar de forma concurrente. Para concluir, se describirá el procesado de la señal realizado mediante los correspondientes diagramas de bloques. Esta última parte se ha separado de las dos previas porque ni los objetos ni los hilos representan de forma completa el procesado.

4.5.1. Diseño orientado a objetos

El uso de la Programación Orientada a Objetos (POO) delimita y simplifica las tareas de programación de forma natural por lo que en general es una buena elección. Sin embargo, debe hacerse constar que muchas de las operaciones realizadas en el software de WFS se han programado deliberadamente en contra de sus mandatos pues, a cambio de la facilidad que proporciona introduce cierta «burocracia» en el programa, por lo que, en determinadas situaciones, puede reducir el rendimiento de forma dramática. Así, aunque la programación general está orientada a objetos, las partes especificas de procesado de señal no se han representado con objetos, o por lo menos no ha sido así a la hora de realizar los cálculos.

A continuación se detallarán los objetos principales que componen el software. La funcionalidad de los objetos descritos va desde la representación de elementos físicos de la escena, elementos de la cadena de procesado o simplemente piezas menores creadas para facilitar las tareas de programación.

Línea de retardo

Este objeto encapsula en su interior todas las funciones que permiten recrear una línea de retardo. Para realizar la implementación se ha seguido el método más común: emplear un *buffer* circular con dos punteros, uno de lectura y otro de escritura, además de un filtro interpolador FIR de primer orden.

El funcionamiento de la línea de retardo no reviste de especial dificultad, sin embargo, la posibilidad de acceder a ella de forma simultánea desde distintos hilos si puede acarrear serios problemas.

La línea de retardo convencional encaja con la plantilla de los productores y consumidores (véase la Sección 4.4.4), sin embargo, por necesidades de diseño, debe permitir acceder a los datos sin «consumirlos», ya que para una misma fuente sonora necesitamos generar varios retardos (uno por altavoz). Por tanto, la plantilla que mejor se ajusta al caso es lectores/escritores con prioridad a los lectores. Esta plantilla asegura que se pueden leer muestras de la línea de retardo desde varios hilos simultáneamente, a la vez que asegura que únicamente un único escritor en solitario pueda modificar el *buffer* de datos. A la plantilla es necesario hacerle una ligera modificación que no permita seguir escribiendo una vez esté lleno el *buffer*, ni leer cuando esté vacío. La implementación de esta plantilla ha requerido emplear tres semáforos, uno que asegura la exclusión mutua en el acceso a los datos y otros dos que sirven de sendas colas de espera, una para los lectores y otra para los escritores.

Fuente Virtual

Este objeto representa cada una de las fuentes virtuales de las escenas, por lo que en su interior contiene todos los objetos y variables necesarios

para realizar este cometido. Entre otros objetos, internamente hace uso de líneas de retardo, *buffers*, filtros, etc.

Para que este objeto funcione de forma segura en un entorno concurrente, necesita emplear mecanismos de protección a dos niveles, por un lado emplea un semáforo de exclusión mutua cuando se accede a sus funciones y parámetros internos, por otro lado, emplea el mecanismo de sincronización interno de las líneas de retardo.

Auralización

Este objeto encapsula en su interior todo el procesado de las señales de reverberación, que proporcionan el ambiente de la sala.

Puesto que el cálculo del procesado de la auralización se realizará en paralelo con el de las fuentes virtuales, y teniendo en cuenta que el tamaño de bloque empleado para este procesado es mayor que el tamaño de bloque que emplea la fuente virtual, es necesario un mecanismo de sincronización independiente que asegure la buena coordinación con el resto de hilos.

En este caso, la sincronización se ha realizado por dos vias. Por un lado, se emplea el paso de mensajes para señalizar la disponibilidad de datos. Por otro se emplean semáforos para asegurar que el intercambio de los datos tanto de entrada como de salida sea realizado correctamente.

Subwoofer

Este objeto se encarga de centralizar el procesado dedicado al cálculo de la señal de los *subwoofers*. Debido al diseño de la lógica de procesado, no ha sido necesario aplicar ningún método de sincronización especial, ya que, como se verá en la sección siguiente, su procesado se realiza en el mismo hilo de las fuentes virtuales.

Filtro

Este objeto se encarga de centralizar la descripción de los filtros tanto FIR como IIR, así como de determinar las funciones de procesado específicas en la librería Intel según el tipo de filtro. Su funcionamiento no reviste de especial atención en cuanto a la protección de los datos, ya que su protección está prevista dentro de los mecanismos de los objetos que hacen uso de él.

Segmento

Representa cada uno de los segmentos de altavoces que componen el array. Entre otras cosas mantiene los parámetros típicos como son: número de altavoces, separación entre estos, posición y orientación del segmento, etc. Este objeto tampoco reviste de especial dificultad de protección, ya que sus datos son modificados *off-line*, permaneciendo constantes durante todo el procesado de WFS.

Array

Representa el array físico de altavoces, el cual está compuesto a su vez por una lista de segmentos de altavoces. La construcción del array se realiza de forma incremental segmento a segmento, lo que simplifica bastante su descripción en el interfaz de usuario.

En el uso de este objeto se puede observar fácilmente la «burocracia» que introduce la POO, y cómo se ha evitado.

Una de las operaciones que más veces se debe calcular en el procesado de WFS es el cálculo de la distancia desde la posición de una fuente hasta cada uno de los altavoces. Como el array está compuesto por varios segmentos, las posiciones de los altavoces no se encuentran en el array sino en los segmentos, y ni siquiera en estos, pues las posiciones las almacena el objeto

Punto el cual representa un punto en el espacio, los segmentos tienen en su interior un vector de puntos, por lo que el cálculo de las distancias implica cierto trabajo extra (*overhead*) al tener que recorrer la lista de segmentos, y dentro de estos los vectores de posiciones, para determinar las posiciones de los altavoces. Como se puede ver, calcular todas las distancias requiere de bastante trabajo extra por el simple hecho de describir los datos con objetos.

Con el fin de eliminar este trabajo extra innecesario, una vez construido el array se realiza un duplicado de las posiciones de los altavoces en dos vectores de números reales, uno mantiene las posiciones en el eje X y el otro en el Y. Así es posible calcular las distancias entre la fuente y todos los altavoces en unas pocas operaciones vectoriales, sin tener que recorrer listas de segmentos.

El objeto array no necesita ningún mecanismo especial de sincronización pues una vez construido el resto de objetos e hilos se limitan a emplearlo únicamente como lectores, operación que puede realizarse de forma concurrente sin problemas.

Lista de fuentes

La lista de fuentes mantiene un inventario de todas las fuentes virtuales que hay en la escena, tanto si están en uso como si no.

Este objeto si tiene un fuerte mecanismo de sincronización, su utilidad va más allá de una mera colección de fuentes, pues permite centralizar y sincronizar la ejecución de comandos que van dirigidos a todas las fuentes a la vez. Pongamos por ejemplo, que se recibe la orden `Pause`, esta orden debe ser ejecutada por todas las fuentes simultáneamente, lo que, en la práctica implica que todas las fuentes deben entrar en pausa al comenzar

el siguiente bloque de datos.

La plantilla que más se aproxima al trabajo que desarrolla este objeto es la de lectores/escritores con prioridad a lectores. Sin embargo requiere de ciertas modificaciones especiales puesto que debe sincronizar tres tipos de procesos en vez de dos. En concreto la emplean dos procesos lectores con prioridades distintas y un proceso escritor.

El proceso escritor será aquel que añada o elimine fuentes a la lista. Los otros dos procesos lectores, la emplean y pueden trabajar concurrentemente. El lector tradicional emplea la lista durante un tiempo muy corto, mientras que el proceso lector de alta prioridad emplea la lista durante un tiempo muy superior, en concreto, este proceso es el encargado de bloquear el acceso a los escritores con el fin de que no puedan hacer aparecer o desaparecer fuentes durante el procesado de un bloque.

Objetos menores

Como es normal, aparte de todos los objetos comentados en la realización del software se han empleado varios objetos más, los cuales son esenciales para el funcionamiento del software pero no revisten de especial importancia en cuanto al tratamiento de WFS se refiere. En la Tabla 4.1 se puede ver la relación resumida de estos objetos.

4.5.2. Paralelización de WFS

En esta sección se mostrará la división que se ha realizado de procesado de WFS en diferentes tareas, las cuales permitirán realizar gran parte del trabajo de forma concurrente.

Como ya se comentó anteriormente (véase la Sección 4.4.3), toda aplicación tiene un proceso principal o proceso padre, aquel proceso con el que

Objeto	Función
DriverASIO	Encapsula las funciones de acceso al *hardware* de sonido.
Estado	Permite llevar un seguimiento del estado en el que se encuentra el software.
ListaO	Mantiene una lista ordenada de todos los comandos que deben ser ejecutados por una fuente virtual.
Orden	Representa cualquier comando que pueda ser empleado sobre una escena.
Punto	Representa un punto en el espacio.
Vector	Representa un vector.
WAV	Encapsula las funciones de acceso y comprobación de los ficheros WAV.

Tabla 4.1. Relación de objetos menores.

el sistema arranca la ejecución de la aplicación, el cual puede crear varios hilos hijos para dividir y repartir el trabajo entre ellos. En el caso del motor de renderizado de WFS propuesto, este proceso se encarga de gestionar el interfaz de usuario el cual permite establecer la configuración, así como de realizar todos los trabajos *off-line* necesarios como son la creación de las estructuras que definen el array, la reserva y liberación de memoria, etc. También es el encargado de dar vida al resto de hilos, así como de esperar a que finalicen para proceder a su destrucción. Por todo esto, podemos decir que básicamente, la función de este hilo es la de realizar tareas de mantenimiento.

Todo el procesado de WFS lo realizan los hilos hijos, en concreto, se ha dividido todo el trabajo en cinco hilos distintos, aparte del proceso padre:

1. Procesador de las fuentes virtuales.

2. Procesador de la auralización.
3. Actualización sincronizada de la escena.
4. Gestión del acceso a disco.
5. *Streaming* de audio.

En las siguientes secciones se describirá el funcionamiento interno de cada uno de los hilos programados.

Procesador de las fuentes virtuales.

El procesador de las fuentes virtuales es el hilo principal de procesado. Su función es la de calcular las muestras de sonido correspondientes a cada canal y por tanto es el que tiene asociado el *hardware* de sonido. En la Figura 4.1 se puede ver en detalle la lógica de trabajo de este hilo. En primer lugar es necesario actualizar la escena, por tanto, se deben ejecutar todos los comandos que estén pendientes de ejecución. Después será necesario procesar todas las fuentes sonoras virtuales que haya en la escena, se recalcularán sus parámetros si es necesario (por ejemplo si se han movido al actualizar la escena) y seguidamente se obtendrán las muestras de sonido que deberemos enviar a cada canal para esa fuente.

Una vez hecho esto el siguiente paso es incluir el efecto de la auralización, esto es, la reverberación de la sala, por medio de varias fuentes virtuales que emitirán ondas planas, el sonido que emitirán estas fuentes se calcula en otro hilo aparte por lo que aquí se tratarán como cualquier otra fuente virtual con la salvedad de que estas fuentes no se van a mover, por lo que no será necesario recalcular sus parámetros.

A continuación se realizará un procesado opcional que, en caso de ser necesario, permite ecualizar cada canal por separado, aunque generalmente

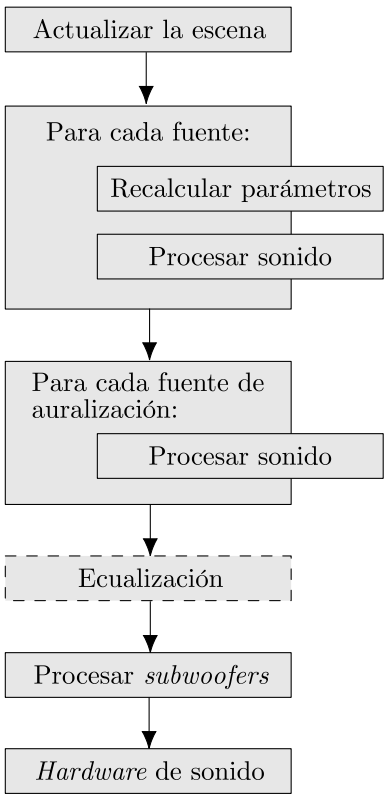


Figura 4.1. Lógica de trabajo del procesador de fuentes virtuales.

no se aplica pues con la ecualización global es suficiente.

Por último se calcula la señal que alimentará a los *subwoofers* y se sobreescribe la señal correspondiente a los canales en los que se han colocado los *subwoofers*. La señal resultantes es la que posteriormente se enviará al *hardware* de sonido.

Procesador de la auralización.

La inclusión del efecto de la sala es esencial para hacer creíble una escena sonora. El efecto de auralización generalmente se consigue filtrando

el sonido de cada canal con la respuesta al impulso de la sala a simular pero, en el caso de WFS, este proceso es más elaborado puesto que se intenta representar la naturaleza espacial de la sala.

Las reflexiones que se producen en una sala son diferentes dependiendo de la dirección, por tanto, no basta con emplear una única respuesta al impulso, será necesario emplear varias para recoger estos matices. Como se puede intuir, este procesado es una de las partes más costosas de la generación de escenas sonoras. Para minimizar el impacto computacional en [de Vries and Huselbos, 2004] se propone emplear un máximo de ocho respuestas al impulso, que representarán el carácter direccional de las reflexiones. Las respuestas al impulso se emplearán en otras tantas fuentes virtuales que emitirán ondas planas alrededor de la escena en las direcciones registradas (véase la Figura 4.2). El hecho de emplear fuentes con ondas planas permite mantener el carácter difuso de la reverberación. Además de la medición directa de las respuestas al impulso de una sala, es posible simular el comportamiento de la sala y obtener respuestas al impulso sintéticas, en [Escolano, 2008] se realiza un estudio riguroso sobre la implementación de mejoras en la técnica FDTD (*Finite Difference Time Domain*), las cuales pueden permitir caracterizar salas virtuales [Escolano et al., 2004a,b].

En general el tamaño de bloque empleado en la auralización suele ser mayor que en el procesado del resto de señales. Por un lado, un tamaño mayor permite disponer de más tiempo para procesar el bloque, que como se verá en la sección de procesado de la señal correspondiente, requiere de un procesado muy intensivo. Por otro lado, no es necesario dividir la respuesta al impulso de la sala en demasiados bloques, lo que al final conlleva un exceso de trabajo en la división y posterior unión de todos los bloques.

Esta diferencia de tamaño de bloque entre el hilo del procesador de

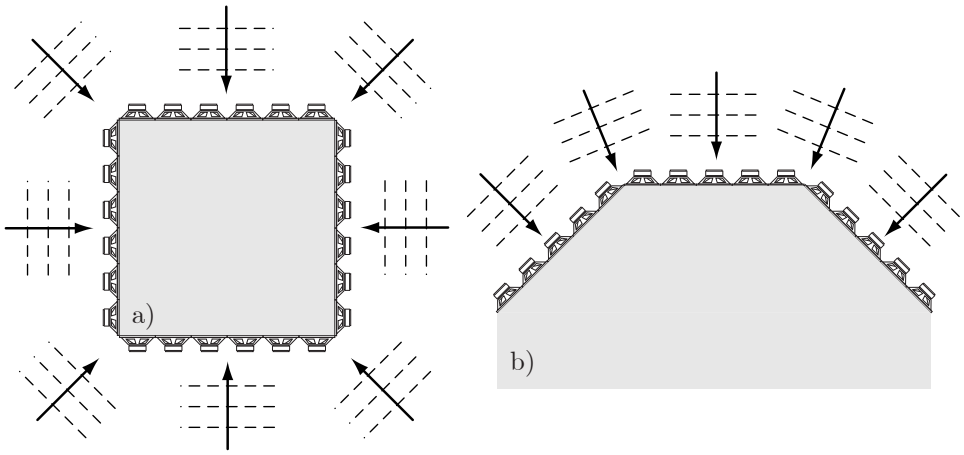


Figura 4.2. Ejemplo de distribución de las fuentes de auralización, a) Configuración en array cuadrado, b) Configuración en U abierta.

la auralización y de las fuentes virtuales obliga a que trabajen de forma asíncrona, por lo que es necesario crear un protocolo de sincronización entre ambos. Este protocolo es sencillo pues, cada vez que se procesa un bloque de datos de las fuentes virtuales, se envía ese bloque al *buffer* de entrada de la auralización. En caso de que con los datos disponibles más los que acaban de llegar formen más de un bloque de datos de auralización, se avisa al hilo de auralización para que despierte y lo procese. Como es normal todo este proceso va protegido por medio de un protocolo que emplea semáforos.

En la Figura 4.3 se puede ver la lógica de funcionamiento del hilo de auralización. Como se puede apreciar, el hilo permanece a la espera de recibir los datos suficientes para formar un bloque de datos. Una vez recibido un bloque completo de datos, se realiza su procesado, y el resultado es acumulado con el con el de los bloques previos. En la siguiente sección se describirá en profundidad el procesado realizado y el porqué de la acumulación con los datos anteriores.

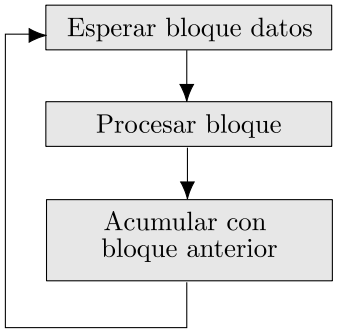


Figura 4.3. Lógica de trabajo del hilo de la auralización.

Actualización sincronizada de la escena.

Una escena sonora puede llegar a ser realmente compleja, y la sincronización de todos los elementos que la componen puede ser extremadamente complicada si no se planifica bien.

Un mecanismo que permite describir de forma natural la evolución de una escena es el paso de mensajes. Siempre que uno de los objetos se debe mover o cuando suceda cualquier evento, se genera el mensaje adecuado que avisará de los cambios que deben producirse.

Ya que los hilos de procesado realizan sus tareas en base a un conjunto de parámetros que pueden evolucionar con el tiempo, es necesario prestar especial atención a la forma en que estos pueden modificarse puesto que su alteración durante el procesado suele resultar nefasta.

Para simplificar al máximo las tareas de sincronización, se ha optado por la técnica de la centralización, la cual consiste en supeditar toda modificación o actualización a un gestor. Para ello se ha creado un doble juego de parámetros para todo objeto, estos juegos serán el actual y el futuro. Conforme se reciben comandos de actualización de la escena, se realizan los cambios pertinentes sobre el juego de parámetros futuro, los cambios nunca

se realizan sobre el juego actual por lo que no se ejecutan inmediatamente. Llegado el momento adecuado, el gestor intercambia todos los juegos de parámetros de forma simultánea.

Tal y como se comentó en la lógica de trabajo del procesador de fuentes virtuales, el primer paso que se realiza al comenzar un bloque es el cambio del juego de parámetros (véase la Figura 4.1), el juego futuro pasa a ser el actual y viceversa. De esta forma, aunque los comandos de actualización lleguen de forma asíncrona, se ejecutan todos a la vez, haciendo uso para ello del objeto *Lista de Fuentes*, antes de procesar un bloque, impidiendo de esta forma que se realize cualquier cambio durante su procesado. Cabe la posibilidad de que algunos comandos no lleguen a ser ejecutados nunca al ser sobrescritos por comandos posteriores, por ejemplo, si antes de cambiar el juego de parámetros llegan varios comandos de movimiento de una fuente sonora, solamente permanecerá el último de ellos puesto que la escena únicamente se actualiza al comienzo de cada bloque.

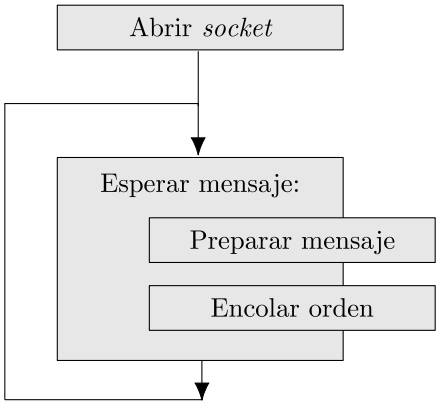


Figura 4.4. Lógica del gestor de recepción órdenes.

Ya se ha descrito cuando se procede a realizar el cambio del juego de parámetros, pero falta por determinar cómo se reciben los comandos. Los

comando de cambio o actualización de la escena se reciben por red, por ello, se ha creado otro hilo de funcionamiento asíncrono el cual se encarga de gestionar la recepción de los mensajes de actualización de la escena. Este hilo crea un socket UDP y espera hasta recibir algún mensaje. Una vez recibido un mensaje comprueba si está completo o es solo una parte, si el mensaje está incompleto, espera hasta obtener la parte faltante. Una vez se tiene un mensaje completo lo introduce en el juego de parámetros futuro del objeto correspondiente. En la Figura 4.4 se describe la lógica de funcionamiento del gestor de recepción de órdenes.

Los mensajes de actualización de la escena se envían en formato de texto plano, un ejemplo del formato de los mensajes es el siguiente:

```
source 1 play 1

source 2 gain 3.2
```

El primer mensaje reproduce la fuente 1, mientras que el segundo cambia la ganancia de la fuente 2 a 3.2.

En el apéndice B se detalla la lista con todos los mensajes soportados junto con su formato. El formato general de los mensajes se ha extraído de [Potard, 2006] ya que resulta ser un sistema muy útil e intuitivo, y sobre todo fácil de depurar, aunque por conveniencia se han añadido y/o eliminado determinados mensajes y se han hecho modificaciones sobre otros.

Gestión del acceso a disco.

En general el acceso a disco suele convertirse en un cuello de botella que disminuye notablemente el rendimiento de las aplicaciones y esto, en un sistema de reproducción de WFS no puede permitirse. Además de una gran potencia de cálculo es necesario acceder a disco para leer las muestras

sonoras de cada fuente virtual y en determinadas escenas puede haber gran cantidad de fuentes virtuales simultaneas. Por lo tanto, es necesario agilizar al máximo la gestión del acceso a disco y con ello evitar el deterioro del rendimiento.

Como ya se comentó en la Sección 4.2, cuando una tarea hace una petición de acceso a disco, independientemente de si esta petición es de lectura o escritura, el hilo desde el que se ha hecho la petición pierde el uso del procesador hasta que esta se ejecuta por completo. Como es de suponer, no deberían hacerse este tipo de peticiones en hilos que sean críticos para el funcionamiento de la aplicación en tiempo real. Así, se ha creado un hilo independiente que se encarga de gestionar el acceso a disco y cuyo funcionamiento es asíncrono con respecto al resto de hilos.

De todos los objetos que podemos encontrar en el sistema de síntesis WFS únicamente las fuentes virtuales necesitan acceder a disco durante la recreación de la escena sonora, el resto de elementos acceden a disco antes de empezar la reproducción por lo que no se les tendrá en cuenta aquí. Así, se ha añadido un *buffer* (al que denominaremos *buffer* previo) a cada fuente sonora, el gestor de acceso a disco se encarga de rellenar estos *buffers* con el audio de cada fuente sonora leyendo de disco. Su lógica de trabajo es bastante simple, recorre la lista de fuentes virtuales de forma cíclica comprobando el nivel de llenado del *buffer* previo. En caso de que quede sitio libre lee de disco un bloque de sonido y lo introduce en el *buffer*. Mientras alguna fuente haya necesitado datos continúa con este comportamiento, pero, si en la última pasada no ha realizado trabajo alguno, descansa unos milisegundos (medio bloque de tiempo) para no sobrecargar inútilmente al procesador. En la Figura 4.5 se puede ver la lógica de trabajo del gestor de acceso a disco.

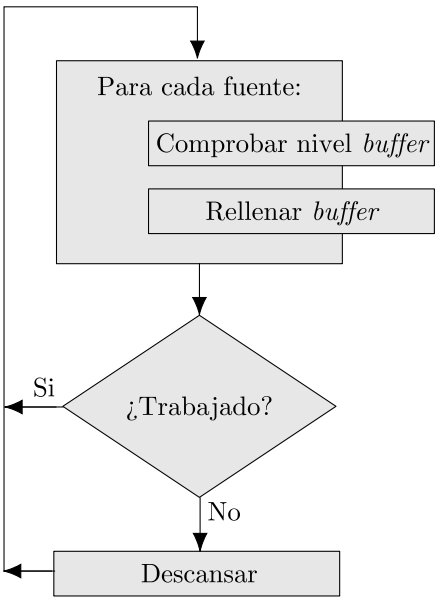


Figura 4.5. Lógica del gestor del acceso a disco.

La primera vez que se plantea este problema, se suele pensar, por inercia, como si se estuviera tratando con un sistema de reproducción multipista:

> «Se dispone de muchas fuentes sonoras (pistas) simultaneas que reproducir y se produce un cuello de botella en los accesos a disco, al haber tantos accesos aleatorios. Conclusión: la forma natural de reducir estos accesos es multiplexar todo en una única pista.»

Esta solución no es viable tratándose de un sistema de WFS ya que puede funcionar en directo, esto es, no se sabe cuando va a sonar determinada fuente ni si va a sonar tres veces seguidas por lo que la multiplexión a priori no es posible.

Una pregunta que cabe hacerse es por qué se ha empleado un único hilo para esta tarea cuando podrían haberse empleado más, incluso uno

por cada fuente virtual. El razonamiento que se ha empleado para elegir un único hilo depende de varios factores los cuales intentaré explicar a continuación:

- En primer lugar se debe intentar mantener la simplicidad ante todo. Es más fácil sincronizar un único hilo que multitud de ellos, y cuanto más sencilla es una tarea existe menor probabilidad de fallos.

- Otro factor a tener en cuenta es que cuantos más hilos tiene una aplicación más tiempo de procesador se gasta en los cambios de contexto, por tanto siempre hay que intentar mantener el número de hilos lo más bajo posible, esta recomendación se encuentra con bastante asiduidad en las páginas de información para programadores de la web de Microsoft. El número ideal es dos o tres hilos por procesador (o núcleo), actualmente los procesadores suelen tener un máximo de seis núcleos, aunque esto es sólo cuestión de tiempo que aumente.

- Por otro lado está el número de discos duros disponibles. Lo ideal es que hayan al menos dos, uno para el sistema operativo (y su fichero de intercambio, *swap*) y otro para los ficheros de sonido. Trabajar con un único disco puede acarrear situaciones problemáticas ya que el sistema tiene preferencia de acceso a disco sobre nuestra aplicación. Por el contrario, disponer de más discos puede aumentar el rendimiento al poder repartir los archivos y acceder a ellos concurrentemente.

- En cuanto al ratio de hilos por disco duro debería de ser uno. Es inútil intentar realizar dos lecturas de disco simultaneas si los datos se encuentran en posiciones distintas. Emplear más hilos suele obligar a que los hilos deban esperar a que termine no solo su petición sino

también todas las realizadas anteriormente por el resto, por lo que es un gasto inútil de recursos a la par que complica el funcionamiento.

Repasados todos los factores, se optó por emplear un único hilo ya que generalmente no se dispone más que de un disco duro, a lo sumo dos y, emplear más hilos no iba a mejorar el rendimiento. En todas las pruebas realizadas el gestor de un único hilo funciona sin problemas incluso con muchas fuentes en escena. Para conseguir esto, el gestor no espera a que las fuentes soliciten datos para ir a buscarlos, sino que se dedica a mantener lleno el *buffer* previo de las fuentes, por lo que dispone así de un tiempo de ventaja prudencial sobre las necesidades de estas.

Si se emplearan más discos se podrían repartir los archivos entre los discos, pero nunca se tiene la certeza de cual será la mejora conseguida pues depende de qué fuentes se reproduzcan y en qué momento.

Otra forma de mejorar el rendimiento del acceso a disco sería empleando un sistema de discos RAID de nivel 0 los cuales distribuyen la información entre varios discos, pudiendo de esta forma mover de manera simultánea cada cabezal por separado, sería «casi» como tener un disco duro con múltiples cabezas.

Streaming de audio.

Aparte del acceso a disco, se ha previsto otra vía por la cual la aplicación puede recibir el sonido de una fuente sonora y es por red en forma de un *stream* de audio.

De nuevo se ha optado por la máxima de la simplicidad y se ha construido el gestor de los *streams* empleando un único hilo, el cual actúa de servidor esperando a que le lleguen los *streams* por red, como es de esperar, su funcionamiento también es asíncrono puesto que no se sabe en que

momento se recibirán datos por la red. El protocolo de red empleado es el UDP (*User Datagram Protocol*) que, aunque no proporciona control frente a fallos, es mucho más ligero en cuanto a procesamiento se refiere que el TCP (*Transmission Control Protocol*). Además, trabajando en una red local no hemos detectado ningún problema de pérdida de paquetes, salvo en el caso de congestión extrema de la red.

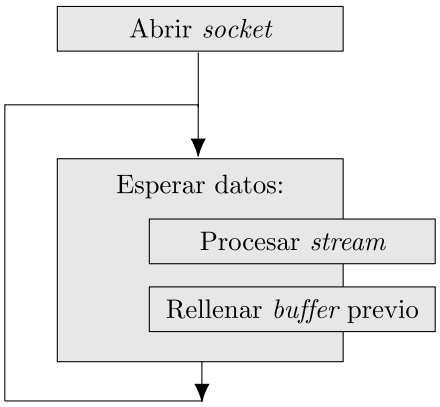


Figura 4.6. Lógica del gestor de *streams* de audio.

En cuanto al procesado a realizar, conforme se van recibiendo los *streams* se almacenan temporalmente hasta conseguir un bloque de audio completo y se introduce en el *buffer* previo de la fuente sonora, todo este procedimiento se muestra en la Figura 4.6. La forma de determinar a qué fuente pertenece un determinado *stream* es por el puerto IP (*Internet Protocol*) por el que se envía.

4.5.3. Procesado de la señal

Si en las dos secciones anteriores se ha mostrado la arquitectura interna del software, representando por un lado la división de la información mediante objetos y por otro la división del trabajo mediante hilos, los cua-

les se ejecutarán de forma concurrente pero sincronizada, en esta sección se describirá el procesado de señales que debe realizar el software de WFS.

En primer lugar se mostrará el diagrama de bloques general para posteriormente ir profundizando en cada uno de los bloques que componen el diagrama.

Procesado global de WFS

En la Figura 4.7 se puede ver el diagrama de bloques que representa el procesado global realizado por la aplicación.

Antes de proceder a explicar el diagrama, es necesario indicar que en esta figura hay dos tipos de flechas distintas. El tipo de flecha que se representa por líneas finas indica que las diferentes señales que viajan por ella lo hacen en serie, mientras que los tipos de flechas gruesas representan el paso de señales en paralelo.

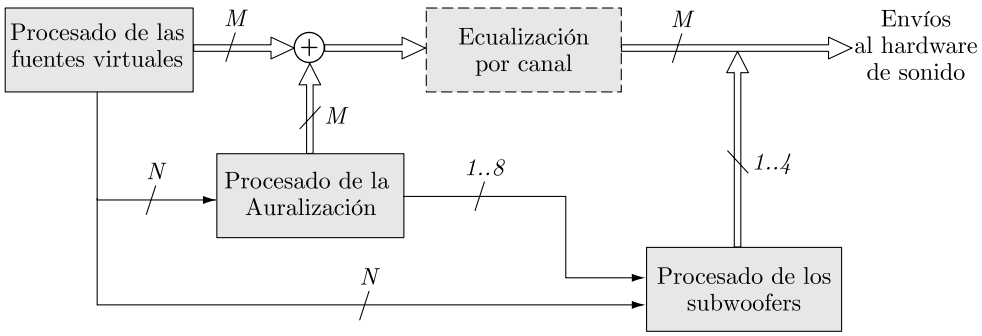


Figura 4.7. Diagrama de bloques del procesado global.

Tal y como está descrito en el diagrama, por un lado, se procesan las fuentes virtuales, el sonido de todas ellas se envía al procesador de la auralización y al procesador de los *subwoofers*.

El procesador de la auralización realiza su cometido y la señal resultan-

te se mezcla con la de las fuentes virtuales. Seguidamente se puede aplicar una ecualización opcional independientemente para cada canal. Al resultado se le superpone la señal procesada por el módulo de los *subwoofers* y en último lugar se envía la señal al *hardware* de sonido.

Procesado de una fuente virtual

El procesado particular de cada fuente virtual se realiza de acuerdo al diagrama de bloques de la Figura 4.8.

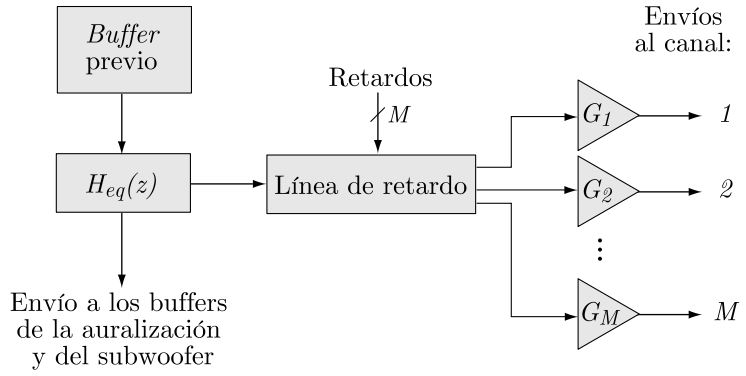


Figura 4.8. Procesado de la señal de las fuentes virtuales.

En primer lugar se obtiene un bloque de muestras de sonido del *buffer* previo de la fuente virtual. Este bloque se filtra por el filtro de ecualización, el cual se encarga de asegurar una respuesta en frecuencia plana por parte del sonido emitido por el array, por lo que incluye el filtro de WFS de +3 dB por octava. En la implementación realizada el filtro es de tipo IIR pues el filtrado es mucho más eficiente que con los FIR. Los filtros empleados en la ecualización han sido diseñados con la técnica descrita en [Ramos and López, 2006].

La señal filtrada se introduce en la línea de retardo y se envía tanto

al *buffer* del *subwoofer* como al de la auralización. Conocidas la posición de la fuente y las posiciones de los altavoces, se pueden calcular todos los parámetros necesarios: distancias, retardos y amplitudes. El siguiente paso es leer tantas veces de la línea de retardo como canales hayan, aplicándose la ganancia correspondiente a cada canal. El resultado se envía al *buffer* de cada canal. Las lecturas de la línea de retardo que impliquen el movimiento de la fuente se realizarán de acuerdo a lo descrito en la Sección 3.6, teniendo en cuenta si se debe o no aplicar el efecto *doppler*.

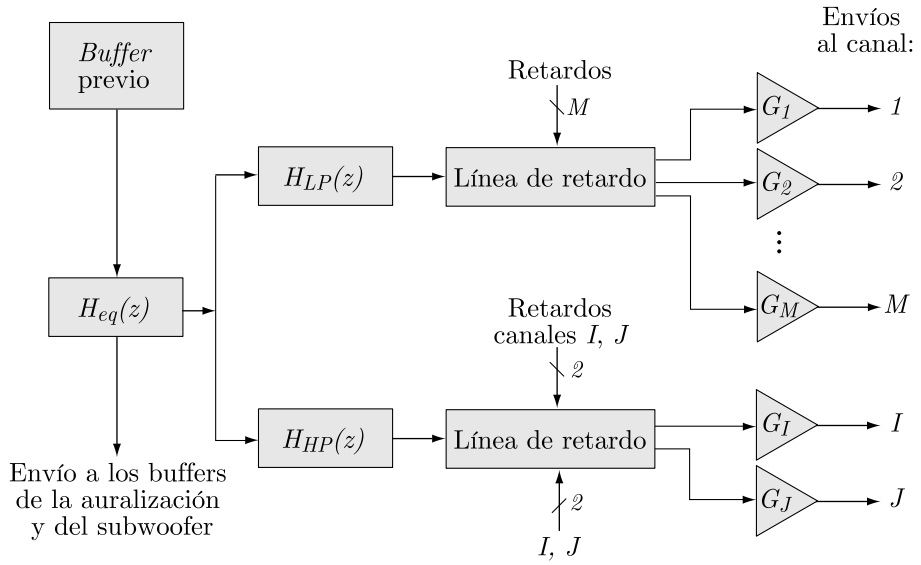


Figura 4.9. Procesado de la señal de las fuentes virtuales mediante la aproximación por subbandas.

En caso de que la fuente emplee la aproximación por subbandas para minimizar el *aliasing* espacial, en vez del diagrama anterior, se emplea el diagrama de la Figura 4.9. Tal y como se describe en la Sección 3.7.2 la señal se divide en dos bandas, la paso alto y la paso bajo. El procesado de la banda paso bajo es idéntico al de las fuentes virtuales normales, mientras que la

paso alto se procesa de forma completamente distinta. Para la banda paso alto se localizan los dos altavoces más representativos, representados en el diagrama por las letras I y J, y se determinan los parámetros únicamente para ellos. Seguidamente se leen las muestras de la línea de retardo, se aplican las ganancias y se envía la señal al *buffer* del canal, en los cuales se sumarán a las señales existentes.

Procesado de los *subwoofers*

El procesado de los *subwoofers* es bastante sencillo pues sólo requiere de un filtrado paso bajo, el cual se ha impuesto que sea común a todos los *subwoofers* para no sobrecargar al procesador (véase la Figura 4.10).

Todo este procesado se realiza en el mismo hilo que procesa las fuentes virtuales pues el procesado de estas debe haber finalizado para poder comenzar este. Debe tenerse en cuenta que realizar el procesado en un hilo a parte no hubiera mejorado la eficiencia pues ambos procesados se deben realizar en serie.

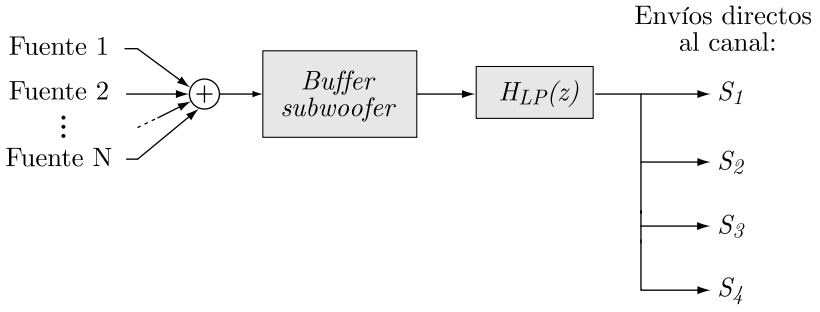


Figura 4.10. Procesado de la señal de los *subwoofers*.

Auralización

Para generar la auralización es necesario realizar el procesado que se muestra en la Figura 4.11.

En primer lugar se acumulan las señales que habrán ido llegando de todas las fuentes virtuales. Seguidamente se filtran por el filtro de WFS de +3 dB por octava, ya que la auralización emplea fuentes de ondas planas, el efecto del filtro de WFS es completo.

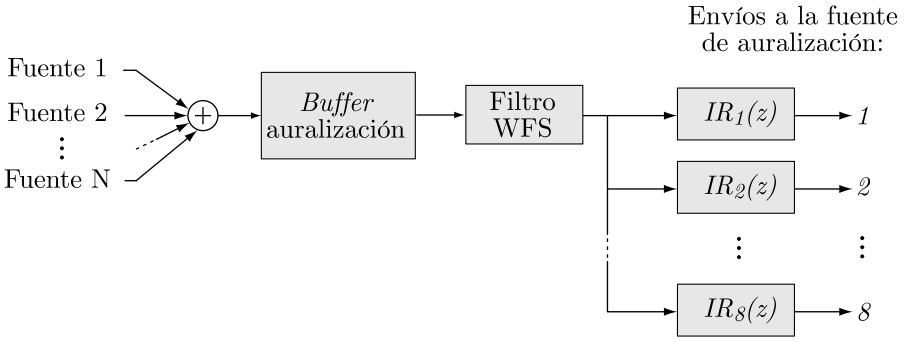


Figura 4.11. Procesado de la señal realizado en la auralización.

A continuación se filtra la señal por cada una de respuestas al impulso direccionales. Las señales resultantes serán empleadas por separado para alimentar a cada una de las fuentes virtuales de auralización.

La señal de las fuentes de auralización se procesa de forma análoga al de las fuentes virtuales, salvo por el hecho de que la señal no entra al *buffer* previo como en las otras, sino que entra directamente a la línea de retardo ya que, la ecualización ya se ha realizado anteriormente.

Tal y como se desprende de los diagramas de bloques, el procesado de la auralización se realiza en dos hilos distintos, en el hilo de auralización se determinan únicamente las señales que deben alimentar a las fuentes

virtuales de auralización. El procesado de estas, se realiza en el procesador de las fuentes virtuales, justo a continuación del procesado del resto de fuenttes, tal y como se comentó en el esquema de la lógica de trabajo del hilo de la auralización (véase la Figura 4.8).

Convolución particionada

Ya que el procesado de la auralización precisa de la convolución de la señal de audio con las respuestas al impulso de la sala, las cuales pueden alcanzar una duración de varios segundos, es necesario realizar estas convoluciones por medio de un algoritmo eficiente. La opción evidente es realizar la convolución en frecuencia multiplicando espectros, pero esta solución introduce una latencia igual a la duración de la respuesta al impulso, la cual es excesiva.

Para solucionar este problema, en la implementación del software se optó por emplear la convolución particionada [Torger and Farina, 2001], la cual consiste en dividir la respuesta al impulso en N particiones uniformes de menor tamaño, convolucionar la señal con cada una de ellas y luego unir los productos intermedios de forma que al final obtenemos el mismo resultado que con la convolución directa pero con una latencia N veces menor. En concreto, el funcionamiento de la convolución particionada es el siguiente:

Sea $h[n]$ de tamaño $L = N \cdot M$, $h[n]$ se puede expresar como:

$$h[n] = \sum_{i=0}^{N-1} h_i \left[n - iM\right] \tag{4.1}$$

donde $h_i[n]$ es de tamaño M y se define como:

$$h_i[n] = h[n] \cdot w[n - iM] \tag{4.2}$$

siendo $w[n]$ una ventana rectangular de tamaño M. Por tanto, la convolución de x con h se puede expresar como

$$\begin{aligned} y[n] &= x[n] * h[n] && (4.3)\\ &= x[n] * \sum_{i=0}^{N-1} h_i\left[n - iM\right] && (4.4)\\ &= x[n] * \sum_{i=0}^{N-1} h_i[n] * \delta\left[n - iM\right] && (4.5)\\ &= \sum_{i=0}^{N-1} h_i[n] * x\left[n - iM\right]. && (4.6) \end{aligned}$$

Para minimizar el coste computacional el cálculo de las convoluciones intermedias se realiza en frecuencia empleando el algoritmo *overlap-save*. Aunque el uso de la convolución particionada no reduce la carga computacional, de hecho la incrementa ligeramente, en general es más rápida de calcular que la convolución directa ya que al realizar las operaciones con bloques de menor tamaño se producen muchos menos fallos de caché en el procesador [Torger and Farina, 2001].

En la Figura 4.12 se muestra en detalle el diagrama de bloques de la implementación realizada, en concreto, se muestran los pasos necesarios para convolucionar el bloque i-ésimo de la señal $x[n]$ con el filtro $h[n]$, el resultado es la señal $y_i[n]$ la cual tendrá que solaparse con el resultado del bloque siguiente $y_{i+1}[n]$. Por simplicidad, en esta figura no se muestran los pasos intermedios de la técnica *overlap-save*, es decir, no se muestran ni el aumento del tamaño de los bloques al pasar del dominio del tiempo al de frecuencia mediante *zero padding* ni tampoco el desechar las primeras muestras al regresar al dominio del tiempo.

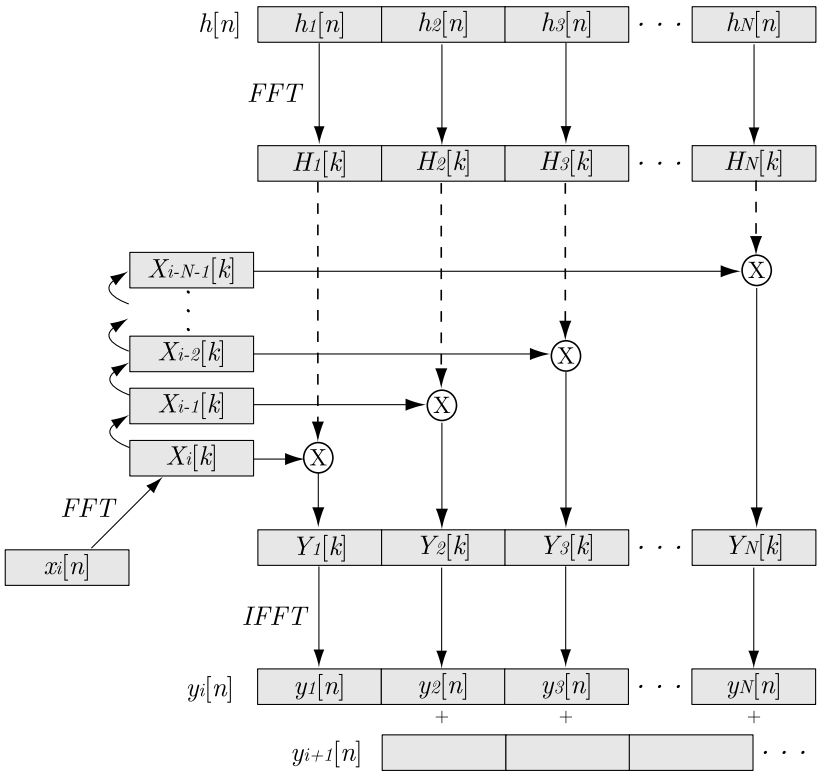


Figura 4.12. Cálculo de la convolución particionada.

4.5.4. Comentarios sobre la eficiencia

Algunas implementaciones de sistemas de WFS, por ejemplo [Baalman, 2004], crean un mallado por todo el área en la cual se permite el movimiento de las fuentes sonoras. Para cada punto del mallado se precalculan los filtros necesarios que se emplearán posteriormente. Con estos filtros se aplicará la ecualización junto con el retardo fraccionario deseado para cada altavoz. Así, al mover una fuente por el mallado se selecciona el filtro adecuado y, en caso de ser una posición intermedia se interpolan los coeficientes del filtro. Para realizar el filtrado de forma eficiente emplean la librería de

procesado *BruteFIR* [Torger, 2001].

Al aplicar de forma combinada el retardo fraccionario y la ecualización se necesitan filtros con muchos coeficientes, y además, el filtrado debe realizarse de forma independiente por cada par fuente-altavoz (véase el caso a) de la Figura 4.13). Así, es necesario hacer muchos filtrados costosos por cada fuente, adicionalmente, los filtros deben ser FIR para que se puedan interpolar los coeficientes de los filtros de las posiciones intermedias del mallado sin tener problemas de estabilidad. Como ya se ha comentado, esta aproximación obliga a realizar un filtrado muy costoso.

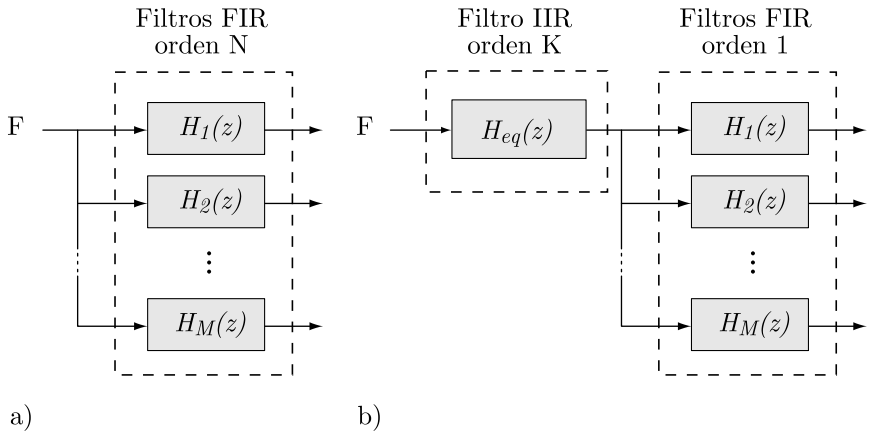


Figura 4.13. Esquemas de filtrado de una fuente. a) Filtrado conjunto. b) Separación de la ecualización y del retardo fraccionario. Órdenes de los filtros: $N >> K > 1$.

En su lugar, se ha preferido separar los filtros de ecualización de los de retardo fraccionario pues se mejora notablemente el rendimiento (véase el caso b) de la Figura 4.13). Por un lado se puede aplicar la ecualización a la señal de la fuente, la cual es relativamente costosa, una única vez y sobre esta señal ya ecualizada se aplica el retardo fraccionario individual para cada canal. Además al no tener que interpolar los coeficientes de los

filtros de ecualización, es posible sustituir el filtro FIR por uno equivalente IIR mucho más eficiente, ya que este permite conseguir el mismo resultado con muchos menos coeficientes.

4.6. Prototipos

Aunque esta sección se podría haber situado en un apéndice, se ha creído conveniente situarla aquí pues el desarrollo de la arquitectura de implementación propuesta ha ido ligada estrechamente con la evolución de los prototipos empleados.

Los diferentes prototipos han sido desarrollados en conjunto por los miembros del grupo de trabajo del *i*TeAM de la Universidad Politécnica de Valencia y del grupo de Teoría de la Señal de la Universidad de Alicante.

En primer lugar se mostrará el *hardware* que hace que sea posible gestionar 96 altavoces a la vez de forma sincronizada, para, a continuación mostrar los distintos prototipos construidos. Todos los prototipos construidos mantienen la misma separación entre altavoces (18 cm) lo que implica que su frecuencia de *aliasing* es común y tiene un valor aproximado de 1 kHz.

4.6.1. *Hardware* de conversión D/A y amplificación

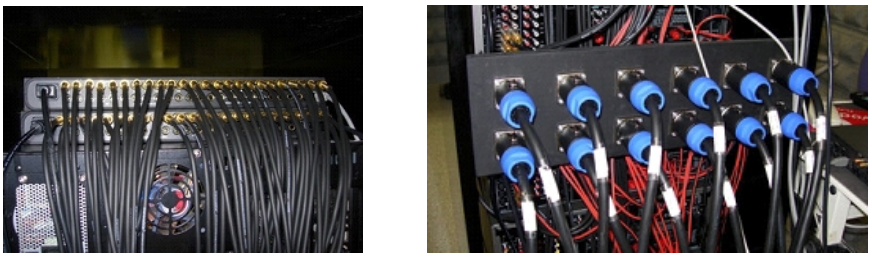
El *hardware* de conversión D/A empleado es un conjunto de cuatro tarjetas 24I/O de MOTU (véase la Figura 4.14). Una sola de estas tarjetas proporciona 24 canales de salida y otros tantos de entrada, siendo gestionada por una tarjeta controladora que emplea el interfaz *AudioWire*[3]. Esta tarjeta controladora permite combinar hasta cuatro módulos, con lo que

[3]Protocolo propietario de la casa MOTU que emplea la capa física del *firewire* aunque no es compatible con este.

se consiguen 96 canales simultáneos. La primera prueba que se realizó con la controladora fue la comprobación con un osciloscopio de la buena sincronización de los cuatro módulos al funcionar en paralelo. Las pruebas realizadas fueron más que satisfactorias, por lo que el problema de tener que sincronizar varias tarjetas quedó resuelto con la elección.

Figura 4.14. *Hardware* de conversión D/A, MOTU 24I/O.

Como equipos de amplificación se emplearon 14 amplificadores Yamaha de 7.1 canales cada uno, lo que proporciona soporte para los 96 canales de sonido y deja dos libres para suplir posibles fallos. En la Figura 4.15 se muestra un detalle del cableado del *hardware* de sonido con los amplificadores. Con el fin de minimizar el número de conexiones, se empleó conectores *speakon* de ocho polos para conectar los amplificadores a los arrays.



a) b)

Figura 4.15. Cableado del prototipo. a) Conexión con el *hardware* de sonido. b) Detalle de los *Speakons*.

El ordenador empleado en el prototipo es un Pentium IV a 3.4 GHz con un único núcleo, el cual, aún disponiendo solamente de *Hyper-Threading* (véase la Sección 4.2), es capaz de gestionar fácilmente los 96 canales con

30 fuentes virtuales simultáneas.

4.6.2. Arrays de altavoces

Para mostrar los distintos arrays se ha establecido la división según el tipo de altavoces empleados: dinámicos y planos (DML).

Arrays de altavoces dinámicos

- Prototipo de array cerrado compuesto por 96 canales (véase la Figura 4.21) cuyas características son:

  - Altavoces de medios de 5" y *tweeters* de 1"
  - Filtro *cross-over* pasivo.
  - Compuesto por 12 segmentos de 8 altavoces.
  - Separación entre altavoces: 18 cm.

  El array está montado en configuración de octógono aunque se puede establecer cualquier otra forma. Este prototipo se encuentra en los laboratorios del *i*TeAM en la Universidad Politécnica de Valencia.

Figura 4.16. Prototipo de array de 96 canales dinámicos.

- Prototipo de array abierto compuesto por 32 canales (véase las Figuras 4.17 y 4.18) cuyas características son:

  - Altavoces de medios de 5" y *tweeters* de 1".
  - Filtro *cross-over* pasivo.
  - Compuesto por 4 segmentos de 8 altavoces.
  - Separación entre altavoces: 18 cm.

  El array se suele montar en configuración de U abierta, U cerrada o como array lineal. Este prototipo se encuentra en los laboratorios del *i*TeAM en la Universidad Politécnica de Valencia.

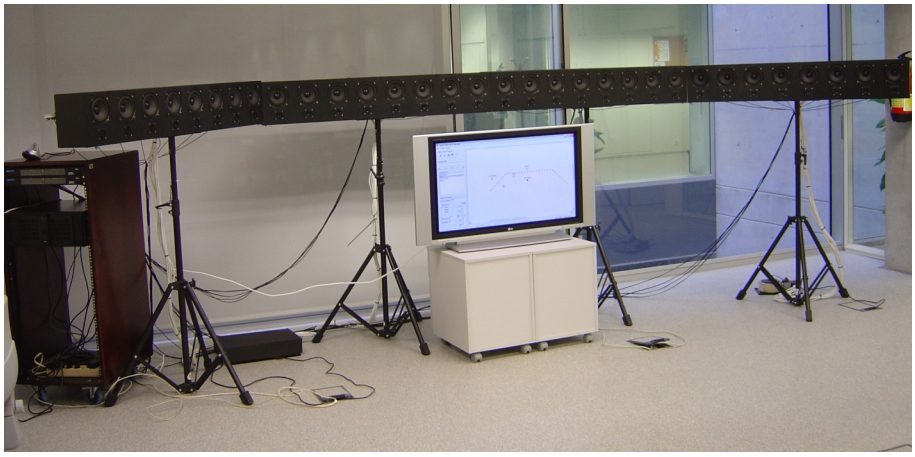


Figura 4.17. Prototipo de array de 32 canales dinámicos.

- Prototipo de array abierto compuesto por 24 canales (véase la Figura 4.19) cuyas características son:

  - Altavoces de medios de 5".
  - Compuesto por 3 segmentos de 8 altavoces.
  - Separación entre altavoces: 18 cm.

Figura 4.18. Detalle de los altavoces del prototipo de 32 canales.

El array se suele montar en configuración de U abierta, U cerrada o como array lineal. Este prototipo se encuentra en el Laboratorio de Teoría de la Señal del Departamento de Física, Ingeniería de Sistemas y Teoría de la señal (DFISTS) de la Universidad de Alicante.

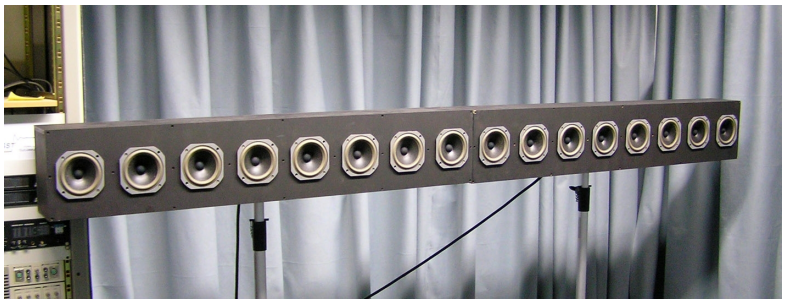


Figura 4.19. Prototipo de array de 24 canales dinámicos.

Arrays de altavoces planos DML

- Prototipo de cinco paneles de altavoces DML (véase la Figura 4.20) cuyas características son:

  - Tamaño pequeño (54x35 cm).

- Paneles de policarbonato en forma de celda de abeja de 5mm de espesor.
- 3 excitadores por panel.
- Separación entre excitadores: 18 cm.

El array está montado en configuración lineal y se encuentra en el Laboratorio de Teoría de la Señal del DFISTS en la Universidad de Alicante.

Figura 4.20. Prototipo de array con 5 paneles de altavoces planos.

- Prototipo de tres paneles de altavoces DML (véase la Figura 4.20) cuyas características son:

  - Tamaño medio (90x79.5 cm).
  - Paneles de policarbonato en forma de celda de abeja de 5mm de espesor.
  - 5 excitadores por panel.
  - Separación entre excitadores: 18 cm.

  El array está montado en configuración lineal y se encuentra en el Laboratorio de Teoría de la Señal del DFISTS en la Universidad de Alicante.

Figura 4.21. Prototipo de array con 3 paneles de altavoces planos.

4.7. Conclusiones

En el presente capítulo se ha descrito la arquitectura software propuesta para implementar un motor de renderizado de WFS en tiempo real.

Esta arquitectura permite crear el motor sin necesidad de emplear conjuntos DSPs ni una combinación de ordenadores personales en red. La elección de esta estrategia se debe principalmente a los inconvenientes de las otras, por un lado, el uso de DSPs encarece de forma notable la instalación requerida, mientras que por otro lado, el uso de una red de ordenadores requiere un mecanismo de sincronización muy complejo. Al realizar todo el procesado en un mismo ordenador se reduce tanto el coste como la complejidad de la instalación, pero a cambio es necesario exprimir al máximo sus capacidades de cómputo.

Para ello se han revisado los diferentes aspectos sobre los cuales los procesadores actuales consiguen alcanzar eficiencias tan elevadas y se ha enfocado la programación a potenciar estas características. Por un lado se aprovecha la capacidad de procesado en paralelo, a la vez que se minimizan los tiempos muertos de los procesadores, mediante la programación con-

currente, teniendo en cuenta la buena sincronización de todos los procesos simultáneos, para evitar las condiciones de carrera y los interbloqueos. Por otro lado se aprovecha las técnicas de procesado vectorial minimizando a la vez el esfuerzo requerido para ello. Se emplea la librería de procesado de señal de Intel, la cual está programada por la propia casa que diseña los procesadores, por lo cual consigue sacar el máximo rendimiento de estos. Por último, se ha tenido en cuenta el hecho de trabajar con arquitecturas segmentadas, que, aunque en un primer momento pueda parecer que su uso no conlleva más que ventajas, si no se es cuidadoso en la programación puede tener un impacto negativo importante de cara al rendimiento. Por ello se ha estudiado con detenimiento todas las situaciones susceptibles de provocar un vaciado del cauce de procesado de instrucciones, y se ha actuado en consecuencia.

Aparte de enfocar la programación a sacar el máximo rendimiento de los procesadores, también ha sido necesario planificar las tareas de forma eficiente, pues aunque los procesadores tengan mucha capacidad de cálculo, un mal planteamiento puede anular por completo esta capacidad. Así, se han dividido los datos en unidades funcionales coherentes, mediante objetos, y la realización del trabajo se ha llevado a cabo en diversos hilos. Por último, se ha optimizado el procesado las señales de WFS, siendo esta parte crítica para WFS.

Con todo ello se ha conseguido implementar el procesado de forma tan eficiente que permite generar escenas sonoras complejas que emplean la totalidad de los canales, a la par que se añade la auralización de forma dinámica, con reverberaciones que pueden durar varios segundos.

La arquitectura diseñada ha obtenido el reconocimiento de expertos en el área, los cuales han podido comprobar su buen funcionamiento en los

laboratorios que el *i*TeAM tiene en la Universidad Politécnica de Valencia.

Capítulo 5

Arquitectura software para la producción de sonido espacial en WFS

WFS es una técnica de reproducción de sonido espacial muy prometedora a la par que reciente, y es esta novedad, esta falta de madurez, uno de sus mayores inconvenientes para ser una técnica ampliamente utilizada.

En la actualidad toda la industria discográfica está dedicada a la producción de sonido basada en el modelo multipista, el cual no concuerda con la filosofía de una producción de WFS. Cuando a un ingeniero de sonido se le presenta un prototipo de WFS se asombra de las posibilidades, pero, se encuentra con varios problemas que es necesario solventar. Toda su experiencia profesional está ligada al modelo multipista, sus herramientas de trabajo se quedan desfasadas y además, su concepción de lo que es una mezcla sonora debe cambiar.

Con el fin de abordar este problema, en este capítulo se mostrará la

arquitectura software desarrollada la cual podrá ser empleada en cualquier estudio de grabación para la realización de composiciones con la técnica de WFS sin dejar de lado las técnicas existentes de grabación multipista.

5.1. Introducción

Cuando en un estudio de grabación se realiza un proyecto, se concluye con la realización del denominado «máster». El máster es el producto a desarrollar, es el objeto del cual se realizarán innumerables copias en diversos formatos para proceder a su distribución y venta.

Los másters en la actualidad van orientados a un conjunto fijo de pistas, si es un CD de música, dos pistas, si es una película en DVD, seis pistas, y así sucesivamente. Durante el proceso de producción, el ingeniero emplea una DAW (*Digital Audio Workstation*) o un software que realiza las mismas funciones para realizar las mezclas (véase la Figura 5.1), cada sonido es asignado a una pista diferente para obtener al final una mezcla con el número de pistas deseado. El equipo reproductor únicamente debe leer las pistas tal cual están almacenadas y enviarlas a cada canal, es decir, la reproducción consiste en un simple proceso de asignar cada pista a un canal. En el caso de que no se disponga de suficientes canales en reproducción, simplemente se realiza una mezcla (*downmix*) para acomodar todas las pistas en los canales disponibles. Además, para que la reproducción sea correcta, los altavoces deben situarse alrededor del oyente en unas posiciones concretas (para más información véase el Capítulo 2).

Si nos centramos ahora en cómo sería el máster de un proyecto de WFS nos damos cuenta de que es completamente distinto. Ahora, una escena sonora se representa mediante objetos que aparecen en determinados instantes, se desplazan por la escena y eventualmente desaparecen. Además,

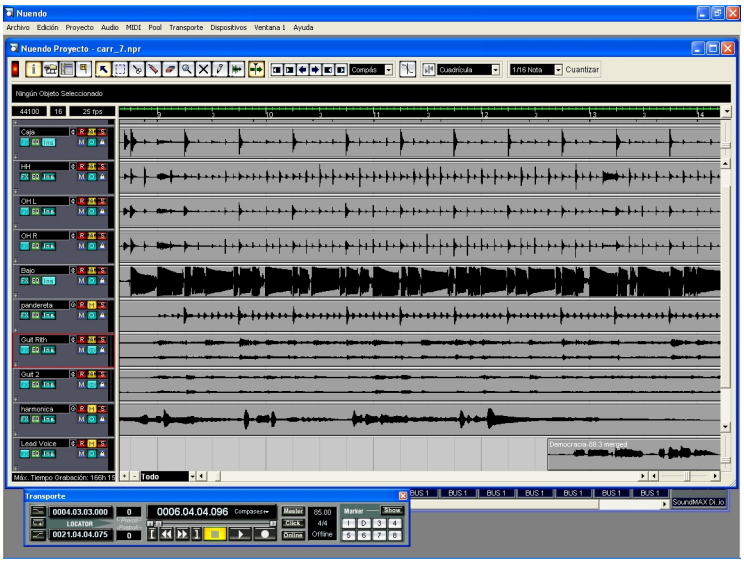


Figura 5.1. Ejemplo de un programa de producción multipista.

el sistema de reproducción no está orientado a un número fijo de pistas, el sonido que va a ser enviado a cada canal se determina dinámicamente durante la reproducción, una misma composición se puede reproducir tanto con 32 canales como con 100. Esto es, en este caso el equipo reproductor debe ser inteligente, debe realizar un procesado muy complejo.

Como podemos ver, son dos visiones completamente diferentes de un mismo problema, el proceso de autoría es radicalmente distinto está orientado a objetos en vez de a pistas, por tanto, es necesario adaptar los sistemas de producción actuales para que la industria pueda producir proyectos de WFS en masa [Pellegrini and Kuhn, 2004].

5.2. Estrategias para el desarrollo de aplicaciones de autoría de WFS

Como se ha descrito en la sección anterior, WFS necesita que se desarrollen herramientas que permitan realizar el proceso de autoría de una forma sencilla. En la actualidad existe una total carencia de herramientas intuitivas, las pocas que hay no son triviales de utilizar y en su contra también tienen el coste económico el cual es muy elevado. En la presente sección vamos a evaluar las posibles estrategias a abordar para llegar a producir proyectos de WFS de forma confortable en cualquier estudio de grabación.

5.2.1. Aplicación autónoma

Como primera aproximación al problema se puede pensar en crear una aplicación cerrada que permita la autoría de proyectos de WFS, esto es, una aplicación que sustituya a las DAW actuales a la hora de producir proyectos de WFS dejando a los proyectos multipista su sistema de producción intacto. Como todos los métodos de trabajo, tiene sus ventajas e inconvenientes, una gran ventaja es que minimizamos el impacto en la cadena de producción musical puesto que los proyectos multipista no se ven afectados, únicamente es necesario adaptarse para producir proyectos de WFS. Sin embargo, el número de desventajas es muy superior, la creación de esta aplicación requiere de un gran esfuerzo pues para ser una alternativa seria es necesario añadir además de los elementos de WFS todos los recursos que permiten las DAW, como son las herramientas de edición, filtros, *plug-ins*, etc. Por otro lado, se va a sobrecargar de trabajo al equipo que debe recrear la escena ya que la reproducción de escenas de WFS requiere de una gran potencia de cálculo y además debemos añadir todos los elementos que

contienen las DAW actuales que también necesitan una potencia de cálculo nada despreciable. Otro factor a tener en cuenta es que estamos obligando a duplicar el trabajo para producir un mismo producto en formato multipista y WFS. Por último, nos encontramos otro inconveniente añadido, y es que para reproducir un proyecto ya creado necesitamos emplear el propio software de autoría. Sopesando los pros y los contras queda claro que esta no es la aproximación adecuada.

5.2.2. Actualización de las aplicaciones existentes

Una segunda aproximación es adaptar los sistemas de producción actuales para que puedan producir proyectos de WFS. En [Pellegrini and Kuhn, 2004] se propuso realizar esta adaptación a través de *plug-ins*. La mayor ventaja de este sistema es la facilidad que encuentra el ingeniero de sonido para emplearlo ya que se sigue trabajando con la misma aplicación usada hasta el momento. Sin embargo, también encontramos un buen número de desventajas debido a la naturaleza del proceso de producción multipista. Los sistemas actuales están orientados a pistas y no a objetos (o fuentes virtuales), por lo que este último concepto lo debe introducir el *plug-in*. Cada vez que se desee introducir una nueva fuente virtual en la escena, se intercala una nueva instancia del *plug-in* en la cadena de proceso de *plug-ins* de la pista deseada, por lo que habrá tantos *plug-ins* como fuentes virtuales tenga la escena. Esto último introduce un inconveniente pues será necesario sincronizar todas estas instancias del *plug-in* ya que únicamente una de ellas debe tener acceso al *hardware* de audio. Esta instancia hará las funciones de instancia maestra y realizará todo el procesado de WFS, el resto de instancias actuarán como esclavas y únicamente transmitirán el sonido, y en su caso los parámetros necesarios, a la instancia

maestra. Por último, nos encontramos dos inconvenientes que ya tenía la aproximación anterior, seguimos sobrecargando la máquina al obligarla a soportar además del procesado típico de la DAW todo el necesario para WFS y, además, para reproducir un proyecto ya creado necesitamos emplear el propio software de autoría.

Como se puede deducir fácilmente, este tampoco es el camino a seguir, es necesario encontrar una forma distinta de trabajar que evite todos los inconvenientes encontrados.

5.2.3. Estrategia híbrida

En tercer lugar, se puede abordar el problema desde una aproximación híbrida de las dos anteriores, intentando solventar de esta forma los inconvenientes de cada una. Para esta aproximación se creará una aplicación para realizar el procesado de WFS la cual residirá en una máquina independiente y se conectará con las DAW actuales a través de *plug-ins*. La aplicación independiente ya no será tan autónoma y hará la función del *plug-in* maestro por lo que los *plug-ins* que se emplean en la DAW serán todos esclavos, es decir, únicamente transmitirán los parámetros y las muestras de sonido a la aplicación autónoma. Desde este punto de vista, los *plug-ins* son un simple *bypass* por lo que su interferencia en la aplicación en la que residen es mínima.

Siguiendo esta filosofía se eliminan muchos de los problemas encontrados en las anteriores. En primer lugar, podemos solventar la sobrecarga del procesador al separar el procesado de la DAW del específico de WFS empleando para ello dos máquinas diferentes. En segundo lugar, para reproducir un proyecto basta con emplear la aplicación autónoma por lo que no es necesario incluir la DAW.

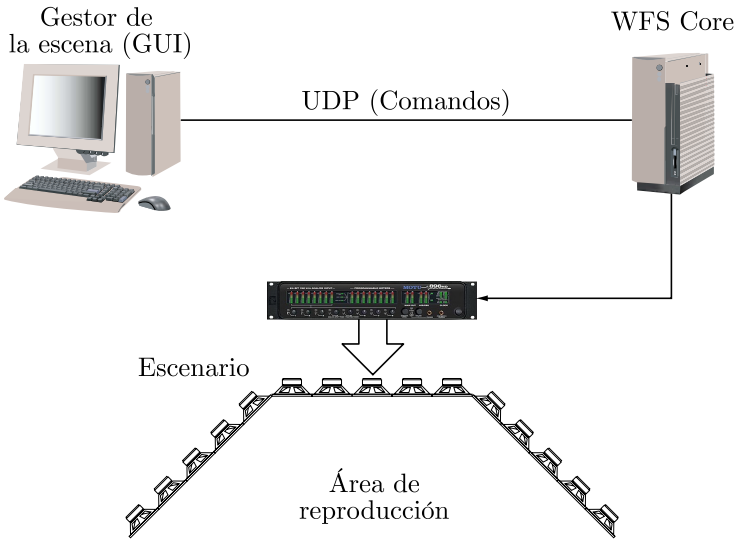


Figura 5.2. Estrategia híbrida.

5.3. Arquitectura de producción de sonido propuesta

Con el fin de flexibilizar al máximo el proceso de autoría de un proyecto de WFS se ha optado por una arquitectura abierta que sea capaz de adaptarse a todos los posibles escenarios de trabajo. Con estos objetivos se han desarrollado tres aplicaciones o módulos independientes, cada uno de los cuales tiene una función bien diferenciada que se basa en la estrategia híbrida pero con algunas modificaciones que la hacen más flexible.

Los tres módulos creados son:

- Núcleo de renderizado

- *Plug-in* VST

- Gestor de las escenas

El primer módulo es el núcleo de renderizado, al que denominamos «render». Su función es la de recrear la escena sonora a partir de la señal de cada fuente virtual y las órdenes de actualización de la escena que se le vayan indicando, esto es, es el motor de generación de las escenas de WFS, su funcionamiento interno se ha descrito en profundidad en el Capítulo 4. Con el fin de maximizar su utilidad se le ha añadido la posibilidad de renderizar escenas mediante las configuraciones de altavoces estándar, esto es, permite emplear las configuraciones 5.1, 6.1 y 7.1 o cualquier otra que se desee, siempre y cuando los altavoces residan en el mismo plano horizontal. También se le ha añadido la posibilidad de emplear como motor de renderizado el VBAP, aunque con algunas limitaciones, por el momento no se ha añadido la posibilidad de emplear altavoces en elevación.

El *plug-in* es el módulo que se introduce sobre la aplicación DAW permitiendo de esta forma su comunicación con el render. De él se emplearán tantas instancias como fuentes virtuales tenga la escena, cada una de las cuales transmitirá por red al render el sonido correspondiente a la pista en la que se encuentra en forma de un *stream* de audio. Esta forma de comunicación permite emplear todas las herramientas disponibles en la DAW para modificar el sonido a voluntad antes de ser enviado al render. La comunicación por red entre la DAW y el render permite la ejecución de ambas aplicaciones en máquinas distintas. La tecnología empleada para desarrollar el *plug-in* ha sido la VST© [Steinberg, 2009], ya que la gran mayoría de las aplicaciones de autoría de audio profesionales son compatibles con esta tecnología.

Por último, queda el módulo del gestor de las escenas, el cual es un añadido que se ha incluido sobre la estrategia híbrida. Este módulo es una herramienta de composición y ejecución de escenas sonoras y a su vez se

encarga de gestionar al render. El render siempre permanece a la espera de recibir comandos que ejecutar, el gestor se encarga de hacer llegar estos comandos en el momento adecuado según la escena. Este módulo también se comunica con cada una las instancias del *plug-in* para enviar y/o recibir comandos de actualización, el gestor no maneja en ningún momento muestras de sonido. El resumen de esta arquitectura se puede ver en la Figura 5.3.

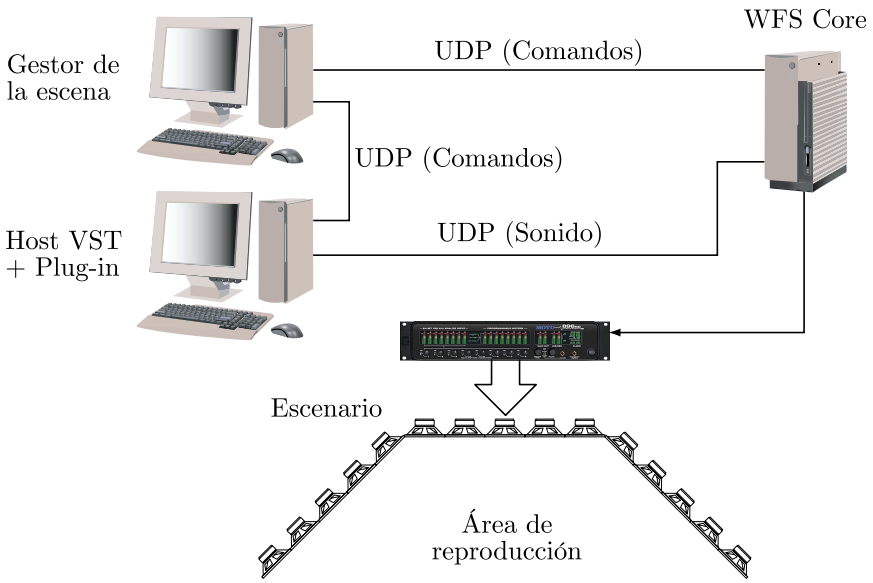


Figura 5.3. Arquitectura de producción de sonido espacial.

5.3.1. Codificación de las escenas

Uno de los aspectos cruciales a la hora de realizar una aplicación de síntesis de sonido espacial es la elección del formato sobre el cual codificar toda la información de las escenas sonoras. La mayoría de los formatos actuales están orientados a pistas por lo que no son adecuados para nuestro

trabajo.

A continuación se realizará una pequeña revisión de los formatos candidatos a ser empleados entre los cuales se incluye uno desarrollado por el autor, aunque finalmente se optó por emplear uno de los formatos estándar. Si se desea profundizar en el tema, en [Potard, 2006] se puede encontrar una descripción y comparación en profundidad de estos formatos.

VRML

El primer formato estandarizado mediante el cual se puede describir una escena compuesta por objetos es el VRML (*Virtual Reality Modeling Language*) [ISO/IEC, 1997, 2004].

VRML se ideó como lenguaje para describir escenas de imágenes tridimensionales para la web, por tanto, el formato en el que se describen las escenas es un fichero de texto. Este fichero contiene los denominados «gráfos de escena», los cuales describen de forma jerárquica las relaciones existentes entre los diferentes objetos. Los nodos de los grafos representan objetos o agrupaciones de estos, los cuales tienen sus propiedades y eventos propios. La relación entre los objetos se describe mediante «rutas». Aunque esta forma de representación permite describir una escena sonora, está pensada desde un punto visual, no sonoro, por lo que la descripción queda muy limitada y, debido al tipo y propiedades de los objetos sonoros existentes, la composición puede resultar redundante y difícil de realizar incluso para escenas sonoras sencillas [Potard, 2006].

A partir de VRML se creó posteriormente el estándar X3D [X3D, 2004], una versión XML de VRML, por lo que mantiene las mismas propiedades que éste en cuanto a la descripción de una escena sonora se refiere, por tanto, no se tomará en consideración.

MPEG4 Advanced AudioBIFS

El siguiente formato estandarizado que permite describir una escena mediante objetos es el MPEG-4 (*Moving Pictures Expert Group*) [ISO/IEC, 1999, 2001], el cual difiere en gran medida de sus predecesores MPEG-1 y MPEG-2 [ISO/IEC, 1993, 1996] los cuales únicamente se encargaban de describir los formatos de compresión del contenido multimedia incluido. MPEG-4 divide el proceso a realizar sobre el contenido en tres capas, las cuales son: entrega, sincronización y compresión. Esta forma de dividir el proceso permite asegurar el envío y recepción de los contenidos incluidos, pero no indica nada de como debe ser generada la escena sonora a partir de los contenidos, dejando así una puerta abierta para que cada sistema adapte los contenidos a su motor de renderizado [Potard, 2006].

Dentro del estándar, el mecanismo dedicado a la descripción de las escenas tridimensionales es el MPEG-4 BIFS (*BInary Format for Scenes*). Una escena puede ser una simple página web, un vídeo, o un completo entorno tridimensional con sonido e imagen. BIFS se creó a partir de VRML [Koenen, 2002], por lo que para describir una escena emplea también los grafos de escenas. Todos los nodos y características de VRML están disponibles en BIFS, además se incluyen varios tipos de nodos nuevos. La principal diferencia entre BIFS y VRML es que el primero almacena la escena en formato binario mientras que el segundo emplea un fichero de texto. Dentro de BIFS, el subconjunto de herramientas dedicadas al sonido es el AudioBIFS y su versión mejorada Advanced AudioBIFS (AABIFS).

Tanto VRML como MPEG-4 AABIFS emplean el mecanismo de los grafos para describir las escenas. La representación mediante grafos no es nada intuitiva, es difícil ver un grafo e interpretar claramente la escena que representa. Aunque es un mecanismo apto para desarrollar la tarea,

no es intuitivo, así, construir una escena es complicado, pero realizar una modificación sobre una escena ya compuesta es extremadamente difícil.

Por todo ello, desechamos estas dos aproximaciones y se decidió desarrollar un formato propio.

Formato propio

Una vez estudiados los formatos previos y comprobado que no eran adecuados para la tarea requerida, se decidió crear un formato propio que solventara muchos de los inconvenientes de los formatos estándar, aunque esto supuso la pérdida de la estandarización.

Para describir una escena se crea un proyecto el cual esta compuesto por un fichero de texto y varios ficheros binarios, además de los ficheros `wav` donde se almacena el audio de cada fuente. En el fichero de texto, el fichero principal del proyecto, se indica el número de fuentes virtuales y sus parámetros iniciales: el nombre, el fichero con las muestras de sonido, la posición, la ganancia, si es una fuente puntual o de ondas planas, etc. Adicionalmente, para cada fuente se crea un fichero binario el cual contiene una lista ordenada con los parámetros y sus modificaciones que se deben aplicar a la fuente. Cada una de las entradas de la lista contiene el instante de tiempo en el que se debe aplicar, el parámetro a cambiar y su valor. Para realizar modificaciones sobre más de un parámetro a la vez se añade más de una entrada con la misma referencia temporal.

Se decidió emplear un formato binario para la lista de órdenes con el fin de minimizar el espacio en disco así como el tiempo de interpretación de los ficheros. Esta forma de describir las escenas permite realizar modificaciones de forma sencilla sobre ellas, pues basta cortar la lista ordenada por el punto indicado e ir añadiendo cualquier modificación a partir de este punto.

Este formato se estuvo empleando hasta que a mediados de 2007 llegó a nuestras manos [Potard, 2006], donde se introduce un formato nuevo, el XML3DAudio, el cual, debido a su sencillez y efectividad, adoptamos como sistema de descripción de las escenas.

XML3DAudio

XML3DAudio es un novedoso sistema que permite la creación de escenas sonoras tridimensionales sin utilizar los grafos de escenas, en su lugar emplea una aproximación que imita la forma de describir una composición en el lenguaje de síntesis de sonido CSound [Vercoe, 2008]. Así, una escena se describirá indicando por un lado la «orquesta» y por otro la «partitura».

La orquesta contiene todos aquellos objetos susceptibles de ser empleados en la escena, esto incluye: las fuentes sonoras, agrupaciones de fuentes, salas, oyentes, etc. Por otro lado, la partitura describe el número de instancias que se van a emplear de cada objeto, los parámetros iniciales de cada instancia y las variaciones que van a sufrir estos parámetros con el paso del tiempo.

Esta forma de describir una escena es muy ventajosa ya que separa la descripción de los elementos de su acción [Potard, 2006]. Si es necesario aplicar permanentemente acciones sobre un conjunto de objetos, basta con definir este conjunto en la orquesta y posteriormente hacer referencia al conjunto en vez de a cada componente por separado en la partitura. Si la agrupación no va a ser permanente se puede definir un grupo temporal en la partitura y emplearlo mientras sea necesario. Si se ha definido un objeto *violín* en la orquesta, en la partitura se podrán emplear tantas instancias de este objeto como se desee o reutilizar una determinada instancia varias veces.

Los objetos más útiles para la descripción de una escena de WFS son las fuentes sonoras y los oyentes (*listeners*). Este último tipo de objeto permite definir el punto de referencia desde el que se «escuchará» la escena. En un sistema binaural indicaría la posición y orientación de la cabeza de la persona. Este objeto introduce una gran versatilidad a la hora de componer una escena, ya que al moverse o rotar podemos cambiar toda la escena sin ninguna dificultad, así, lo que en un sistema binaural es un inconveniente, aquí es una ventaja. Pongamos por ejemplo una escena de una película en la cual hay dos personajes, A y B, la cual se ve desde el punto de vista de A. Al componer la escena sonora colocaremos todos los objetos en el sitio correspondiente y colocaremos al oyente en la posición del personaje A con su misma orientación. Si el siguiente plano emplea el punto de vista del personaje B, basta con cambiar al objeto oyente a la posición y orientación de B, todos los objetos cambiarán de posición relativa sin tener que mover nada.

Otro de los aspectos más importantes de esta aproximación es que la partitura permite aplicar sobre los objetos tanto órdenes directas como órdenes programadas. Una orden directa realiza cambios automáticos sobre determinados parámetros, por ejemplo: subir el volumen, cambiar de posición, rotar, etc. Una orden programada permite describir el movimiento de la fuente como en un lenguaje de programación, ejemplos de estas órdenes son: desplazarse en línea recta durante cinco segundos hasta una posición dada, realizar una trayectoria circular indefinidamente, etc. Estas órdenes al ser interpretadas generan un conjunto de ordenes directas de forma transparente al compositor de la escena.

La forma elegida para plasmar las escenas es el lenguaje de marcas XML, con él es posible describir los objetos, sus propiedades y las acciones

de forma sencilla, además de ser fácilmente interpretable por el compositor al ver el fichero. En el siguiente ejemplo podemos identificar fácilmente como una escena está compuesta de la orquesta y la partitura (*score*).

```
<?xml version="1.0" encoding="UTF-8"?>
<AUDIO_SCENE>
  <ORCHESTRA>
        <Listener>
            <Id>Oyente</Id>
            <Position>
                <Xl>3</Xl>
                <Yl>5</Yl>
                <Zl>0</Zl>
            </Position>
            <Orientation>
                <Azimuth>0</Azimuth>
                <Elevation>0</Elevation>
                <Roll>0</Roll>
            </Orientation>
        </Listener>
        <Source>
            <Id>Guitarra</Id>
            <URL>guitarra.wav</URL>
            <Position>
                <X>1</X>
                <Y>-0.5</Y>
                <Z>0</Z>
```

```
            </Position>
        </Source>
  </ORCHESTRA>
  <SCORE>
        <Performance_Score>
            <Line_of_Score>
                <start_time>0</start_time>
                <Duration>100</Duration>
                <Command>play</Command>
                <Object>voz</Object>
                <Object>Guitarra</Object>
                <Parameter>loop</Parameter>
            </Line_of_Score>
            <Line_of_Score>
                <start_time>20</start_time>
                <Duration>10</Duration>
                <Command>move</Command>
                <Object>Guitarra</Object>
                <Parameter>2</Parameter>
                <Parameter>-0.2</Parameter>
                <Parameter>0</Parameter>
            </Line_of_Score>
        </Performance_Score>
  </SCORE>
</AUDIO_SCENE>
```

Inspeccionando el código XML anterior es fácil comprobar que la orquesta contiene dos objetos: un oyente y una fuente sonora. El oyente tiene

una posición y orientación por defecto y la fuente tiene como parámetros la posición inicial y el nombre del fichero `wav` asociado.

Por otro lado, la partitura contiene dos órdenes. La primera hace sonar la guitarra al comenzar la reproducción y la detiene a los 100 segundos. El comando `loop` indica que si la reproducción llega al final del fichero, el sonido se debe repetir en un bucle. La segunda orden mueve la guitarra en línea recta desde su posición inicial hasta la posición (2; -0,2; 0), el movimiento comienza en el segundo 20 y llega a su destino 10 segundos después.

5.3.2. Adaptación de XML3DAudio para su uso en WFS

XML3DAudio está pensado para describir escenas sonoras tridimensionales de forma independiente del sistema de sonido con el que se vaya a reproducir la escena, por lo cual cabe pensar que habrá que realizar muy pocas variaciones para conseguir representar una escena de WFS. Una característica de XML3DAudio que es necesario destacar es que el uso de XML como formato para plasmar las escenas permite realizar las modificaciones necesarias sin prácticamente ningún esfuerzo.

Pasamos a continuación a describir las modificaciones que el autor cree oportuno realizar para poder reflejar todas las características de WFS con XML3DAudio.

La modificación más común es la de incluir algunas propiedades nuevas sobre determinados objetos, esto es, son modificaciones sobre la orquesta, aunque posteriormente se podrá variar su valor con los comandos de la partitura.

Fuentes de ondas planas

En WFS existen dos tipos de fuentes: las puntuales y las de ondas planas, las primeras están perfectamente reflejadas por el objeto `Source` pero no sucede así con las segundas.

Una fuente de ondas planas es una fuente puntual lejana a la cual se le ha incrementado el nivel sonoro para compensar la atenuación que produce la distancia. Así, la forma más sencilla de simularlas es esa, alejar una fuente puntual y aumentar su volumen. Esta técnica es muy sencilla pero obliga al compositor a realizar tareas que podríamos denominar «trucos» para conseguir que la fuente suene adecuadamente. Además deben hacerse cálculos manuales para realizar la compensación del volumen, por lo que no es una técnica conveniente de realizar a mano.

Con el fin de que todo el trabajo se realice de forma transparente al compositor, será necesario añadir el parámetro `Plane_Wave` al objeto `Source`, el cual hará la función de un valor booleano que indicará si debe tratarse o no a la fuente como una onda plana.

Para conseguir que una fuente sonora puntual se comporte como una fuente de ondas planas aún cuando no está así definida en la orquesta, basta con añadir el parámetro `Plane_Wave` a la orden `play` correspondiente de la partitura.

Entre los parámetros que posee el objeto `Source` está el parámetro `dimensions` el cual puede llegar a inducir el error de creer que con él se puede simular una onda plana. Este parámetro permite indicar el tamaño de una fuente, por ejemplo, para simular el oleaje de una playa se puede definir una línea de 100 metros de longitud. Al provenir el sonido desde toda la extensión de la línea podemos suponer que es como una onda plana, pero, esta suposición es errónea pues, según [Potard, 2006], para simular la

extensión lo que se hace es repetir la fuente sonora a lo largo de la superficie indicada, previo paso de cada una de ellas por un filtro paso-todo de fase aleatoria para decorrelar las señales y evitar así que el oído las integre. Por tanto, este parámetro tiene que ver con la psicoacústica y no debe usarse para simular ondas planas.

Efecto *doppler*

Dentro de XML3DAudio se posible simular el efecto *doppler*, se puede hacer que sea el adecuado al medio, que la modulación en frecuencia sea mayor de la que debiera e incluso eliminarla. El inconveniente que nos encontramos es que este efecto no está asociado a las fuentes sonoras sino al medio en el que se mueven.

Ya se ha indicado en la Sección 3.6 que en determinados escenarios, no debería aplicarse el efecto *doppler* sobre determinadas fuentes sonoras. Al asociar los parámetros del efecto doppler sobre el medio en el que se mueven las fuentes en vez de sobre las fuentes se obliga a que todas las fuentes sonoras de la escena tengan el mismo tratamiento del *doppler*, esto es, o se les aplica a todas o a ninguna.

Para conseguir que la aplicación del efecto *doppler* fuera selectiva por fuente, una característica muy deseable en WFS, sería necesario añadir el parámetro `No_Doppler` en la descripción de las fuentes sonoras dentro de la orquesta. De esta forma, independientemente del medio es posible inhabilitar el efecto *doppler* al aparecer este parámetro. Para emplear directamente este parámetro en la partitura basta con añadirlo al comando `play`.

Tratamiento del *aliasing* espacial

Otro aspecto casi exclusivo de WFS es el tratamiento que se puede dar a las altas frecuencias, con el fin de minimizar el efecto del aliasing espacial. En este caso no hay ninguna herramienta en XML3DAudio que nos permita tener en cuenta esta característica. Por tanto será necesario añadir otro parámetro al que denominaremos `High_Frequencies`, el cual podrá tener varias opciones según la técnica a aplicar. Las opciones serán:

- `None`: No se aplica ningún tratamiento especial, este será el valor por defecto.

- `OPSI`: aplica la técnica OPSI para la banda de altas frecuencias.

- `Subband`: aplica la aproximación por subbandas para las altas frecuencias.

Con el fin de poder discriminar el tratamiento en función de la fuente sonora es preferible incluir este parámetro dentro del objeto `Source`, además es un comportamiento independiente del resto de objetos disponibles: `Medium`, `Room`, etc.

Sistema de coordenadas

Una de las características más peculiares y versátiles de WFS es que independiza la escena sonora recreada de la forma física del array de altavoces empleado. Esto es, este sistema de reproducción adapta las señales que hay que enviar al array de altavoces durante la reproducción.

Esta versatilidad supone un ligero inconveniente a la hora de plasmar la escena con XML3DAudio. Este estándar fue diseñado para recrear escenas sonoras de forma independiente del sistema de altavoces empleado,

pero, en su concepción se empleo como reproductor un sistema basado en ambisonics al cual su autor denominó CHESS (*Configurable Hemispheric Environment for Spatialised Sound*) [Potard, 2006]. El sistema CHESS empleaba un array de altavoces tridimensional en forma de media esfera de unos dos metros de radio, y, aún cuando pudiera ser usado con cualquier otro array, durante el desarrollo de XML3DAudio se empleo este array como base. La consecuencia directa de esto es que en XML3DAudio las coordenadas empleadas en las escenas son exactas.

Supongamos ahora que una escena creada con el sistema CHESS se reproduce en un cine de grandes dimensiones mediante WFS, toda la escena estará situada en el centro de la sala, cuando generalmente debería suceder alrededor de ella, esto es, es necesario aplicar de alguna forma un escalado de la escena.

Dado que las coordenadas empleadas por XML3DAdudio son exactas y debemos conseguir que sean relativas, será necesario incluir un nuevo objeto que defina el tamaño de la escena con el fin de que el sistema que reproduzca la escena sea capaz de realizar el escalado de forma automática. Para ello hemos propuesto incluir un nuevo objeto que defina las dimensiones del *setup* empleado en la composición de la escena:

```
<Setup_dimensions>
    <X>6</X>
    <Y>6</Y>
    <Z>3</Z>
</Setup_dimensions>
```

Como se puede apreciar el objeto `Setup_dimensions` contiene tres parámetros `X`, `Y` y `Z`, los cuales definen las dimensiones de un cubo que enmarca al array de altavoces empleado. Conociendo las dimensiones del

array original, esto es, las indicadas por `Setup_dimensions`, y las del array empleado es muy sencillo escalar las posiciones de las coordenadas para adecuarlas al tamaño de la sala de reproducción.

De nuevo es necesario hacer un inciso sobre la idoneidad de emplear uno de los objetos ya existentes en XML3DAudio para proporcionar esta utilidad, en concreto el objeto `Room`. Aunque este objeto puede usarse con este fin, no debe emplearse ya que su función es la de proporcionar la descripción de una sala con el fin de incluir reverberación en la escena, por lo que generalmente se emplearán varios objetos `Room` durante la reproducción.

5.4. Descripción de los módulos desarrollados

5.4.1. Gestor de las escenas

El gestor de las escenas es el módulo que nos va a permitir controlar el funcionamiento del núcleo de renderizado, es su panel de control. En la Figura 5.4 se puede ver una captura de pantalla del gestor, en la cual hay una escena cargada que contiene cuatro fuentes virtuales, representadas por cuatro guitarras, cada una de las cuales se puede controlar de forma independiente.

El interfaz está dividido en varias zonas, las cuales hemos marcado con un número sobre la imagen, que pasamos a describir a continuación. En primer lugar tenemos el menú principal, marcado con el número uno, que nos va a permitir realizar las operaciones típicas de la gestión de los ficheros, crear un proyecto nuevo, cargar uno ya creado, limpiar el actual, etc.

El número dos está situado sobre el panel `Master Controls` el cual contiene tres controles de ganancia cuyo ajuste repercute de forma global

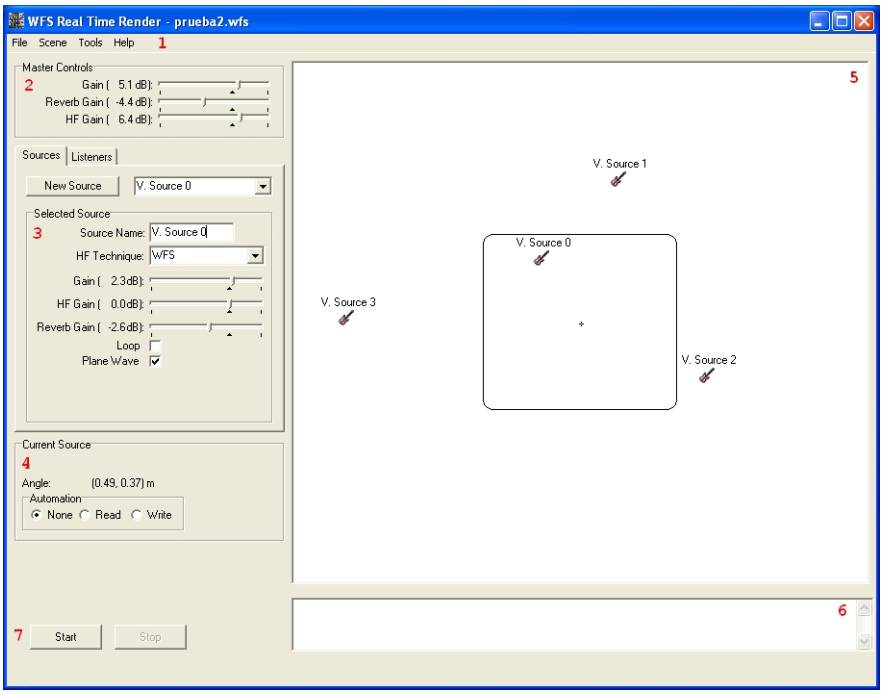


Figura 5.4. Pantalla principal del gestor de las escenas.

sobre todo el proyecto. Las ganancias que se pueden ajustar son:

- `Gain`: Es el control de volumen maestro.
- `Reverb`: Permite controlar el nivel de la reverberación.
- `HF`: Permite controlar el nivel empleado en la banda de alta frecuencia en todas las fuentes que emplean la división en dos subbandas.

El número tres está situado sobre el panel `Selected Source` que, como su propio nombre indica, permite realizar modificaciones sobre el comportamiento de la fuente seleccionada. Para seleccionar una fuente basta con elegirla con el *combobox* que hay en el panel, o pulsando con el botón izquierdo del ratón sobre el icono correspondiente en el panel de la derecha

(marcado con el número cinco).

Si seguimos los controles por su disposición en el panel podemos ver que es posible controlar los siguientes parámetros:

- `Source Name`: Es el nombre con el que nos referiremos a la fuente virtual, el cual se muestra sobre el icono en el panel de la derecha.

- `HF Technique`: Con este control podemos indicar la técnica con la que se tratará el problema del *aliasing* espacial, es decir, como se tratará el contenido en alta frecuencia de la fuente. Se dispone de tres opciones: `None`, `OPSI` y `Sub-band`. Si se elige `None` se mantiene el tratamiento estándar de WFS por lo que no se realiza ningún procesado especial. Las otras dos opciones realizan una división del espectro en dos subbandas pero cada una emplea una técnica distinta para la banda de alta frecuencia.

- `Elevation`: Este control permite indicar la técnica a emplear para la simulación de los efectos de elevación. Las posibilidades son tres: `None`, `Panning` y `HRTF`. Como se podrá deducir, `None` desactiva el tratamiento en elevación y los otros dos seleccionan la técnica a emplear.

- `Gain`: Es el control de volumen de la fuente.

- `Reverb Gain`: Con este control podemos modificar la presencia de esta fuente en la reverberación aplicada. Si lo reducimos a $-\infty$ la fuente quedará excluida de la reverberación.

- `HF Gain`: Este control permite modificar la ganancia de la subbanda de alta frecuencia, siempre y cuando esté seleccionada una técnica que emplee subbandas.

- `Loop`: Si se habilita este control, la reproducción de la fuente se reiniciará automáticamente una vez haya terminado.

- `Plane Wave`: Al marcar este control estamos indicando que la fuente pasa a ser una onda plana, en vez de una onda esférica.

- `Doppler`: Con este control podemos seleccionar el tratamiento del movimiento de la fuente. Al marcarlo se aplicará el algoritmo de movimiento que simula el efecto *doppler*, en caso contrario se aplicará el algoritmo de movimiento sin *doppler*.

El número cuatro está situado sobre el panel del control encargado de la automatización. Este control actúa sobre la fuente seleccionada y tiene tres posibles estados:

- `None`: la fuente no se controla de forma automática. Su control se realiza de forma manual a través del panel marcado con el número 5.

- `Read`: la fuente se controla de forma automática. Todas las modificaciones de los parámetros almacenadas para la fuente se ejecutarán en el instante adecuado.

- `Write`: todas las modificaciones que se realicen sobre los controles de la fuente son almacenadas junto con el instante de tiempo en el cual se han producido para su uso posterior.

Así, el funcionamiento de este control es idéntico al empleado por las mesas de mezclas digitales para automatizar una mezcla, la cual se realiza de forma progresiva. En primer lugar se añade una fuente a la escena, se coloca en la posición inicial deseada y se marca el control `Write`, a continuación se pulsa el botón `Start` con lo que comienza la reproducción durante la cual podemos mover la fuente de forma manual por la trayectoria

deseada y/o modificar los controles de volumen, reverberación, etc. Una vez concluido el tramo que se quería automatizar se pulsa `Stop` y concluye la memorización de los parámetros, pasando el control de forma automática a `Read`. Este procedimiento se repite para cuantas fuentes queramos que tenga la escena, reproduciéndose las fuentes anteriores mientras se registra la fuente actual, así se consigue completar la escena de forma progresiva. Si en algún momento se desea modificar la automatización almacenada de una fuente, basta con cambiar el control de `read` a `write` en el momento adecuado y realizar las modificaciones oportunas.

El panel blanco que ocupa casi toda la ventana, marcado con el número cinco, sirve para mostrar el estado de la escena así como su evolución. El rectángulo redondeado que hay en él representa la «sala» tal y como se describió en la Sección 3.1. Su inclusión en la pantalla sirve para determinar la posición de una fuente con respecto al array, aunque las posiciones siempre serán relativas ya que el gestor de las escenas desconoce la configuración concreta del array, este puede ser tanto un único segmento lineal como una configuración en forma de octógono, para el gestor esto es indiferente. El punto central del panel, marcado con una pequeña cruz representa el centro de la sala.

Sobre este panel se muestra la posición de las fuentes, las cuales podemos mover arrastrando su icono. Al seleccionar uno de los iconos, seleccionamos la fuente a la que representa por lo que los controles de los paneles de la izquierda se deben actualizar. Durante la reproducción, si las fuentes cambian de posición los iconos se irán moviendo por el panel para reflejar este cambio.

El pequeño panel blanco que hay en la parte inferior, marcado con el número seis, sirve como control de depuración, todas las órdenes que se

van generando muestran en el panel, siempre y cuando se haya activado a través de la opción correspondiente del menú principal. También se pueden mostrar aquí los mensajes recibidos desde los *plug-ins* VST. Toda esta información sirve para en determinadas situaciones localizar fallos.

Por último quedan los botones `Start` y `Stop`, marcados con el número siete en la pantalla, los cuales sirven para poner en marcha y parar la reproducción.

Control del tiempo

Un aspecto importante del gestor de las escenas que hay que tener en cuenta es que no dispone del *hardware* de sonido, lo cual es un gran inconveniente como veremos a continuación. Al trabajar con el *hardware* de sonido, el paso del tiempo es muy sencillo de determinar pues el propio *hardware* avisa cada vez que ha transcurrido el tiempo que dura un bloque de datos (véase la Sección 4.3.2), gracias a él todo se mantiene sincronizado. Así, al carecer del *hardware* deberemos emplear otros recursos para controlar el paso del tiempo.

El mecanismo estándar de control del tiempo bajo Windows es el temporizador, el cual se activa con la función `SetTimer`. La resolución del temporizador es de 1 ms y su funcionamiento es bastante sencillo, una vez ha transcurrido el tiempo designado se envía el mensaje `WM_TIMER`, el cual entra el la cola de mensajes pendientes, por lo que hasta que no se procesen los mensajes previos el aviso no llegará al proceso. Como podemos ver este mecanismo no es fiable pues no nos asegura la duración del retardo el cual puede dilatarse de forma considerable.

Para aplicaciones que requieren de un temporizador preciso Windows proporciona el temporizador multimedia. La resolución de este tempori-

zador también es de 1 ms pero la notificación se realiza a través de una función *callback*, empleando así el mismo método que el *hardware* de sonido. Al saltarse la cola de mensajes la duración del retardo es mucho más precisa, además, en caso necesario se reduce la duración de los *quantum* de tiempo que se asignan a los procesos para asegurar la precisión (véase la Sección 4.4.3).

5.4.2. *Plug-in* VST

El *plug-in* VST es la utilidad que se ha diseñado para poder emplear las herramientas de producción multipista actuales en la producción de sonido espacial basada en objetos. Al emplear la tecnología VST de Steinberg se consigue extender enormemente el abanico de posibilidades en cuanto a la elección de la DAW software, la gran mayoría de DAW actuales permite emplear este estándar de *plug-ins*, no solamente aquellas basadas en Windows, también las hay Unix y Mac. De esta forma aquellos usuarios acostumbrados a trabajar en un entorno Mac pueden emplear el sistema de producción orientado a objetos, eso si, el motor de renderizado debe residir en una máquina Windows.

Aquel software que admite *plug-ins* VST es un *host* VST, esto es, la DAW va a actuar como *host* del *plug-in*. El formato de un *plug-in* VST bajo Windows es el de una librería DLL la cual se construye heredando de una clase en C++, proporcionada por el SDK del VST. La estructura de la clase es muy sencilla ya que, aparte de los constructores y destructores típicos, únicamente debemos completar el contenido de una función[1] a la cual llama el *host* cada vez que desea procesar un bloque de sonido.

El protocolo de funcionamiento simplificado del *plug-in* es el siguiente:

[1] En realidad dos: `process` y `processReplacing`.

- En el constructor se establece una comunicación inicial con el gestor de la escena y el núcleo de renderizado. Así estos tienen en cuenta que en la escena va a haber una fuente virtual más. Además se crea un hilo que permanecerá a la espera de recibir mensajes del gestor de la escena a través de la red.

- En la función de procesado se envía al núcleo de renderizado las muestras de sonido recibidas, en forma de un *stream* de audio.

- En el destructor aparte de eliminar todas las estructuras internas, se avisa tanto al gestor de la escena como al núcleo de renderizado que la fuente va a dejar de existir.

Esto es, el *plug-in* actúa como una conexión vampiro en cuanto al sonido se refiere, toda muestra de sonido que recibe es copiada y enrutada por red hasta el núcleo de renderizado.

El *plug-in* desarrollado carece de interfaz, el *host* en el que se aloja crea uno «de oficio» que permite modificar todos los parámetros empleados en el *plug-in*. Los parámetros que maneja el plug-in son únicamente dos: las coordenadas (X, Y) de la posición de la fuente virtual. Los valores de estas coordenadas son relativos al tamaño de la sala, ya que todo parámetro de un plug-in VST está confinado al rango [0..1].

Ya sólo resta comentar la utilidad del hilo de comunicación con el gestor de la escena. Este hilo se encarga de enviar y recibir las actualizaciones de las coordenadas de la fuente. Generalmente la fuente se moverá de forma manual en el gestor de la escena, por lo que este último notificará al *plug-in* el cambio de coordenadas. También es posible mover la fuente a través del interfaz simple del *plug-in*, aunque esto no es recomendable, por lo que será el *plug-in* el que enviará el aviso al gestor de la escena.

Este comportamiento se diseñó con un propósito claro: que la evolución de la escena se almacenase en el propio proyecto de la DAW de forma independiente al proyecto que registra el gestor de la escena. Todo parámetro de un *plug-in* puede ser automatizado por la propia DAW, basta con indicarle a esta que siga su evolución. Por ello es necesario que el gestor notifique todos los cambios al *plug-in* y que el *plug-in* notifique en su caso al gestor de las escenas.

5.4.3. Núcleo de renderizado

El interfaz principal del render, el cual se puede ver en la Figura 5.5, tiene el propósito de facilitar la definición del array físico de altavoces que se va a emplear. Una vez se hayan indicado todos los parámetros del array comenzará la ejecución, quedando el render a la espera de la llegada de órdenes a través de la red, esto es, el interfaz no va a permitir controlar la aplicación, únicamente permite configurarla. El panel blanco de la derecha muestra una figura con la configuración actual del array de altavoces, con una pequeña cruz que indica el punto que se está empleando como «centro» del array. Este panel es meramente ilustrativo y sólo sirve para verificar la configuración establecida.

Arriba a la izquierda disponemos del panel `Master` el cual contiene tres deslizadores que proporcionan las únicas herramientas de control de que dispone el interfaz sobre la reproducción. Estos tres deslizadores permiten manipular de forma directa tres ganancias:

- `Gain`: permite modificar la ganancia global de la reproducción, esto es, es el control de volumen maestro.

- `Reverb`: permite modificar la ganancia global de la reverberación.

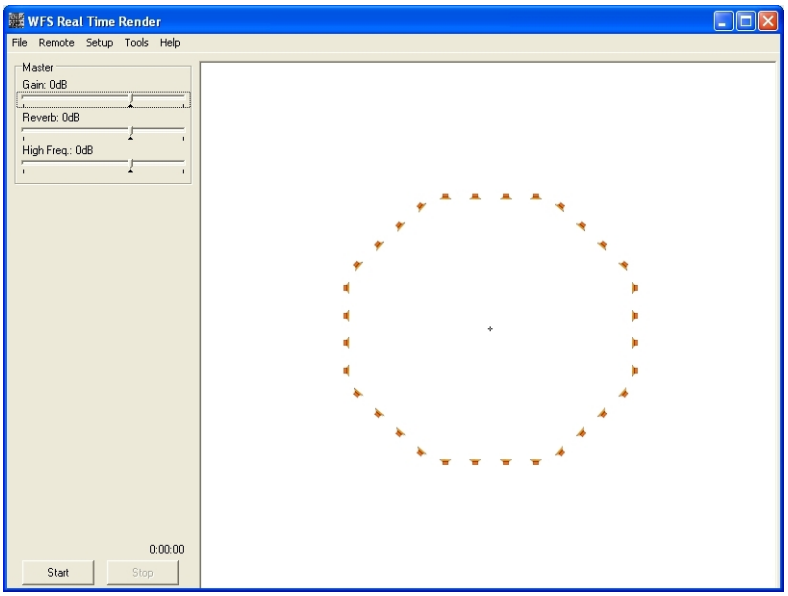


Figura 5.5. Pantalla principal del núcleo de renderizado.

- `High Freq`: permite modificar la ganancia global del contenido de la banda de alta frecuencia para todas aquellas fuentes que realicen la división del espectro en subbandas.

Estos tres controles se han previsto como un ajuste fino de la configuración del array de altavoces, una vez establecidos no suelen modificarse.

Selección de la *hardware* de sonido

Si elegimos la opción del menú `Tools->ASIO Configuration` nos aparece el panel de selección de la tarjeta de sonido (véase la Figura 5.6). En este panel aparecerán en el *combobox* todas las tarjetas de sonido de las que esté disponible un *driver* ASIO, una vez seleccionada la tarjeta podemos indicar el tamaño de bloque que vamos a emplear en la reproducción.

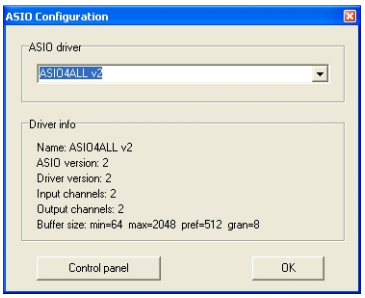


Figura 5.6. Pantalla de selección de la tarjeta de sonido.

Definición del array

De entre las opciones de configuración que permite el interfaz del render, la definición del array físico a emplear es la más importante. Se accede a ella a través de la opción del menú `Setup->Array Setup`, y lo primero que nos presenta es un asistente (véase la Figura 5.7) con una variedad de configuraciones típicas que permiten definir un array sin mucha dificultad. A parte de las configuraciones típicas en línea, U abierta, cuadrado, octógono y circular, se nos presenta una opción para emplear una configuración estándar 5.1, 6.1 o 7.1 y por último una opción para diseñar un array a medida.

En la Figura 5.8 se muestra el asistente que se nos presenta para definir un array en U abierta. La figura que se muestra en el asistente es meramente orientativa y sólo sirve de ejemplo. Tal y como se puede apreciar, el array se va a definir por segmentos, de cada uno de ellos es necesario indicar: el número de altavoces que contiene, la separación entre ellos, el ángulo de inclinación con respecto a la horizontal y el canal del primer altavoz.

Si analizamos el ejemplo de la figura podemos ver que el array estará compuesto por tres segmentos, cada uno de los cuales contendrá ocho altavoces (24 canales en total) todos ellos separados 18 cm entre sí. El pri-

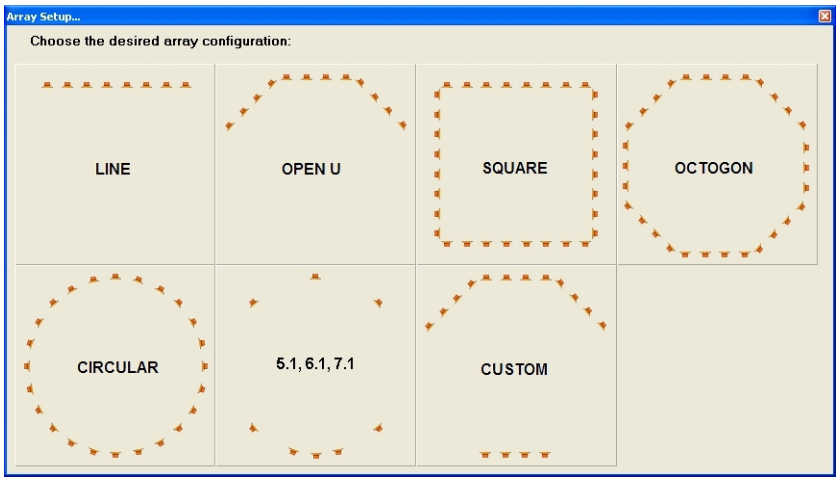


Figura 5.7. Asistente para la creación del array.

mer segmento tiene una inclinación de -45º con respecto a la horizontal, el segundo es horizontal (0º) y el tercero tiene una inclinación de 45º. En cuanto a los canales que empleará cada segmento son: del 0 al 7 el primero, del 8 al 15 el segundo y del 16 al 23 el tercero.

Al haber seleccionado el asistente de la configuración en U abierta, no es necesario indicar la posición exacta de cada segmento, ni que el segundo está situado a continuación del primero y el tercero del segundo. El asistente ya realiza todos estos cálculos por nosotros.

Uno de los asistentes más útiles es el que permite crear la configuración de un array circular (véase la Figura 5.9). En este caso basta con indicar el radio del circulo y el número de altavoces que se van a emplear. Si tenemos en cuenta que el array debe crearse a través de segmentos de altavoces, este es un caso degenerado en el cual cada segmento posee únicamente un altavoz.

En el caso de que el array que vayamos a emplear no se ajuste a ninguno de los asistentes podemos emplear el asistente «a medida», el cual

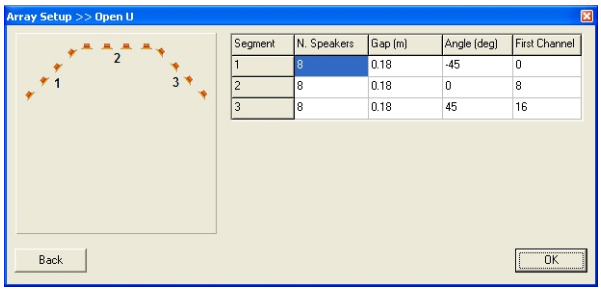


Figura 5.8. Asistente para la creación de un array en configuración U abierta.

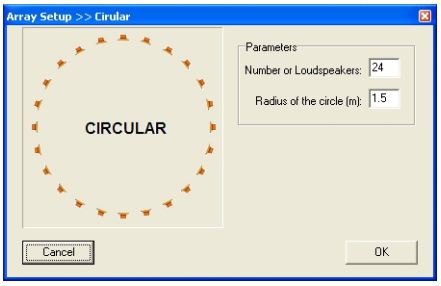


Figura 5.9. Asistente para la creación de un array circular.

está representado por la opción `Custom`. En este asistente se puede indicar el número de segmentos deseados y la posición exacta de cada uno de ellos. En este caso no se presupone ninguna forma determinada lo cual permite crear cualquier figura a cambio de dificultar bastante su creación al tener que calcular la posición de cada segmento a mano, por ello se han añadido unas cuantas herramientas al asistente con el fin de simplificar la creación de una array de forma arbitraria.

Generalmente un segmento siempre va concatenado con el anterior, esto es, el punto de origen del segmento es el extremo del anterior, no existe un hueco entre segmentos. Si en las casillas que contienen la posición X e Y del segmento se coloca un signo '+', el asistente entiende que va a ir a

continuación del segmento anterior y realiza el cálculo automáticamente. Aún así, crear un array desde cero es algo tedioso y como generalmente la configuración a medida deseada será una pequeña variación sobre alguna configuración típica, se ha añadido otra pequeña utilidad: al entrar al asistente se muestra la configuración del array actual sobre la cual se pueden realizar las modificaciones oportunas. Así, es posible emplear uno de los asistentes previos para crear una configuración típica que se asemeje a la deseada y editar posteriormente esta configuración con el asistente `Custom` sin tener que realizarla desde cero. En la Figura 5.10 se puede ver la vista del asistente `Custom` en la cual se muestra un array en forma de octógono creado con el asistente propio.

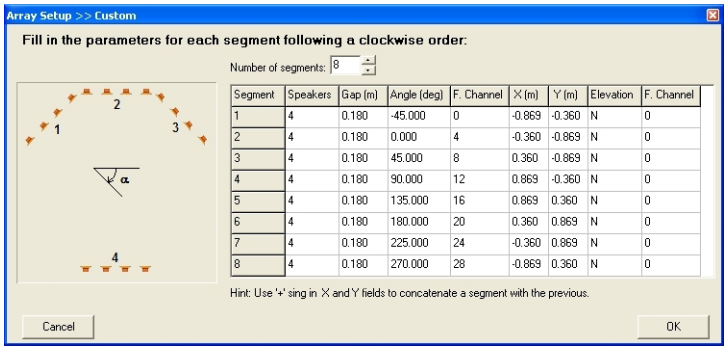


Figura 5.10. Asistente para la creación de un array a medida.

Como ya se ha comentado en la Sección 5.3, se pretende que la arquitectura propuesta permita la reproducción de escenas sonoras descritas por objetos, no únicamente la reproducción de escenas para WFS. Así el render también permite emplear las configuraciones típicas de altavoces de la actualidad, como son las configuraciones 5.1, 6.1 y 7.1. Para tal caso se ha añadido un asistente que facilita la construcción de estos arrays. La pantalla del asistente se puede ver en la Figura 5.11. En este caso basta con

indicar cual de las tres configuraciones estándar se va a emplear y el radio del circulo sobre el que residirán los altavoces. Para dar una mayor flexibilidad al colocar los altavoces, el asistente permite modificar los ángulos en los que deben situarse los altavoces así como el canal de la tarjeta de sonido asignado a cada uno de ellos.

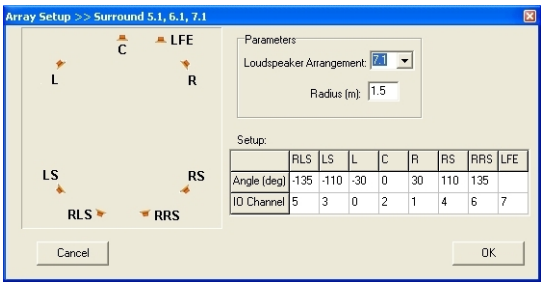


Figura 5.11. Asistente para la creación de una configuración *Surround* 5.1, 6.1 y 7.1.

Subwoofers

El acceso al panel de control de los subwoofers se encuentra en el menú `Setup->Subwoofer Setup`. Si se desea activar los subwoofers es necesario marcar el *checkbox* 'Enable Subwoofer Processing'.

Luego hay que indicar el número de *subwoofers* que se van a utilizar. Para cada *subwoofer* hay que indicar el número de canal de la tarjeta de sonido sobre el que estará conectado.

Por último hay que seleccionar el filtro paso bajo que se utilizará para filtrar el sonido de los *subwoofers*. El filtro indicado será empleado para todos los subwoofers independientemente de la cantidad elegida, y debe estar expresado como un IIR en Secciones de Segundo Orden (SOS) [Ramos and López, 2006].

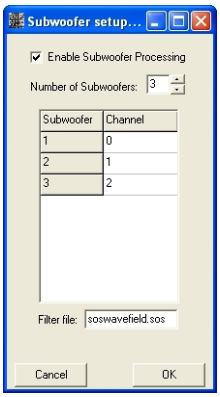


Figura 5.12. Asistente para la configuración de los subwoofers.

Ecualización

En determinadas situaciones es necesario ecualizar el array para que el sonido emitido sea correcto. En general si los altavoces empleados en el array son dinámicos no es necesario ecualizarlos, aunque puede ser conveniente. Al ecualizar el array conseguimos que su respuesta en frecuencia sea más plana, extendemos ligeramente la radiación de las bajas frecuencias y eliminamos el resto del contenido en baja frecuencia que no son capaces de emitir.

En el caso de arrays de MAPs si es obligatorio realizar la ecualización del array pues su respuesta en frecuencia dista bastante de ser plana y, además, el contenido en baja frecuencia de la señal hace que distorsionen fácilmente al ser la excursión máxima de los excitadores muy pequeña.

En la Figura 5.13 se puede ver el asistente que permite seleccionar los filtros de ecualización que se van a emplear, todos los filtros deben ser IIR descritos como secciones de segundo orden.

A la hora de ecualizar el array disponemos de dos opciones, o bien se

Figura 5.13. Asistente para la configuración de los filtros de ecualización del array.

pasan todas las señales por un filtro de ecualización global o bien se aplica un filtro distinto para cada canal. En caso de emplear un array de altavoces dinámico y siempre y cuando todos los altavoces sean iguales basta con emplear un único filtro de ecualización global ya que las diferencias en la respuesta en frecuencia de los altavoces es mínima. En el caso de arrays de MAPs esta regla ya no se cumple, depende en gran medida del tamaño de los MAP empleados, por lo que puede ser necesario realizar un filtrado por canal en vez de uno global.

5.5. Conclusiones

En el desarrollo del presente capítulo se ha descrito la problemática que conlleva la creación de un sistema de autoría de WFS. La industria discográfica esta totalmente orientada a la edición multipista, por lo que se hace necesario cambiar la filosofía de trabajo: en vez de pistas ahora debe haber objetos sonoros que interactúan con su entorno.

En primer lugar se han repasado las estrategias posibles para realizar el cambio de filosofía: modificar el software existente, desarrollarlo desde cero o una estrategia híbrida. Una vez repasadas las estrategias, se ha optado por la única que tiene posibilidades de ser llevada a cabo y luego ser empleada de forma amplia por la industria discográfica.

Con esta estrategia en mente se ha procedido a describir la implementación propuesta, la cual está compuesta por tres módulos diferentes:

- Gestor de las escenas

- Plug-in VST

- Motor de renderizado

El gestor de las escenas hace las veces de interfaz de usuario centralizando toda la comunicación con el usuario final. El *plug-in* VST permite conectar las estaciones *DAW* actuales con el sistema propuesto, por lo que cualquier recurso disponible para estas puede ser empleado a su vez para WFS. Por último, el núcleo de renderizado es el encargado de procesar todos los comandos enviados por el gestor de las escenas, así como los *streams* de audio enviados por el plug-in, de forma que permite recrear el campo sonoro sintético de forma dinámica.

Esta implementación se encuentra disponible en los laboratorios del *i*TeAM en la UPV. Para que el sistema descrito funcione en tiempo real se necesitan tres ordenadores personales, cada uno de los cuales deberá ejecutar uno de los tres módulos. Es posible hacer funcionar el sistema en un único ordenador si no se emplea el *plug-in* VST, pues el *plug-in* se aloja en un *host* VST, el cual suele consumir muchos recursos del ordenador. El gestor de la escena y el motor de renderizado pueden coexistir en el mismo ordenador, aunque es recomendable separarlos.

Capítulo 6

Conclusiones y líneas abiertas

La finalidad de la realización de esta tesis doctoral ha sido la de conseguir mejorar la creación de los sistemas de *Wave Field Synthesis* desde multiples puntos de vista: su aplicación práctica, la implementación eficiente y la posibilidad de hacer viable su uso por el público en general.

En el presente capítulo se revisarán los objetivos propuestos en la introducción y se resumirán las conclusiones a las que se ha llegado después de todos estos años de trabajo. Seguidamente se irán detallando una por una todas las aportaciones realizadas que pueden contribuir a mejorar la implementación de sistemas de WFS. A continuación se describirá un conjunto de problemas abiertos en los que convendría seguir trabajando. Por último se incluirá una lista de las comunicaciones presentadas en las que ha contribuido el autor con el trabajo desarrollado en esta tesis.

6.1. Conclusiones

Tal y como se ha descrito en el principio del Capítulo 3, el desarrollo teórico de WFS no tiene en cuenta diversos aspectos que hacen que su implementación no sea directa. Algunos de estos aspectos provienen de la suposición de campo lejano que se hacen en la simplificaciones de las ecuaciones teóricas, lo que conlleva que cuando las fuentes están en posiciones cercanas al frente de ondas secundario, las ecuaciones dejan de ser válidas.

Algunos de estos aspectos problemáticos han sido resueltos por otros autores, pero sus resultados no se han hecho públicos pues están explotando de forma comercial la implementación. En el Capítulo 3 se ha realizado un análisis exhaustivo de diversos casos particulares que deben solventarse, dándose a cada uno de ellos una solución conveniente. Todo ello ha permitido implementar un sistema de WFS completamente funcional.

En el Capítulo 4 se ha propuesto una arquitectura software capaz de implementar en tiempo real un motor de renderizado de WFS en un único ordenador de propósito general, sin necesidad de emplear un grupo de DSPs o una red sincronizada de ordenadores. La estrategia por la que se apostó al comienzo de esta tesis ha probado ser acertada. Al realizar todo el procesado en la misma máquina se reduce drásticamente el coste de la instalación. La arquitectura software proyectada ha sido alabada por expertos en el campo de WFS que han podido escuchar los resultados con el prototipo de 96 canales que está disponible en los laboratorios del *i*TeAM en la Universidad Politécnica de Valencia. En general todo aquel que escucha el prototipo y conoce los fundamentos de WFS se lleva una grata sorpresa al comprobar que es posible realizar todo el procesado en un único ordenador de propósito general.

En el Capítulo 5 se han discutido las diferentes estrategias que pueden

permitir introducir un sistema de producción de WFS en un estudio de producción de sonido. Se han revisado las ventajas e inconvenientes de cada una de las estrategias y se ha concluido que la mejor opción es una estrategia híbrida. En base a esta estrategia se ha implementado un sistema de autoría de WFS el cual está compuesto por tres módulos: el gestor de las escenas, el *plug-in* VST y el motor de renderizado. El sistema permite la producción de sonido espacial de forma sencilla aprovechando, gracias al *plug-in* VST, todas las herramientas disponibles en las estaciones DAW actuales.

6.2. Contribuciones

En la siguiente sección se describen las contribuciones al desarrollo de sistemas de WFS realizadas por el autor durante el desarrollo de esta tesis doctoral.

- Se ha propuesto una técnica que permite determinar de forma sencilla y eficiente los altavoces que deben ser activados para simular una fuentes sonora. Esta técnica es sistemática por lo que permite ser empleada incluso con topologías complejas. Se ha realizado la comparación con otra técnica existente y se ha comprobado su buen funcionamiento pues evita en la medida de lo posible la radiación en sentido contrario al avance del frente de onda. Recientemente se ha desarrollado una técnica formal basada en la intensidad del campo sonoro que proporciona los mismos resultados que la técnica propuesta.

- Se ha propuesto un método para minimizar los problemas que afectan a las fuentes sonoras próximas al array. Aún siendo un método

empírico, se ha demostrado que es posible sintetizar fuentes próximas sin perjuicio de la calidad del sonido. En este caso no ha sido posible comparar el método con otros pues, hasta donde conoce el autor, no se ha hecho público ningún otro.

- Se ha propuesto una técnica de mejora de la síntesis de las fuentes focalizadas. Al igual que sucedía con la síntesis de las fuentes virtuales, las fuentes focalizadas también sufren el problema de la radiación en sentido contrario al avance del frente de onda. En este caso el problema se acentúa al estar delante del array, pero con la técnica propuesta se ha conseguido mitigar sus efectos en la medida de lo posible. Comparando el resultado obtenido con otras técnicas, se ha comprobado que para fuentes no centradas en la sala consigue reducir el *aliasing* espacial inherente a las fuentes focalizadas. Cuando las fuentes se encuentran cercanas al centro de la sala, el peor caso posible para fuentes focalizadas, permite su síntesis pero sin conseguir eliminar la radiación en sentido contrario.

- Se ha descrito un método para simular fuentes en movimiento en WFS. La aportación realizada en este punto no es la técnica de simulación de movimiento, sino el hecho de conseguir aplicar una técnica de simulación del movimiento de forma muy eficiente, aplicando a la vez la modulación en frecuencia debida al efecto *doppler*. También se ha previsto otra técnica basada en el efecto *panning* para el caso de querer simular el movimiento sin aplicar el efecto *doppler*.

- Se ha propuesto una técnica de mejora de la síntesis del campo sonoro en alta frecuencia. La técnica propuesta no permite reducir el *aliasing* espacial del campo sintético generado por WFS, en su lugar lo que se

busca es sustituir esta parte del espectro por otra que no sufra de los efectos indeseables del *aliasing*. En determinadas situaciones, la aplicación de esta técnica es muy beneficiosa, pues erradica por completo el efecto de filtrado peine producido por el aliasing espacial. Se ha realizado la comparación de la técnica propuesta con otra existente, y se ha llegado a la conclusión de que ninguna es capaz de mejorar de forma definitiva el problema del *aliasing*. En determinados casos concretos una técnica es mejor que la otra, por lo que, de momento, es preferible emplear en cada caso la más adecuada.

- Se ha descrito una arquitectura software capaz de implementar un motor de renderizado de WFS en tiempo real con un único ordenador personal. Esto se ha conseguido enfocando el problema tanto desde el punto de vista de la programación informática como del tratamiento digital de la señal. Por un lado se ha buscado obtener la máxima eficiencia de cómputo del *hardware*, por otro, se han optimizado todas las ramas de procesado para reducir al máximo sus necesidades.

- Se ha presentado una división de las tareas necesarias para sintetizar un campo sonoro mediante WFS, la cual permite aprovechar el paralelismo inherente de los procesadores actuales de varios núcleos. Esta división permite incluso realizar el procesado necesario en un procesador de un único núcleo, siempre y cuando se limiten los tiempos de reverberación de las salas a simular con la auralización.

- Se ha propuesto una forma eficiente de realizar el filtrado en WFS separando la ecualización y el retardo fraccionario en dos filtrados independientes. La ventaja de dividir el procesado radica en que la parte más costosa, la ecualización, solamente se aplica una vez por

fuente, mientras que el retardo fraccionario, mucho más liviano, debe aplicarse una vez por cada par fuente-altavoz. Además, así es posible realizar el filtrado de ecualización con filtros IIR con lo que se reduce todavía más la necesidad de cálculo.

- Se han estudiado las posibles estrategias para introducir un sistema de autoría de WFS sencillo y cómodo de usar, en un estudio de grabación en el que se emplean los actuales sistemas multipista. A partir de este estudio se ha obtenido una estrategia híbrida, que permite el acercamiento de los sistemas WFS a los estudios de producción de sonido actuales.

- Se ha propuesto una arquitectura software, basada en la estrategia híbrida, capaz de introducirse en los estudios de producción de sonido actuales de forma sencilla, eficiente y sobre todo económica. La arquitectura está compuesta por tres módulos: un gestor de las escenas, un *plug-in* VST y el motor de renderizado.

- Se ha realizado un estudio de la viabilidad del uso de XML3DAudio como base para la codificación de las escenas de WFS. Como conclusión de este estudio, el autor ha propuesto una serie de modificaciones a realizar sobre XML3DAudio que permiten expresar determinados aspectos únicos de WFS, así como la necesidad de poder representar coordenadas relativas. Lo que permite mantener la integridad de una escena aunque se reproduzca en instalaciones de WFS con tamaños muy distintos.

6.3. Líneas abiertas de trabajo

A continuación se describen las líneas de investigación que quedan abiertas después del trabajo realizado en esta tesis:

- El efecto *doppler* incluido en la implementación es una aproximación, en realidad se aplica una modulación en frecuencia sobre la señal que es equivalente a la producida por el efecto *doppler*. Recientemente [Ahrens and Spors, 2008b] se ha formalizado la representación de fuentes en movimiento, por lo que sería conveniente estudiar la viabilidad de su implementación en tiempo real.

- La técnica propuesta para sintetizar fuentes focalizadas indica que la apertura completa se obtiene al alcanzar la zona central del array, de tamaño justo la mitad que el array. Experimentalmente se ha comprobado que este tamaño es adecuado para salas de tamaño medio o pequeño. Sería conveniente comprobar como se comporta la técnica en salas de mayor tamaño.

- Es necesario incluir los efectos de elevación dentro del software, los cuales se han obviado en el desarrollo de esta tesis.

- Uno de los mayores inconvenientes que ha aparecido al emplear un único ordenador para WFS es la dificultad que existe en encontrar un *hardware* de sonido que proporcione la cantidad necesaria de canales de salida. Sería conveniente estudiar la posibilidad de exteriorizar este *hardware* y gestionarlo de forma sincronizada por medio de una red local.

6.4. Publicaciones

Durante el desarrollo de esta tesis doctoral se han ido publicando diversos aspectos de la investigación en forma de comunicaciones tanto en congresos nacionales como internacionales. A continuación se muestra una lista de aquellas publicaciones que ha realizado el autor o en las que ha contribuido con el trabajo desarrollado en esta tesis.

- S. Bleda, J. J. López y J. Escolano, "Simulación y Análisis de Prestaciones para Configuraciones de Arrays para Wave Field Synthesis", XVIII Simposium de la unión científica internacional de radio (URSI), La Coruña, Sept. 2003.

- S. Bleda, J. J. López and B. Pueo, "*Software for the simulation, performance analysis and real-time implementation of Wave Field Synthesis systems for 3D-Audio*", *6th International Conference on Digital Audio Effects DAFx-03*, Londres, Reino Unido, Sept. 2003.

- J. J. López, S. Bleda, B. Pueo and J. Escolano, "*A Sub-band Approach to Wave-Field Synthesis Rendering*", *118th Convention of the Audio Engineering Society*, Barcelona, May 2005.

- S. Bleda, J. J. López, J. Escolano and B. Pueo, "*Design and Implementation of a Compatible Wave Field Synthesis Authoring Tool*", *118th Convention of the Audio Engineering Society*, Barcelona, May 2005.

- J. Escolano, B. Pueo, S. Bleda and J. J. López, "*Virtual rooms recreation for Wave Field Synthesis*", *118th Convention of the Audio Engineering Society*, Barcelona, May 2005.

- J. Escolano, J. J. López, B. Pueo and S. Bleda, "Síntesis en Tiempo Discreto de Espacios Virtuales para Aplicaciones en Sonido Multicanal", XX Simposium de la Unión Científica Internacional de Radio (URSI), Gandia, Sept. 2005.

- S. Bleda, J. J. López, J. Escolano and B. Pueo, "*A flexible authoring tool for Wave Field Synthesis*", *International Computer Music Conference, ICMC*, Barcelona, Sept. 2005.

Además de los congresos descritos, el autor ha co-publicado un artículo en revista el cual, aun no siendo el autor principal, el trabajo realizado en esta tesis ha sido de vital importancia para su desarrollo.

- B. Pueo, J. J. López, J. Escolano and S. Bleda, "*Analysis of Multiactuator Panels in Space-Time Wavenumber Domain*", *Journal of the Audio Engineering Society*, vol. 5, páginas 1092-1106, Dec. 2007.

Apéndice A

Notación

En las siguientes secciones se mostrará la notación empleada en esta tesis.

A.1. Convenciones

Las convenciones que se han empleado son las siguientes:

- Los escalares en el dominio del tiempo se representan con las letras en minúsculas, e.g., $x(t)$.
- Los escalares en el dominio de la frecuencia se representan con las letras en mayúsculas, e.g., $X(f)$.
- Los vectores en el dominio del tiempo se representan con letras en negrita y en minúsculas, e.g., $\boldsymbol{x}(t)$.
- Los vectores en el dominio de la frecuencia se representan con letras en negrita y en mayúsculas, e.g., $\boldsymbol{X}(f)$.
- El código fuente se representa con letras monoespaciadas, e.g., `x[n]`.

A.2. Lista de símbolos

$A(\boldsymbol{r}_n, w)$	FFT de la modulación en amplitud de la señal *driving*.
c	Velocidad del sonido en el aire, 343 m/s a 20ºC.
Δx	Separación entre los altavoces.
f	Frecuencia lineal.
f_s	Frecuencia de muestreo.
j	Número complejo.
k	Número de onda.
$\boldsymbol{n}$	Vector normal.
N	Tamaño de bloque empleado.
$q(\boldsymbol{r}_n, t)$	Señal *driving* del altavoz n.
$Q(\boldsymbol{r}_n, w)$	Transformada de fourier de $q(\boldsymbol{r}_n, t)$.
r_0	Distancia perpendicular oyente-segmento.
$\boldsymbol{r}_n$	Vector que va desde la fuente hasta el altavoz n.
r_n	Módulo de $\boldsymbol{r}_n$ (distancia fuente-altavoz n).
$s(t)$	Señal emitida por la fuente virtual.
$S(w)$	Transformada de fourier s(t).
s_0	Distancia perpendicular fuente-segmento.
t_n	Tiempo que tarda el sonido en recorrer r_n.
T	Periodo de muestreo, equivalente a $1/f_s$.
θ_n	Ángulo entre el segmento fuente-altavoz y el eje del altavoz.
w	Frecuencia angular.
$\boldsymbol{x}$	Posición de la fuente virtual.

A.3. Abreviaturas y acrónimos

AABIFS	*Advanced Audio BIFS*
ALSA	*Advanced Linux Sound Architecture*

AMD	*Advanced Micro Devices*
ASIO	*Audio Stream Input/Output*
BIFS	*Binary Format For Scenes*
CD	*Compact Disk*
CISC	*Complex Instruction Set Computer*
DAW	*Digital Audio Workstation*
DLL	*Dynamic Link Library*
DMA	*Direct Memory Access*
DML	*Distributed Mode Loudspeaker*
DSP	*Digital Signal Processor*
DVD	*Digital Versatile Disk*
FDTD	*Finite Difference Time Domain*
FIR	*Finite Impulse Response Filter*
HOA	*Higher Order Ambisonics*
HRIR	*Head Related Impulse Response*
HRTF	*Head Related Transfer Function*
IID	*Inter-aural Intensity Difference*
IIR	*Infinite Impulse Response Filter*
ILD	*Inter-aural Level Difference*
IP	*Internet Protocol*
ITD	*Inter-aural Time Difference*
LFE	*Low Frequency Extension/Effects*
MAP	*Multiactuator Panel*
MME	*Multimedia Extensions*
MMX	Conjunto de instrucciones SIMD
MPEG	*Motion Pictures Expert Group*
OPSI	*Optimised Phantom Source Imaging*

POO	Programación Orientada a Objetos
RIR	*Room Impulse Response*
RISC	*Reduced Instruction Set Computer*
SDK	*Software Development Kit*
SIMD	*Single Instruction Multiple Data*
SOS	*Second Order Sections Filter*
SSE	*Streaming SIMD Extensions*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
VBAP	*Vector Base Amplitude Panning*
VRML	*Virtual Reality Modeling Language*
VST	*Virtual Studio Technology*
WFS	*Wavefield Synthesis*
XML	*Extended Markup Language*

Apéndice B

Protocolo UDP de gestión de las escenas

A continuación se muestra la lista de comandos que admite el núcleo de renderizado a través de red mediante el protocolo UDP. Los comandos están divididos en grupos según su función.

B.1. Comandos globales

- `start`

  Arranca la reproducción. No debe confundirse con el comando `play`.

- `stop`

  Detiene la reproducción y apaga el reproductor.

B.2. Creación de fuentes sonoras.

- `create source <id> <file>`

Crea una fuente sonora con el identificador `<id>` proporcionado (número entero). `<file>` indica el fichero wav (incluida la ruta) del cual se obtendrá el sonido.

- `kill source <id>`

  Borra la fuente cuyo identificador sea `<id>`.

B.3. Control de una fuente sonora.

Todo mensaje que haga referencia a una fuente sonora tiene el formato: `source <id> <orden> <param1> <param2> ...`

Donde:

- `<id>` es el identificador de la fuente, un número entero que se corresponde con el orden de creación de la fuente.

- `<orden>` es una cadena de texto con la orden que se quiere enviar a la fuente.

- `<param1>, <param2>, ...` son los parámetros que acompañan a la orden.

A continuación se muestra la lista de comandos que se pueden enviar a las fuentes:

- `pos_cart <x> <y> <z>`

  Indica el cambio de posición de una fuente. La posición se indica en metros y en coordenadas cartesianas mediante tres números en coma flotante.

- `play [0 1]`

Indica el comienzo o parada de la reproducción de una fuente. 0 parar la reproducción, 1 comenzar a reproducir.

- `loop [0 1]`

  Permite reproducir de forma reiterada el sonido de una fuente. 0 desactivar, 1 activar.

- `gain <g>`

  Indica el valor de la ganancia a aplicar a la fuente. `<g>` será un valor en coma flotante comprendido entre 0 y 5 (de -Inf. a +15 dB aproximadamente.)

- `plane_wave [0 1]`

  Indica si la fuente es onda plana o esférica. 0 onda esférica (por defecto), 1 onda plana.

- `focused [0 1]`

  Indica el tratamiento a emplear con las fuentes focalizadas. 0 para el tratamiento normal, 1 para emplear el cono de apertura.

Ejemplos:

- Cambiar la posición de la fuente 1: `source 1 pos_cart 0.5 1.2 0.0`
- Cambiar la ganancia de la fuente 2: `source 2 gain 0.9`
- Indicar que la fuente 2 es una onda plana: `source 2 plane_wave 1`

Apéndice C

Software de simulación numérica

Con la finalidad de minimizar el tiempo de trabajo, antes de comenzar a programar la aplicación de WFS en tiempo real, se creó una aplicación de simulación numérica de WFS en Matlab [Bleda et al., 2003b]. La mayoría de las gráficas mostradas en esta tesis se han construido mediante esta aplicación.

Esta aplicación simula el campo sonoro mediante un modelo analítico de la ecuación de síntesis de WFS, discretizando el espacio con una rejilla ajustable.

En la Figura C.1 se puede ver el interfaz gráfico de la aplicación, con el cual es posible describir de forma sencilla la configuración de la escena. Entre otras cosas permite indicar la posición de la fuente virtual y la frecuencia del tono que va a emitir, la cantidad de segmentos (máximo 3) que componen el array, el número de altavoces que los componen, la separación entre estos y el tamaño del mallado que se va a emplear.

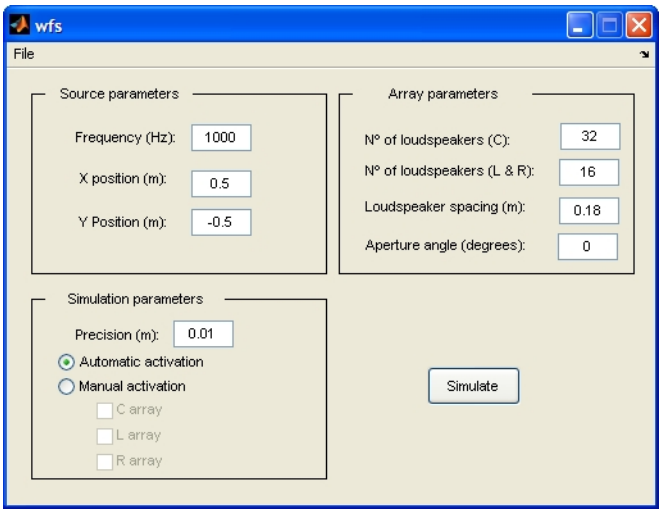


Figura C.1. Interfaz gráfica de la aplicación de simulación.

El diagrama de bloques de la Figura C.2 muestra la lógica de trabajo que sigue la aplicación. Y tal y como se indica, en primer lugar se colocan los altavoces en el espacio. Seguidamente se obtienen las señales driving de cada altavoz, para, a continuación, obtener el campo sonoro en toda la sala. El campo sonoro en la sala se obtiene como combinación lineal del campo sonoro generado por cada fuente secundaria por separado. El campo generado por cada fuente secundaria viene dado por

$$P(\boldsymbol{x},t) = \frac{1}{|\boldsymbol{x}_n - \boldsymbol{x}|} e^{-jk|\boldsymbol{x}_n - \boldsymbol{x}|} e^{jwt}, \tag{C.1}$$

en la cual $\boldsymbol{x}$ es un punto cualquiera de la sala, t es el tiempo, $\boldsymbol{x}_n$ es la posición de la fuente secundaria (altavoz) enésima, w es la frecuencia angular del tono puro emitido y k es el número de onda. Como se puede apreciar la Ecuación C.1 se puede dividir en dos términos, uno que depende únicamente del espacio y otro del tiempo. El término espacial será constante una vez calculado, mientras que el temporal habrá que recalcularlo para

cada instante de tiempo.

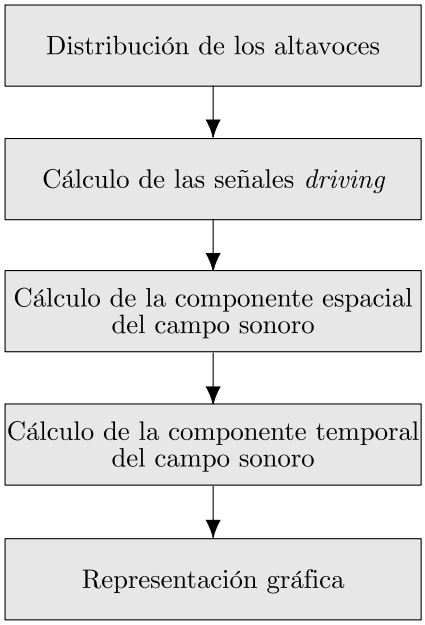


Figura C.2. Diagrama de bloques de la aplicación.

Una vez se obtiene el campo de presiones en toda la sala para dos instantes de tiempo, es posible calcular la velocidad de las partículas a partir de la diferencia entre ambos. Esta forma de calcular la velocidad de las partículas viene a ser la empleada por la técnica FDTD.

Bibliografía

M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. Dover Publications, 1972.

J. Ahrens and S. Spors. Focusing of virtual sound sources in higher order ambisonics. In *Proc. of the 124th Audio Eng. Soc. Convention*, Amsterdam, The Netherlands, May 2008a.

J. Ahrens and S. Spors. Reproduction of moving virtual sound sources with special attention to the doppler effect. In *Proc. of the 124th Audio Eng. Soc. Convention*, Amsterdam, The Netherlands, May 2008b.

V. R. Algazi, R. Duda, and M. Thompson. Motion-tracked binaural sound. In *Proc. of the 116th Audio Eng. Soc. Convention*, Berlin, Germany, May 2004.

ALSA. Advanced linux sound architecture (alsa) project homepage, 2009. URL `http://www.alsa-project.org`.

G. R. Andrews. *Concurrent Programming: Principles and Practice*. Addison-Wesley, 1991.

B. S. Atal and M. R. Schroeder. Simulation of room acoustics using electronic computers. *Gravesaner Blätter*, 27/28:124–137, 1966.

M. A. J. Baalman. Wonder: Wave field synthesis of new dimensions of electronic music in realtime. http://gigant.kgw.tu-berlin.de/∼baalman/, 2004.

D. W. Batteau. The role of the pinna in human sound localization. *Proceedings of Royal Society*, 168:158–180, 1967.

B. B. Bauer. Phasor analysis of some stereophonic phenomena. *J. Acoust. Soc. Am.*, 33:1536–1539, Nov. 1961.

B. B. Bauer, G. A. Budelman, and D. W. Gravereaux. Recording techniques for sq matrix quadraphonic discs. *J. Audio Eng. Soc.*, 1:19–26, 1973.

D. R. Begault. Binaural auralization and perceptual veridicality. In *Proc. of the 93rd Audio Eng. Soc. Convention*, San Francisco, California, USA, Oct. 1992.

D. R. Begault. *3-D Sound for Virtual Reality and Multimedia*. Academic Press Professional, 1994.

J. C. Bennett, K. Barker, and F. O. Edeko. A new approach to the assessment of stereophonic sound system performance. *J. Audio Eng. Soc.*, 33: 314–321, 1985.

A. J. Berkhout. *Applied Seismic Wave Theory*. Elsevier Science, 1987.

A. J. Berkhout. A holographic approach to acoustic control. *J. Audio Eng. Soc.*, 36(12):977–995, Dec. 1988.

A. J. Berkhout, P. Vogel, and D. de Vries. Use of wave field synthesis for natural reinforced sound. In *Proc. of the 92nd Audio Eng. Soc. Convention*, Vienna, Austria, Mar. 1992.

A. J. Berkhout, D. de Vries, and P. Vogel. Acoustic control by wave field synthesis. *J. Acoust. Soc. Am.*, 93(5):2764–2778, 1993.

J. Blauert. *Spatial hearing: the psychophysics of human sound localization.* MIT Press, Cambridge, 1997.

S. Bleda, J. J. López, and J. Escolano. Simulación y análisis de prestaciones para configuraciones de arrays para wave field synthesis. In *XVIII Simposium de la unión científica internacional de radio (URSI'03)*, La Coruña, España, Sep. 2003a.

S. Bleda, J. J. López, and B. Pueo. Software for the simulation, performance analysis and real-time implementation of wave field synthesis systems for 3d-audio. In *Proc. of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*, London, UK, Sep. 2003b.

S. Bleda, J. J. López, J. Escolano, and B. Pueo. Design and implementation of a compatible wave field synthesis authoring tool. In *Proc. of the 118th Audio Eng. Soc. Convention*, Barcelona, Spain, May 2005.

N. Bleistein. *Mathematical methods for wave phenomena.* Academic Press Inc., 1984.

M. M. Boone. Acoustic rendering with wave field synthesis. In *Proc. of the ACM Siggraph and Eurographics Campfire on Acoustic Rendering for Virtual Environments*, pages 37–45, May 2001.

M. M. Boone and W. P. J. de Bruijn. On the applicability of distributed mode loudspeaker panels for wave field synthesis based sound reproduction. In *Proc. of the 108th Audio Eng. Soc. Convention*, Feb. 2000.

M. M. Boone and E. N. G. Verheijen. Multi-channel sound reproduction based on wave field synthesis. In *Proc. of the 95th Audio Eng. Soc. Convention*, New York, USA, Oct. 1993.

M. M. Boone, E. N. G. Verheijen, and P. E. Van Tol. Spatial sound-field reproduction by wave-field synthesis. *J. Audio Eng. Soc.*, 43(12):1003–1012, 1995.

W. Bucholtz. *Planning a computer system: Project Stretch.* McGraw-Hill, 1962.

T. Caulkins, E. Corteel, and O. Warusfel. Wave field synthesis interactionwith the listening environment, improvements in the reproduction of virtual sources situated inside the listening room. In *Proc. of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*, London, UK, Sep. 2003.

P. Damaske and B. Wagener. Investigations of directional hearing using a dummy head. *Acustica*, 21:30–35, 1969.

J. Daniel. *Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia.* PhD thesis, Université Paris 6, 2001.

J. Daniel, J. B. Rault, and J. D. Polack. Ambisonics encoding of other audio formats for multiple listening conditions. In *Proc. of the 105th Audio Eng. Soc. Convention*, San Francisco, California, USA, Sep. 1998.

J. Daniel, S. Moreau, and R. Nicol. Further investigations of high order ambisonics and wavefield synthesis for holophonic sound imaging. In *Proc. of the 114th Audio Eng. Soc. Convention*, Amsterdam, The Netherlands, Mar. 2003.

D. de Vries. Sound reinforcement by wavefield synthesis: Adaptation of the synthesis operator to the loudspeaker directivity characteristics. *J. Audio Eng. Soc.*, 44(12):1120–1131, 1996.

D. de Vries and E. Huselbos. Auralization of room acoustics by wave field synthesis based on array measurement of impulsive responses. In *Proc. of the XII European Signal Processing Conference (EUSIPCO'04)*, Vienna, Austria, Sep. 2004.

D. de Vries, A. J. Reijnen, and M. A. Schonewille. The wave field synthesis concept applied to generation of reflections and reverberation. In *Proc. of the 96th Audio Eng. Soc. Convention*, Amsterdam, The Netherlands, Mar. 1994.

G. DellaSala. Introducing the 10.2 surround format, Sept. 2006. URL `http://www.audioholics.com/news/on-location-articles/on-location-with-audyssey-laboratories/introducing-the-10-2-surround-format`.

C. Doppler. Uber das farbige licht der doppelsterne und einiger anderer gestirne des himmels. *Abhandlungen der königlichen böhmischen Gesellschaft der Wissenschaften*, 2:465–482, 1842.

J. Escolano. *Contributions to Discrete-Time Domain methods in Room Acoustic Simulations*. PhD thesis, Universidad Politécnica de Valencia, Jul. 2008.

J. Escolano, S. Bleda, B. Pueo, and J. J. López. Wave field synthesis 3d simulator based on finite-difference time-domain method. In *Proc. of the 116th Audio Eng. Soc. Convention*, Berlin, Germany, May 2004a.

J. Escolano, S. Bleda, B. Pueo, and J. J. López. Wave field synthesis simulation by means of finite-difference time-domain technique. In *12th European Signal Processing Conference*, Vienna, Austria, Sep. 2004b.

EventHelix. Optimizing c and c++ code, 2009. URL `http://www.eventhelix.com/RealtimeMantra/Basics/OptimizingCAndCPPCode.htm`.

P. B. Fellget. From quadro to surround sound. *J. Audio Eng. Soc.*, 5:19–25, 1977.

A. Fog. Software optimization resources, 2009. URL `http://www.agner.org/optimize/`.

A. Franck, A. Gräfe, T. Korn, and M. Strauß. Reproduction of moving sound sources by wave field synthesis: An analysis of artifacts. In *Proc. of the 32nd AES International Conference*, Hillerød, Denmark, Sep. 2007.

B. Gardner and K. Martin. Hrtf measurements of a kemar dummy-head microphone. Technical Report 280, M.I.T. Media Lab Perceptual Computing, May 1994.

M. A. Gerzon. The principles of quadraphonic recording. part 1: Are four channels really necessary? *Studio Sound*, 12:338–342, Aug. 1970a.

M. A. Gerzon. The principles of quadraphonic recording. part 2: The vertical element. *Studio Sound*, 12:380–384, Sept. 1970b.

M. A. Gerzon. Periphony: With-height sound reproduction. *J. Audio Eng. Soc.*, 21:2–10, Jan./Feb. 1973.

M. A. Gerzon. What's wrong with quadraphonics? *Studio Sound*, 16:50–56, May 1974.

M. A. Gerzon. Ambisonics: Part two: Studio techniques. *Studio Sound*, pages 24–30, Aug. 1975.

B. S. Gottfried. *Programación En C*. McGraw-Hill, 2005.

C. Huygens. *Traité de la Lumière*. 1678.

Intel. *Intel® Signal Processing Library. Reference Manual.* Intel Corporation, 2000.

Intel. *Intel 64 and IA-32 Architectures Software Developer's Manual. Volume 3A - System Programming Guide, Part 1.* Intel Corporation, Mar. 2009a.

Intel. Intel® hyper-threading technology (intel® ht technology). http://www.intel.com/technology/platform-technology/hyper-threading/, 2009b.

ISO/IEC. Mpeg-1. information technology – coding of moving pictures and associated audio for digital storage media at up to about 1,5 mbit/s – part 1: Systems. ISO/IEC 11172-1, 1993.

ISO/IEC. Mpeg-2. information technology – generic coding of moving pictures and associated audio information: Video. ISO/IEC 13818-2, 1996.

ISO/IEC. Vrml97 - virtual reality modeling language. part 1: Functional specification. ISO/IEC 14772-1, 1997.

ISO/IEC. Mpeg-4. information technology – coding of audio-visual objects – part 1: Systems. ISO/IEC 14496-1, 1999.

ISO/IEC. Mpeg-4. information technology – coding of audio-visual objects – part 1: Systems. ISO/IEC 14496-1, 2001.

ISO/IEC. Vrml97 - virtual reality modeling language. part 2: External authoring interface. ISO/IEC FDIS 14772-2, 2004.

M. Kleiner, B. Dalenbäck, and P. Svensson. Auralization an overview. In *91st Audio Eng. Soc. Convention*, New York, USA, Oct. 1991. Preprint 3119.

R. Koenen. Overview of the mpeg-4 standard. Technical report, ISO/IEC, Mar. 2002.

R. Kürer, G. Plenge, and H. Wilkens. Correct spatial sound perception rendered by a special 2-channel recording method. In *Proc. of the 37th Audio Eng. Soc. Convention*, New York, USA, Oct. 1969.

T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine. Splitting the unit delay - tools for fractional delay filter design. *IEEE Signal Processing Magazine*, 13:30–60, 1996.

H. S. Linkwitz. Passive crossover networks for noncoincident drivers. *J. Audio Eng. Soc.*, 24:2–8, 1976.

J. J. López. *Reproduction of 3D-Sound using local control techniques*. PhD thesis, Communications Department, Technical University of Valencia, 1999.

J. J. López, A. González, and F. Orduña-Bustamante. Measurement of cross-talk cancellation and equalization zones in 3-d sound reproduction under real listening conditions. In *Proc. of the 16th Audio Eng. Soc. International Conference*, Arktikum, Rovaniemi, Finland, Apr. 1999.

J. J. López, S. Bleda, B. Pueo, and J. Escolano. A sub-band approach to

wave-field synthesis rendering. In *Proc. of the 118th Audio Eng. Soc. Convention*, Barcelona, Spain, May 2005.

Microsoft. Understanding scheduling, thread context, and irql. Application note, Microsoft Corporation, July 2004.

Microsoft. Locks, deadlocks, and synchronization. Application note, Microsoft Corporation, April 2006.

M. Mittal, A. Peleg, and U. Weiser. Mmx technology architecture overview. *Intel Technology Journal*, 1(1), 1997.

H. Møller. Fundamental of binaural technology. *Applied Acoustics*, 36: 171–218, 1992.

G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, April 1965.

F. M. Márquez. *Unix Programacion Avanzada*. Ra-Ma, 2004.

NHK. Super hi-vision, 2002. URL `http://www.nhk.or.jp/digital/en/superhivision/index.html`.

R. Nicol and M. Emerit. Reproducing 3d-sound for video conferencing: A comparison between holophony and ambisonic. In *Proceedings of the First COST-G6 Workshop on Digital Audio Effects (DAFX-98)*, Barcelona, Spain, Nov. 1998.

K. Ono and S. Komiyama. A method of reproducing concert hall sounds by "loudspeaker walls". In *Proc. of the 102nd Audio Eng. Soc. Convention*, Munich, Germany, Mar. 1997.

J. Ortega, M. Anguita, and A. Prieto. *Arquitectura de computadores*. Thomson, 2005.

R. Pellegrini and C. Kuhn. Wave field synthesis: Mixing and mastering tools for digital audio workstations. In *Proc. of the 116th Audio Eng. Soc. Convention*, Berlin, Germany, May 2004.

A. D. Pierce. *Acoustics: An Introduction to Its Physical Principles and Applications*. Acoustical Society of America, 1991.

Platon. La republica (politeia). Grecia, 395 AC.

G. Potard. *3D-Audio object oriented coding*. PhD thesis, University of Wollongong, Australia, Sept. 2006.

B. Pueo. *Analysis and enhancement of Multiactuator Panels for Wave-Field Synthesis reproduction*. PhD thesis, Universidad Politécnica de Valencia, Sep. 2008.

B. Pueo and M. Romá. *Electroacústica, Altavoces y Micrófonos*. Pearson Prentice Hall, 2003.

B. Pueo, J. J. López, J. Escolano, and S. Bleda. Analysis of multiactuator panels in space-time wavenumber domain. *J. Audio. Eng. Soc.*, 5:1092–1106, 2007.

V. Pulkki. *Spatial sound generation and perception by amplitude panning techniques*. PhD thesis, Helsinki University of Technology, Finland, 2001a.

V. Pulkki. Localization of amplitude-panned virtual sources ii: Two- and three-dimensional panning. *J. Audio Eng. Soc.*, 49(9):753–767, Sept. 2001b.

V. Pulkki and M. Karjalainen. Localization of amplitude-panned virtual sources i: Sterophonic panning. *J. Audio Eng. Soc.*, 49(9):739–752, Sept. 2001.

G. Ramos and J. J. López. Filter design method for loudspeaker equalization based on iir parametric filtres. *J. Audio Eng. Soc.*, 54(12):1162–1178, 2006.

J. W. S. Rayleigh. Our perception of sound direction. *Philosophical Magazine*, 13:214–232, 1907.

D. Rocchesso. Fractionally addressed delay lines. *IEEE Transactions On Speech And Audio Processing*, 8(6):717–727, 2000.

F. Rumsey, D. Griesinger, T. Holman, M. Sawaguchi, G. Steinke, G. Theile, and T. Wakatuki. Multichannel surround sound systems and operations. Technical Document AESTD1001, Audio Engineering Society, 2001.

J. O. Smith. *Physical audio signal processing for virtual musical instruments and audio effects*. Center for Computer Research in Music and Acoustics (CCRMA), 2008.

J. O. Smith, S. Serafin, J. Abel, and D. Berners. Doppler simulation and the leslie. In *Proc. of the 5th Int. Conference on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, Sep. 2002.

W. B. Snow. Basic principles of stereophonic sound. *J. Soc. of Motion Picture and Television Engineers*, 61(11):922–940, Mar 1953.

J. J. Sonke. *Variable Acoustics by wave field synthesis*. PhD thesis, Technical University Delft, The Netherlands, 2000.

T. Sporer, J. Plogsties, and S. Brix. Carrouso - an european approach to 3d-audio. In *Proc. of the 110th Audio Eng. Soc. Convention*, Amsterdam, The Netherlands, May 2001.

S. Spors. An analytic secondary source selection criterion for wavefield synthesis. In *Proc. Annual German Conf. on Acoustics (DAGA)*, Stuttgart, Germany, Mar. 2007a.

S. Spors. Extension of an analytic secondary source selection criterion for wave field synthesis. In *Proc. of the 123rd Audio Eng. Soc. Convention*, New York, USA, Oct. 2007b.

S. Spors and J. Ahrens. Spatial sampling artifacts of focused sources in wave field synthesis. In *NAG-DAGA International Conference on Acoustics*, Rotterdam, The Netherlands, Mar. 2009.

S. Spors, H. Teutsch, and R. Rabenstein. High-quality acoustic rendering with wave field synthesis. *Vision, Modelling and Visualization*, pages 101–108, 2002.

S. Spors, R. Rabenstein, and J. Ahrens. The theory of wave field synthesis revisited. In *Proc. of the 124th Audio Eng. Soc. Convention*, Amsterdam, The Netherlands, May 2008.

E. W. Start. *Direct Sound Enhancement by Wave Field Synthesis*. PhD thesis, Delft University of Technology, 1997.

E. W. Start, V. G. Valstar, and D. de Vries. Application of spacial bandwidth reduction in wave field synthesis. In *Proc. of the 98th Audio Eng. Soc. Convention*, Paris, France, Feb. 1995.

Steinberg. *ASIO 2.2 - Audio Streaming Input Output Development Kit*, 2006.

Steinberg. Vst plug-ins sdk 2.3, 2009. URL `www.steinberg.net`.

M. J. Strauß and M. Munderloh. Influence of loudspeaker displacement on the reproduction quality of wave field synthesis systems. In *Proc. of the 19th International Congress on Acoustics*, Madrid, Spain, Sep. 2007.

A. S. Tanenbaum and A. S. Woodhull. *Sistemas Operativos: Diseño e Implementacion.* Prentice Hall, 1998.

G. Theile. On the performance of two-channel and multi-channel stereophony. In *Proc. of the 88th Audio Eng. Soc. Convention*, Montreux, Switzerland, Mar. 1990.

G. Theile and H. Wittek. Wave field synthesis: A promising spatial audio rendering concept. *Acoustical Science and Technology*, 25(6):393–399, 2004.

G. Theile, H. Wittek, and M. Reisinger. Potential wave-field synthesis application in the multichannel stereophonic world. In *24th Audio Eng. Soc. International Conference on Multichannel Audio*, 2003.

A. Torger. Brutefir. http://www.ludd.luth.se/∼torger/brutefir.html, 2001.

A. Torger and A. Farina. Real-time partitioned convolution for ambiophonics surround sound. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, USA, Oct. 2001.

B. Vercoe. *The Canonical Csound Reference Manual.* MIT Media Lab, http://www.csounds.com/manual/html/index.html, Dec. 2008.

E. N. G. Verheijen. *Sound Reproduction by Wave Field Synthesis.* PhD thesis, Delft University of Technology, 1997.

H. Wilkens. Head-related stereophony, an aid in comparing and judjing various spatial impresions. *Acustica*, 26:213–221, 1972.

E. J. Williams. *Fourier Acoustics: Sound Radiation and Nearfeld Acoustical Holography.* Academic Press, 1999.

H. Wittek. Opsi: A hybrid wfs / phantom source solution to avoid spatial aliasing. Technical report, IRT, 2002.

H. Wittek. Perception of spatially synthesized sound fields - literature review about wfs. Technical report, http://www.hauptmikrofon.de/wittek.htm, Dec. 2003.

X3D. X3d: Open standards for real-time 3d communication. Web3D Consortium. http://www.web3d.org/x3d/specifications/, 2004.