Implementación de un Sistema de Sonido Envolvente para Ambientes Tipo CAVE

John Jairo Palma Robayo
Universidad de los Andes
Ingeniería de Sistemas y Computación
joh-palm@uniandes.edu.co

Proyecto de Grado para Ingeniería de Sistemas y Computación

Contenidos

Abstract **1**

Glosario **2**

1. Introducción **3**
- 1.1. ¿Por qué audio en aplicaciones inmersivas? Motivación
- 1.2. Objetivos del sistema
- 1.3. Usuarios del Proyecto
- 1.4. Guía del proyecto
- 1.5. Agradecimientos

2. Marco Teórico **5**
- 2.1. Investigación previa.
  - 2.1.1. Spatialized Audio Rendering for Immersive Virtual Environments
  - 2.1.2. Perceptual Audio Rendering of Complex Virtual Environments
  - 2.1.3. Echology
  - 2.1.4. Dirt-Cheap 3D Spatial Audio
  - 2.1.5. Otros proyectos relevantes.
- 2.2. Algoritmos de Espacialización
  - 2.2.1. VBAP

3. Herramientas para Sonido Espacializado y Desarrollo **11**
- 3.1. Pruebas a realizarse en los sistemas
- 3.2. Descripción de las herramientas
  - 3.2.1. SuperCollider
  - 3.2.2. Sonix
  - 3.2.3. OpenAL
  - 3.2.4. CSound
  - 3.2.5. PortAudio
- 3.3. Herramientas adicionales de desarrollo

4. Diseño de la Herramienta **16**
- 4.1. Requerimientos del Sistema
  - 4.1.1. Requerimientos Funcionales
  - 4.1.2. Requerimientos No Funcionales
  - 4.1.3. Casos de uso.
- 4.2. Diseño del Sistema

5. Implementación del Sistema **29**
5.1. Herramientas Seleccionadas
5.2. Características del Sistema Desarrollado
5.2.1. Desarrollo con OpenAL
5.2.2. Desarrollo con PortAudio

6. Uso del Sistema **34**
6.1. Instalación/Uso del sistema
6.2. Inicialización del sistema
6.2.1. Configuración del sistema de Audio
6.3. Creación y manipulación de objetos en el sistema
6.4. Instalación física posible del sistema
6.5. Pruebas Realizadas
6.6. Trabajo por desarrollar
6.7. Problemas en la versión actual
6.8. Comparación con otros sistemas

7. Conclusiones **44**
7.1. Trabajo Futuro

8. Bibliografía **46**

Anexos **49**
Diseño detallado de las implementaciones del sistema
Tablas de Comparación entre los documentos investigados
Tablas de Comparación entre APIs
Demos Realizados.
Proceso de instalación de las herramientas analizadas.
Tabla de Comparación del Sistema Finalizado con otros APIs

Abstract

Este documento describe el desarrollo incremental de un sistema de Audio 3D a usarse en ambientes interactivos donde se necesitan pistas auditivas de la ubicación del oyente. El sistema permite la creación de un conjunto de fuentes asociadas cada una a archivos de audio para su reproducción y son ubicadas en el espacio físico ya sea mediante el API subyacente o implementando un algoritmo de espacialización como VBAP [4]. Este documento ofrece una guía a los aportes realizados por este sistema de manera que provee una aproximación al desarrollo de sistemas de audio en general, y su especialización en particular.

Glosario

- **API:** Application Programming Interface.
- **Azimuth:** Ángulo en el plano xy tomado desde el eje x.
- **CAVE:** CAVE Automatic Virtual Environment.
- **Clipping:** Distorsión de una señal de audio causada, en el caso de señales digitales, por sumas de elementos superando los valores mínimos de un rango de datos.
- **CVS:** Concurrent Versión System.
- **Downsampling:** Proceso que reduce la sample rate de una señal
- **DSP:** Digital Signal Processing.
- **HRTF:** Head Related Transfer Function.
- **IDE:** Integrated Development Environment.
- **Resampling:** Proceso en el cual se cambia la tasa de muestreo (sample rate) de una señal de audio.
- **Sample Rate:** Número de muestras por segundo que se toman de una señal continua (en este caso, audio) para hacer una señal discreta.
- **Tracker:** Dispositivo que permite identificar el movimiento de un elemento incorporado en el espacio.
- **Upsampling:** Proceso que incrementa la sample rate de una señal.
- **VBAP:** Vector Base Panning Algorithm.

Capítulo 1

Introducción

1.1. ¿Por qué audio en aplicaciones inmersivas? Motivación

El uso de audio en aplicaciones virtuales inmersivas (Ej. *CAVE* [1]) es de gran importancia si se quiere obtener un ambiente realista para el usuario. Un elemento auditivo correctamente implementado permite darle a éste pistas sobre su ubicación en el mundo virtual, así como de los elementos que lo rodean. Dado lo anterior, es necesario el desarrollo de alternativas que le den una sensación acústica relevante a una persona que se encuentre interactuando con un proyecto particular que requiera su atención y constante manipulación.

Dentro de las motivaciones personales en primer lugar se encuentra la oportunidad de poder desarrollar una plataforma capaz de integrarse con un sistema más grande y estructurado, como es InTml [15]. Esta arquitectura debe ser flexible y simple, de manera que pueda ser extendida de acuerdo a las necesidades del sistema con el que se use.

Por otro lado, también está el hecho de poder hacer una investigación exhaustiva sobre distintas herramientas disponibles para la manipulación de audio, con el fin de poder utilizar estos conocimientos en otras áreas de interés, como son las aplicaciones multimedia, música y videojuegos.

1.2. Objetivos del Sistema

El sonido es un componente complejo de modelar. Lo que escuchamos está influenciado irremediablemente por un sinnúmero de factores, desde los acústicos –como la reverberación, las diversas formas de atenuación, reflexiones, etc.-, pasando por la forma como recibimos el sonido dada la forma de la cabeza humana, hasta la misma ubicación espacial de la fuente sonora.

Sin embargo, muchas de las variables mencionadas arriba son, en mayor o menor medida complejas de implementar o podrían impactar de manera negativa el desempeño del sistema, de forma que se ha optado por reducir el alcance del proyecto, enfocándose primero en la distribución espacial del sonido, de todas maneras dejando abierta la posibilidad a nuevas modificaciones que permitan incorporar todos los elementos posibles de los nombrados arriba. De acuerdo con esto, se han definido varios objetivos para el sistema, definiendo éstos la funcionalidad de la herramienta desarrollada.

En primer lugar está el permitir la asociación de fuentes gráficas a eventos sonoros, así como su reproducción de éstos últimos. De esta manera se tiene una relación directa del sonido con un objeto gráfico que puede o no estar siendo manipulado por la persona interactuando con el sistema. De esto se desprende también la muy posible existencia de múltiples fuentes sonoras, cada una asociada a los objetos existentes en el mundo, que deben ser reproducidas fielmente.

En segundo lugar, el movimiento de las fuentes en el espacio debe incidir en la ubicación espacial del sonido, de manera que la fuente se reproduce con mayor amplitud en un parlante más cercano a la misma.

En tercer lugar, el sistema debe ser configurable, de manera que se puedan escoger tanto el número de cajas acústicas a usarse en el sistema, como su distribución en el espacio, a fin de proveer mayor flexibilidad en el momento de crear las instalaciones físicas del proyecto.

1.3. ¿A quién está orientado? Usuarios del Proyecto

El proyecto se encuentra orientado a todo aquel interesado en ambientes interactivos que requieran el uso de audio, así que el sistema puede ser usado en simulaciones, instalaciones artísticas y musicales.

El primer uso inmediato, sin embargo, es un ambiente tipo CAVE a implementarse próximamente en el Laboratorio de Visualización Inmersiva y Sistemas Autónomos de la Universidad de los Andes, dirigido por el Grupo de Investigación IMAGINE [27].

1.4. Guía del Proyecto

En este documento se describen tanto el desarrollo como las principales características de un sistema de sonido espacializado, el cual ha sido desarrollado como Proyecto de Grado para Ingeniería de Sistemas y Computación, durante el segundo semestre de 2007, en un proceso dirigido por Pablo Figueroa Ph.D. Se espera que lo consignado aquí –junto con el código fuente del sistema [26]- sea una referencia del sistema desarrollado y una introducción a futuros trabajos relacionados con la interacción con audio. A continuación se presenta una descripción de cada uno de los capítulos restantes del documento:

- **Capítulo 2:** Este capítulo presenta varios proyectos relacionados con ubicación de audio en el espacio, así como un resumen de los más relevantes.

- **Capítulo 3:** En este capítulo se describen las diversas herramientas investigadas para la implementación de sistemas de audio espacializado, así como algunas de las pruebas usadas en ellas.

- **Capítulo 4:** Se refiere al diseño realizado para el sistema, desde la formulación de requerimientos hasta el diagrama de clases final del sistema. También expone los problemas encontrados durante esta fase.

- **Capítulo 5:** Este capítulo ahonda en los detalles de la implementación del sistema, así como algunos desarrollos paralelos del sistema usados como pruebas.

- **Capítulo 6:** Describe el uso del sistema finalizado, así como posibles implementaciones físicas del mismo y algunas de las pruebas desarrolladas. También describe los problemas actuales del sistema y alternativas de solución. Finalmente hace una comparación del sistema con otras alternativas existentes y algunos de los desarrollos investigados en el capítulo 2.

- **Capítulo 7:** El capítulo final del documento, presenta las conclusiones obtenidas del desarrollo de este proyecto, así como una guía para trabajos futuros sobre el mismo.

- **Anexos:** Presenta información de utilidad recopilada durante la fase de investigación del proyecto, así como diagramas detallados de las implementaciones realizadas y guías de

instalación de las herramientas utilizadas. Finalmente se incluye una tabla comparativa del sistema final con otros sistemas implementados existentes.

1.5. Agradecimientos

Quisiera tomar un momento para agradecer a las personas sin cuya cooperación, oportuno consejo o simple presencia habría sido mucho más difícil llegar a esta instancia definitiva de mi carrera. En primer lugar me gustaría agradecer a mi familia; a mis padres por el esfuerzo enorme que hicieron para darme estudio en la Universidad de los Andes y junto a ellos a mis hermanas quienes siempre estuvieron ahí para brindarme ánimo y recordarme lo importante de este proceso.

De igual manera me gustaría agradecer a mi director de proyecto, Pablo Figueroa Ph.D, quién estuvo pendiente de los avances del mismo y de hacer las sugerencias y correcciones necesarias para orientarlo, además de darme el tiempo necesario para lograr un buen resultado. Además me gustaría agradecer a todos mis compañeros del laboratorio por sus consejos no sólo de carácter técnico sino por el buen trato que me ofrecieron durante el tiempo que estuve allí. De manera especial me gustaría agradecer a Oscar Javier Chavarro por proveerme el demo del tracker, el cual fue de gran utilidad a la hora de hacer pruebas del sistema.

Capítulo 2

Marco Teórico

La primera etapa del proyecto estuvo enfocada a la investigación de desarrollos realizados anteriormente en el área de audio 3D en ambientes envolventes y de alta interacción con el usuario, así como posibles herramientas que pudiesen ser de ayuda en el desarrollo de un sistema similar, todo esto dado el poco conocimiento que se tenía sobre el tema antes de empezar, convirtiéndose toda esta información en una guía vital para la orientación del proyecto.

El capítulo se divide en dos secciones. La primera reseña varios de los proyectos investigados, a fin de ilustrar las diversas aproximaciones al problema de audio espacializado (también llamado audio 3D), así como sus aplicaciones. La segunda sección trata de algoritmos de especialización, particularmente VBAP (*Vector Base Panning Algorithm*), el cual es desarrollado en detalle.

2.1. Investigación previa

Existen diversas implementaciones anteriores de sistemas similares a lo deseado con distintos objetivos, con el fin de identificar problemáticas comunes asociadas con éstas. Los factores principales que se tuvieron en cuenta para revisar los proyectos fueron los siguientes:

- *Generalidades del Proyecto:* Se tuvieron en cuenta no solo implementaciones en ambientes tipo CAVE, sino también otros proyectos realizados para instalaciones artísticas, y también propuestas de otros sistemas interactivos que usan audio de manera intensa.

- *Hardware y Software Utilizado:* Se tuvo en cuenta desde el número de cajas acústicas utilizadas hasta los APIs subyacentes, incluyendo tecnologías propietarias y OpenSource. Cada una de estas implementaciones varío de acuerdo a los elementos utilizados, así como en el modelado riguroso de fenómenos acústicos como por ejemplo la reveberación.

2.1.1. *Spatialized Audio Rendering for Immersive Virtual Environments*

Este proyecto [12] -desarrollado por Martin Naef y Markus Gross, de ETH Suiza, en conjunto con Oliver Staadt de la Universidad de California, Davis-, consiste en la creación de un sistema para hacer procesamiento y presentación (*rendering*) de audio espacializado. Para cumplir con este objetivo se presenta una arquitectura capaz de manejar múltiples fuentes, ubicadas dentro de nodos de escena. Las fuentes son distribuidas espacialmente usando VBAP, y diversos procesos les son aplicados (atenuación y retraso (*delay*) por distancia, absorción del aire y reverberación usando unidades externas), siendo finalmente mezcladas. Una vez desarrollada la arquitectura ésta es integrada con un API denominado *blue-c*, el cual se encarga de proveer servicios para ambientes virtuales compartidos.

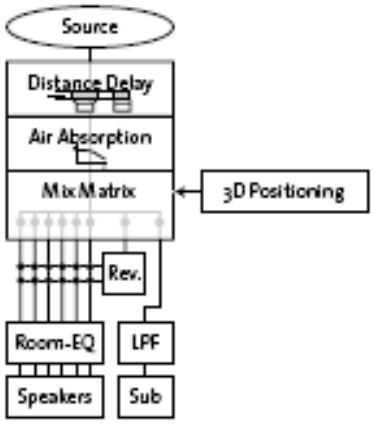


Figura 1: Cadena de procesamiento de Audio propuesto por Naef et al.

2.1.2. *Perceptual Audio Rendering of Complex Virtual Environments*

Tsingos *et al*. [7] proponen un sistema capaz de hacer rendering de escenarios con un gran número de fuentes de audio presentes. El desarrollo consiste en agrupar las fuentes en grupos (*clusters*) de acuerdo a la distancia entre ellas y un representante del cluster. De esta manera, el procesamiento restante se enfoca solamente sobre el representante (el cual contiene la mezcla de las fuentes relevantes al escenario, seleccionadas previamente mediante enmascaramiento). El procesamiento espacial se hace de manera previa a la creación del cluster, para mantener la coherencia de las pistas auditivas para el usuario, pero al final son los clusters los que pueden ser ubicados espacialmente. De esta manera se puede manejar un gran número de fuentes sonoras con un menor impacto en el desempeño del sistema.

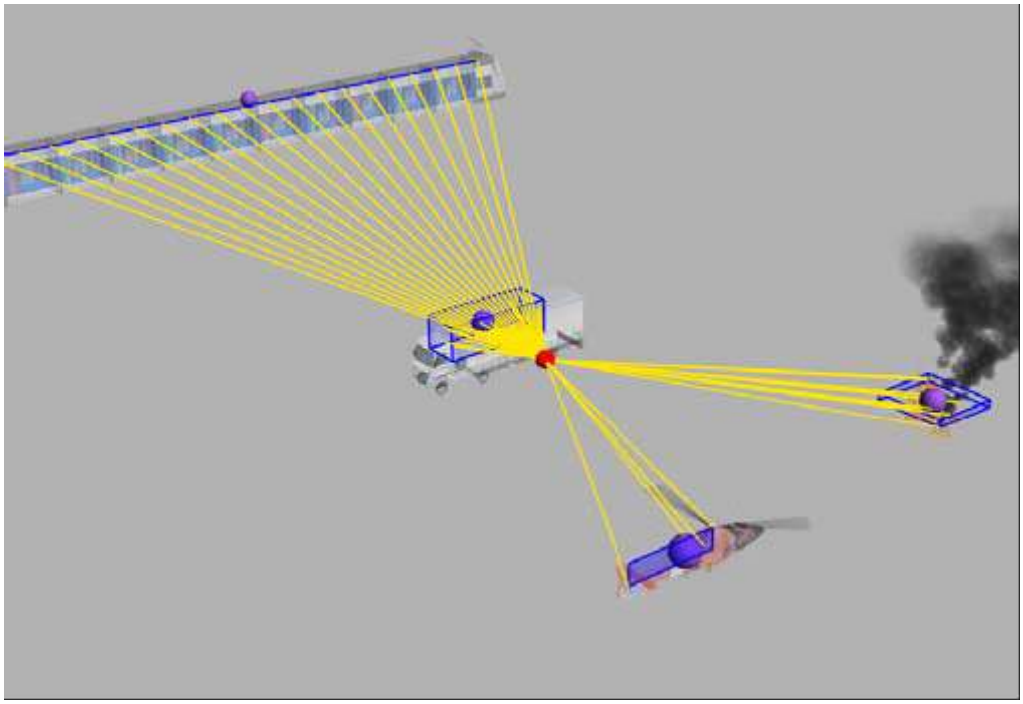


Figura 2: Ejemplo de la formación de clusters respecto a una escucha (en rojo).

2.1.3. *Echology*

Este proyecto, desarrollado por Deutscher et al. [3], presenta el desarrollo de una instalación en la cual los participantes pueden interactuar con ballenas Beluga en tiempo real mediante un sistema tipo *table-top*. Una de las premisas del sistema es ofrecer un ambiente relajado de interacción, usando audio 3D mediante la implementación de VBAP, y usando Max/Msp/Jitter, como herramienta de desarrollo. Este proyecto evidencia la importancia de tener audio espacializado como parte de un sistema interactivo, así como su implementación.

2.1.4. *Dirt-Cheap 3-D Spatial Audio*

El proyecto presentado por Klein *et al*. [8] describe una implementación simple de un sistema de audio 3D para una aplicación tipo CAVE. La aplicación está construida sobre software libre (OpenSource), más específicamente Linux y la librería Mustajuuri (para C++), usando alternativas poco costosas, pero de todas maneras permitiendo el uso de un sistema configurable que soporta hasta ocho parlantes.

2.1.5. *Otros proyectos relevantes*

Existen otros proyectos que vale la pena mencionar, por que dan una luz sobre el uso del audio 3D en un sistema interactivo, así como otras propuestas de implementaciones que pueden ser útiles como guía para el desarrollo de una aplicación como la que se busca:

- *A VR Interface for Collaborative 3D Audio Performance* [11]*:* Este proyecto tiene como objetivo apoyar la interpretación musical y separarla espacialmente del proceso de mezcla, permitiendo al usuario manipular las fuentes sonoras en el espacio, usando *blue-c*.

- *Spatial Sound Localization in an Augmented Reality Environment* [13]*:* El objetivo del sistema es permitir a un usuario localizar espacialmente fuentes sonoras en una escena, dándole pistas mediante la correcta ubicación de las mismas en un espacio 3D simulado usando HRTF (*Head-Related Transfer Function*).

- *Immersive Sound Field Simulation in Multi-screen Projection Displays* [9]*:* Desarrolla una tecnología para la simulación de campos sonoros usando filtros de convolución calculados con la ecuación de onda de Kirchhoff, el cual es usado en un Avatar Virtual en un entorno tipo CAVE (CABIN).

2.2. Algoritmos de Espacialización

Existen también diversos algoritmos de especialización del sonido que son usados de manera recurrente en las implementaciones de los sistemas mencionados anteriormente, tales como VBAP *(Vector Base Amplitude Panning)* y HRTF (*Head-Related Transfer Function*). Mientras que HRTF simular audio tridimensional en sistemas binaurales (stereo o audífonos), VBAP permite ubicar una fuente espacialmente en un sistema configurable de *n* cajas acústicas disponibles. La parte restante de esta sección es dedicada a una descripción de VBAP, a manera de familiarización con un algoritmo espacial.

2.2.1 *VBAP*

VBAP [4] es un algoritmo para crear sonido 3D desarrollado por Ville Pulkki en 1997 en *Helsinki University of Technology* (HUT). El algoritmo consiste en lo siguiente:

- Se define un vector (*p*) partiendo de la escucha hacia la posición virtual de la fuente sonora.

- Se define una base vectorial donde existen vectores unitarios que parten de la posición de la escucha y apuntan a la posición física de los parlantes del sistema. En el caso de una ubicación tridimensional de los mismos (donde cada uno de los vectores tiene un ángulo de elevación), tres vectores definen un triángulo dentro del cual se puede ubicar la fuente (l_1, l_2, l_3).

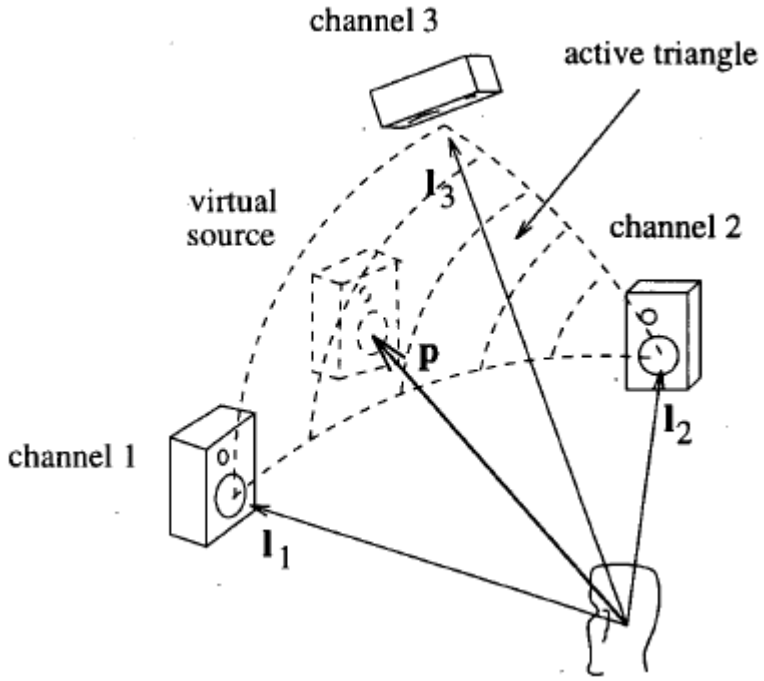


Figura 3: Triángulo dentro del cual se puede ubicar la fuente sonora.[4]

- Una vez se han definido estos vectores, se calculan las ganancias para cada uno de los parlantes en el triángulo, donde *p* es una combinación lineal de los vectores l_1, l_2 y l_3, multiplicados por un vector de ganancia (*g*).

$$\boldsymbol{p} = g_1\boldsymbol{l}_1 + g_2\boldsymbol{l}_2 + g_3\boldsymbol{l}_3$$

$$\boldsymbol{p}^{\mathrm{T}} = \boldsymbol{g}\boldsymbol{L}_{123}\,.$$

$$\boldsymbol{g} = \boldsymbol{p}^{\mathrm{T}}\boldsymbol{L}_{123}^{-1} = [p_1 \quad p_2 \quad p_3] \begin{bmatrix} l_{11} & l_{12} & l_{13} \\ l_{21} & l_{22} & l_{23} \\ l_{31} & l_{32} & l_{33} \end{bmatrix}^{-1}$$

Figura 4: Ecuaciones fundamentales de VBAP.[4]

El sistema se puede extender fácilmente para el uso de más de tres parlantes, siempre y cuando la definición de los triángulos sea correcta -es decir, no haya triángulos que se crucen, ni parlantes dentro de algún triángulo-. Una implementación del sistema es propuesta en [5].

Capítulo 3

Herramientas para Sonido Espacializado y Desarrollo

Una vez se estudiaron diversas implementaciones del sistema se pasó a la búsqueda de herramientas para el desarrollo de la aplicación. Se tuvieron en cuenta diversos aspectos, principalmente la capacidad de la herramienta de manipular los diversos canales de salida, así como la carga de archivos y la facilidad de integración con aplicaciones ya desarrolladas, ya que ese es uno de los objetivos principales. Este capítulo describe varias de las herramientas investigadas, así como sus principales características y las pruebas que se aplicaron a cada una de ellas. Finalmente describe otras herramientas a usarse para el desarrollo del proyecto

3.1 Pruebas a realizarse en los sistemas

A fin de determinar que herramienta es más apta para el desarrollo requerido, se decidió hacer un conjunto de pruebas que abarcaron desde las fases tempranas de instalación hasta la carga de archivos, pensando también en la implementación futura del sistema. De esta manera se desarrolló el plan siguiente:

- Realizar carga de archivos: De esto se desprende el usar las herramientas para carga de archivos integradas a cada uno de los sistemas, u otras librerías en aso que no haya tales integraciones.
- Crear múltiples fuentes a partir de los archivos cargados en el sistema.
- Reproducir las fuentes de acuerdo a su posición en el espacio.

3.2 Descripción de las Herramientas

3.2.1. *SuperCollider*

SuperCollider [17], desarrollado por James McCartney, es un lenguaje de programación orientado a objetos enfocado principalmente en la síntesis sonora. Disponible para Linux, Windows y Mac, la primera versión fue presentada en Internacional Computer Music Conference (ICMC), en 1996, la cual se llevo a cabo en Hong Kong, y actualmente es un proyecto Open Source, de manera que es

mantenido no solo por su creador sino por cualquiera dispuesto a colaborar.

Entre las características relevantes del sistema se encuentra el hecho que permite programación interactiva (es decir, se pueden escribir partes del programa mientras está en ejecución). Permite la ubicación de sonidos espacialmente y puede trabajar como un sistema distribuido, bajo una arquitectura cliente-servidor.

Algunos problemas que se encontraron analizando el sistema fueron, por ejemplo, la instalación, ya que aún no se han podido resolver algunos problemas que impiden la ejecución de las pruebas. Entre otras cosas, la carga de archivos tiene menor prioridad con respecto a la síntesis sonora y no es claro el proceso de integración con otros lenguajes, aunque plug-ins para el lenguaje pueden ser escritos usando C.

3.2.2. *Sonix*

Vr Juggler [18] es un proyecto iniciado en 1997 por la Dra. Carolina Cruz-Neira y un equipo de estudiantes en Iowa State University, a fin de desarrollar un marco (*framework*) de desarrollo para aplicaciones de realidad virtual de carácter Open Source, aunque desde el 2003 recibe apoyo comercial por parte de Infiscape Corporation.

Sonix es el sistema de audio de VR Juggler. Provee un conjunto de interfases que permiten la implementación de múltiples APIs de audio para la operación del sistema, lo cual permite tener un sistema portable y cambiar de API durante la ejecución. Dentro de las características relevantes del sistema se resaltan las siguientes:

- Permite la selección del número de altavoces disponibles para el sistema, así como del *sample rate* y el tamaño de palabra (*bitrate*) de los archivos a usar.

- Permite la carga de archivos.

- Asocia nombres a las fuentes para su identificación.

- Permite asociación de sonidos ambientes, de manera que se mueven junto con la escucha.

Sin embargo, a pesar de lo poderosa que es la idea detrás de Sonix, hay algunas limitaciones notables, entre ellas el hecho que depende de VRJuggler para su funcionamiento, lo cual limita su instalación, y

todo lo que puede hacer el sistema depende finalmente del API que realice la implementación.

3.2.3. *OpenAL*

OpenAL (acrónimo para Open Audio Library) actúa como una "interfaz software para hardware de audio" [16]. Empezó como un proyecto sobre un API de audio como complemento a OpenGL [24] hacia 1998. Desarrollado en primera instancia por Loki Entertainment Software en 1999, Loki habló con Creative Labs para agregar soporte de hardware y estandarizar el sistema. En 2000 salieron los dos primeros juegos que hacen uso de la librería, y después que Loki cerró en 2002, Creative realiza el desarrollo de OpenAL.

OpenAL funciona bajo el concepto de fuente (*source*), buffer de audio y escucha (*listener*). Las fuentes son asociadas a los buffers (los cuales contienen los datos de audio), y al reproducirse son ubicadas espacialmente de manera automática y se le aplican atenuaciones y procesos de acuerdo a la posición de la escucha.

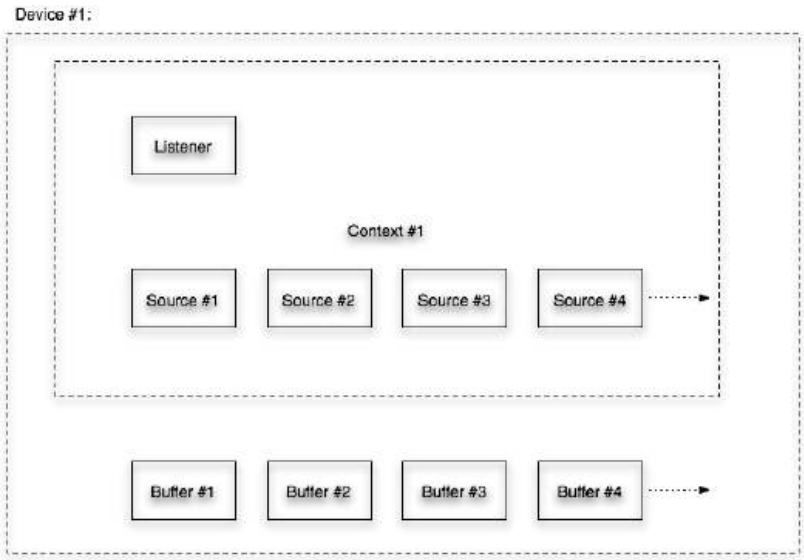


Figura 5: Relación entre objetos OpenAL.[4]

Entre las características relevantes de OpenAL, se encuentra el hecho de ser portable a varios sistemas operativos (Windows, MacOS. BeOS, Linux, entre otros), y gracias al desarrollo en Creative Labs es soportado por un gran número de tarjetas especializadas de audio. También posee una sintaxis similar a OpenGL, lo que facilita la integración con programas similares, así como una librería auxiliar (ALUT) que ayuda con la creación de fuentes a partir de archivos. Sin embargo, un defecto particular consiste en que No se puede

manipular la ubicación de las fuentes con respecto a los altavoces físicos.

3.2.4. *CSound*

CSound [15] es un lenguaje de programación orientado al manejo de sonido, especialmente la síntesis y diseño sonoros. Fue desarrollado por Barry Vercoe en Music and Cognition Group de M.I.T Media Laboratory hacia 1985. A partir de allí el proyecto ha visto un desarrollo continuado bajo diversos desarrolladores –siendo actualmente software libre-, desembocando en la versión 5.0, lanzada en 2005.

La premisa del funcionamiento de Csound se basa en la existencia de instancias divididas en una orquesta (la cual define los instrumentos virtuales a usarse) y una interpretación (*score*) la cual envía notas y otros parámetros a los instrumentos definidos en la orquesta, con la salida siendo generalmente un archivo creado por el sistema, o en tiempo real. Dentro de las características del lenguaje están opcodes dentro del lenguaje que implementan algoritmos de ubicación espacial, como HRTF y VBAP, además de la carga de algunos formatos de archivos de audio.

Para integrarse con otros lenguajes (particularmente C/C++) existe un API, el cual es usado para la manipulación de una instancia de CSound, a la cual se envían eventos en tiempo real a través del API. Sin embargo, algunas limitaciones presentes están en el uso de algunas herramientas, como es el envío de eventos en tiempo real, desarrollado usando *Cscore* como parte de API, pero que no ha podido ser usado actualmente.

3.2.5. *PortAudio*

PortAudio [14] nace de una propuesta y posterior desarrollo realizada por Ross Bencina a la lista *music-dsp*. Es también una herramienta de software libre y se encuentra actualmente en la versión v19, cuyo último desarrollo fue lanzado en Diciembre 7 de 2007.

PortAudio es un API diseñado sobre C, el cual permite el acceso por parte del programador a los dispositivos de audio disponibles en la máquina, así como a sus configuraciones válidas. Asimismo permite manipular tanto entrada como salida de audio, usando el concepto de de *streams* (flujos), asociados a una función de *callback* (implementada por el usuario), la cual se encarga de pasar los datos al sistema de audio correspondiente, así como recibirlos en la entrada. Al poder seleccionar el dispositivo de audio que se desea

usar, también se pueden seleccionar el número de salidas y manipular independientemente, así como el *sample rate* y el *bitrate* del sistema, que es algo deseado para el proyecto. Al ser un sistema básico, no está implementada la carga de archivos, por lo cual existe la necesidad de apoyarse en otro API y tampoco se puede abrir más de un stream por dispositivo, por que necesitaría una coordinación entre los flujos adecuada, ya que la fundón de callback es llamada a nivel de interrupción.

3.3 Herramientas Adicionales de Desarrollo

El sistema fue construido usando C++ como lenguaje de desarrollo a fin de ser integrado en el futuro con otros proyectos en desarrollo actualmente en el grupo IMAGINE. Para los cálculos matemáticos requeridos se optó por la librería GMTL (*Generic Math Template Library*) [20], a fin de tener más control y facilidad sobre algunos de los cálculos matemáticos del sistema.

Para el manejo de versiones y control de fuentes se usó un repositorio en la Universidad de Los Andes -perteneciente al grupo IMAGINE- usando CVS (Concurrent Versión System). El proyecto fue desarrollado usando Eclipse CDT (C/C++ Development Tooling) [21] como IDE para desarrollo, y se documentó usando Doxygen [22]. Finalmente, el sistema operativo utilizado fue Linux (usando Fedora Core 6).

Capítulo 4

Diseño de la herramienta

Una vez se evaluaron las distintas herramientas se optó por desarrollar un sistema lo suficientemente flexible para admitir implementaciones futuras. Con esto en cuenta se formularon los requerimientos que se presentan a continuación y se hizo un diseño principal presentado a continuación, basado siempre en los objetivos del sistema desarrollados anteriormente.

4.1. Requerimientos del Sistema

4.1.1. Requerimientos funcionales.

A continuación se listan los requerimientos funcionales encontrados para el sistema, divididos en cada una de las funcionalidades propuestas al sistema.

- Inicialización y Destrucción del Sistema
  - **R1:** Crear el sistema
  - **R2:** Activar audio del sistema usando parámetros espaciales.
  - **R3:** Salir del sistema.

- Fuentes:
  - **R4:** Crear una fuente.
  - **R5:** Asociar una fuente a un archivo disponible en el sistema.
  - **R6:** Reproducir una fuente.
  - **R7:** Mover una fuente.
  - **R8:** Pausar una fuente.
  - **R9:** Detener una fuente.
  - **R10:** Eliminar una fuente.
  - **R11:** Modificar una fuente.
  - **R12:** Disociar una fuente.

- Archivos
  - **R13:** Cargar un archivo de audio al sistema.
  - **R14:** Eliminar un archivo del sistema

- Escucha
  - **R15:** Orientar la escucha en el espacio.

4.1.2. Requerimientos no funcionales

- **R16:** Creación de múltiples fuentes sobre un mismo archivo.
- **R17:** La reproducción física de la fuente está ligada a su posición en el espacio.
- **R18:** La reproducción de la fuente cambia de manera relativa a la posición de la escucha.
- **R19:** Se debe proteger el acceso al sistema, de manera que éste exista solamente una vez.

4.1.3. Casos de Uso

El actor principal del sistema es el programador encargado de usar el sistema, de manera que esto se refleja en los mensajes enviados por el sistema. A continuación se listan las tablas con los casos de uso para cada uno de los requerimientos listados arriba.

Requerimientos Funcionales

R1: Crear el sistema

Identificador: R.1	**Indispensable/Deseable:** Indispensable	**Prioridad:** Alta
Nombre Caso de Uso: Crear el Sistema.		
Categoría (Visible/No visible): Visible		
Resumen:	Inicializar el administrador del sistema	
Curso Básico Eventos:	1. El programador solicita una nueva instancia del sistema 2. Si ya existe una instancia creada, el sistema devuelve la instancia existente	
Caminos Alternativos:	2.1 Si no existe una instancia, el sistema crea una nueva instancia vacía del sistema.	
Caminos de Excepción:		
Puntos de Extensión:		
Pre – Condiciones:		
Post- Condiciones:	Existe una nueva instancia vacía del sistema. No hay fuentes ni archivos cargados y el sistema de audio no está funcionando. Existe una escucha activa.	
Criterios de Aceptación		

R2: Activar audio del sistema usando parámetros espaciales.

Identificador: R.2	**Indispensable/Deseable:** Indispensable	**Prioridad:** Alta
Nombre Caso de Uso: Activar audio usando parámetros espaciales.		
Categoría (Visible/No visible): Visible		
Resumen:	Organiza el sistema de audio a ser usado de acuerdo a los parámetros entregados.	
Curso Básico Eventos:	1. El programador solicita una nueva instancia del sistema 2. El programador inserta los parámetros del sistema y solicita activación. 1. Número, posición y orientación de los parlantes. 2. Distribución del sonido (Mono, Stereo, Distribuido) 3. Sample Rate del sistema 3. Se organiza la salida de audio del sistema de acuerdo a los parámetros anteriores	
Caminos Alternativos:	2.1 Si el sistema de audio está funcionando al momento de usar este caso, éste es detenido y finalizado.	
Caminos de Excepción:	3.1 Si los parámetros del sistema son incorrectos el sistema no reproduce audio.	
Puntos de Extensión:		
Pre – Condiciones:		
Post- Condiciones:	El sistema de audio está en funcionamiento. Las fuentes creadas y en reproducción deben ser escuchadas.	
Criterios de Aceptación	Los parámetros entregados al sistema son correctos.	

R3: Salir del sistema.

Identificador: R.3	**Indispensable/Deseable:** Indispensable	**Prioridad:** Alta
Nombre Caso de Uso: Salir del sistema.		
Categoría (Visible/No visible): No Visible		
Resumen:	Finaliza el uso del sistema.	
Curso Básico Eventos:		
Caminos Alternativos:		
Caminos de Excepción:		

Puntos de Extensión:	
Pre – Condiciones:	El sistema está en funcionamiento.
Post- Condiciones:	El sistema es finalizado. Las fuentes existentes y los archivos son destruidos y el sistema de audio es detenido.
Criterios de Aceptación	

R4: Crear una fuente.

Identificador: R.4	**Indispensable/Deseable:** Indispensable	**Prioridad:** Alta
Nombre Caso de Uso: Crear una fuente.		
Categoría (Visible/No visible): Visible		
Resumen:	Crea una nueva fuente en el sistema	
Curso Básico Eventos:	1. El programador ingresa los parámetros para una nueva fuente ▪ Identificador de la fuente ▪ Identificador del archivo asociado. ▪ Tipo de fuente (ambiente, no ambiente) 2. Se ha creado una nueva fuente.	
Caminos Alternativos:		
Caminos de Excepción:	1.2. Ya existe una fuente con ese identificador. No se crea una nueva fuente.	
Puntos de Extensión:	1.1 El nombre del archivo es vacío. La fuente es creada sin una asociación a un archivo, de manera que no será reproducida.	
Pre – Condiciones:		
Post- Condiciones:	Una nueva fuente es creada, ubicada en el origen, con ganancia de 1 y detenida actualmente.	
Criterios de Aceptación	Los parámetros introducidos son correctos.	

R5: Asociar una fuente a un archivo disponible en el sistema.

Identificador: R.5	**Indispensable/Deseable:** Indispensable	**Prioridad:** Alta
Nombre Caso de Uso: Asociar una fuente a un archivo disponible en el sistema.		
Categoría (Visible/No visible): Visible		

Resumen:	Asocia una fuente a un archivo para que pueda ser reproducida.
Curso Básico Eventos:	1. El programador ingresa los parámetros para una la asociación ▪ identificador de la fuente ▪ identificador del archivo asociado. 2. Se ha asociado una nueva fuente a un archivo.
Caminos Alternativos:	
Caminos de Excepción:	1.2. No existe una fuente con ese identificador. No hay asociación. 1.3. No existe un archivo con ese identificador. No hay asociación.
Puntos de Extensión:	
Pre – Condiciones:	
Post- Condiciones:	La fuente tiene un archivo asociado y está lista para ser reproducida.
Criterios de Aceptación	Los parámetros introducidos son correctos.

R6: Reproducir una fuente.

Identificador: R.6	**Indispensable/Deseable:** Indispensable	**Prioridad:** Alta
Nombre Caso de Uso: Reproducir una fuente.		
Categoría (Visible/No visible): Visible		
Resumen:	Activa una fuente para reproducción.	
Curso Básico Eventos:	1. El programador ingresa los parámetros. ▪ identificador de la fuente 2. La fuente está en reproducción.	
Caminos Alternativos:		
Caminos de Excepción:	1.2. No existe una fuente con ese identificador. No hay reproducción.	
Puntos de Extensión:		
Pre – Condiciones:		
Post- Condiciones:	La fuente está en reproducción.	
Criterios de Aceptación	Los parámetros introducidos son correctos.	

R7: Mover una fuente.

Identificador: R.7	**Indispensable/Deseable:** Indispensable	**Prioridad:** Alta
Nombre Caso de Uso: Mover una fuente.		
Categoría (Visible/No visible): Visible		
Resumen:	Ubica la fuente en el espacio	
Curso Básico Eventos:	1. El programador ingresa los parámetros. ▪ Identificador de la fuente. ▪ Coordenadas del la fuente. 2. La fuente es ubicada espacialmente.	
Caminos Alternativos:		
Caminos de Excepción:	1.2. No existe una fuente con ese identificador. No hay movimiento.	
Puntos de Extensión:		
Pre – Condiciones:		
Post- Condiciones:	La fuente ha sido movida a una nueva posición especificada por los parámetros.	
Criterios de Aceptación	Los parámetros introducidos son correctos.	

R8: Pausar una fuente.

Identificador: R.8	**Indispensable/Deseable:** Indispensable	**Prioridad:** Alta
Nombre Caso de Uso: Pausar una fuente.		
Categoría (Visible/No visible): Visible		
Resumen:	Detiene la reproducción de una fuente, pero el audio queda en el mismo punto.	
Curso Básico Eventos:	1. El programador ingresa los parámetros. ▪ Identificador de la fuente. 2. La fuente es pausada.	
Caminos Alternativos:		
Caminos de Excepción:	1.2. No existe una fuente con ese identificador.	

Puntos de Extensión:	
Pre – Condiciones:	
Post- Condiciones:	La reproducción de la fuente es detenida. Si se vuelve a reproducir el audio empieza desde el punto antes de la pausa.
Criterios de Aceptación	Los parámetros introducidos son correctos.

R9: Detener una fuente.

Identificador: R.9	**Indispensable/Deseable:** Indispensable	**Prioridad:** Alta
Nombre Caso de Uso: Detener una fuente.		
Categoría (Visible/No visible): Visible		
Resumen:	Detiene la reproducción de una fuente.	
Curso Básico Eventos:	1. El programador ingresa los parámetros. ▪ Identificador de la fuente. 2. La fuente es detenida.	
Caminos Alternativos:		
Caminos de Excepción:	1.2. No existe una fuente con ese identificador.	
Puntos de Extensión:		
Pre – Condiciones:		
Post- Condiciones:	La reproducción de la fuente es detenida. Si se vuelve a reproducir el audio empieza desde el principio del archivo.	
Criterios de Aceptación	Los parámetros introducidos son correctos.	

R10: Eliminar una fuente.

Identificador: R.10	**Indispensable/Deseable:** Indispensable	**Prioridad:** Alta
Nombre Caso de Uso: Eliminar una fuente.		
Categoría (Visible/No visible): Visible		
Resumen:	Borra la fuente	
Curso Básico	1. El programador ingresa los parámetros.	

Eventos:	▪ Identificador de la fuente. 2. La fuente es detenida. 3. Si existe asociación a un archivo de audio, ésta es eliminada. 4. La fuente es eliminada.
Caminos Alternativos:	
Caminos de Excepción:	1.2. No existe una fuente con ese identificador.
Puntos de Extensión:	
Pre – Condiciones:	
Post- Condiciones:	La fuente es eliminada del sistema y su referencia al archivo de audio usado. El identificador queda libre para usarse con otra fuente.
Criterios de Aceptación	Los parámetros introducidos son correctos.

R11: Modificar una fuente.

Identificador: R.11	**Indispensable/Deseable: D**eseable	**Prioridad:** Alta
Nombre Caso de Uso: Modificar una fuente.		
Categoría (Visible/No visible): Visible		
Resumen:	Cambia cualidades de la fuente, como es el hecho de ser ambiente o no, ganancia, altura (pitch).	
Curso Básico Eventos:	1. El programador ingresa los parámetros. ▪ Identificador de la fuente. ▪ Parámetro a modificar 2. La fuente es modifcada.	
Caminos Alternativos:		
Caminos de Excepción:	1.2. No existe una fuente con ese identificador. 1.3. La ganancia tiene un valor incorrecto (0 < ganancia ≤ 1). 1.4. La altura tiene un valor incorrecto (0 ≤ ganancia ≤ 2).	
Puntos de Extensión:		
Pre – Condiciones:	En el caso de la ganancia, la fuente tiene que ser ambiente para registrar cambios.	
Post- Condiciones:	La fuente es alterada. Los cambios son audibles inmediatamente si la fuente está en reproducción	
Criterios de	Los parámetros introducidos son correctos.	

Aceptación	

R12: Disociar una fuente.

Identificador: R.12	**Indispensable/Deseable:** Indispensable	**Prioridad:** Alta
Nombre Caso de Uso: Disociar una fuente.		
Categoría (Visible/No visible): Visible		
Resumen:	La fuente es separada del archivo de audio al que se encuentra asociada.	
Curso Básico Eventos:	1. El programador ingresa los parámetros. ▪ Identificador de la fuente. 2. La fuente es disociada.	
Caminos Alternativos:		
Caminos de Excepción:	1.2. No existe una fuente con ese identificador. 1.3. La fuente no tenía un archivo asociado	
Puntos de Extensión:		
Pre – Condiciones:		
Post- Condiciones:	La fuente es separada del archivo de audio al que estaba asociada y es detenida. Si se intenta reproducir nuevamente no hay sonidos de salida.	
Criterios de Aceptación	Los parámetros introducidos son correctos.	

R13: Cargar un archivo de audio al sistema.

Identificador: R.13	**Indispensable/Deseable:** Indispensable	**Prioridad:** Alta
Nombre Caso de Uso: Cargar un archivo de audio al sistema.		
Categoría (Visible/No visible): Visible		
Resumen:	Se ingresa un nuevo archivo del cual se leerán datos de audio que podrán ser asociados a una o varias fuentes.	
Curso Básico Eventos:	1. El programador ingresa los parámetros. ▪ Identificador del archivo. ▪ Ruta del archivo 2. Se crea un nuevo archivo.	
Caminos		

Alternativos:	
Caminos de Excepción:	1.2. Ya existe un archivo con ese identificador. 1.3. La ruta al archivo es inválida. 1.4. El archivo no es de un formato soportado por el sistema. 1.5. El archivo no se puede reproducir.
Puntos de Extensión:	
Pre – Condiciones:	
Post- Condiciones:	Se crea un nuevo archivo de audio al que se pueden asociar fuentes y del que se pueden leer datos. No hay ninguna fuente asociada
Criterios de Aceptación	Los parámetros introducidos son correctos.

R14: Eliminar un archivo del sistema

Identificador: R.14	**Indispensable/Deseable:** Indispensable	**Prioridad:** Alta
Nombre Caso de Uso: Eliminar un archivo del sistema		
Categoría (Visible/No visible): Visible		
Resumen:	Se elimina un archivo de audio del sistema.	
Curso Básico Eventos:	1. El programador ingresa los parámetros. ▪ Identificador del archivo. 2. Todas las fuentes asociadas al archivo son detenidas y disociadas. 3. El archivo es eliminado.	
Caminos Alternativos:		
Caminos de Excepción:	1.2. No existe un archivo con ese identificador.	
Puntos de Extensión:		
Pre – Condiciones:		
Post- Condiciones:	Se elimina el archivo. Todas las fuentes antes asociadas a éste son detenidas y disociadas. El identificador queda libre para usarse con otro archivo.	
Criterios de Aceptación	Los parámetros introducidos son correctos.	

R15: Orientar la escucha en el espacio

Identificador: R.15	**Indispensable/Deseable:** Indispensable	**Prioridad:** Media
Nombre Caso de Uso: Orientar la escucha en el espacio		
Categoría (Visible/No visible): Visible		
Resumen:	La escucha es ubicada y/o orientada en el espacio.	
Curso Básico Eventos:	1. El programador ingresa los parámetros. ▪ Posición de la escucha. ▪ Orientación de la escucha 2. La escucha es ubicada en el espacio de acuerdo a los parámetros.	
Caminos Alternativos:		
Caminos de Excepción:		
Puntos de Extensión:		
Pre – Condiciones:		
Post- Condiciones:	La escucha es ubicada en el espacio. A partir de esta nueva ubicación son calculadas las ganancias de las fuentes.	
Criterios de Aceptación		

Requerimientos no funcionales

R16: Creación de múltiples fuentes sobre un mismo archivo.

Nombre Creación de múltiples fuentes sobre un mismo archivo	
Tipo: Necesario	**Crítico:** Si
Descripción Los datos de un archivo pueden ser compartidos para que múltiples fuentes puedan leer de ellos.	
Criterios de Aceptación Pueden existir varias fuentes relacionadas con un mismo archivo.	

R17: La reproducción física espacial de la fuente.

Nombre La reproducción física espacial de la fuente	
Tipo: Necesario	**Crítico:** Si
Descripción Una vez configurado el sistema de audio y la ubicación de la fuente, el(los) canal(es) de reproducción es (son) seleccionado(s).	

Criterios de Aceptación Las fuentes son reproducidas en el sistema seleccionado de acuerdo a su posición en el espacio virtual.

R18: La reproducción de la fuente cambia de manera relativa a la posición de la escucha.

Nombre La reproducción de la fuente cambia de manera relativa a la posición de la escucha	
Tipo: Necesario	**Crítico:** Si
Descripción Se realizan atenuaciones necesarias de acuerdo a la posición de la fuente relativa a la escucha, de manera que se logra una correcta ubicación sonora.	
Criterios de Aceptación Las fuentes son reproducidas en el sistema seleccionado de acuerdo a su posición en el espacio y relativas a la escucha.	

R19: Se debe proteger el acceso al sistema, de manera que éste exista solamente una vez.

Nombre Se debe proteger el acceso al sistema, de manera que éste exista solamente una vez	
Tipo: Necesario	**Crítico:** Si
Descripción De manera que múltiples instancias del sistema podrían interferir entre sí y con la implementación de audio subyacente, así como con el manejo de recursos, se requiere la existencia de sólo una existencia activa en el sistema.	
Criterios de Aceptación Sólo existe una instancia del sistema en uso.	

4.2 Diseño del sistema

Una vez definidos estos parámetros, se tenían las dos alternativas siguientes:

- Desarrollo de un sistema nuevo: Se pensó en la creación de un nuevo API que permita la implementación de otros APIs subyacentes. En la figura 4 se encuentra una propuesta de arquitectura para el API, usando los patrones *Abstract Factory* y *Singleton*[6] para permitir múltiples implementaciones.

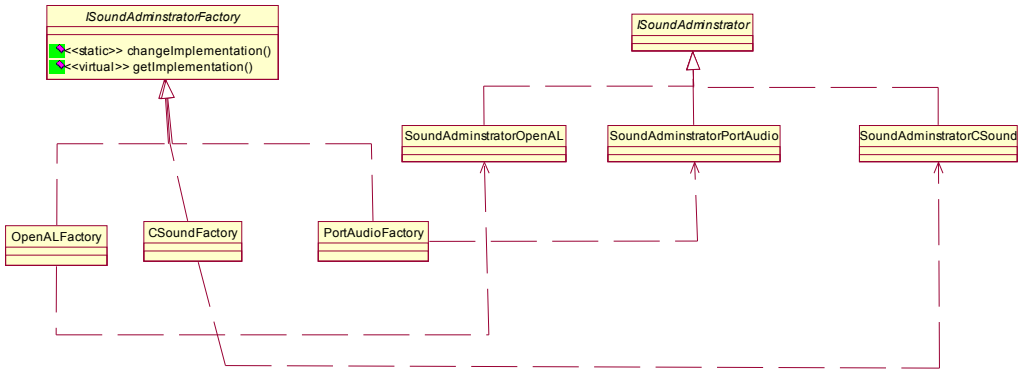


Figura 6: Diseño de la arquitectura del sistema SoundAdministrator

- Sonix como sistema base: Sonix ofrece un conjunto de interfases que pueden ser extendidas para admitir nuevas implementaciones bajo APIs distintos. Éste fue considerado debido a su similitud con la propuesta anterior.

De estas dos posibilidades so optó por la primera, ya que no se encontraba ligada a VR Juggler o algún otro *framework* de desarrollo y de esta manera se garantiza su uso en otras alternativas de desarrollo. Sin embargo, el sistema también podría usarse con Sonix actuando como una capa intermedia entre éste y la implementación final del sistema. En los anexos se puede encontrar un diseño detallado de un sistema desarrollado completamente.

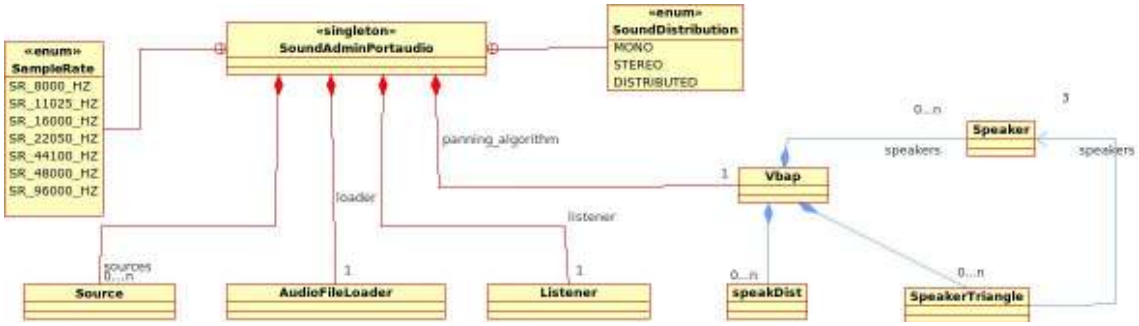


Figura 7: Diseño de una implementación subyacente usando PortAudio

Capítulo 5

Implementación del Sistema

Una vez finalizado el diseño del sistema se pasó a la fase de implementación del mismo, caracterizada principalmente por la intensificación de pruebas sobre las herramientas y la selección de las herramientas finales para trabajar. Este trabajo explica la dinámica del desarrollo del sistema usando las herramientas seleccionadas.

5.1. Herramientas seleccionadas.

Una vez hechas las pruebas de las herramientas se redujo la selección a sólo dos alternativas, OpenAL y PortAudio. OpenAL fue seleccionado por su facilidad de uso, de manera que se presta adecuadamente a una implementación previa del sistema con todas las características requeridas del mismo, aunque su representación espacial no fuese la más exacta.

Por otro lado, PortAudio (en colaboración con *libsndfile* [23] para el manejo de archivos) se seleccionó por su capacidad a bajo nivel de seleccionar diversos dispositivos y los canales de reproducción, de manera que se pudiese tener mayor control sobre la especialización del sonido que en OpenAL.

5.2. Características del sistema desarrollado.

Como parte de la definición del sistema, cada uno fue definido con las siguientes características:

- *Fuentes*:
    - Cada fuente tiene asociada un *identificador* único (una cadena de caracteres), la cual es asignada al momento de su creación.
    - Asimismo, cada fuente tiene asignado un archivo (mediante un identificador como cadenas de texto únicas).
    - Las fuentes (así como la escucha) se ubican en el espacio de acuerdo al sistema de coordenadas usado en OpenGL.

- Las fuentes pueden ser reproducidas. pausadas, detenidas. También pueden ser eliminadas y/o separadas (disociadas) de un archivo.
- La percepción de la fuente cambia de acuerdo a su posición relativa a la escucha.
- Existen dos tipos de fuentes. Las fuentes *ambiente* se mueven junto a la escucha y se reproducen por todas las salidas del sistema. Las fuentes *localizadas* no están ligadas a la escucha y por tanto son distribuidas espacialmente en las salidas disponibles, y atenuadas de acuerdo a su distancia a la escucha.
  - Los archivos pueden reproducirse cíclicamente (*loop*).

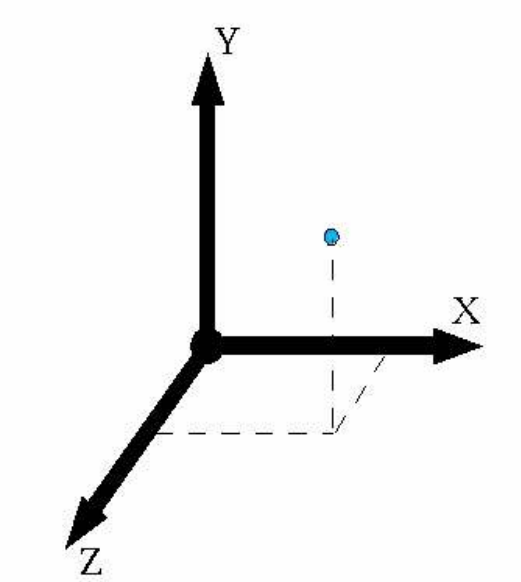


Figura 8: Sistema de coordenadas OpenGL (imagen tomada de http://resumbrae.com/ub/dms423_f06/09/coordSystem.jpg)

- *Escucha:*
  - La escucha puede moverse y orientarse en el espacio. De esta manera afecta la manera en que son escuchadas las fuentes activas en ese momento.

- *Archivos:*
  - Cada archivo se encuentra asociado a un *identificador* único (una cadena de caracteres), de manera que no puede haber dos archivos con el mismo identificador.
  - Dos fuentes pueden compartir el mismo archivo, moviéndose aparte en el tiempo.

5.2.1. Desarrollo con OpenAL

La primera versión del sistema fue desarrollada en OpenAL. Usando esta librería se implementó una primera versión de la interfaz a utilizar y se definieron varios de los métodos que podrían utilizarse en implementaciones distintas.

Dentro de las características propias del sistema está el uso de la parte de carga de archivos para crear *buffers* OpenAL usando archivos diversos (incluyendo Ogg Vorbis [25]) que pueden ser asociados a las fuentes existentes vía una estructura intermedia que independiza las fuentes de los buffers. Asimismo, dado que OpenAL tiene incluido el concepto de la escucha y controla la manera en que las fuentes es especializan no fue necesario hacer implementación de algoritmos o alternativas que ofrecieran estos servicios.

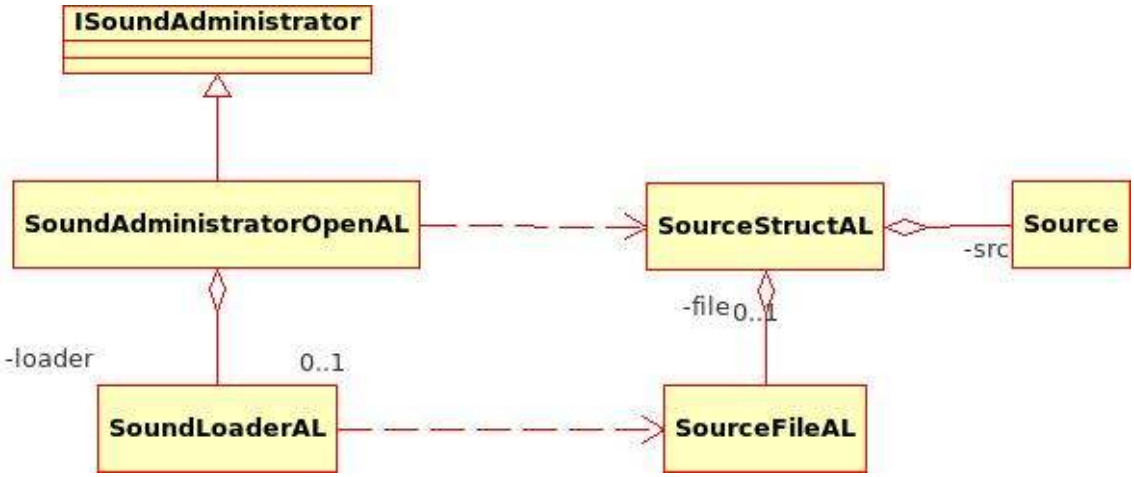


Figura 9: Diagrama de la implementación realizada usando OpenAL

5.2.1. Desarrollo con PortAudio

El desarrollo con PortAudio tuvo una aproximación diferente. Se realizaron un conjunto de pruebas pequeñas orientadas a probar más a fondo el sistema, organizadas de esta manera:

- *Carga de archivos:* Se organizaron pruebas destinadas a la reproducción de archivos cargados desde libsndfile y libvorbis, donde se tuvieron en cuenta problemas relacionados con la relación entre el tamaño de palabra y el sample rate de los archivos contra los usados por un dispositivo particular seleccionado por PortAudio.

- *Selección de dispositivos:* De acuerdo a un sample rate particular y un número de salidas se despliegan un conjunto de dispositivos de los cuales el usuario puede escoger uno para

usar como salida, a través del cual se reproducían archivos u ondas sinusoidales.

- *Mezcla:* Sobre el proyecto anterior se creó el concepto de una fuente -cada una asociada a una archivo independiente, o a una onda diferente-, y se crearon dos fuentes simultáneas que eran mezcladas en la salida. Se tuvo en cuenta el concepto de *clipping,* ya que la suma de las señales podía saturar la salida de audio.

- *Tiempo Real:* Se creó un sistema en el que se pudieran cambiar ciertos parámetros de una fuente, como ganancia y altura (dentro de una tabla sinusoidal) cuando el usuario presionara una tecla, o un cambio en la ubicación espacial usando los canales stereo del sistema.

- *Canales de salida*: Se probaron las salidas de un sistema enviando una onda sinusoidal por cada canal seleccionado del sistema.

- *VBAP*: Como PortAudio es un API para el procesamiento de audio a bajo nivel, se decidió realizar una implementación de VBAP, la cual estuvo basada en un desarrollo previo [5].

Una vez realizadas todas estas pruebas se pasó a la implementación del sistema. Éste se realizó independiente del desarrollo realizado con OpenAL por que integra diversos elementos que no estaban en ese sistema, incluyendo la función de callback (necesaria para PortAudio) y la implementación del patrón *Singleton*.

```
enum SampleRate {
        SR_8000_HZ,
        SR_11025_HZ,
        SR_16000_HZ,
        SR_22050_HZ,
        SR_44100_HZ,
        SR_48000_HZ,
        SR_96000_HZ
};

enum SoundDistribution {MONO, STEREO, DISTRIBUTED};
```

Figura 10: Distribuciones y sample rate posibles del sistema.

Asimismo se agregó soporte para sistemas stereo y monofónicos con *n* parlantes. También se agregó soporte para diversas sample rate de salida (desde 8 kHz hasta 96 kHz). El dispositivo seleccionado puede ser cualquiera que soporte las características solicitadas.

En el caso de tener múltiples fuentes, se aplica un factor de escalamiento para evitar *clipping* pronunciado al momento de sumar las señales. Este factor reduce la ganancia del sistema de la siguiente manera:

$$scale = \begin{cases} \frac{1}{10} \ si\ No.Fuentes\ en\ reproducción < 10 \\ \frac{1}{2 * No.Fuentes\ en\ reproducción} \ si\ No.Fuentes\ en\ reproducción \geq 10 \end{cases}$$

Figura 11: Factor de escalamiento de la mezcla.

Se realizaron cálculos de atenuación por distancia para cada una de las fuentes con respecto a su posición relativa a la escucha de acuerdo a la *ley del inverso al cuadrado.*

$$atten = \frac{1}{d^2}$$

$$d = \| listener - src \ \|$$

$$d^2 < 1.0 \rightarrow la\ fuente\ se\ reproduce\ como\ una\ señal\ ambiente$$

Figura 12: Aplicación de la ley del inverso al cuadrado en el sistema desarrollado

Para la ubicación espacial en un sistema stereo lo único que se hace es enviar la fuente de acuerdo al valor de su coordenada x (siendo positivo el canal derecho, y negativo el izquierdo).

Capítulo 6

Uso del Sistema

Este capítulo realiza una descripción del uso del sistema, desde la instalación del mismo, pasando por las funciones principales y una enumeración de las pruebas realizadas al mismo, esto centrado en la implementación realizada con PortAudio, por ser la que tiene funcionalidad más completa.

6.1. Instalación/Uso del sistema

El sistema fue desarrollado como un proyecto Eclipse, el cual tiene que ser enlazado con las librerías necesarias para el sistema. Al no tenerse una versión del sistema que permita usarlo fuera de Eclipse como un sistema independiente, las pruebas se hacen importando pequeños proyectos dentro del mismo, y acomodándolo usando instrucciones de preprocesamiento en C++:

```
/**
 * Variable de pruebas del sistema
 * */
#ifndef PRUEBA
#define PRUEBA (0)
#endif
```

```
/**
 * @brief  Archivo de prueba que permite verificar lo siguiente: <dl>
 * <dt> Asociar / Disociar una fuente a un archivo
 * <dt> Reproducir una fuente en loop / Reproducirla una vez
 * <dt> Destruir una archivo / Disociar las fuentes asociadas
 * </dl>
 * @author John Jairo Palma Robayo
 * @example TestLooped.cpp
 * */
#include "pruebas.h"
#if PRUEBA == 0
#include "../soundadmin/SoundAdminPortaudio.h"
#include <string>
#include <sstream>

using namespace std;

/**
 * Administrador de prueba del sistema
 * */
SoundAdminPortaudio* testadmin;
```

Figura 13: Instrucciones de prueba y ejemplo.

6.2. Inicialización del sistema

Para hacer uso del sistema tiene que usarse una instancia del mismo. De acuerdo a lo expresado en el diseño, sólo puede existir una instancia en ejecución, la cual tiene que ser obtenida mediante el uso del patrón *Singleton*:

```
/**
 * Obtención de la instancia
 * */
testadmin = SoundAdminPortaudio::getSoundAdminInstance();
```

Figura 14: Uso de método estático para la obtención de la instancia.

6.2.1. Configuración del sistema de Audio

Una vez obtenida la instancia se puede configurar el sistema usando el método *setConfiguration*, el cual recibe todos los parámetros necesarios para usar el sistema de audio, como son el número de parlantes a usar, el sample rate del sistema, la distribución espacial a usar y un archivo de configuración. Este archivo contiene los ángulos de cada caja (*azimuth, elevación*), con lo cual es configurado VBAP (en el caso de utilizar un sistema distribuido):

```
-30 0
30 0
-90 0
90 0
180 0
180 40
-30 40
30 40
```

```
/**
 * Método de configuración de audio
 * */
testadmin->setConfiguration(SoundAdminPortaudio::SR_44100_HZ, channels, argv[3],
        SoundAdminPortaudio::DISTRIBUTED);
```

Figura 15: (*arriba*) Archivo de carga del sistema de audio para 8 cajas. El primer número corresponde al ángulo de orientación (azimuth) respecto al eje x, y el segundo la elevación (respecto al eje y).
(*abajo*) Ejemplo de uso del método de configuración.

Es importante resaltar que los parlantes no pueden ser todos coplanares, ya que dicha configuración resultaría en la no salida de sonido dada la implementación de VBAP usada, la cual se espera sea usada en ambientes donde se espera los parlantes rodeen desde diferentes ángulos a la persona que interactúa con el sistema.

6.3. Creación y manipulación de objetos en el sistema

Desde que el sistema es solicitado se pueden crear, actualizar y eliminar archivos y fuentes, así como manipular a la escucha. No es necesario que se haya configurado el sistema de audio para hacer todo esto, pero esto resultará en no poder escuchar nada del sistema.

```
testadmin->createSource("testsrc", "testFile", false);

testadmin->playSource("testsrc",false);

testadmin->updateListener(pos, up);

char c;
while(c != 'o'){
    cin >> c;
    switch(c){
    case 'a':
        testadmin->attachSource("testsrc", "testFile");
        cout << "Fuente asociada" << endl;
        break;
    case 'd':
        testadmin->detachSource("testsrc");
        cout << "Fuente Disociada" << endl;
        break;
    case 'w':
        if(testadmin->deleteFile("testFile"))
            cout << "archivo borrado" << endl;
        else
            cout << "archivo no se pudo borrar" << endl;
        break;
    case 'p':
        if(testadmin->isPlaying("testsrc"))
            testadmin->pauseSource("testsrc");
```

```
case 'f':
    if(testadmin->addFile("testFile", file))
        cout << "archivo creado" << endl;
    else
        cout << "archivo no se pudo crear" << endl;
    break;
case 's':
    if(testadmin->destroySource("testsrc"))
        cout << "Fuente destruida" << endl;
    else
        cout << "Fuente no se pudo destruir" << endl;
    break;
case 'r':
    testadmin->createSource("testsrc", "testFile", false); break;
case 'y':
    if(testadmin->isPlaying("testsrc"))
        testadmin->stopSource("testsrc");
    else
        if(testadmin->isLooped("testsrc"))
            testadmin->playSource("testsrc");
        else
            testadmin->playSource("testsrc", false);
    break;
case 'l':
    if(!testadmin->isLooped("testsrc"))
        testadmin->loopSource("testsrc", true);
    else
        testadmin->loopSource("testsrc", false);
```

Figura 16: Ejemplo de manipulación de fuentes, archivos y escucha.

6.4. Instalación física posible del sistema

El sistema está planeado para implementarse en un ambiente tipo CAVE, como el ilustrado en la Figura 1. Cada una de las pantallas es controlada por un equipo diferente. Se busca también que los diversos dispositivos de interacción disponibles en el laboratorio se integren a las pruebas realizadas hasta el momento, de manera que se pueda tener una distribución como la prevista en las figuras 15 y 16.

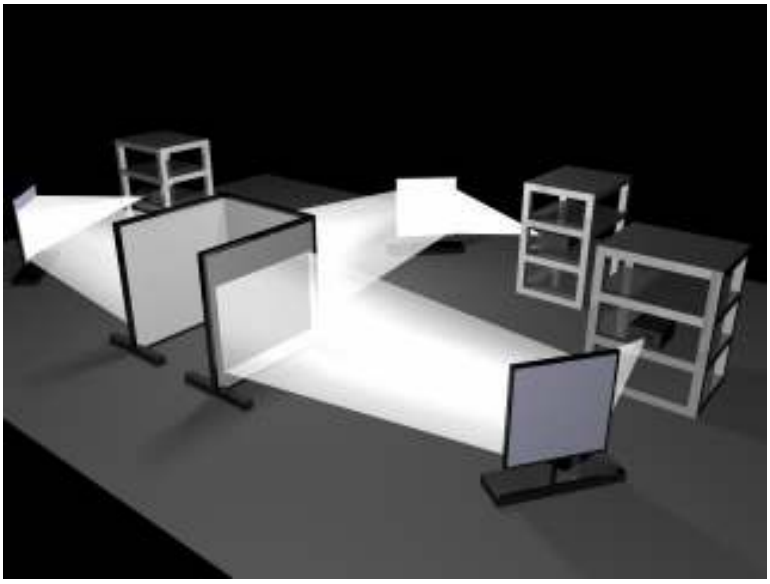


Figura 17. CAVE objetivo del sistema

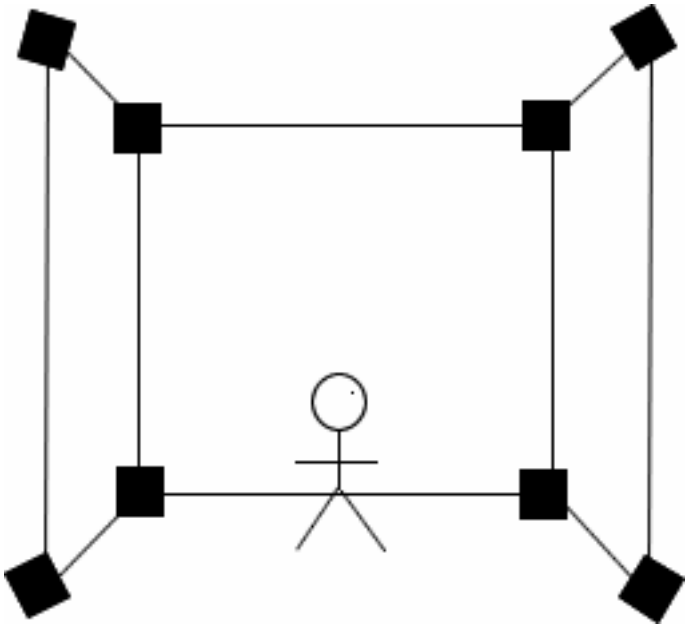


Figura 18. CAVE: Vista para una configuración de 8 parlantes

6.5. Pruebas Realizadas

Se realizaron un conjunto de pruebas destinadas a revisar las distintas condiciones del sistema.

La primera prueba (desarrollada bajo en el archivo MultipleSources.cpp), permite crear un sistema configurable en el que se crean y reproducen tantas fuentes como el usuario necesite o hasta que el sistema llegue al 80% de uso de CPU. Con esta prueba se busca determinar el límite del sistema respecto al número de

fuentes a usar y la latencia introducida al momento de leer del archivo por muchas fuentes, llegando a un pico entre 300 y 350 fuentes, donde la latencia introducida es grande y el factor de escalamiento silencia la mezcla.

Se realizó una segunda prueba con el fin de probar el desempeño del algoritmo de especialización (movement.cpp). El usuario define el tamaño de uno de los lados de un prisma rectangular (si definimos *l* como la longitud de los lados más cortos, su volumen sería $2l^3$). Luego, la fuente es movida a lo largo del prisma rectangular ubicado en el centro del plano cartesiano. Con esto también se probó que la disposición espacial de los parlantes incide en la aparición de puntos donde no hay sonido, ya que no habría ningún triángulo dentro del cual ubicar la fuente.

La última prueba (TestTracker.cpp) crea dos fuentes que pueden ser usadas también como sonido ambiente, pero que son controladas en el espacio por *trackers*, permitiendo una interacción más fluida con el sistema, usando un sistema similar al presentado abajo.

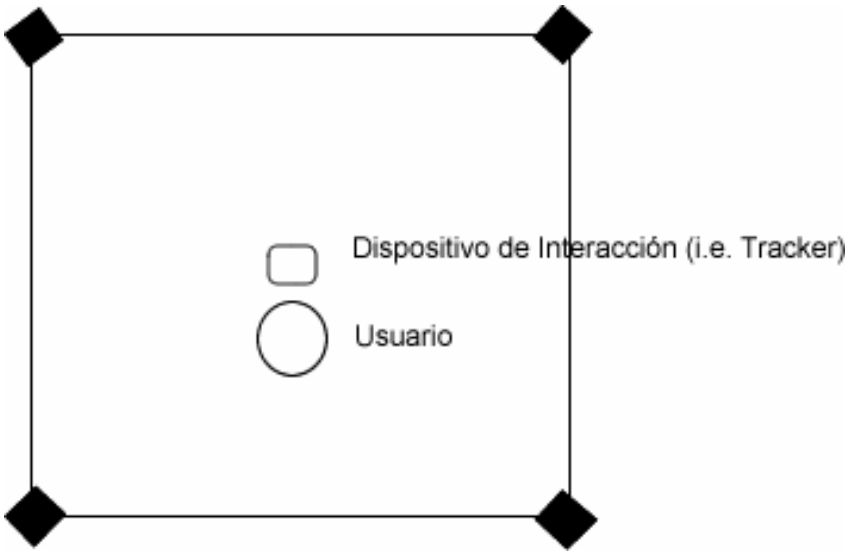


Figura 19. CAVE: Vista superior con elemento de interacción presente

6.6. Trabajo por desarrollar

Aunque el sistema puede ser utilizado sin complicaciones cumpliendo los objetivos propuestos existen algunos elementos que pueden mejorar la funcionalidad del sistema y podrían ser implementados en futuras iteraciones del mismo.

En primer lugar está el hecho de integrar los proyectos desarrollados en PortAudio y OpenAL para crear una interfaz común de manera que

se puedan usar las implementaciones de manera independiente y ofrecer un sistema más robusto y funcional.

En segundo lugar está la carga de archivos Ogg (en el caso de la versión PortAudio) y de archivos de otras extensiones (aiff, wav) usando *libsndfile* para crear buffers de OpenAL y así poder ampliar la paleta sónica del sistema.

En tercer lugar está la aplicación de espacialización en stereo, la cual es demasiado simple y se puede mejorar usando implementaciones de sistemas como HRTF o emulando un potenciómetro como los usados en consolas de audio para distribuir la señal entre los dos canales.

6.7. Problemas en la versión actual

Existen varios problemas que deben ser solucionados para próximas versiones del sistema, de manera que no afecten el desempeño del sistema.

El principal problema está en la relación del sample rate de los archivos y la usada por el sistema PortAudio. Cuando ambos datos no coinciden pueden ocurrir efectos indeseados en la reproducción del archivo. Para remediar esto, se tienen que aplicar procesos de *resampleo,* donde el sample rate de los datos del archivo son convertidos al sample rate del dispositivo reproductor, ya sea aumentándolo (*Upsampling*) o disminuyéndolo (*Downsampling*). Al no tener una alternativa para afrontar este problema, tampoco se puede hacer uso de la alteración de altura (*pitch shift*), ya que implica también un proceso de resampling.

En el caso de la versión de OpenAL se encuentra la limitación de no poder ubicar el sonido espacialmente como se desea, causando errores de ubicación de las fuentes, así como una limitación en el número de canales que se pueden usar.

6.8. Comparación con otros sistemas

Una vez explicada la funcionalidad del sistema éste puede ser comparado con otros proyectos que han abarcado el mismo problema, así como con herramientas que se puedan utilizar para el mismo fin.

El sistema se puede definir en primera instancia como una simplificación del sistema propuesto por Naef y Gross [12], ya que la implementación PortAudio tiene en cuenta la ubicación de cada uno

de las cajas acústicas en el sistema mediante VBAP [4], así como cálculos acústicos –en el caso de atenuación por distancia-, aunque de menor complejidad. El sistema también comparte similitud con el trabajo desarrollado por Klein *et al*. [8], dado el énfasis en un sistema fácil de usar y de implementar físicamente. El sistema puede servir como posible herramienta para implementar instalaciones artísticas como Echology [3] o propuestas similares a la desarrollada por Naef y Collicott [11].

El sistema comparte las mayores similitudes con Sonix [18], no sólo en el uso de identificadores para diferenciar fuentes y archivos, sino en la idea de definir un sistema que permita múltiples implementaciones, y, a diferencia de Csound o SuperCollider está orientado principalmente a la reproducción y ubicación de las fuentes en el espacio mientras que éstos están orientados a la síntesis sonora y al apoyo de procesos de composición musical computarizados.

Capítulo 7

Conclusiones

A lo largo de este documento se ha presentado el desarrollo de un sistema inmersivo de audio que se puede usar en diversas aplicaciones de realidad virtual, ya sean éstas instalaciones artísticas, videojuegos o simulaciones. El proyecto ha superado varias dificultades, como es el adentrarse en una disciplina poco abordada en el ámbito educativo circundante como es el manejo de audio y, todavía más, el hecho mismo de usar software para su ubicación espacial adecuada. Los aportes principales del proyecto se pueden listar de la siguiente manera:

- El hecho de formular un sistema que, acompañado del código fuente [26], provee una arquitectura básica pero sólida sobre la cual se pueden basar nuevos desarrollos en el área de audio 3D, e implementar nuevos proyectos que usen esa característica.

- La implementación de un algoritmo de espacialización de fuentes permite arrojar una luz sobre los procesos de ubicación de fuentes sonoras en el espacio.

- Se ha creado una guía comprensiva de aspectos relacionados no sólo con audio tridimensional, sino también con otras ramas dentro del área de audio computarizado, la cual se espera impulse la investigación y el desarrollo conjunto con áreas del conocimiento distintas a la ingeniería, como el arte y la música.

7.1 Trabajo futuro

Hay diversas direcciones en las cuales se puede orientar el trabajo futuro con base en lo desarrollado, de manera que se pueda crear un producto más robusto y/o con otras funcionalidades. Aquí se enumeran otras alternativas a evaluar para el proyecto:

- **Uso de otras herramientas para hacer implementaciones**: El hecho de no haber podido usar otras herramientas para implementar, como es el caso de SuperCollider y Csound, no las descarta para usos futuros sino que, dado el carácter flexible del proyecto, pueden servir para agregar nuevas funcionalidades.

- **Síntesis sonora:** Agregar asimismo algoritmos de síntesis que permitan a los usuarios crear los sonidos asociados a las fuentes sonoras, proveyendo así un poco más de control creativo al usuario.

- **Características acústicas más realistas:** Integrar cálculos acústicos más complejos, como atenuaciones por aire y suelo, así como el uso de reverberación para simular recintos virtuales, a fin de ofrecer una experiencia más realista al usuario.

Capítulo 8

Bibliografía

[1] Carolina Cruz-Neira, Daniel J. Sandin, Thomas A. DeFanti. *Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE.* SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, 1993.

[2] Anthony Savidis, Constantine Stephanidis, Andreas Korte, Kai Crispien, Klaus Fellbaum. *A Generic Direct-Manipulation 3D-Auditory Environment for Hierarchical Navigation in Non-visual Interaction*. ACM SIGACCESS Conference on Assistive Technologies, 1996.

[3] Meghan Deutscher, Reynald Hoskinson, Sachiyo Takashashi, Sidney Fels. *Echology: An Interactive Spatial Sound and Video Artwork.* International Multimedia Conference, 2005.

[4] Ville Pulkki. *Spatial Sound Generation and Perception by Amplitude Panning Techniques.* Helsinki Institute of Technology, Laboratory of Acoustics and Audio Signal Processing, Espoo 2001

[5] Ville Pulkki, Tapio Lokki. *Creating Auditory Displays with Multiple Loudspeakers Using VBAP: A Case Study with DIVA Project.* International Conference on Auditory Display (ICAD), 1998.

[6] Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[7] Nicolas Tsingos, Emmanuel Gallo, George Drettakis. *Perceptual Audio Rendering of Complex Virtual Environments.* ACM SIGGRAPH 2004 Papers.

[8] Eric Klein, Greg S. Schmidt, Erik B. Tomlin, Dennis G. Brown. *Dirt-Cheap 3-D Spatial Audio.* Linux Journal, Octubre 2005.

[9] T.Ogi, T. Kayahara, M. Kato, H. Asayama, M. Hirose. *Immersive Sound Field Simulation in Multi-screen Projection Displays.*

[10] Erik Kabisch, Falko Kuester, Simon Penny. *Sonic Panoramas: Experiments with Interactive Landscape Image Sonification.* ACM International Conference Proceeding Series; Vol. 157. 2005

[11] Martin Naef, Daniel Collicott. *A VR Interface for Collaborative 3D Audio Performance.* New Interfaces For Musical Expression. 2006.

[12] Martin Naef, Oliver Staadt, Markus Gross. *Spatialized Audio Rendering for Immersive Virtual Environment.* Virtual Reality Software and Technology, 2002.

[13] Jaka Sodnik, Saso Tomazik, Raphael Grasset, Andreas Duesner, Mark Billinghurst. *Spatial Sound Localization in an Augmented Reality Environment.* Proceedings of the 20th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: design: activities, artefacts and environments, 2006.

[14] www.portaudio.com. Sitio oficial de PortAudio

[15] www.csounds.com. Sitio oficial de csound.

[16] www.openal.org. Sitio oficial de OpenAL.

[17] supercollider.sourceforge.net. Sitio oficial de SuperCollider

[18] www.vrjuggler.org. Sitio oficial de VRJuggler, incluyendo Sonix.

[19] http://www.cs.ualberta.ca/~pfiguero/InTml/. Sitio oficial de InTml

[20] http://ggt.sourceforge.net/html/main.html Sitio oficial de GMTL.

[21] http://www.eclipse.org/cdt/ Eclipse IDE para C/C++.

[22] http://www.doxygen.org Herramienta de documentación para varios lenguajes de programación.

[23] http://www.mega-nerd.com/libsndfile/ Sitio oficial de Libsndfile. Herramienta para lectura/escritura de archivos de audio.

[24] http://www.opengl.org/ Sitio oficial de OpenGL.

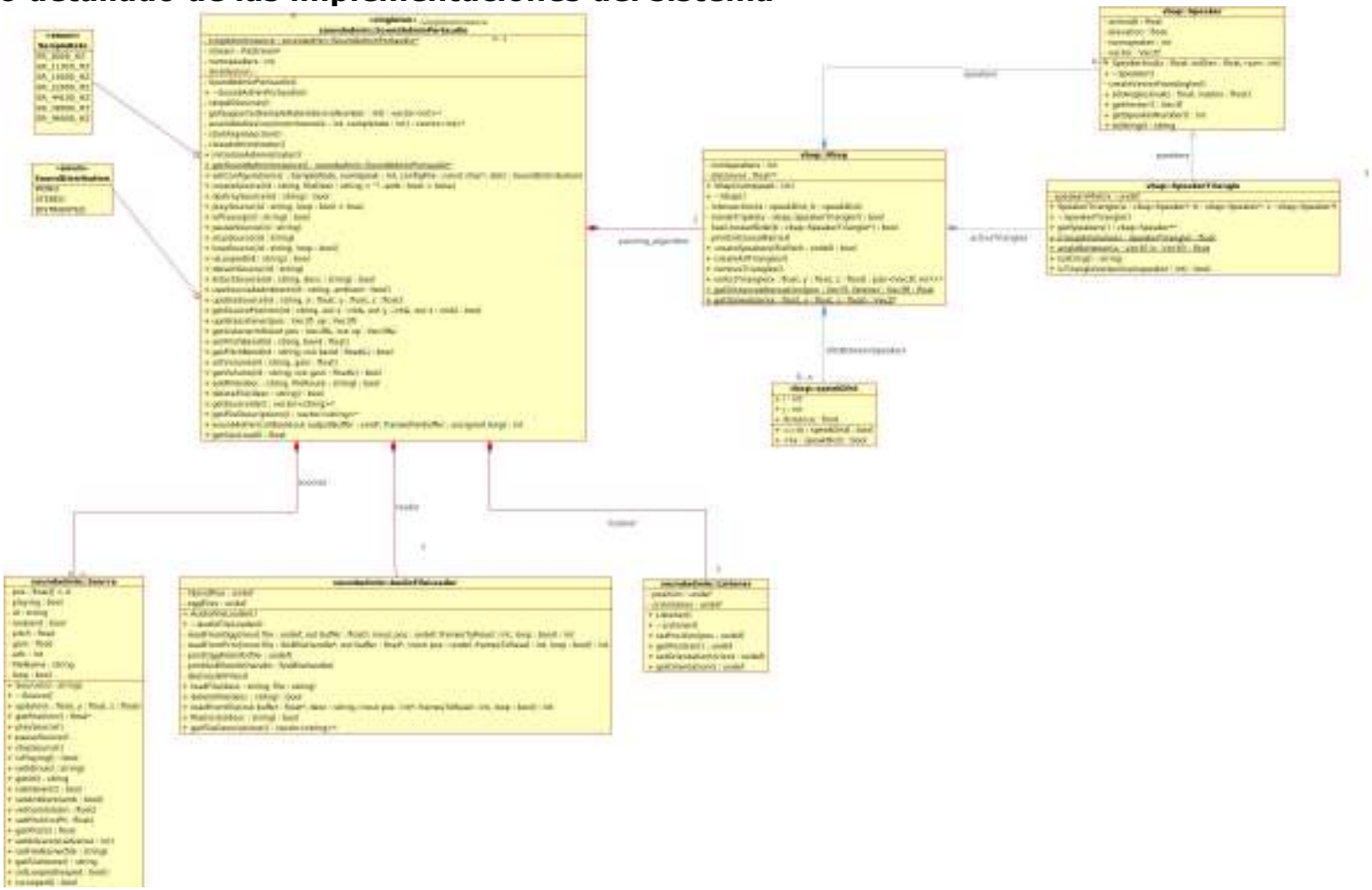
[25] http://www.vorbis.com/ Sitio oficial de Ogg Vorbis. Contiene documentación de las librerías usadas para manipular archivos ogg.

[26] John Jairo Palma Robayo*, Código fuente del administrador de sonido*. Disponible en CD ROM en la Universidad de los Andes.

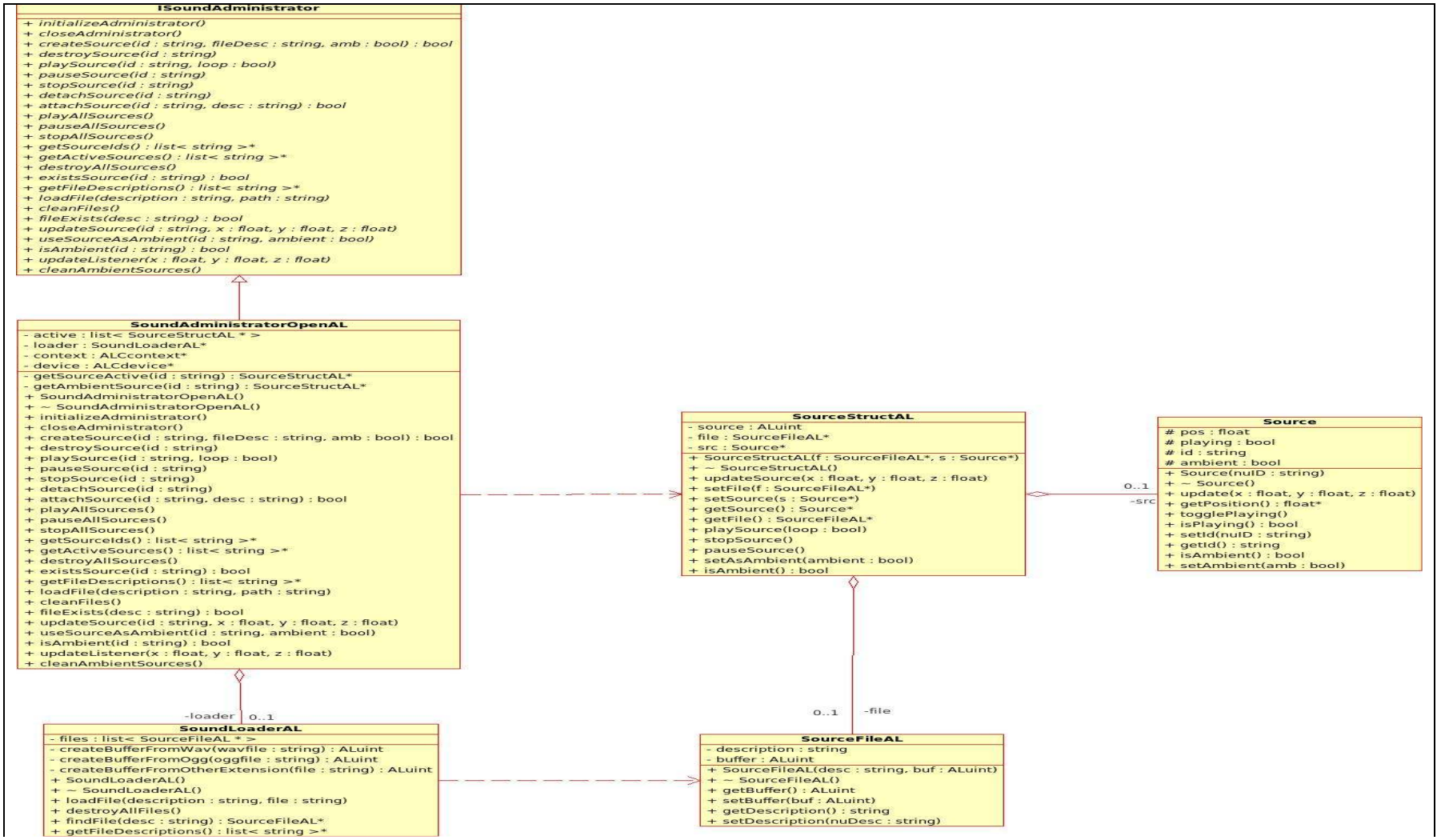
[27] http://imagine.uniandes.edu.co/ Sitio oficial del Grupo IMAGINE.

Anexos

Diseño detallado de las implementaciones del sistema



Diseño Detallado de una implementación del sistema usando PortAudio



Diseño Detallado de una implementación del sistema usando OpenAL

Tablas de comparación entre documentos de investigación

Nombre	Autor (es)	Resumen	No. Parlantes/ Audífonos	Hw utilizado	Sw utilizado	Notas Interesantes
Spatialized Audio Rendering for Immersive Virtual Environments	Martin Naef, Oliver Staadt, Markus Gross	Define una arquitectura para hacer render de audio en un sistema inmersivo. Esto para integrarse con un API (blue-c) desarrollado para este tipo de sistemas.	Configurable a través del API (6 u 8)	▪ Alesis AI-3 (Interfaz AD) ▪ SG Onix 3200 ▪ T.c. electronic M-One XL (reverberación) ▪ Yamaha MSP5 (speakers)	▪ SGI Digital Media Library (comunicación con el HW) ▪ Windows 2000 (OS para la primera prueba)	▪ Fuentes sonoras como nodos de escena ▪ Procesamiento (Doppler, Reverberación, atenuación) ▪ Utilizan HW de audio especializado ▪ Usa VBAP para la localización 3D.
Perceptual Audio Rendering of Complex Virtual Environments	Nicolas Tsingos, Emmanuel Gallo, George Drettakis	Define un pipeline para render de audio 3D que permite manejar un gran número de fuentes. Presenta un algoritmo que reduce el número de fuentes y el procesamiento mediante enmascaramiento	Audífonos (Sennheiser HD600)	▪ Pentium 4 3GHz ▪ Audigy2 platinum Ex	▪ DirecSound3D	▪ Evalúa la relevancia de las fuentes y las acomoda en clusters, que se convierten en fuentes puntuales localizadas en su representante. ▪ Audio de 1200-muestras a 44.1 kHz
Dirt-Cheap 3-D Spatial Audio	Eric Klein, Greg S. Schmidt,	Implementación de audio 3d para un sistema	▪ 8 (configurables	▪ Creative Audigy 2 ZS (tarjeta 7.1)	▪ Linux Red Hat ▪ ALSA	▪ 1 Sólo computador controla todas las fuentes sonoras ▪ Usa VBAP

	Erik B. Tomlin, Dennis G. Brown	inmersivo tipo CAVE usando sw libre y hw no especializado	)		▪ (drivers) ▪ Mustajuuri (API)	
Spatial Sound Localization in an Augmented Reality Environment	Jaka Sodnik, Saso Tomazik, Raphael Grasset, Andreas Duesner, Mark Billinghurst	Desarrollo de un sistema inmersivo tipo tabletop dedicado a la identificación de fuentes sonoras por parte de un usuario.	Audífonos (AKG K44)	▪ Creative SoundBlaster X-Fi Extreme Music. ▪ eMagin Z800 3DVISOR (visualización)	▪ OpenAL. ▪ OpenSceneGraph ▪ OSGART (Framework para tracking y calibración) ▪ OS: Windows	▪ Usa HRTF mediante OpenAL. ▪ Introduce la idea de *artificial coding of elevation* con el fin de mejorar la percepción de elevación ofrecida por OpenAL
Immersive Sound Field Simulation in Multi-screen Projection Displays	T.Ogi, T. Kayahara, M. Kato, H. Asayama, M. Hirose	Desarrolla una tecnología para la simulación de campos sonoros usando filtros de convolución calculados con la ecuación de onda de Kirchhoff. Lo usan en un Avatar Virtual en un entorno tipo CAVE (CABIN)	16	▪ SGI Origin2000 (supercomputador para calculo de *impulse response*)	▪ WaveEngine (DSP Multicanal propietario)	▪ Los parlantes se ubican detrás de las pantallas de manera que se tiene que compensar la atenuación que éstas causan. ▪ Se calcularon todas las posiciones posibles para la fuente y el receptor de acuerdo al modelo, pero esto requirió enrome computación. ▪ Desarrollaron su propio DSP ▪ Usan VBAP para aplicar la *respuesta a impulsos* a los speakers.
A Generic Direct-Manipulation 3D-Auditory Environment for Hierarchical	Anthony Savidis, Constantine Stephanidis, Andreas Korte, Kai	Propuesta de un ambiente de navegación no visual mediante pistas auditivas, usando audio, reconocimiento	Audífonos	▪ Guantes de interacción	N/A	▪ Define en parte un protocolo, pero no hay una implementación clara. ▪ Se usa una estructura tipo anillo para la inmersión del usuario.

Navigation in Non-visual Interaction	Crispien, Klaus Fellbaum	de voz e interacción (pointing) en 3D.				
A VR Interface for Collaborative 3D Audio Performance	Martin Naef, Daniel Collicott.	Define un sistema para el apoyo de la interpretación musical – incluyendo ubicación en el espacio- usando elementos virtuales	14	▪ Guantes de interacción. ▪ Sistema audio de 8 canales.	▪ Windows (PC) ▪ Blue-c (API)	▪ Permite el movimiento de las fuentes sonoras en tiempo real, así como la manipulación de audio en tiempo real y desde archivo. ▪ Orientado a la interpretación musical y a la separación del manejo espacial del proceso de mezcla.
Sonic Panoramas: Experiments with Interactive Landscape Image Sonification	Erik Kabisch, Falko Kuester, Simon Penny	Define un sistema de sonificación de imágenes de gran tamaño dentro de un sistema inmersivo.	6	▪ Interfaz 6 canales (usando FireWire)	▪ Max/MSP/Jitter (Framework programación)	▪ usaban detección de movimiento por parte del oyente y de ahí hacían la localización del sonido.
Echology: An Interactive Spatial Sound and Video Artwork	Meghan Deutscher, Reynald Hoskinson, Sachiyo Takashashi, Sidney Fels	Presenta el diseño de una instalación para interactuar con ballenas Beluga usando múltiples parlantes y un tabletop.	8	▪ N/A (Se supone una tarjeta 7.1)	▪ Windows (PC) ▪ Max/MSP/Jitter	▪ Usa VBAP (dentro de Max/MSP) para la localización 3D

Tablas de comparación entre APIs

Nombre	Objetivo	Tipo (API/Lenguaje)	Lenguaje de Desarrollo	Múltiples Parlantes?
Open Source Audio Library	Manejo de funciones de Audio	API	C++	N/A
Sonix	Audio para VRJuggler	API	C++	Si. Soporta hasta 8.
CSound	Procesamiento de señales y rendering sonoro. Apoyo a la composición musical.	Lenguaje / API	C++	Si. Hasta 16
Mustajuuri	Audio para ambientes virtuales inmersivos	API	C++	Si
PortAudio	I/O de audio.	API	C, C++	Si
OpenAL	Generación de audio 3D para integrar con aplicaciones diversas	API	C, C++, Java (mediante JOAL)	Si
SuperCollider	Síntesis sonora	Lenguaje	C, C++	Si

- ***Comparación de carga de archivos en los sistemas***

Nombre	Carga archivos?	Extensiones Soportadas
CSound	Si	AIFF, WAV, MIDI
SuperCollider	Si	WAV, AIFF. RIFF, Sun, IRCAM, MIDI. Otros formatos son soportados como solo lectura.
OpenAL, OpenAL++, ALUT	Si, pero a través de ALUT	Puede cargar múltiples formatos pero los convierte a soportado por OpenAL

PortAudio	No	Recomendado el uso de libsndfile.
Mustajuuri	Si	Mediante libsndfile.
Libsndfile	Si	WAV, AIFF, RAW, PAF, entre otros. No soporta MP3 por patentes.
Libvorbis, libogg	Si	Ogg, Vorbis
Audio File Library	Si	AIFF, AIFF-C, WAVE, NeXT/Sun, entre otros.
Open Source Audio Library	Si	Mediante Audio File Library
Sonix	Si	Depende del API inferior.

Demos Realizados

En [26] se incluyen diversos demos y pruebas realizadas para probar cada uno de los APIs.

NOTA: Estas pruebas no incluyen archivos de audio.

- CsoundTest: Proyecto realizado para probar el desempeño de Csound con carga de archivos al score. Usa threads para modificar los parámetros de la función hrtf. No es estable por que no reconoce CScore.

- PortAudioTest: Incluye una prueba de mezcla de ondas sinusoidales y salida por medio de PortAudio. Asimismo incluye una prueba de salida por cada canal de forma individual que puede servir para organizar sistemas en torno a PortAudio.

- PortAudioTest2: Proyecto realizado para probar la funcionalidad de PortAudio con múltiples streams.

- VRPN: Prueba que incluye la interacción de dos fuentes de audio con el tracker disponible en el laboratorio IMAGINE.
    - Esta prueba se encuentra disponible tanto en OpenAL (bajo el nombre OpenALTest), como en la versión PortAudio del sistema (bajo el nombre TestTracker.cpp).

- LoadOggFilesOpenALTest: Librería que permite cargar un archivo .ogg y convertirlo a un buffer OpenAL.

- OpenALTest: Incluye una implementación previa de la interfaz para OpenAL.

- MixFileProject: Permite mezclar dos archivos usando PortAudio.

Instalación de herramientas utilizadas.

La siguiente tabla ofrece una descripción de los procesos de instalación de cada una de las herramientas utilizadas para el desarrollo del sistema actual:

Herramienta	**Proceso de Instalación**

CSound	CSound es instalado de maneras distintas en cada sistema operativo.
	• Windows: De la página oficial del proyecto se pueden descargar instaladores para procesadores de 32 y 64 bits.
	• Linux: Se puede instalar de la manera siguiente:
	o *Usando un paquete rpm*: Se descarga el paquete en cuestión y se ejecuta usando el comando:
	`rpm -Uhv csound-5.04.0-1.i586.rpm`
	• *Compilando el sistema usando el código fuente*: Esta aproximación requiere descargar un paquete .tar.gz con el código fuente, descomprimirlo y ejecutar los siguientes comandos en el directorio raíz.
	`./configure`
	`make`
	`make install`

<table>
<tr><td></td><td>

A través de otro sistema: En este caso, fue instalado mediante un conjunto de paquetes disponibles para Fedora Core 6 enfocado a aplicaciones de audio llamado Planet CCRMA (http://ccrma.stanford.edu/planetccrma/software/planetccrma.html). En este caso había que instalar el sistema completo, el cual requería de un nuevo kernel para Fedora. El proceso fue el siguiente:

Actualizar el sistema Operativo

yum update
Instalar la firma del paquete CCRMA

rpm --import http://ccrma.stanford.edu/planetccrma/RPM-GPG-KEY.planetccrma.txt
Agregar los repositorios CCRMA

rpm -Uvh
http://ccrma.stanford.edu/planetccrma/mirror/fedora/linux/planetccrma/6/i386/planetccrma-repo-1.0-3.fc6.ccrma.noarch.rpm
Instalar el núcleo y reiniciar el sistema

yum install planetccrma-core
reboot
Instalar las aplicaciones del sistema.

yum install planetccrma-apps
yum install planetccrma-menus

</td></tr>
<tr><td>SuperCollider</td><td>Linux: Se puede obtener mediante un live cd como pure:dyne, o mediante Planet CCRMA (ver Csound para instalación).

Windows: Se encuentra disponible en el paquete PsyCollider. Sin embargo, es una versión beta.</td></tr>
<tr><td>OpenAL</td><td>Windows: Se puede descargar el instalador desde el sitio oficial, junto con el SDK (Software Development Kit).</td></tr>
</table>

	Linux: Se puede descargar de un repositorio de Fedora Core 6:
	
	`yum install openal`
	`yum install openal-devel`
	`yum install freealut`
	
	También puede ser descargado desde la página oficial de OpenAL e instalado como un rpm (usando el comando `rpm –Uvh <nombre_rpm>`)

PortAudio	Descargar del sitio oficial la última rama de desarrollo y compilar e instalar usando los comandos en el directorio raíz de PortAudio
	
	`./configure`
	`make`
	`make install`

GMTL	GMTL puede ser instalado fácilmente desde los repositorios disponibles para Fedora Core, usando `yum install gmtl`.

Sonix	Como Sonix se encuentra ligado a VR Juggler, es necesario instalar esta suite primero. Lo recomendable es instalar todos los prerrequisitos de la librería antes de probar la instalación del sistema. Algunos de los prerrequisitos incluyen:
	• OpenSceneGraph (http://www.openscenegraph.org/)
	  • Pueden obtenerse binarios de instalación para Windows
	• GMTL
	• Boost C++ Libraries (http://www.boost.org/)
	  • Se pueden instalar usando yum.

libsndfile	Descargar del sitio oficial el código fuente como .tar.gz y ejecutar estos comandos en el directorio raíz.
	
	`./configure`
	`make`
	`make install`
	
	Incluye también una librería auxiliar (SndfileHandle) para usar sndfile con C++.

Libvorbis	Las librerías de manejo de archivos Ogg se pueden instalar desde yum:

	yum install libvorbis yum install libvorbis-devel

Tabla de Comparación del Sistema Finalizado con otros APIs

En este caso el sistema finalizado se refiere a la implementación realizada usando PortAudio, dado que la otra implementación usa un sistema establecido como es OpenAL.

Nombre	**Objetivo**	**¿Múltiples Parlantes?**	**Algoritmo Audio 3D**	**Consideraciones Acústicas**
Sistema PortAudio	*Reproducción y espacialización de múltiples fuentes sonoras*	*Si. Hasta 8. Pueden ser más dependiendo del hardware subyacente.*	*VBAP*	*Atenuación por distancia fuente/escucha.*
CSound	Apoyo a composición musical	Si. Hasta 16	VBAP, HRTF	Depende de los instrumentos creados por el usuario.
OpenAL	Generación de audio 3D para integrar con aplicaciones diversas	Si. Dependiente del hardware.	Depende del hardware subyacente.	Efecto Doppler, atenuación por distancia, cambios de altura (pitch shift).
SuperCollider	Síntesis sonora	Si. Seleccionables.	Potencia constante (4 canales). Ambisonics (stereo)	Depende de la instrumentación creada.