Plug-in de audio para mezclas binaurales utilizando HRTF

Conference Paper · November 2018

CITATIONS
0

READS
194

2 authors:

Estefanía Bergaglio
Universidad Nacional de Lanús
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Jorge Petrosino
Universidad Nacional de Lanús
20 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Simposio Internacional de Arte Sonoro

"Mundos Sonoros: cruces, circulaciones, experiencias"

UNTREF / Instituto de Investigaciones en Arte y Cultura "Dr. Norberto Griffa"

"*Plug-in de audio para mezclas binaurales utilizando HRTF*"

Estefanía Bergaglio
Jorge Petrosino

Resumen

Se presenta el proceso de desarrollo de un plug-in de audio en formato VST usando herramientas que facilitan su realización requiriendo solamente conocimientos básicos y generales de programación así como del tipo de procesos que se desean implementar.

Este plug-in permite realizar mezclas binaurales a partir de fuentes monoaurales que pueden ser ubicadas virtualmente en el plano horizontal con ángulos de acimut que varían entre los 90º a la izquierda y los 90º a la derecha. Las pistas binaurales obtenidas respetan las alteraciones producidas por la función de transferencia de la cabeza (HRTF).

Palabras Clave:
Plug-in de audio, mezclas binaurales, HRTF, procesamiento de señales.

"Plug-in de audio para mezclas binaurales utilizando HRTF"

Mezclas en el plano horizontal utilizando la función de transferencia de la cabeza (HRTF)

El plug-in de audio que se presenta aquí ha sido producido en el marco de una beca de iniciación a la investigación del Consejo Interuniversitario Nacional, cuyo título es "Desarrollo de complemento de software de audio para mezclas binaurales con espacialidad". Se incluyen entre los productos desarrollados un texto explicativo donde se detalla el proceso para crear y modificar un plug-in incluyendo la instalación y configuración de los distintos softwares, así como el modo de obtener las respuestas al impulso necesarias para la convolución, extraídas de la base de datos del IRCAM. Toda la información producida, incluyendo pistas de audio para probar su utilización, se encuentra disponible en una carpeta de Google Drive a la que se puede acceder mediante el siguiente enlace http://cor.to/LUOo. Tanto el plug-in como el código fuente se ponen a disposición bajo licencia de Creative Commons para su uso libre.

Mezclas binaurales utilizando HRTF

Se denomina mezcla de audio al proceso que toma una serie de registros de audio para generar un único producto sonoro. El proceso típico parte de registros monoaurales y obtiene un producto estéreo. La mezcla pretende, entre otras cosas, dar lugar a una distribución aparente de las fuentes sonoras incluidas en los registros monoaurales. Para lograr que una fuente monoaural sea percibida hacia la izquierda, por ejemplo, se genera una copia del archivo en ambos canales otorgando mayor nivel al canal izquierdo. Este método se conoce como paneo por nivel y fue propuesto por Alan Bloomlein a principios del siglo XX (Burns, 1999).

Es importante mencionar que dicho procedimiento da lugar a una sensación de localización en una dirección, pero no refleja la situación física que se produciría si efectivamente la fuente se encontrase un poco más a la izquierda. Desde un punto de vista físico, la propagación de ondas esféricas al aire libre produce una disminución de 6 dB cada vez que se duplica la distancia. Si la fuente se encuentra en un recinto cerrado dicha caída puede resultar aún menor. En una situación de mezcla mediante una consola, no es nada extraño que se genere artificialmente una diferencia de nivel de 6 dB o más para localizar un determinado instrumento hacia la izquierda. Para que esto se correspondiese con la situación física la fuente sonora debería estar ubicada de modo que la distancia al oído derecho fuese el doble que la distancia al izquierdo. Si la fuente se encuentra a 2 metros del oído izquierdo, debería simultáneamente encontrarse a 4 metros del oído derecho, lo que resulta físicamente absurdo.

Los indicios de ubicación de una fuente sonora que se corresponden con la propagación física de las ondas sonoras al aire libre se relacionan con tres tipos de fenómenos (Figura 1):

a) la diferencia de tiempos de llegada del sonido a cada oído en función de la ubicación de la fuente que produce un retardo de hasta 700 microsegundos,

b) el efecto de sombra acústica en el oído contralateral, producido por la cabeza del oyente que provoca una disminución de las componentes de alta frecuencia (difracción de la cabeza),

c) la interferencia debida a diversas reflexiones de las ondas en el pabellón auricular y el torso del oyente, que dependen fuertemente de la ubicación física de la fuente.

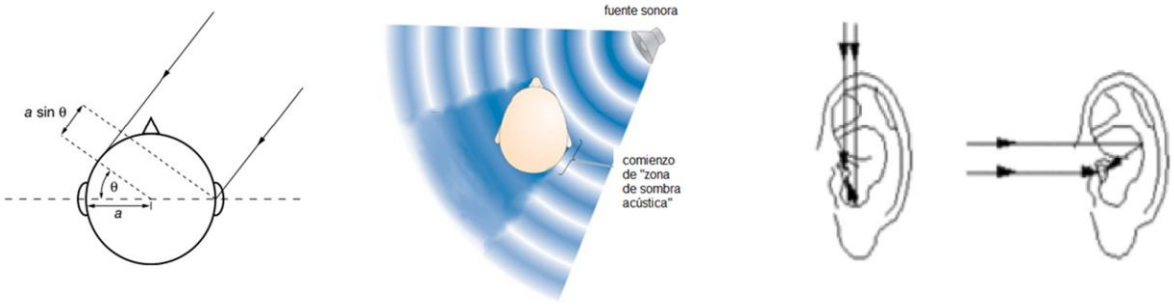

Figura 1 - Representación de los tipos de indicios de localización de una fuente sonora

Estos tres tipos de alteraciones que el entorno físico provoca a las ondas pueden ser representadas matemáticamente mediante lo que se conoce como función de transferencia de la cabeza (Head-Related Transfer Function, o HRTF). Si se procesa una fuente sonora y se conoce la información de HRTF correspondiente a la ubicación de la fuente respecto de cada oído, pueden obtenerse las señales que llegarían al oído izquierdo y al oído derecho. Suele utilizarse el término audio binaural para referirse al producto obtenido mediante este procedimiento que genera una mayor sensación de localización de la fuente sonora. Para generar la sensación dicho registro binaural debe ser reproducido mediante auriculares. La reproducción del mismo mediante altavoces no logra generar la misma sensación clara de localización.

El término mezcla de audio binaural hace referencia al procesamiento de una serie de fuentes sonoras monoaurales para obtener un producto sonoro en dos canales con el fin de localizar las fuentes en distintos puntos del espacio respetando los distintos tipos de indicios. El procesamiento mediante HRTF puede ser utilizado para generar resultados que ubiquen a la fuente en posiciones que estén al mismo nivel, por encima o por debajo del plano horizontal (según el ángulo de elevación de la fuente). Por una cuestión de simplificación del procedimiento matemático, si se desea recrear principalmente la ubicación virtual de una serie de instrumentos musicales es posible restringir la zona de localización al plano horizontal frente al oyente. Esto se corresponde con una fuente de ángulo de elevación nulo y con variaciones del ángulo de acimut entre -90º y 90º.

Existen varias bases de datos de HRTF de acceso libre, como la producida por el IRCAM (Institute for Research and Coordination in Acoustics/Music) que se encuentra disponible en http://recherche.ircam.fr/equipes/salles/listen/index.html. Esta base de datos contiene un conjunto de archivos de audio de unos pocos milisegundos de duración que contiene grabaciones realizadas desde diferentes localizaciones de fuentes. Estos archivos de audio contienen las respuestas al impulso que llegan a cada oído (Head-Related Impulse Responses, o HRIR). La respuesta al impulso HRIR y la función de transferencia HRTF son representaciones alternativas del mismo fenómeno. La función de transferencia HRTF no es más que la representación espectral (transformada de Fourier) de las respuestas al impulso HRIR. Las bases de datos contienen la información de respuestas al impulso para cada oído a partir de mediciones con una fuente localizada en una gran variedad de ubicaciones. La base del IRCAM contiene información de 187 puntos de localización para el oído izquierdo y otras tantas para el oído derecho. La localización de las fuentes de esta base de datos no incluye variación en la distancia a la fuente, sino en el ángulo de arribo (combinando el ángulo de acimut y de elevación). Para obtener los archivos se utilizaron micrófonos intra-aurales registrando las respuestas al impulso de 51 personas. Para cada persona existen 187 archivos estéreo de 24 bits para cada posición de la fuente. El nombre de cada archivo incluye información básica que puede explicarse en forma sencilla mediante un caso tomado como ejemplo. El archivo IRC_1002_C_R0195_T030_P000.wav corresponde al sujeto identificado como 1002, la C indica que se trata de un archivo compensado en el que se han corregido las variaciones producidas por el sistema de amplificación, el número que sigue a R es la distancia de la fuente en centímetros, el que sigue a la T es el ángulo de acimut en grados y el que sigue a la P es el ángulo de elevación en grados.

El proceso necesario para obtener el registro binaural a partir de un archivo monoaural se conoce como convolución, y resulta ser una función muy utilizada en procesamiento de señales. La convolución entre la señal monoaural y la respuesta al impulso del oído derecho, dará como resultado la señal correspondiente al oído derecho. El mismo proceso se aplica para obtener la señal del oído izquierdo.

Esta intrincada descripción deja claro que el proceso necesario para realizar una mezcla binaural a partir de la información de las bases de datos es demasiado engorroso. Es aquí donde la posibilidad de disponer de un plug-in que tenga incorporada la información de las respuestas al impulso y la capacidad de realizar la convolución se convierte en algo conveniente.

Plug-in de audio

Un plug-in de audio es un complemento de software que incrementa las capacidades del sistema al que se añade, conocido como anfitrión o *host*, al proveer funcionalidades específicas (Pinos Vargas, 2010). Está compuesto por un algoritmo DSP, es decir un bloque de código a través del que se realiza el procesamiento digital de las señales de audio (Pinos Vargas, 2010; McPherson y Reiss, 2015), y por una interfaz gráfica o GUI, que es la traducción del código a textos, objetos y texturas interpretables por los usuarios (Goudard y Muller, 2003). Esta idea permite que los usuarios experimentados creen nuevos procesos para un software de uso general, mientras que los usuarios no experimentados pueden utilizar estos complementos agregándolos al programa que utilizan.

El funcionamiento general de un plug-in varía según distintos atributos, siendo uno de ellos la función que cumple. Ésta puede ser de análisis, como mostrar el espectro de Fourier, de síntesis, generando un sonido en particular, o de transformación, como un ecualizador (Goudard y Muller, 2003). Otra característica que influye en su funcionamiento es su formato o arquitectura, ya que de éste depende su compatibilidad con las aplicaciones anfitrionas y los sistemas operativos. Entre las arquitecturas más conocidas se encuentran RTAS y AAX de Avid Technology, AudioUnit de Apple y VST, desarrollado por la compañía Steinberg. Esta última es una de las más utilizadas ya que es compatible con la mayoría de plataformas DAW profesionales (McPherson y Reiss, 2015), a la vez que es una de las más populares entre los programadores independientes debido a que las licencias

necesarias para el desarrollo de este tipo de plug-in son gratuitas (Berrío Bernal, 2012).

Uso del plug-in

El plug-in desarrollado es de uso libre y gratuito y está disponible en formato VST para Windows. Para utilizar el complemento, en primer lugar se debe copiar el archivo (de extensión .dll) en el directorio de plug-ins VST correspondiente al software anfitrión que se utilizará, o bien incorporar un nuevo directorio a la lista de las carpetas desde donde el anfitrión accede a dichos complementos. Este paso dependerá del *host* que se utilice y de cómo éste trabaje el uso de plug-ins. Luego de esto, basta con insertarlo en algún canal que contenga una pista de audio monoaural. Al hacerlo, se abre un canal de comunicación que envía datos del audio monoaural al complemento y devuelve al host la información de ambos canales de audio monoaural.

La interfaz de este complemento de audio está compuesta por una lista desplegable que contiene las opciones de los ángulos de acimut, que varían entre los 90º a la izquierda y los 90º a la derecha, con diferencias de 15º entre cada una. También cuenta con un control deslizante del volumen de salida para compensar el nivel en caso de que sea necesario (Figura 2).

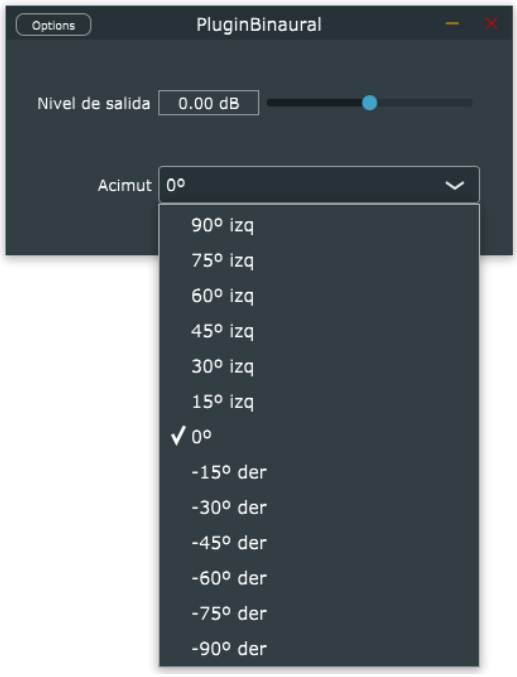

Figura 2: "PluginBinaural" con la lista de opciones desplegada.

El plug-in tiene incorporado dentro de su código la información de la base de datos de respuestas al impulso del IRCAM, por lo que al elegir una de las opciones de la lista, el complemento realiza la convolución en tiempo real entre dicha pista de audio y la respuesta al impulso correspondiente al ángulo de acimut seleccionado, obteniendo como resultado el audio con paneo binaural. Cabe aclarar que tanto los valores del ángulo de acimut como los del nivel de salida pueden ser automatizados según las opciones disponibles en el *host* utilizado.

Para poner a prueba el complemento durante el proceso de beca se utilizó el software anfitrión Reaper, que es gratuito en etapa de evaluación.

Programación de un plug-in de audio

Los plug-ins de audio suelen desarrollarse utilizando el lenguaje C++ (Rumsey, 2004). La programación en C++ se basa en escribir una serie de instrucciones, conocidas como código fuente, en uno o más archivos de texto. Éstos, denominados archivos o ficheros fuente, son luego convertidos en lenguaje de máquina por el procesador de la computadora, obteniendo como resultado el software final en su formato correspondiente (Davis, 2004). Para llevar esto a cabo se utiliza un entorno de desarrollo integrado o IDE, que es un programa que permite la edición del código fuente y su compilación para la generación del software. Se requiere de conocimientos avanzados en programación y de mucha práctica para llevar a cabo el desarrollo de plug-ins, (Collins, 2003). Sin embargo, existen diversas herramientas que facilitan el proceso de programación en menor o mayor medida, permitiendo generar complementos de audio con alguna funcionalidad específica sin contar con grandes conocimientos sobre el tema.

El desarrollo del plug-in que se presenta aquí fue realizado en su totalidad con herramientas de uso gratuito. Por un lado se utilizó JUCE, un entorno de trabajo compatible con Windows, macOS y Linux para el desarrollo de distintos tipos de software, que consta de una librería con una amplia gama de clases o módulos que resuelven problemas comunes de desarrollo de software, simplificando de gran manera el trabajo a realizar por el usuario. Para utilizar estos módulos el *framework* cuenta con una aplicación llamada Projucer, cuyas opciones de configuración permite hacer ajustes en relación a los formatos, la compatibilidad con el sistema operativo, la arquitectura, entre otras cosas, facilitando aún más el proceso. Otra gran ventaja de JUCE es que con su descarga viene incluida una serie de ejemplos de softwares desarrollados con dicho *framework*, junto con sus respectivos archivos fuentes para que el usuario pueda modificarlos si así lo desea. Por otra parte, en su sitio web se ofrece una gran variedad de tutoriales de uso, junto con un índice de todas las clases y módulos que contiene. Para la edición del código y la compilación se utilizó el IDE denominado Visual Studio Community, de la compañía Windows, mientras que para la puesta a prueba del complemento de audio se utilizó la estación de trabajo digital Reaper. Por último, las respuestas al impulso utilizadas para la convolución fueron extraídas de la base de datos pública del Institut de Recherche et Coordination Acoustique/Musique (IRCAM). Dicha base de datos está compuesta por las respuestas al impulso de 51 sujetos recibidas desde una gran variedad de posiciones de la fuente, con diferencias de 15° entre cada una.

Es pertinente aclarar que si se desea generar el complemento de audio en los formatos VST3, RTAS o AAX es necesario adquirir el kit de desarrollo de software (o SDK) correspondiente a cada uno, mientras que para los formatos VST, AudioUnit y AudioUnit v3, así como para la versión Standalone, no es necesario agregar ningún otro componente. Por otra parte, también es necesario realizar algunos ajustes de configuración en la aplicación Projucer. Éstos se encuentran especificados en el documento “Desarrollo de plug-ins de audio con JUCE” al que se puede acceder mediante el siguiente enlace: http://cor.to/LSaE.

La aplicación Projucer permite comenzar con un proyecto en blanco o bien con una plantilla basada en el tipo de software que se desee desarrollar, en este caso un plug-in de audio, creando automáticamente los archivos con el código fuente necesario para una interfaz gráfica de una ventana y funciones de entrada y salida de audio. Los ficheros “PluginEditor.cpp” y “PluginEditor.h” son para las instrucciones relacionadas a los elementos de la interfaz gráfica, mientras que “PluginProcessor.cpp” y “PluginProcessor.h” son para el código correspondiente al procesamiento que el plug-in realiza. Si bien esta segunda opción facilita el proceso en cierta medida, igualmente se requieren de conocimientos avanzados de programación y de C++ para lograr obtener como resultado un plug-in con alguna funcionalidad específica. Sin embargo, existe otro modo que fue el elegido en este caso y que se basa en partir de un proyecto existente para luego realizar las modificaciones necesarias.

Cuando se trabaja con un proyecto de Projucer, sea cualquiera de los casos mencionados anteriormente, la carpeta donde se encuentra dicho archivo (de extensión .jucer), contiene además otras tres carpetas: “Builds” tiene los archivos necesarios para abrir el proyecto en el IDE elegido y será el destino del plug-in cuando se realice la compilación; “JuceLibraryCode” posee los archivos de código fuente necesarios para el funcionamiento del complemento (como módulos, código relacionado a la

configuración y archivos binarios); “Source” contiene los cuatro ficheros fuente mencionados anteriormente, en los que se desarrollará el código del plug-in.

Para desarrollar el complemento presentado aquí se partió del ejemplo de JUCE denominado “DSP module plugin demo” ya que contenía las líneas de código correspondientes a los elementos necesarios, es decir una lista desplegable con opciones de respuestas al impulso y un control deslizante del volumen de salida.

En la carpeta de Google Drive mencionada anteriormente se encuentra la información acerca de los cambios realizados en el código fuente del plug-in, además de la explicación detallada de todo el proceso, para que cualquiera que lo desee pueda seguir los pasos indicados y obtener como resultado un complemento de audio funcional.

Referencias bibliográficas

- Berrío Bernal, Juan David (2012). *Plug in VST de reverberación basado en el modelo de fuentes imagen*. Trabajo de investigación, Ingeniería de Sonido, Facultad de Ingenierías, Universidad de San Buenaventura, Medellín, Colombia.
- Burns, R. (1999). Blumlein and the birth of stereo. IEE review, 45(6), 269-273.
- Collins, Mike (2003). *A Professional Guide to Audio Plug-ins and Virtual Instruments*. Italia: Editorial Focal Press.
- Davis, Stephen Randy (2004). *C++ For Dummies, 5th Edition*. Indianápolis, Estados Unidos: Editorial Wiley Publishing, Inc.
- Goudard, Vincent y Muller, Remy (2003). *Real Time Audio Plugin Architectures*.
- McPherson, Andrew y Reiss, Joshua (2015). *Audio Effects: Theory, Implementation and Application*. Estados Unidos: Editorial CRC Press.
- Pinos Vargas, José Luis (2010). *Implementación de Tratamiento Digital de la Dinámica de Señales de Audio en C++ Utilizando Especificación VST*. Tesis de grado, Universidad San Francisco de Quito, Ecuador.
- Rumsey, Francis (2004). *Desktop Audio Technology: Digital Audio and MIDI Principles*. Holanda: Editorial Focal Press.