Universidad Nacional de Quilmes

DEPARTAMENTO DE CIENCIAS SOCIALES
V JORNADA DE BECARIOS Y TESISTAS 2015

Proyecto de Investigación
Cartografías de Espacio-Tiempo y Arte Sonoro
(Programa Perspectiva Acústica)

Autor: Carlos Daniel Cattaneo
Director: Lic. Hernán Kerlleñevich
Co-Director: Dr. Pablo E. Riera

1) Título: Desarrollo de una Consola de Mezcla de Sonido en Pure Data.

2) Autor: Carlos Daniel Cattaneo.

3) Direccion electrónica: carlo1112@gmail.com

4) Formación de grado en curso: Licenciatura en Música y Tecnología.

5) Tipo de beca: Beca de Formación en Docencia e Investigación.

6) Tema de la tesis en preparación:

7) Director de la beca: Lic. Hernán Kerlleñevich.

Co-director: Dr. Pablo E. Riera

8) Denominación del programa y proyecto: Programa: "Perspectiva Acústica", Director: Dr. Manuel Eguía. Proyecto: "Cartografías de Espacio-Tiempo y Arte Sonoro", Director: Dr. Pablo E. Riera.

9) Denominación de Grupo: LAPSo. (Laboratorio de Acústica y Percepción Sonora)

Desarrollo de una Consola de Mezcla de Sonido en Pure Data.

Cattaneo, Carlos. Kerlleñevich, Hernán. Riera, Pablo.

Resumen

El marco de trabajo de la Luthería Digital propone mejorar el vínculo entre las herramientas de programación (entornos y lenguajes de *creative coding*) y el desarrollo de aplicaciones musicales. Dentro de este marco se viene trabajando con herramientas de código abierto para el desarrollo de software para la creación musical, en base a la generación, control y análisis de materiales sonoros y visuales.

Como punto de contacto entre el marco de trabajo de la Luthería Digital y el entorno de programación de audio Pure Data (Pd), se viene desarrollando una consola de mezcla de sonido, que permite poner a prueba ciertos conceptos e ideas sobre la programación en Pd, haciendo hincapié en el diseño de módulos y su documentación.

La consola que se está desarrollando permite agregar, de manera sencilla, canales y auxiliares que poseen procesadores de ecualización, compresión y envíos a canales con efectos. A su vez, la consola permite la utilización de los protocolos de comunicación OSC y MIDI para ser controlada de forma externa con otros dispositivos como tabletas y controladores MIDI.

Una consola de estas características permite trabajar con interfaces de audio multicanal para manipular audio de entrada y de salida. Esto permite, por ejemplo, realizar mezclas en tiempo real, como también grabaciones multicanal y cualquier procesamiento de audio que sea posible

realizar mediante Pd. En un futuro se realizarán las pruebas de latencia necesarias para establecer las configuraciones óptimas que permitan trabajar en tiempo real.

Introducción

La principal función de una consola de mezclas es el enrutamiento de señales. Suele poseer múltiples entradas que pueden ser copiadas o direccionadas a otros canales, y mezcladas en uno o más canales de salida (*master*). Cada consola posee varios tipos de limitaciones, un ejemplo son los tipos de canales que poseen, que pueden ser mono, estéreo o multicanales. Algunas consolas permiten enrutamientos de señales complejos y otras permiten hacerlo de forma versátil.

Por lo general, dependiendo de la consola, un canal tiene una entrada de micrófono o línea, donde se adaptan las impedancias, una salida directa, ecualización, envios a mezclas auxiliares, opción de *mute* y *solo,* enrutamiento a grupos, un amplificador controlado por voltaje y salida general al canal *master*.

En los comienzos, las capturas de sonido de bandas u orquestas eran mezcladas de forma acústica, el balance del sonido se conseguía mediante la posición de los músicos, la acústica del recinto y la posición de un sólo micrófono. Con la aparición de las consolas de mezcla se comenzó a registrar más de un micrófono a la vez y equilibrar (o mezclar) las señales eléctricamente, en lugar de acústicamente.

El concepto de mezcla como lo conocemos hoy en día, surge en la década del 60 cuando aparecen los primeros grabadores multi-canal de cinta. Usualmente tenían 4 u 8 canales, y se utilizaron consolas de mezcla muy simples para mezclar los canales luego de ser grabados.
Hoy en día se cuenta con consolas digitales y con los sistemas DAW (*Digital Audio Workstation*) que corren en computadoras.

Consola de Audio en Pure Data

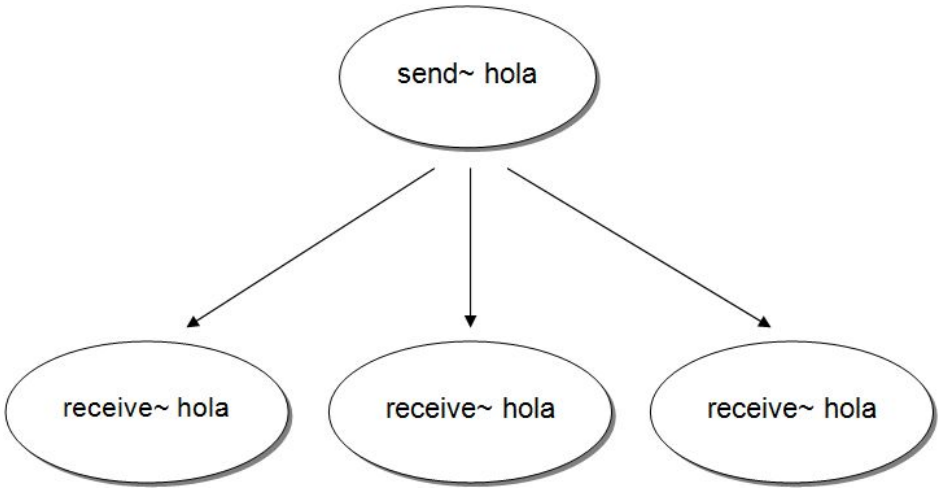
En este trabajo realizamos una consola en el lenguaje PureData (Pd). Para esto generamos un conjunto de herramientas con el propósito de facilitar la programación y uso de la consola.
Pd tiene una forma de programación donde se conectan "objetos" mediante "cables", estos cables transportan señales de audio o de control de un objeto a otro en un solo sentido. Esto hace que este tipo de entorno de programación sea idóneo para desarrollar una consola, y cualquier sistema de procesamiento de audio.

Se consideró un diseño del programa donde el flujo de información de la consola estaba dividido en dos tipos de abstracciones. Por un lado, se encuentran las abstracciones que manejan las señales de audio y por otro las que manejan las señales de control. Estas dos abstracciones se comunican entre sí mediante mensajes de control.. La división del flujo de información permite que el audio de la consola funcione de forma autónoma sin la necesidad de ejecutar todos los componentes del control, permitiendo que se puedan utilizar programas alternativos para realizar

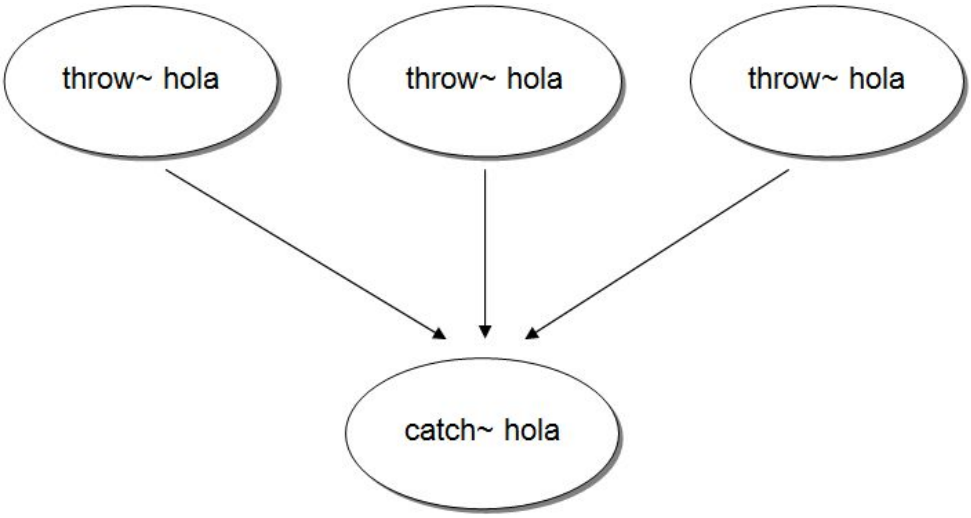
esta función. Esta separación permite que dentro del programa haya una estructura clara y resultó muy cómoda a la hora de trabajar.

A la hora de diseñar un conjunto de abstracciones, u objetos de Pd que facilitasen el trabajo para la confección de la consola surgieron algunas limitaciones en lo relacionado al enrutamiento de audio. Esta limitación está dada por cómo funcionan los objetos *send~*, *receive~* y *catch~*, *throw~*, los cuales permiten conectar señales de audio entre dos objetos sin la necesidad de utilizar un *cable*.

El par *send~*, *receive~*: desde un punto, se puede hacer un sólo envío (*send~*) y se puede recibir (*receive~*) en múltiples puntos, como muestra la imagen:



En el par *catch~*, *throw~*, puede haber múltiples envíos (*throw~*), pero solo un receptor (*catch~*) en el que se hace la sumatoria (mezcla) de todas las señales de audio, como se puede ver en la imagen:



En un primer momento se desarrollaron 3 tipos de canales, de entrada, de envíos y de salida principal (*master*), cada uno con un objeto distinto de Pd, pero nuestra idea era tener una "clase" de canal padre que los unificara, como suele suceder en los lenguajes de programación orientados a objetos.

Originalmente los canales tenían la siguiente estructura:

- Canal de entrada:

  En la entrada, un objeto *adc~* (señal desde interfaz de *hardware*) y uno *receive~*. Un envío y una salida (*master)* mediante los objetos *throw~*.

- Canal de envíos:

  En la entrada, un objeto *catch~*. En la salida, un objeto *throw~*.

- Canal de salida (*master*):

  En la entrada, un objeto *catch~*. En la salida, un objeto *throw~*.

Para que todos los canales fueran compatibles necesitaban tener *catch~* y *throw~* o *receive~* y *send~* en la entrada y en la salida, respectivamente.

Para poder implementar una "clase" de canal genérica era necesario que desde múltiples emisores se pudiera enviar a múltiples receptores, lo cual no era posible con los objetos estándar de Pd, por lo que se decidió construir un nuevo objeto capaz de hacer esta tarea.
Primero implementamos un objeto con *dynamic patching* e *instancias*, donde la primera instancia crea un *catch~* y la segunda crea un *receive~*, y si se decide eliminar alguna instancia, se corrigen los patches restantes, pero esto generó problemas ya que los archivos originales de los objetos se modificaban, al salir del programa principal Pd preguntaba si se querían guardar los cambios y se desplegaban ventanas emergentes no deseadas.

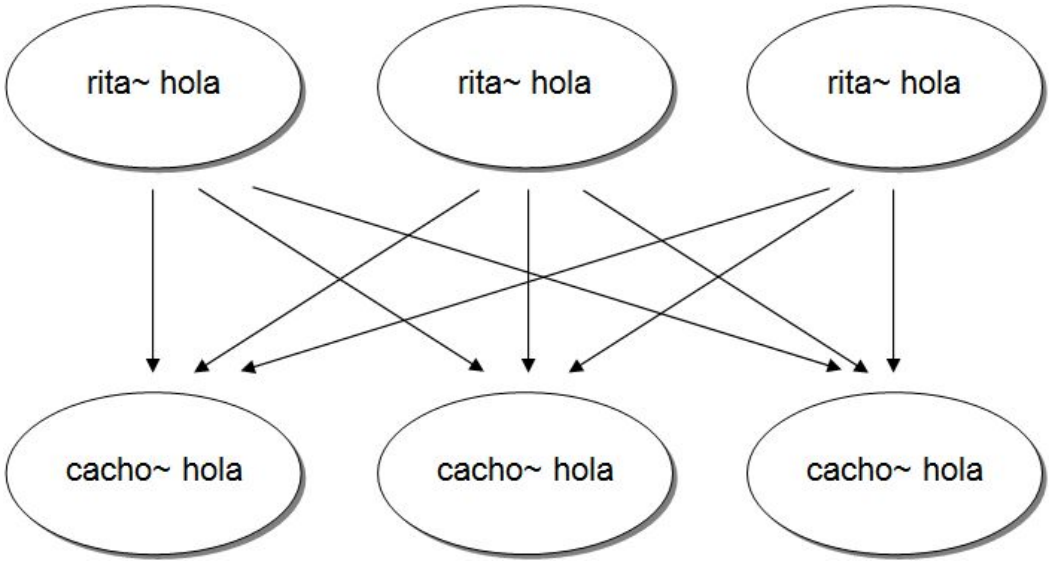
Desarrollo de los *externals rita~* y *cacho~*

Para profundizar en el funcionamiento interno de los objetos *send~*, *receive~* y *catch~*, *throw~*, decidimos observar el código fuente de ellos. En el caso del *send~*, este escribe en una dirección de bloque de memoria fija y el *receive~* lee desde esa dirección, por este motivo no se pueden tener multiples *send~*s. A su vez, analizando los códigos de *catch~* y *throw~*, observamos que cuando se crea un *catch~*, se aloja un bloque de memoria que es inicializado en 0 y se dispone a recibir la información de los *throw~*. Cuando los *throw~* envían la señal, lo hacen de forma aditiva y no sobreescribe la memoria, luego, el *catch~* al finalizar la lectura vuelve a poner los

valores en cero. Esto logra que si otro *catch~* estuviese abierto tratando de leer la misma porción de memoria, leería puramente valores en cero.

Decidimos realizar nuestro propio objeto de audio externo (*external*). Para esto tomamos como base el par *throw~*, *catch~* ya que era el más conveniente, debido a que realiza una suma de señales.

Se construyó el par de objetos nuevos denominados *rita~*, *cacho~*.[1] Este par permite tener múltiples *rita~* (envíos) y múltiples *cacho~* (receptores) como se muestra en la siguiente imagen:



En base a los códigos de *throw~* y *catch~*, se realizaron modificaciones de algunos aspectos de los mismos, para que se puedan tener múltiples *catch~*. El principio fue similar al que se mencionó en el caso de *dynamic patching*. El primer *cacho~* que se crea asume el rol de primera

[1] Estos nombres son simplemente ilustrativos y temporarios.

instancia y es el que aloja el espacio de memoria. Los siguientes *cacho~,* ya sabiendo que hay una primera instancia, utilizan el mismo espacio de memoria para la lectura. Para volver los valores a cero se cuenta cuando se realizan cada lectura y se espera hasta que se cumplan todas.

Los objetos *rita~*, *cacho~* pueden ser descargados desde este repositorio *git*: https://github.com/pabloriera/pd/tree/master/cachorita

Luego a partir de estos objetos nuevos, fue relativamente sencillo construir una "clase" padre para los canales que permitió tener un objeto unificado para todos los canales ya sea de entrada, envío o de salida.

Finalmente la consola se completó con la siguientes funcionalidades estandares:

Cada canal cuenta con un fader, vúmetros, inversor de fase, ecualizador con analizador de espectro, solo/mute, panorama (canales mono), balance (canales estéreo), envíos a otros canales.

Sistema de presets

Otra particularidad de la consola que se desarrolló reside en que se utilizó un sistema de presets (*state manager*), desarrollado previamente en el grupo de trabajo, que permite realizar cambios de páginas de canales en la consola y evita tener que ver todos los canales en la pantalla al mismo tiempo. Esto se realiza mediante etiquetas que asignan a cada control una variable global y permite, además, guardar los valores de los variables en un archivo XML externo, para luego

cargarlas de forma instantánea en la consola. A su vez, este sistema administrador de las variables está integrado con comunicación via OSC.

Conclusiones

Se trabajó en el desarrollo de una consola de audio en Pd buscando potenciar la creación de herramientas que permitan un diseño flexible.

El nuevo par de objetos *rita~*, *cacho~* permite una nueva forma de usar Pd que equivale a poder *patchear* o interconectar objetos de audio, a partir del uso de etiquetas o nombres que identifiquen los puertos de entrada y salida de audio. De esta manera es posible tener una lista de canales de entrada y de salida y conectarlos de forma arbitraria sin utilizar cables y el *mouse*.

La utilización de los objetos *send~*, *receive~* , *catch~*, *throw* , *rita~* y *cacho~* introducen en muchos casos una latencia de un bloque de audio, aunque puede ser evitada si el orden de ejecución se realiza de forma correcta entre la escritura y lectura de los bloques de memoria.

Para concluir con el trabajo de la consola será necesario revisar el estado de la latencia de entrada, procesamiento y salida, tanto interna de la consola como la relacionada a Pd, el sistema operativo y la interfaz de audio, tratando de lograr un máximo de latencia aceptable como ocurre en las consolas profesionales.

Si bien el desarrollo de una consola en Pd no reemplaza al uso de los multipista, permite tener a disposición una consola totalmente personalizable que podría ser adaptada de forma rápida para diferentes situaciones de performance o instalaciones sonoras.

Bibliografía

- HOWTO write an External for Pure Data IOhannes m zmölnig March 25, 2014. http://pdstatic.iem.at/externals-HOWTO/
- The Mixing Engineer's Handbook, by Bobby Owsinski. 1999.
- The Recording Engineer's Handbook, by Bobby Owsinski, 2009.
- Mixing Audio - Concepts, Practices and Tools, by Roey Izhaki, 2008.
- The Theory and Technique of Electronic Music, by Miller Puckette, World Scientific, 2007.