Pablo Cetta

WAUL

Entorno visual para Web Audio

Tutorial y Guía de referencia

Enero de 2018

Índice

Librería *WAUL*

Introducción

Web Audio[1] es una API (Interfaz de Programación de Aplicaciones) destinada al procesamiento y síntesis de audio digital en aplicaciones web, desarrollada por el *Audio Working Group* del *W3C Consortium.*

El propósito de la librería *WAUL* es la utilización facilitada de la *Web Audio API* a través de una interfaz gráfica que permite la programación empleando objetos interconectables por medio de cables virtuales. El desarrollo de la interfaz fue realizado partiendo de la librería p5.js[2], creada por Lauren McCarthy, y en particular p5.dom, que permite interactuar con distintos elementos presentes en HTML 5. *WAUL* fue creada en principio con fines didácticos, dado que la posibilidad de ser utilizada desde un navegador hace propicia la tarea de difundir y practicar diversos procesos de tratamiento digital de señales de audio, aplicables en música..

La *Web Audio API* ya se encuentra implementada en gran parte, en las versiones más recientes de los navegadores actuales. Si bien se la considera aún un diseño experimental, su utilización permite ver la enorme potencialidad de este desarrollo. La librería *WAUL* no cubre todas sus características aún, si bien agrega otras, gracias a la posibilidad de extender la API a través de la programación en JavaScript.

Programación con WAUL

La lógica del flujo de datos utilizada por esta librería intenta seguir los criterios empleados en otros lenguajes de procesamiento en tiempo real, como *Pure Data*[3] o *Max-MSP*[4]. Incluso los nombres de los objetos son en muchos casos iguales o similares.

La programación se realiza sobre una ventana (un *canvas*), en cuya parte superior se destaca un menú con cuatro botones y una lista desplegable.

Menú de la ventana de programación

[1] La documentación más reciente puede consultarse en https://webaudio.github.io/web-audio-api/

[2] https://p5js.org/
[3] https://puredata.info/
[4] https://cycling74.com/

El primer botón permite encender o apagar el procesamiento de audio digital, la acción sobre el segundo ícono crea un nuevo *patch* o programa, y los dos botones restantes sirven para cargar o guardar *patches* de forma local, o sea en el ordenador del usuario. La lista desplegable, por último, sirve para acceder a los *patches* almacenados en el servidor. Esos *patches* son los que iremos viendo a lo largo de este tutorial.

Si deseamos editar un *patch* almacenado es preciso ingresar en el modo de edición. Esto se logra presionando *Shift* + *u*, tanto para entrar al modo como para salir de él.

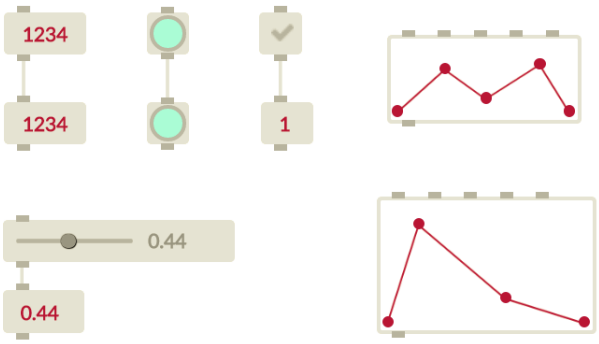
Para comenzar a programar hacemos *click* en el ícono de creación de nuevo *patch*. El modo de edición se seleciona automáticamente. Sobre la ventana de programación creamos un nuevo objeto presionando *Shift* + 1 (ver la lista de *shortcuts* –atajos- al final del documento). A continuación escribimos el nombre del objeto en el interior del gráfico rectangular que aparece. Una *e*, por ejemplo, crea un editor mediante el cual es posible comunicar datos alfanuméricos a otro objeto, o bien visualizar los datos que ingresan al mismo editor, provenientes de otro objeto. Si escribimos *b*, por otra parte, creamos un botón *bang*. El mensaje que emite este botón lleva el mismo nombre y sirve para que el objeto que lo recibe devuelva su contenido o dispare el resultado de una operación.

Objetos de control

Los objetos que conforman la librería se dividen en dos grupos: objetos de control –aquellos que responden a un mensaje particular, en general producido por el usuario- y los objetos de audio –objetos que producen datos a la frecuencia de muestreo, cada vez que es encendido el procesamiento digital de señales.

Objetos de interfaz con el usuario

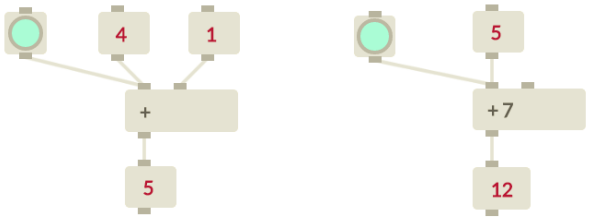
Los objetos de interfaz con el usuario son: *edit*, *bang*, *toggle* (botón que devuelve 0 ó 1), *slider* y *envgen* (editor gráfico de envolventes). Dado que son muy usados tienen alias para sus nombres (***e***: *edit*, ***b***: *bang*, ***g***: *toggle*, ***i*** : *slider*, ***v***, *envgen*)



01 – objetos de interfaz

Debemos considerar que los objetos de control que poseen más de un *inlet* realizan la operación para la que están programados en el momento en que un dato ingresa por el primer *inlet* de la izquierda. En el *patch* siguiente de suma, si modificamos el contenido del editor que ingresa por

la derecha al objeto, veremos que el resultado de la adición no cambia. Para que el resultado se refleje es necesario que luego ingrese otro dato por la izquierda (por el *hot inlet*), o bien que ese *inlet* reciba un mensaje *bang*.



02 - suma

El segundo ejemplo del *patch* introduce el concepto de argumento del objeto. Luego de crear un objeto vacío y de escribir el nombre del objeto (+, en este caso) introducimos un número, separado por un espacio. El valor numérico expresado funciona como valor por defecto, y su vida se prolonga hasta que ingrese un nuevo dato por el *inlet* de la derecha.

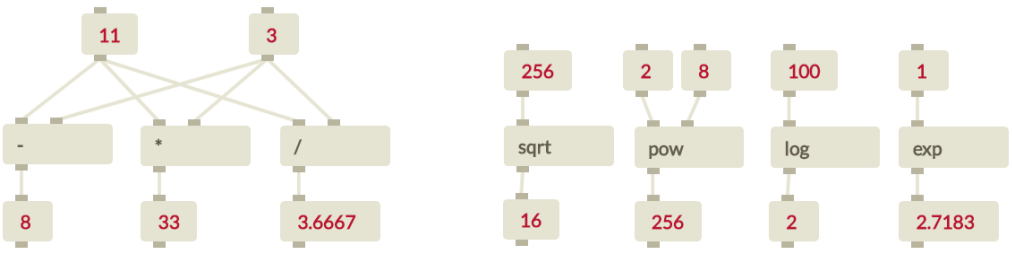
Algunos objetos de interfaz con el usuario admiten argumentos. En el caso del *slider* podemos especificar el rango y el paso. Si escribimos "slider 10 20 2", por ejemplo, el valor mínimo será 10, el máximo 20 y el resultado de moverlo con el mouse irá de dos en dos. Si no especificamos ningún valor, el rango va de 0 a 1, mientras que el paso es 0.01, lo cual es útil para el control de la amplitud en audio digital.

Se observa que cuando deseamos cambiar el valor de un *slider* el objeto se mueve en su conjunto. A fin de evitarlo podemos salir del modo edición presionando *Shift + u*, y regresar luego de la misma forma.

En el caso del objeto *envgen*, podemos especificar dos argumentos, que modifican el ancho y el alto en pixels del objeto (por defecto es de 115 y 50, respectivamente).

Operaciones matemáticas

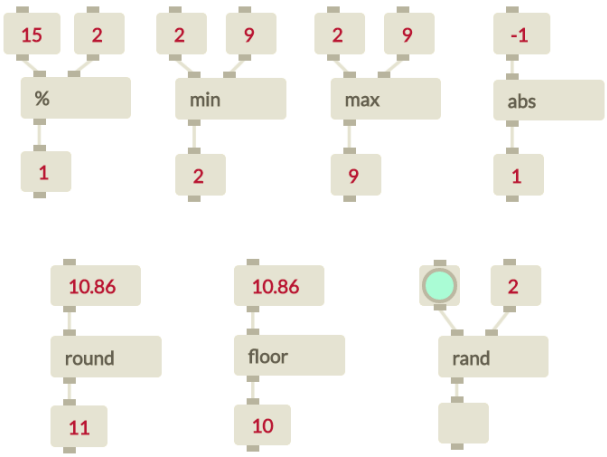
El siguiente *patch* muestra otros objetos de operaciones matemáticas: resta, multiplicación, división, raíz cuadrada, potenciación, logaritmo en base 10 y exponencial (e^x).



03 – otras operaciones

Las operaciones aritméticas de suma, resta, multiplicación y división también son aplicables a las señales de audio, a través de los objetos +~, -~, *~ y /~. Nótese en estos casos el agregado del símbolo "~", propio de los objetos que procesan audio. El objeto *~ de audio tiene otro análogo, denominado *amp~*.

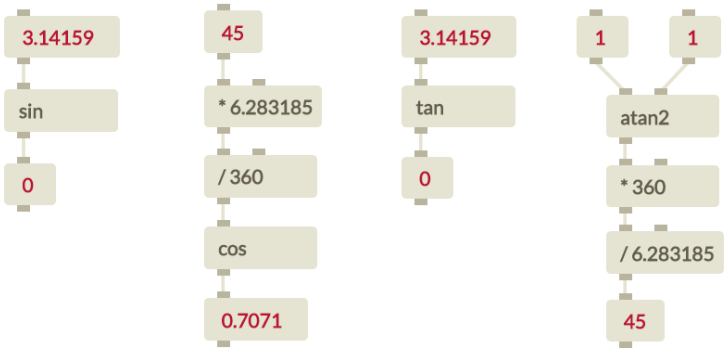
En el *patch* 04 observamos los objetos *%* (resto de una división entera), *min*, *max*, *abs* (valor absoluto), *round* (redondeo), *floor* (truncamiento) y *rand* (obtención de un número pseudoaleatorio entre 0 y $N - 1$). Todos ellos admiten un único argumento.



04 – otros objetos numéricos

Funciones trigonométricas

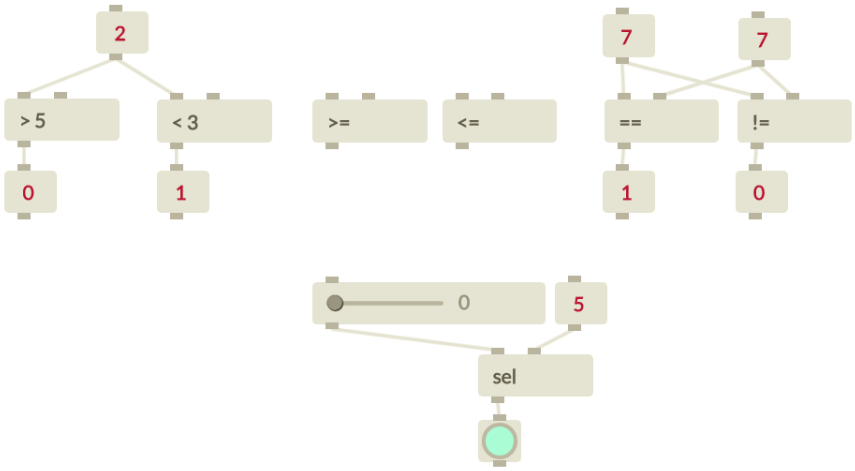
Los objetos que calculan los valores de las funciones trigonométricas son *sin*, *cos*, *tan* y *atan2*. Este último devuelve el arcotangente, en radianes, de y/x (coordenadas, especificadas en ese orden).



05 – operaciones trigonométricas

Operadores relacionales

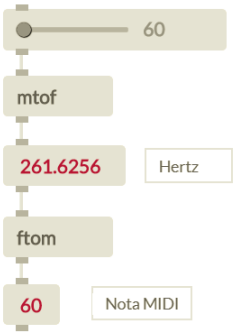
Los operadores relacionales sirven para realizar comparaciones entre números. El objeto *sel*, por ejemplo, detecta el ingreso de un número determinado (especificado en su *inlet* derecho o a través de un argumento) devolviendo un mensaje *bang*.



06 – operadores relacionales

Operadores de conversión

Los objetos *mtof* y *ftom* convierten valores de notas MIDI a frecuencia (Hz), y viceversa.



07 – conversión

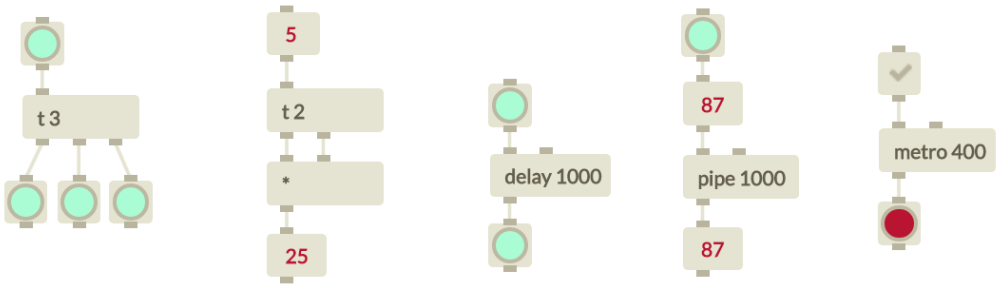
Objetos de control temporal

El objeto *t* (*trigger*) recibe un *bang* y lo devuelve tantas veces lo especifique su argumento, comenzando por el *outlet* de la derecha y terminando por la izquierda. El segundo ejemplo del *patch* ilustra uno de sus posibles usos: determinar el orden de llegada de los datos. Según se observa, multiplicamos un número por sí mismo, pero si ese número ingresa primero por la

izquierda (*hot inlet*), antes de acumularse por la derecha, el resultado que arroja el objeto es erróneo (5 * 0 = 0).

El objeto *delay* sirve para retrasar la salida de los *bangs* que ingresan, mientras que *pipe* hace lo propio con números.

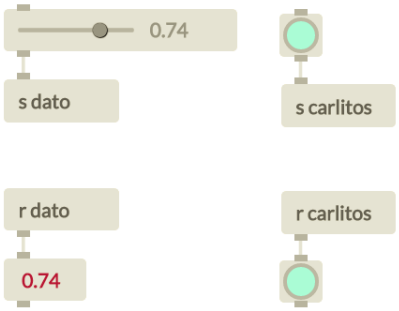
Finalmente, el objeto *metro* actúa como un metrónomo: envía *bangs* periódicamente. El período está determinado por un argumento o un número que ingresa por el *inlet* derecho.



08 – control temporal

Conexiones remotas

Los objetos *s* (*send*) y *r* (*receive*) evitan el cruce innecesario de cables sobre los objetos. A fin de conectar un envío con un receptor, ambos deben tener la misma palabra escrita como argumento.

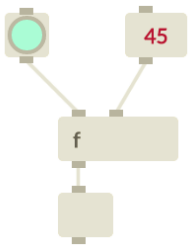


09 – conexiones remotas

Las señales de audio también admiten conexiones remotas, a través de los objetos *s~* y *r~* (*send* y *receive* de audio).

Otros objetos de control

El objeto *f* (*float*) almacena un número que ingresa por el *inlet* derecho, y lo devuelve cuando recibe un *bang* por el izquierdo.



10 – variables

El objeto *print* imprime lo que ingresa en su inlet en la consola del navegador de Internet.

Para escribir comentarios creamos un objeto vacío y escribimos *c*.

lbang envía un *bang* en el momento en el que el *patch* es cargado. De este modo se pueden configurar partes de programa al momento del arranque.

pack empaqueta valores en una lista de datos; tantos como se especifique en su argumento. Sirve al emplear objetos que utilizan listas como parámetros, como es el caso de *line*.

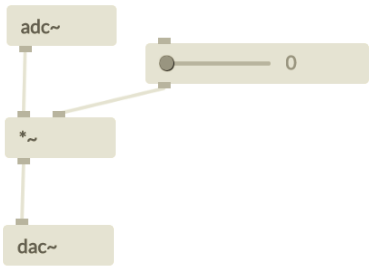
11 – otros objetos de control

Objetos de audio

Los objetos de procesamiento de audio se caracterizan por el uso del tilde (~) al final del nombre.

Conversores

La entrada de y salida de audio del *patch* se realiza mediante los objetos *adc~* (conversor analógico-digital) y *dac~* (conversor digital-analógico), respectivamente. La entrada de señales sólo es posible si la aplicación se ejecuta desde un servidor seguro (*https*). De otro modo, el mismo navegador de Internet lo impide, por cuestiones de seguridad. En el siguiente ejemplo se ilustra la captura de audio a través de un micrófono o por línea (de acuerdo a la configuración por defecto del sistema operativo) y la reproducción a través de parlantes, con control de la amplitud mediante un *slider*. Se recomienda tener especial cuidado en el control de la amplitud, especialmente si se emplea un micrófono *on-board*, dado que produce realimentación con los parlantes del ordenador.



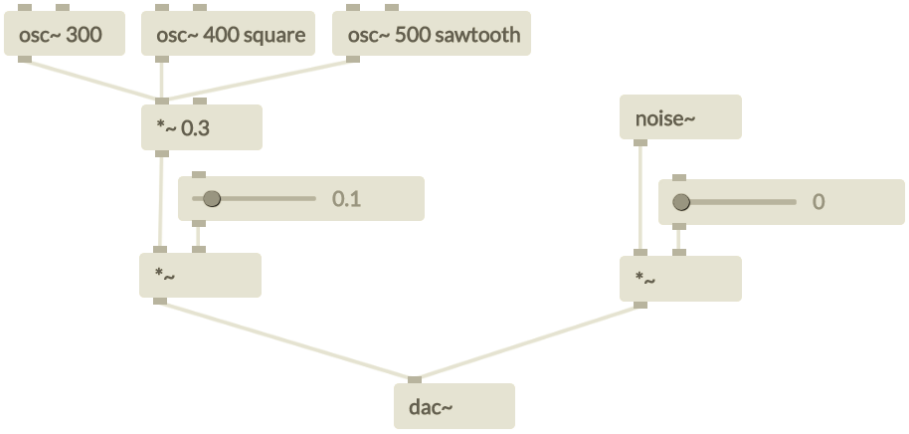
12 – conversores

Generadores de señales

El objeto *osc~* es un oscilador. A través del primer *inlet* se determina la frecuencia de oscilación, y la desafinación en *cents* mediante el segundo *inlet*. Los argumentos válidos del objeto son dos: frecuencia y tipo de forma de onda. Las formas de onda reconocidas son:

sine	onda sinusoidal. Es la elegida por defecto
square	onda cuadrada
sawtooth	onda diente de sierra
triangle	onda triangular

El *patch* muestra, además, el empleo de un generador de ruido blanco.



13 – generadores de audio

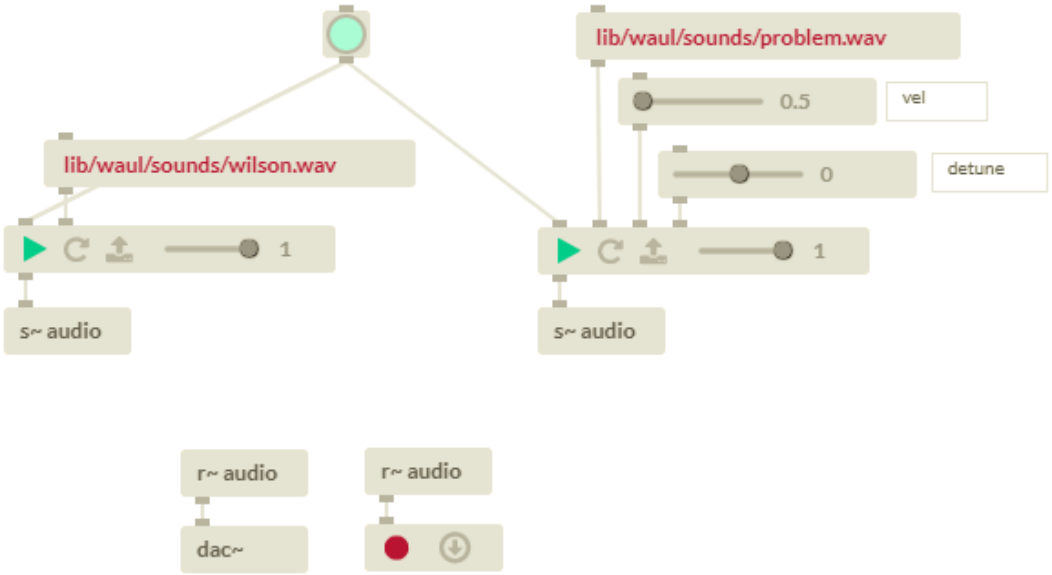
El objeto *k~* sirve para generar una señal constante, cuyo valor de amplitud es el que se especifica en su *inlet*.

Reproducción y grabación de audio

Los objetos *sfplay~* y *bplay~* sirven para la reproducción de archivos de sonido en formato *wav* o *mp3*. Los archivos a reproducir pueden encontrarse tanto en el servidor como en el ordenador del usuario. Si el archivo se aloja en el servidor, se accede al mismo a través de un mensaje que especifica la ruta, conectado al segundo *inlet*. Por el contrario, si se encuentra en la PC del usuario, se accede a través de un explorador de archivos, presionando el botón ubicado sobre el objeto.

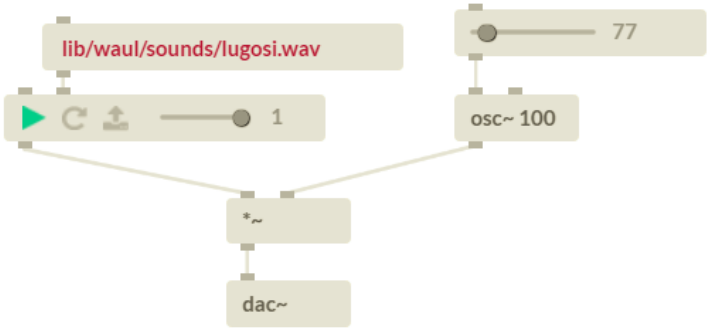
El objeto *sfplay~* lee el archivo directamente desde el soporte de datos, mientras que *bplay~* lo carga en un *buffer*. El segundo objeto se destina, en general, a sonidos cortos y de uso repetitivo. Ambos objetos tienen su propio control de amplitud y un botón para la reproducción continua (*loop*). *bplay~*, por otra parte, permite el cambio de velocidad de lectura del archivo y su desafinación en cents (*detune*).

El objeto *sfrecord~*, por otra parte, permite grabar las señales de audio en un archivo *.wav*, en la computadora del usuario. El nombre del archivo que se descarga al finalizar la operación es por defecto *output.wav*.



14 – reproducción y grabación de audio

Otro ejemplo de utilización de los lectores de archivos de audio lo vemos en el ejemplo 15. La señal de audio registrada en un archivo es multiplicada por una señal sinusoidal. Se produce así un aumento en la cantidad componentes del sonido reproducido y un corrimiento en sus frecuencias, efecto al que denominamos convolución espectral. Las frecuencias resultantes surgen de sumar y restar la frecuencia de la señal sinusoidal con cada uno de los parciales del sonido grabado. Este proceso también es conocido como modulación en anillo, debido al tipo de circuito que se utilizaba en los sintetizadores analógicos para producirlo.

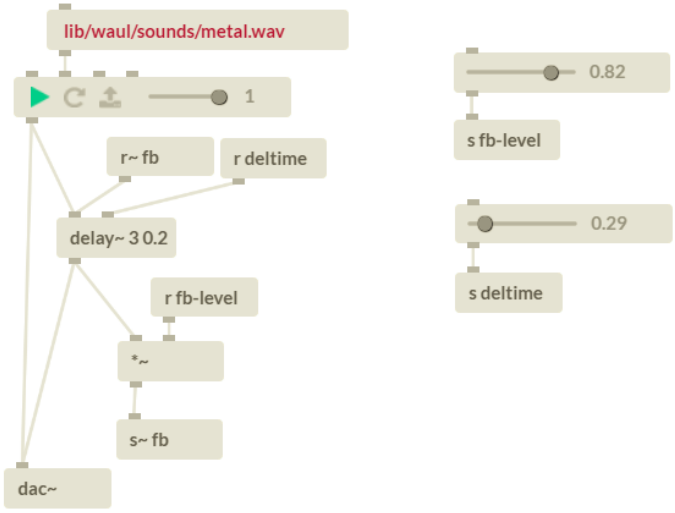


15 – convolución espectral

Retardos

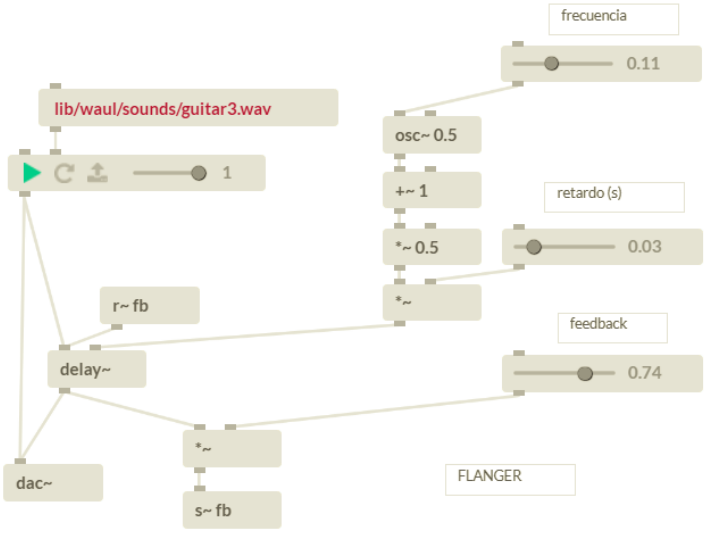
Para efectuar retardos en las señales utilizamos el objeto *delay~*. Sus argumentos son dos: máximo tiempo de retardo y retardo efectivo, expresados en segundos. El retardo efectivo (retardo real que se aplica a la señal) se puede especificar también como parámetro, a través del segundo *inlet*. El máximo nivel de retardo debe anunciarse, dado que el sistema reserva

memoria suficiente para realizar el proceso, y su valor nunca debe ser superado por el retardo efectivo.



16 – retardo con realimentación

El siguiente ejemplo utiliza una línea de retardo para implementar un *flanger*. Emplea un pequeño retardo variable con realimentación, controlado por un oscilador, que conforma un filtro *comb* de retardo variable.



17 – flanger

Filtros

Hasta el momento se han implementado tres filtros. Dos muy básicos, *zero~* y *pole~*, filtros de un cero y de un polo, respectivamente, y el filtro bicuadrático *biquad~*.

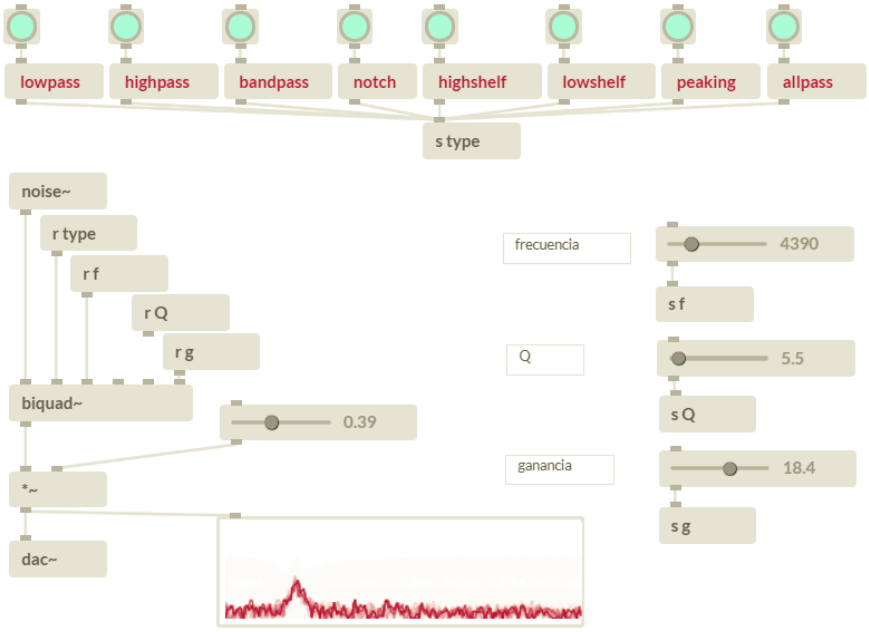
A fin de observar el espectro podemos utilizar el objeto *spect~*, que realiza el análisis espectral de señales.

Los ejemplos 18 y 19 muestran los filtros básicos. A los objetos se les agregó la posibilidad de aumentar el número de muestras de retardo, lo cual permite configurar otros tipos de filtros.

18 – filtro de un cero

19 – filtro de un polo

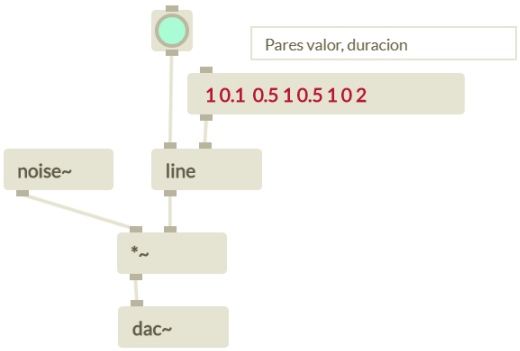
El filtro bicuadrático puede configurarse para obtener distintos tipos de respuesta: *lowpass*, *highpass*, *bandpass*, etc. El objeto *biquad~* admite cuatro argumentos; tipo de filtro, frecuencia de corte o central, ancho de banda (*Q*), y ganancia. Los *inlets* corresponden a 1) señal a filtrar; 2) tipo de filtro; 3) frecuencia de corte; 4) desafinación de la frecuencia de corte en cents; 5) valor de Q y 6) ganancia.



20 – filtro bicuadrático

Generadores de envolventes

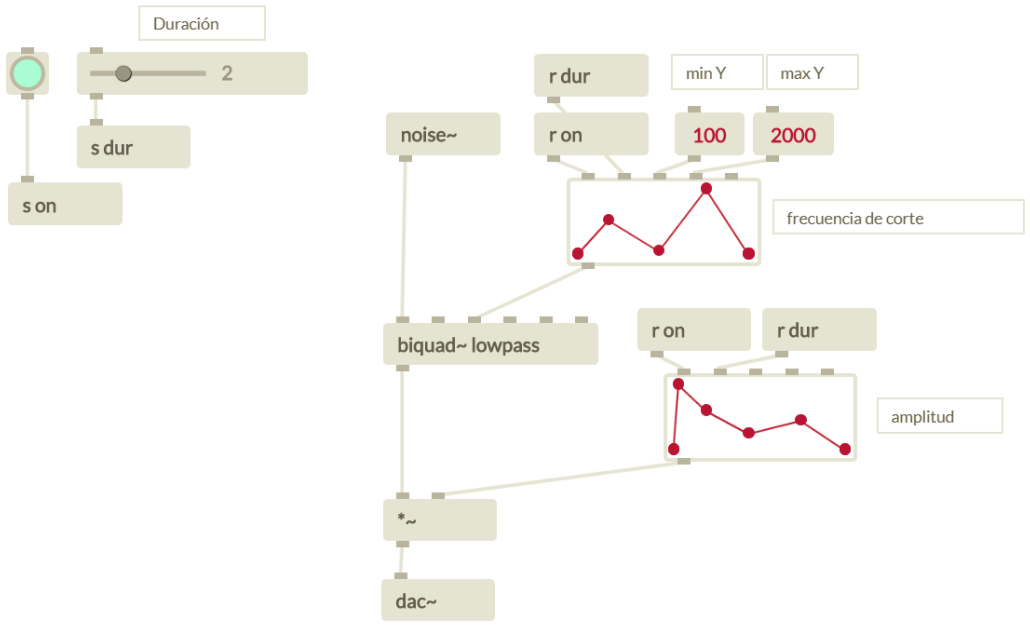
Las envolventes son funciones que permiten modular distintos parámetros en los objetos. La más utilizada es la envolvente dinámica, que nos permite controlar la amplitud de una señal en función del tiempo.



21 – envolvente con line

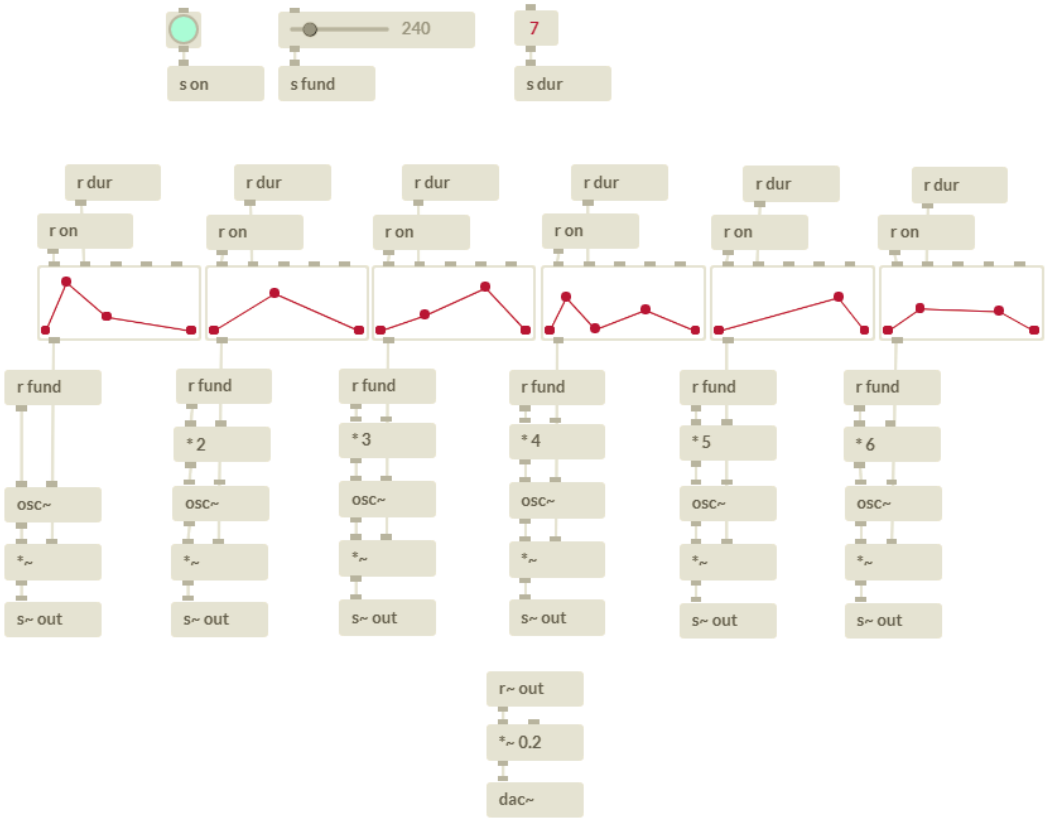
A fin de facilitar el diseño de las envolventes es posible utilizar el objeto *envgen*, ya comentado anteriormente. Los argumentos son ancho y alto en píxeles, y los parámetros, *bang* para disparo de la envolvente (*inlet* 1), duración en segundos de la envolvente (*inlet* 2), valor mínimo del eje *y* (*inlet* 3), valor máximo del eje *y* (*inlet* 4) y tipo de curva (*inlet* 5), que puede ser lineal (valor 0) o exponencial (valor 1).

En el *patch* siguiente *envgen* es aplicado al control de la frecuencia de corte de un filtro pasa-bajos, y luego como control de amplitud de la salida.



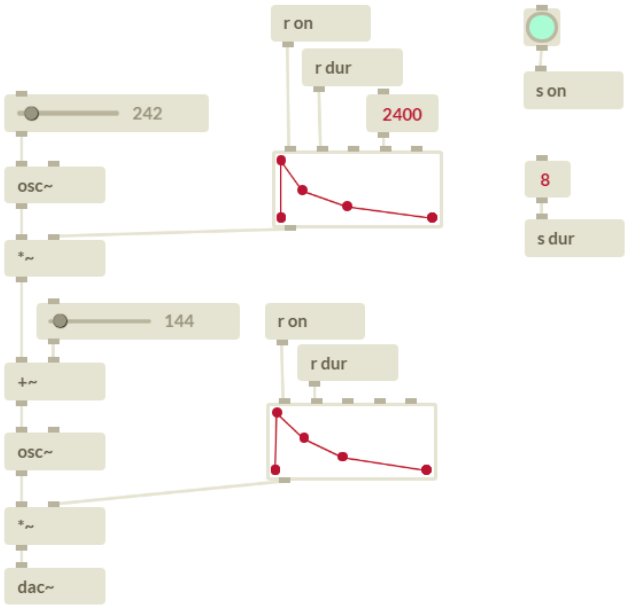
22 – envolvente con envgen

A continuación observamos el uso de envolventes gráficas en dos ejemplos de síntesis. El *patch* 23 implementa una síntesis aditiva elemental, de tan solo seis armónicos.



23 – ejemplo de síntesis aditiva

El *patch* 24 muestra un ejemplo de síntesis por frecuencia modulada.

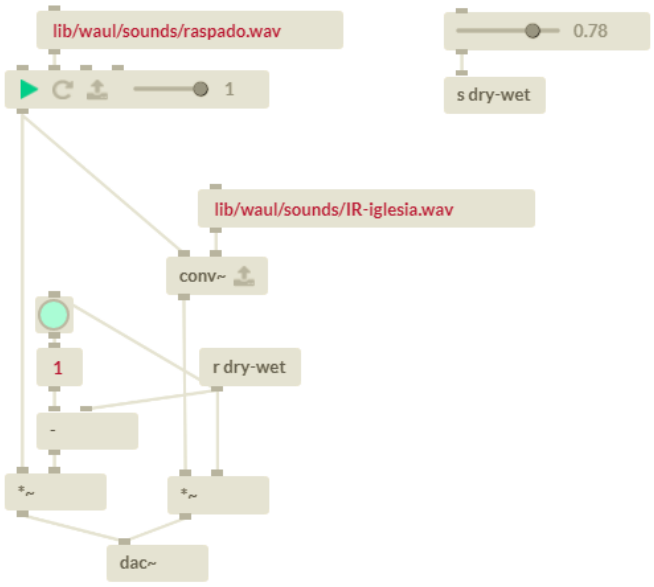


24 – FM simple

Convolución

La convolución de formas de onda suele emplearse en síntesis cruzada y en la generación de reverberación natural, es decir, aquella creada a partir de una respuesta a impulso tomada de un recinto existente.

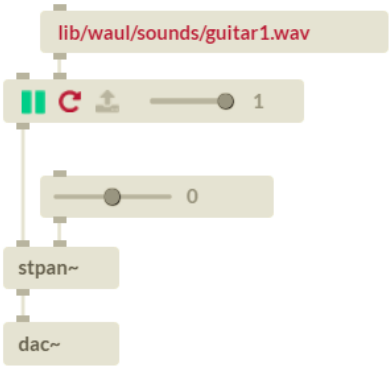
El *patch* 25 es un ejemplo de reverberación aplicada a una señal almacenada en un archivo. El objeto *conv~* recibe en su segundo *inlet* la ruta y nombre de la respuesta a impulso. El archivo de respuesta a impulso puede también ser cargado desde el ordenador del usuario, presionando el botón que se ubica en el objeto. El *slider* ubicado a la derecha del *patch* controla la relación entre sonido directo y reverberado, lo cual actúa como indicador de la distancia entre la fuente sonora y el sujeto que la percibe.



25 – reverberación natural

Espacialización

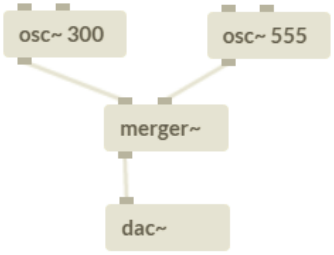
La Web Audio API propone diferentes procedimientos para contribuir a la localización espacial del sonido. Por el momento hemos implementado el paneo estereofónico. En el *patch* 26 se puede observar que, a diferencia de otros lenguajes de procesamiento, la señal estéreo no está representada por dos cables, sino por uno solo. Una conexión de audio puede estar conformada por múltiples canales de audio. Al objeto dac~ siempre ingresa una única conexión, que puede ser mono, estéreo o multicanal.



26 – paneo esteorofónico

Conformación de una señal estéreo a partir de dos señales monofónicas

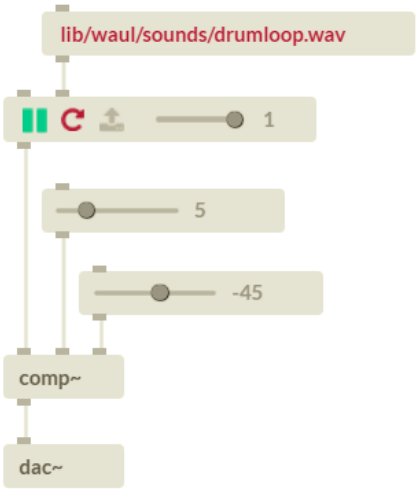
En el siguiente ejemplo vamos a utilizar dos osciladores monofónicos y a destinar cada uno de ellos a un canal distinto de una señal estéreo. Para ello, nos valemos del objeto *merger~*.



27 – conversión a estéreo

Compresión

El objeto *comp~* permite la compresión dinámica de señales de audio, con el objeto de prevenir distorsiones o *clipping*. Admite hasta 5 argumentos, que son *ratio*, *threshold*, *knee*, *attack* y *release*. Los valores por defecto son 12 , -24, 30, 0.003, y 0.25, respectivamente. Por el primer *inlet* ingresa la señal a comprimir, mientras que los dos restantes reciben los parámetros *ratio* (cantidad de dB que se precisan a la entrada para que la señal cambie 1 dB a la salida, entre 1 y 20) y *threshold* (cantidad de dB a partir de la cual la compresión comienza a actuar, entre -100 y 0).



28 – compresión de una señal

Abstracciones

Las abstracciones son *patches* almacenados en el servidor, que pueden ser incluidos dentro de otro *patch*. Para incluir una abstracción creamos un objeto nuevo y escribimos *p*, luego un espacio, y el nombre con el cual figura en la lista del menú (ver *Descarga e instalación* para entender la diferencia entre nombre real de un archivo y nombre asociado).

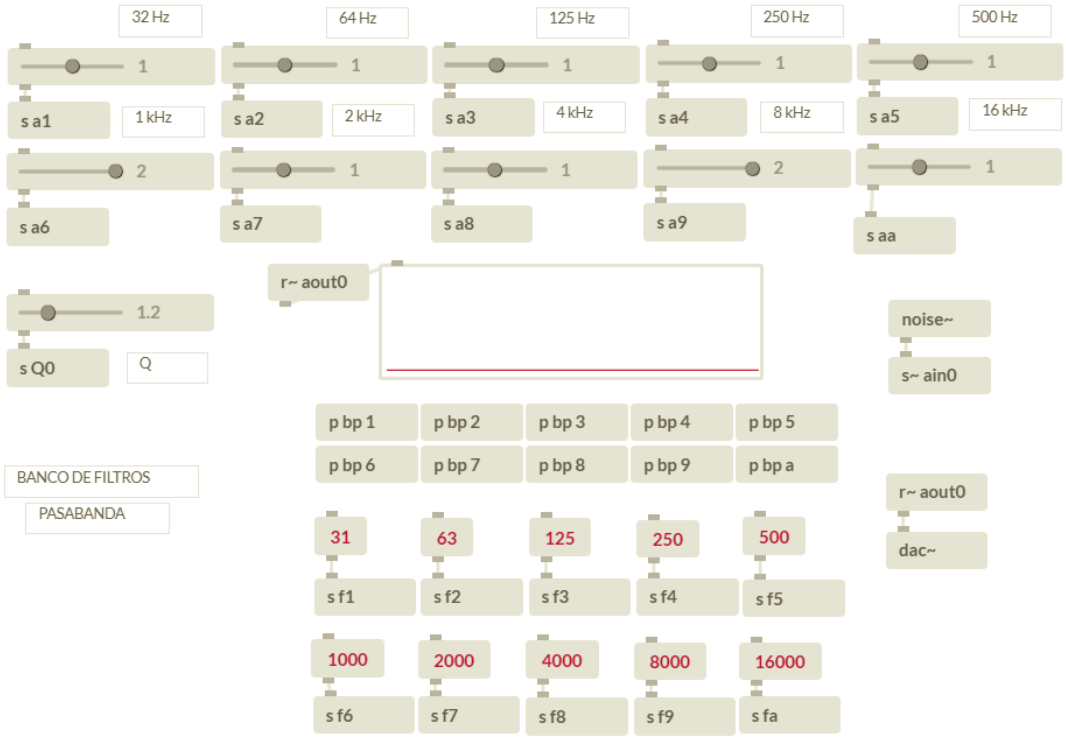
Los objetos que representan a las abstracciones admiten un argumento extra, formado por un número o una letra. Si en el interior del nuevo objeto escribimos *p abst 4*, por ejemplo, significa que llamamos a una abstracción guardada en el servidor, y que identificamos a esa instancia con el número 4. El número de instancia nos sirve para enviar mensajes a ese objeto en particular y no a otros iguales que hayamos incluido.

En las abstracciones, a los nombres propios que ponemos en las conexiones remotas -luego de los nombres de objeto *s, r, s~* o *r~*- se les agrega automáticamente el identificador de instancia que usamos al crear la abstracción. Si, por ejemplo, tenemos dentro de una abstracción un objeto *r resultado*, al especificar el identificador de instancia 4, se convierte internamente en *r resultado4*. De ese modo, desde el *patch* que contiene a nuestra abstracción, nos dirigimos a ella a través de *s resultado4.*

Por otra parte, si especificamos números de instancia en varias abstracciones iguales, pero deseamos enviar un mensaje común a todas ellas, escribimos un cero al final del nombre de variable de la conexión remota. Por ejemplo, *r frec0* en la abstracción y *s frec0* en el *patch* contenedor. De este modo, todos los nombres de variable que terminan con *0* (cero), reciben datos de manera global, e independiente del número de instancia. Estos casos pueden observarse en los *patches* 29 y 30.

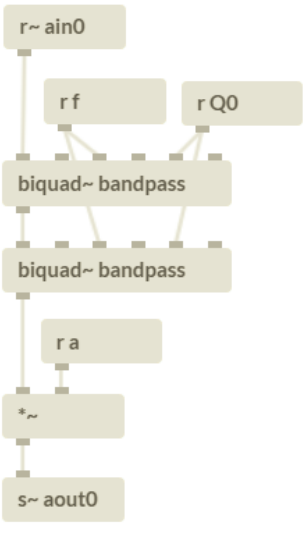
Las abstracciones en *WAUL* solo admiten un único nivel de profundidad. Vale decir, no es posible crear abstracciones dentro de otras abstracciones.

A continuación, un banco de filtros pasabandas de *Q* variable, en el que cada filtro está encapsulado en una abstracción. Según se observa, en las instancias de la abstracción *bp* se agregaron números o letras identificatorias. El envío remoto *s f1* se dirige sólo a la instancia *1*, el *s f2* a la instancia *2*, y así sucesivamente, pero dentro de la abstracción estos números no aparecen, solamente vemos *r f*.



29a – uso de abstracciones. Banco de filtros

Como se ve en el gráfico siguiente, el valor de *Q* es común para todos los filtros, por lo cual se agregó un 0 en el nombre del *send* y del *receive* (*s Q0* y *r Q0*) a fin de que todas las instancias de la misma abstracción reciban el mismo valor, y el número de instancia no sea tenido en cuenta. Lo mismo ocurre con la entrada y salida de audio.

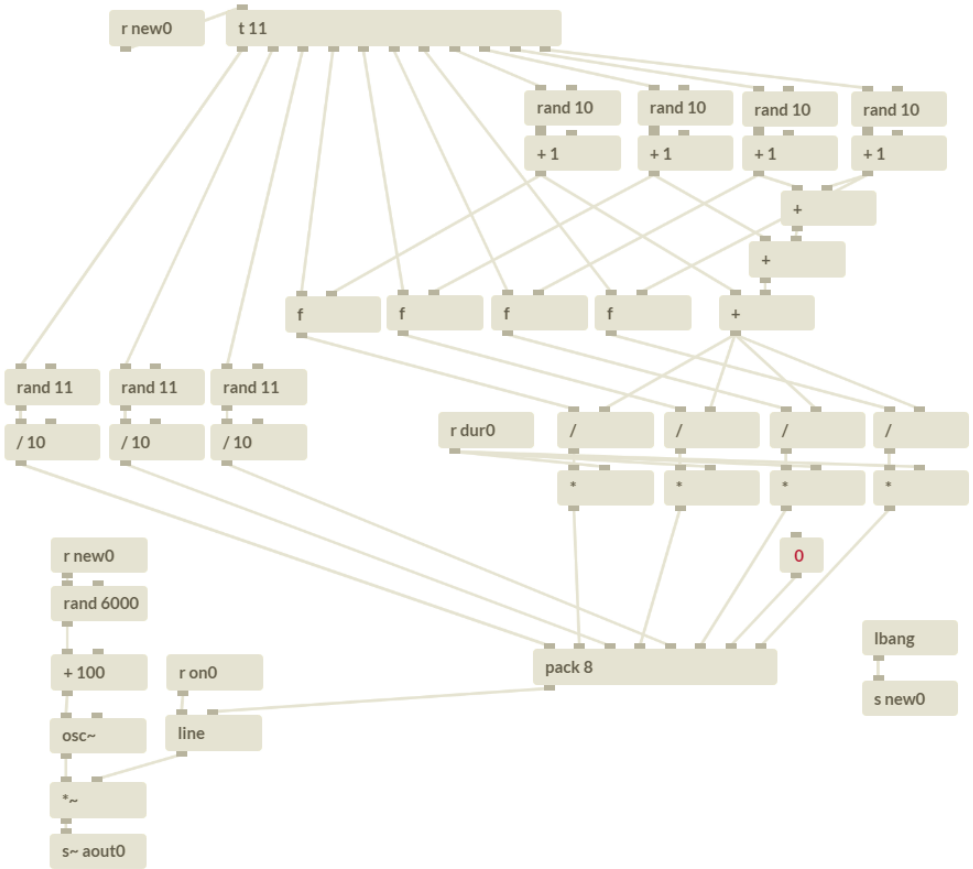


29b – Abstracción del banco de filtros

El siguiente ejemplo presenta un *patch* de síntesis aditiva en el cual los valores de frecuencia de las componentes, y las envolventes dinámicas son generadas de manera aleatoria.

30a – uso de abstracciones. Síntesis aditiva aleatoria

Cada oscilador es representado a través de una abstracción.

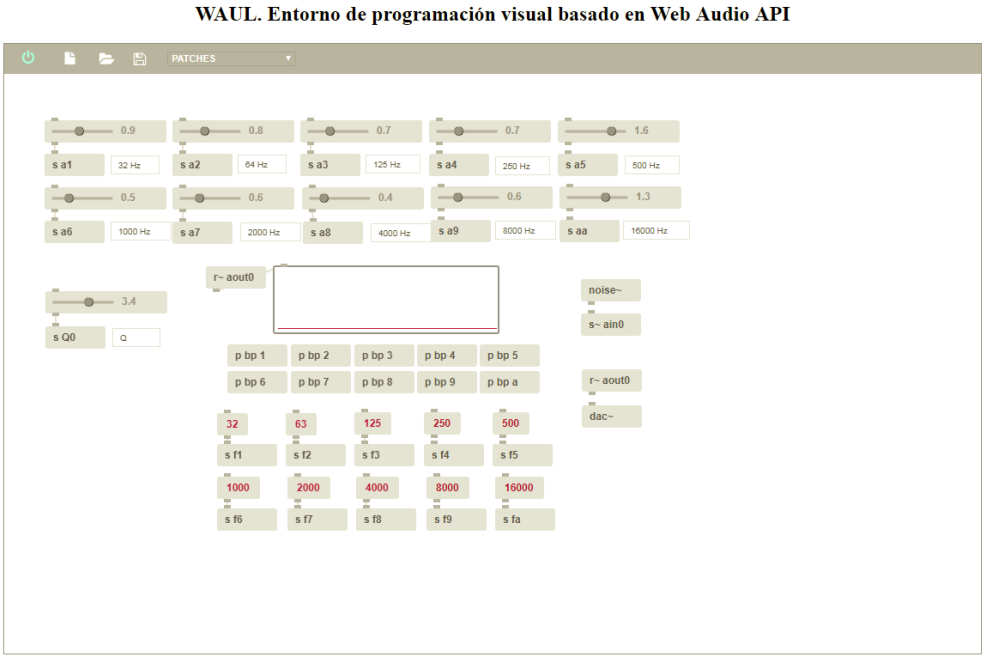


30b – Abstracción de ejemplo de síntesis aleatoria

Descarga e instalación

Descargue la instalación desde https://www.pablocetta.com/waul.php.

Descomprima el archivo *.zip* y copie el contenido en una carpeta de su servidor. A modo de ejemplo: www.pablocetta.com/waul. Al ingresar la URL en el navegador deberá aparecer la aplicación con la siguiente pantalla (ver en *www.pablocetta/waul*).



Página de inicio

En el archivo *index.html* encontrará el elemento *div* que contiene a la ventana de programación. Si deseamos que al inicio la ventana de programación cargue un *patch* de forma automática, debemos incluir en

```
<div id="waulcanvas" data-title="random-aditiva.wau">
```

el nombre del archivo alojado en el servidor, luego del atributo *data-title*. Si borramos ese atributo la ventana se iniciará en blanco.

```
<!DOCTYPE html>
<html>
    <head>
        <meta name="author" content="Pablo Cetta">
        <meta charset="UTF-8">
        <title>WauLib</title>
        <link rel="stylesheet" type="text/css" href="css/style.css">
        <link rel="stylesheet" type="text/css" href="lib/waul/css/waul.css">
        <script src="lib/p5/p5.min.js"></script>
        <script src="lib/p5/p5.dom.js"></script>
        <script src="lib/waul/js/waul.js"></script>
    </head>
    <body ondragstart="return false;" ondrop="return false;">
        <br><br>
        <h2 style="text-align:center">WAUL. Entorno de programación visual basado en Web Audio API</h2>
        <div id="waulcanvas" title="29-banco_de_filtros.wau">
            <!-- Esta es el área de programación -->
        </div>
    </body>
</html>
```

Contenido del archivo index.html

Modificando el archivo *style.css* podrá configurar la posición y tamaño de la ventana de programación.

```
#waulcanvas{
    margin:0 auto;
    width: 90%;
    min-width:330px;
    max-width:1199px;
    height:710px;
    border: 1px solid #999;
}
```

Contenido del archivo style.css

Se recomienda la utilización de WAUL con versiones actuales del navegador Google Chrome, dado que aún no ha sido probado en otros navegadores.

Si bien un sitio puede contener varios entornos de programación en WAUL, Web Audio Api sólo permite un entorno por cada página web.

En la carpeta *config* encontramos dos archivos en formato *json*. El archivo *patches.json* vincula los nombres reales de los archivos con los nombres que deseamos que aparezcan en la lista de *patches* del menú de la ventana. Es importante entender que cuando llamamos a una abstracción no lo hacemos a través del nombre del archivo en el servidor, sino por su nombre asociado (el que aparece en el menú).

```
{
"p" : [
    {
    "name" : "PATCHES",
    "file" : ""
    },
    {
    "name" : "01 - Objetos de interfaz",
    "file" : "01-interfaz.wau"
    },
    {
    "name" : "02 - Suma",
    "file" : "02-suma.wau"
    },
    {
    "name" : "03 - Otras operaciones",
    "file" : "03-otras_ops.wau"
    },
```

Parte del contenido del archivo patches.json

Shortcuts

Shift + 1	Crear objeto
Del – Backspace	Borrar objeto seleccionado
Shift + click	Seleccionar objeto
Shift + a	Seleccionar / deseleccionar todo
Shift + d	Duplicar objeto
Shift + flecha izquierda	Mover objeto seleccionado (lento)
Shift + flecha derecha	
Shift + flecha arriba	
Shift + flecha abajo	
Shift + Control + flecha izquierda	Mover objeto seleccionado (rápido)
Shift + Control + flecha derecha	
Shift + Control + flecha arriba	
Shift + Control + flecha abajo	
Shift + u	Bloquear/desbloquear edición
Shift + e	Encender / detener procesamiento de audio

Guía de referencia

OBJETOS DE CONTROL

Objetos matemáticos

+	Suma de dos números	
	1	**2**
argumentos	sumando 2	
inlets	sumando 1	sumando 2
valores por defecto	0	0

-	Resta de dos números	
	1	**2**
argumentos	sustraendo	
inlets	minuendo	sustraendo
valores por defecto	0	0

*	Multiplicación de dos números	
	1	**2**
argumentos	multiplicador	
inlets	multiplicando	multiplicador
valores por defecto	0	0

/	División de dos números	
	1	**2**
argumentos	divisor	
inlets	dividendo	divisor
valores por defecto	0	0

%	Resto de la división entera	
	1	**2**
argumentos	divisor	
inlets	dividendo	divisor
valores por defecto	0	0

abs	Valor absoluto
	1
argumentos	número
inlets	número
valores por defecto	0

floor	Truncamiento
	1
argumentos	número
inlets	número
valores por defecto	0

round	Redondeo
	1
argumentos	número
inlets	número
valores por defecto	0

min	Mínimo entre dos números	
	1	**2**
argumentos	número 2	
inlets	número 1	número 2
valores por defecto	0	0

max	Máximo entre dos números	
	1	**2**
argumentos	número 2	
inlets	número 1	número 2
valores por defecto	0	0

sqrt	Raíz cuadrada
	1
argumentos	número
inlets	número
valores por defecto	0

pow	Potenciación	
	1	**2**
argumentos	exponente	
inlets	base	exponente
valores por defecto	0	0

log	Logaritmo en base 10
	1
argumentos	número
inlets	número
valores por defecto	0

exp	Exponencial
	1
argumentos	exponente
inlets	exponente
valores por defecto	0

sin	seno
	1
argumentos	ángulo (rad)
inlets	ángulo (rad)
valores por defecto	0

cos	coseno
	1
argumentos	ángulo (rad)
inlets	ángulo (rad)
valores por defecto	0

tan	tangente
	1
argumentos	ángulo (rad)
inlets	ángulo (rad)
valores por defecto	0

atan2	arcotangente de *y / x*	
	1	**2**
argumentos	*x*	
inlets	*y*	*x*
valores por defecto	0	0

rand	Generador de números pseudoaleatorios entre 0 y *N-1*	
	1	**2**
argumentos	*N*	
inlets	*bang*	*N*
valores por defecto		1

Operadores relacionales

>	Mayor que	
	1	**2**
argumentos	número 2	
inlets	número 1	número 2
valores por defecto	0	0

>=	Mayor o igual que	
	1	**2**
argumentos	número 2	
inlets	número 1	número 2
valores por defecto	0	0

<	Menor que	
	1	**2**
argumentos	número 2	
inlets	número 1	número 2
valores por defecto	0	0

<=	Menor o igual que	
	1	**2**
argumentos	número 2	
inlets	número 1	número 2
valores por defecto	0	0

==	Igual que	
	1	**2**
argumentos	número 2	
inlets	número 1	número 2
valores por defecto	0	0

!=	Distinto que	
	1	**2**
argumentos	número 2	
inlets	número 1	número 2
valores por defecto	0	0

sel	Igual que	
	1	**2**
argumentos	número a coincidir	
inlets	número a probar	número a coincidir
outlet	Envía un *bang* cuando hay coincidencia	
valores por defecto	0	0

Objetos de conversión

mtof	Nota MIDI a frecuencia
	1
argumentos	Nota MIDI
inlets	Nota MIDI
valores por defecto	0

ftom	Frecuencia a nota MIDI
	1
argumentos	Frecuencia (Hz)
inlets	Frecuencia (Hz)
valores por defecto	0

Objetos de control temporal

t	Trigger. Recibe un bang y lo replica a través de sus outlets, de derecha a izquierda
	1
argumentos	Número de *outlets*
inlets	*bang*

metro	Metrónomo. Envía un bang cada N milisegundos	
	1	**2**
argumentos	Período en milisegundos	
inlets	0 = encender; 1 = apagar	Período en milisegundos
valores por defecto	0	250 ms

delay	Retarda la salida de un *bang* en *N* milisegundos.	
	1	**2**
argumentos	Retardo en milisegundos	
inlets	*bang*	Retardo en milisegundos
valores por defecto		250 ms

pipe	Retarda la salida de un número, en *N* milisegundos.	
	1	**2**
argumentos	Retardo en milisegundos	
inlets	número	Retardo en milisegundos
valores por defecto		250 ms

Objetos de interfaz con el usuario

slider / i	*Slider*		
	1	**2**	**3**
argumentos	valor mínimo	valor máximo	paso
inlets	número		
valores por defecto	0	1	0.01

edit / e	Caja de entrada/salida de datos alfanuméricos
	1
inlets	datos alfanuméricos

bang / b	Botón de recepción/envío de *bangs*
	1
inlets	*bang*

toggle / g	Botón de recepción/envío 0 ó 1
	1
inlets	*bang*

envgen / v	*Slider*				
	1	**2**	**3**	**4**	**5**
inlets	*bang* de disparo	Duración en segundos	Valor mínimo de *y*	Valor máximo de *y*	lineal: 0 exponencial: 1
valores por defecto		1 seg	0	1	0
argumentos	ancho en px	alto en px	paso		
valores por defecto	115	50	0.01		

Conexiones remotas

s	Envío remoto
	1
argumentos	Nombre de variable
inlet	Datos

r	Recepción remota
	1
argumentos	Nombre de variable
oulet	Datos

Variables

f	Almacenamientode variable numérica	
	1	**2**
argumentos	número	
inlets	*bang*	número a almacenar
valores por defecto		0

Otros objetos

print / p	Impresión en consola del navegador
	1
inlet	Datos

c	Comentarios

lbang	Envía un *bang* cuando se carga el *patch*
	1
oulet	*bang*

pack	Empaqueta números en una lista
	1
argumentos	Número de datos a empaquetar
inlets	Tantos como especifica el argumento
oulet	Lista de números

OBJETOS DE AUDIO

Operadores aritméticos

+~	Suma de dos señales o una señal y un número	
	1	**2**
argumentos	número	
inlets	señal 1	señal 2 o número
valores por defecto	0	0

-~	Resta dos señales o una señal y un número	
	1	**2**
argumentos	número	
inlets	señal 1	señal 2 o número
valores por defecto	0	0

***~**	Multiplica dos señales o una señal y un número	
	1	**2**
argumentos	número	
inlets	señal 1	señal 2 o número
valores por defecto	0	0

/~	Divide dos señales o una señal y un número	
	1	**2**
argumentos	número	
inlets	señal 1	señal 2 o número
valores por defecto	0	0

Conversores

adc~	Conversor analógico-digital
	1
inlets	Señal entrante

dac~	Conversor digital- analógico
	1
outlet	Señal saliente

Generadores de señales

osc~	Oscilador	
	1	**2**
argumentos	Frecuencia	Tipo de onda: *sine:* sinusoidal *square:* cuadrada *sawtooth:* diente de sierra *triangle:* triangular
valores por defecto	0	*sine*
inlets	Frecuencia	*Detune* (cents)
valores por defecto	0	0

noise~	Generador de ruido blanco
	1
outlet	Señal de ruido blanco

k~	Generador de señal constante
	1
argumento	Amplitud
inlet	Amplitud
valor por defecto	0
outlet	Señal constante

Generador de envolventes

line	Generador numérico de envolventes	
	1	**2**
inlets	*bang* de disparo	Lista de pares de números (valor, duración en seg.)
outlet	envolvente	

Retardos, filtros y convolución

delay~	Línea de retardo	
	1	**2**
argumentos	máximo retardo (seg)	Retardo efectivo (seg)
valores por defecto	3 seg	0 seg
inlets	señal a retardar	Tiempo de delay en segundos
outlet	señal retardada	

zero~	Filtro de un cero		
	1	**2**	**3**
inlets	señal a filtrar	ganancia, entre -1 y 1	Cantidad de muestras de retardo. Permite generar otros filtros si es distinta a 1.
valores por defecto		0	1

pole~	Filtro de un polo		
	1	**2**	**3**
inlets	señal a filtrar	ganancia, entre -1 y 1	Cantidad de muestras de retardo. Permite generar otros filtros si es distinta a 1.
valores por defecto		0	1

biquad~	Filtro bicuadrático					
	1	**2**	**3**	**4**	**5**	**6**
argumentos	tipo de filtro: *lowpass, highpass, bandpass, notch, highshelf, lowshelf, pealing, allpass*	frecuencia central o de corte	Q	ganancia		
inlets	señal a filtrar	tipo de filtro	frecuencia	*detune*	Q	ganancia
valores por defecto		*lowpass*	0	0	1	1

conv~	Convolución	
	1	**2**
inlets	señal a convolucionar	ruta y nombre del archivo de respuesta a impulso
controles	Botón para carga de respuestas a impulso grabadas por el usuario (archivos locales)	

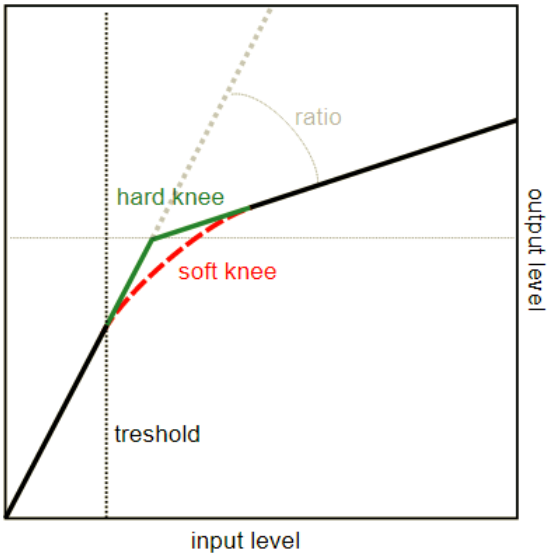
Espacialización

stpan~	Paneo estereofónico	
	1	**2**
inlets	señal a panear	posición de la fuente (-1 a 1)

Amplificación, compresión y mezcla

amp~	Amplificaciónde amplitud	
	1	**2**
argumentos	amplitud	
inlets	señal a amplificar	amplitud
valor por defecto		0

comp~	Compresor dinámico				
	1	**2**	**3**	**4**	**5**
argumentos	*ratio*	*threshold*	*knee*	*attack*	*release*
valores por defecto	12 (1 a 20)	-24 (-100 a 0)	30 (a a 40)	0.003 (0 a 1)	0.25 (0 a 1)
inlets	señal	*ratio*	*threshold*		



Parámetros del compresor

merger~	Conversión de dos señales mono a estéreo	
	1	**2**
inlets	señal L	señal R

Reproducción y grabación

sfplay~	Ejecuta archivos de audio desde el disco	
	1	**2**
inlets	*bang* de disparo	ruta y nombre de archivo a ejecutar
controles	1: botón de *play/ pause*; 2: botón de *loop*; 3:botón de carga de archivos locales; 4: control de amplitud	

bplay~	Ejecuta archivos de audio desde un *buffer*			
	1	**2**	**3**	**4**
inlets	*bang* de disparo	ruta y nombre de archivo a ejecutar	velocidad de reproducción	*detune* en cents
controles	1: botón de *play/stop*; 2: botón de *loop*; 3:botón de carga de archivos locales; 4: control de amplitud			

sfrecord~	Generador de señal constante
	1
inlet	Amplitud
controles	1: botón de *record / stop*; 2: botón de descarga de archivo grabado

Análisis spectral

spect~	Análisis espectral
	1
argumentos	tamaño de la FFT (256, 512 o 1024)
inlet	señal a analizar

Conexiones remotas

s~	Envío remoto de audio
	1
argumentos	Nombre de variable
inlet	señal

r~	Recepción remota de audio
	1
argumentos	Nombre de variable
outlet	señal