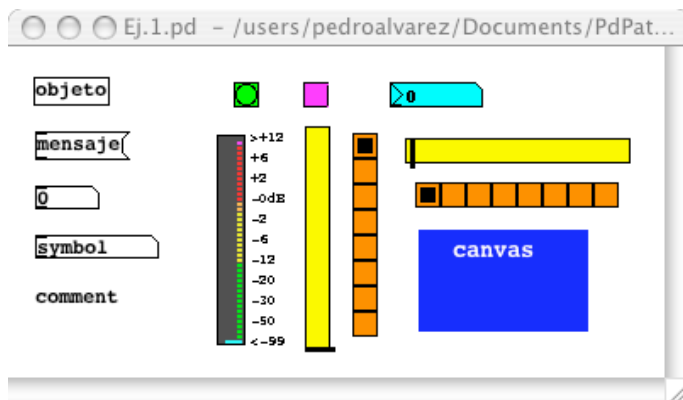


El programa Pd. (PureData)

Pedro Álvarez, Diciembre de 2005

Este programa, así como otros similares de la familia Max (Max/MSP, JMax), proveen un sistema de programación por objetos para el control de datos y audio en tiempo real. La ventaja de Pd es que, además de ser gratuito, funciona en múltiples plataformas (Windows, Linux y Macintosh). Para diseñar ciertos procesos contamos con una ventana denominada “patch” (literalmente “parche”) en la que podemos situar diferentes elementos cumpliendo determinadas funciones e interconectándolos entre sí. Estos elementos se agrupan en 5 categorías básicas: objetos, mensajes, números, símbolos y comentarios. Además existe una categoría especial de objetos que poseen una “interfaz gráfica del usuario” (GUI), permitiendo así controlarlos mediante el mouse y configurar su comportamiento y apariencia.

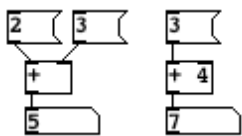


Ej. 1 Un patch con un objeto, un mensaje, un número, un símbolo, un comentario y varios objetos de interfaz gráfica GUI (en colores).

Los mensajes pueden contener cualquier palabra, número o lista que escribamos dentro de ellos, y los envía por su única salida al hacer click con el mouse sobre él. El contenido de un mensaje no cambia, no así los elementos número y símbolo, cuyo contenido puede cambiar según les enviemos mensajes o definamos nuevos valores en su interior.

Los objetos son los elementos del patch que cumplen una función específica según se los defina. Cada objeto específico tiene un número determinado de entradas (por arriba) por las que esperará recibir algún mensaje y salidas (por abajo) por las que envirá el resultado de la operación que realiza y puede tomar algún argumento a continuación de su nombre.

De este modo el objeto “+” realiza la suma del mensaje numérico recibido por cada una de sus dos entradas. Pero si a continuación del “+” escribimos un argumento numérico dentro del objeto, éste reemplazará por defecto al sumando correspondiente a la entrada derecha del objeto.

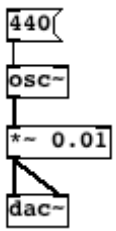


Ej. 2 El objeto “+” con y sin argumento.

En el ejemplo anterior así como en todos los objetos, el proceso que lleva a cabo el objeto en cuestión solo es activado al recibir el mensaje correspondiente por la entrada de la izquierda (llamada la entrada “caliente” por su propiedad de causar un resultado). Las demás entradas sólo definen los parámetros correspondientes, pero no producen resultados.

En Pd. encontramos dos tipos de objetos: de control y de señal. Los de control, por regla general tienen una tasa de muestreo de 1000 veces por segundo, es decir, el objeto “+” presentado anteriormente, podría arrojar hasta mil resultados por segundo. En cambio los objetos de señal, que se identifican por terminar

siempre con la figura “~” tienen una tasa de muestreo de 44.100 muestras por segundo, aunque esto puede variar según lo configuremos en las opciones de audio.

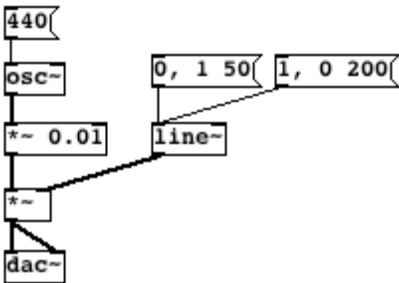


Ej. 3 El objeto “osc~” calcula una señal sinusoidal a la frecuencia que reciba por su entrada izquierda (la derecha modifica la fase). En este caso mediante el mensaje “440” obtenemos una senoide a 440 cps. La señal que sale de “osc~”, que por defecto posee la amplitud máxima en Pd = 1, la multiplicamos por 0,01 para reducir su amplitud y luego la enviamos al objeto “dac~”, que envía la señal que recibe por sus entradas a los respectivos canales de salida de la tarjeta de sonido de nuestro computador.

Nótese que los cables que conectan a los objetos de señal (aquellos entre osc~, *~ y dac~) son más gruesos que el que sale del mensaje. Esta diferencia nos muestra que unos conducen señal (actualizada 44.100 veces por segundo) y los otros conducen mensajes de control (actualizado sólo cuando producimos un mensaje, máximo 1000 veces por segundo). Evidentemente los objetos de señal y sus conexiones consumen bastante más procesamiento del computador que aquéllos de control. Hay que tener presente que, si bien a veces podemos conectar mensajes de control a objetos de audio, la salida de un objeto de audio siempre será una conexión de señal, y por tanto no podemos conectarla a un objeto de control.

Para que Pd. realice los procesos de audio que diseñemos en el patch, es necesario que activemos la opción “compute audio” en la ventana de Pd. (o podemos ctrl-/ en el teclado para encender y ctrl-. para apagar el audio).

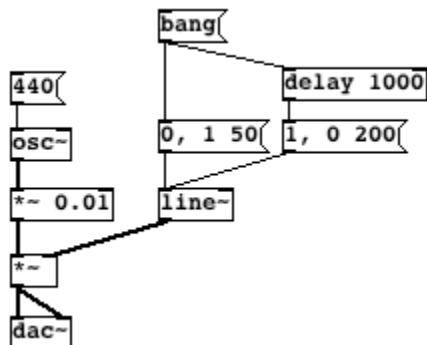
Envolventes



Ej. 4 Al mismo patch anterior le modificamos la señal antes de enviarla a “dac~”. El objeto “line~” crea rampas entre dos valores en un tiempo determinado, con una resolución de audio (44.100 s/s). Haciendo click en el mensaje “0, 1 50” le decimos que vaya del 0 al 1 en 50 milisegundos, con el otro, que vaya de 1 a 0 en 200 milisegundos. Al multiplicar nuestra señal sinusoidal por estas rampas logramos un efecto de “fade in” y “fade out” respectivamente.

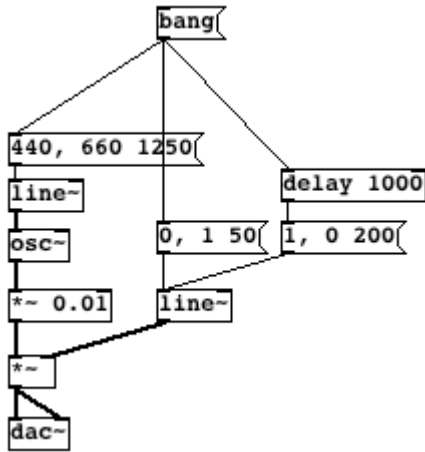
Nótese que ahora usamos un multiplicador de señal “*~” sin argumento numérico, ya que lo utilizamos para multiplicar dos señales de audio.

¿Cómo hacemos ahora para producir, a partir de un solo click del mouse, un sonido de duración definida con su ataque y decaimiento correspondiente? Al ejemplo anterior debemos añadirle una variable que cree un lapso de tiempo entre la aplicación del mensaje “0, 1 50”, nuestro ataque, y el “1, 0 200”, nuestro decaimiento del sonido.



Ej. 5 “bang” es un mensaje especial que gatilla la acción de un mensaje u objeto. En este caso, al hacer click en el mensaje “bang”, éste activa el mensaje “0, 1 50” y a su vez va al objeto “delay”, el cual retrasa este “bang” durante 1000 milisegundos para luego enviarlo al mensaje “1, 0 200”. De este modo, mediante un único click producimos una nota de duración total 1.250 milisegundos, de los cuales los primeros 50 son una rampa del 0 al 1 y los últimos 200, una rampa del 1 al 0.

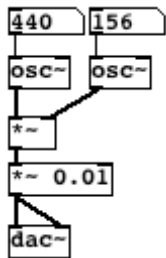
Ahora utilizaremos el mismo objeto line~ para producir un glissando de una 5ª justa.



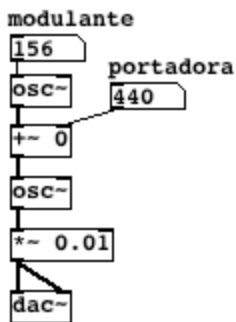
Ej. 6 Ahora el mismo “bang” que produce el ataque de la nota, activa también una rampa del 440 al 660 en los 1250 milisegundos que dura nuestra nota para definir la frecuencia del oscilador. Nótese que ahora el objeto “osc~” está recibiendo una señal para definir su frecuencia (este objeto puede recibir indistintamente mensajes de control o señal en su entrada izquierda).

Modulación

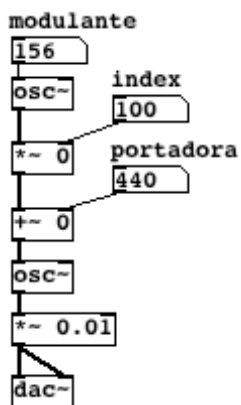
En audio digital, la modulación de amplitud se produce multiplicando dos señales



Ej. 7 Amplitud modulada multiplicando dos señales.

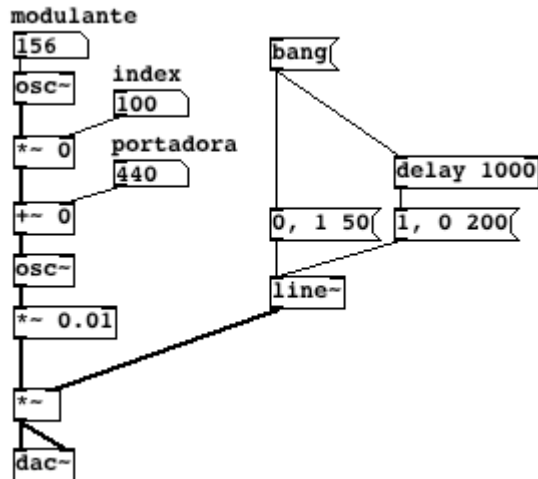


Ej. 8 Modulación de frecuencia. El oscilador de abajo calcula una onda sinusoidal cuya frecuencia central es 440 Hz. modulado por otra onda sinusoidal a 156 Hz. En este caso el grado de desviación de la frecuencia central es la amplitud por defecto del oscilador de arriba: 1. Es decir la frecuencia central se varía entre los 439 y 441 Hz. 156 veces por segundo ¿Cómo cambiamos eso?



Ej. 9 Al multiplicar la señal modulante, cambiamos su amplitud, por tanto modificamos el ámbito de desviación a partir de la frecuencia central. Aquí, al multiplicar por 100, producimos una onda cuya frecuencia varía 156 veces por segundo entre los 340 y los 540 Hz.

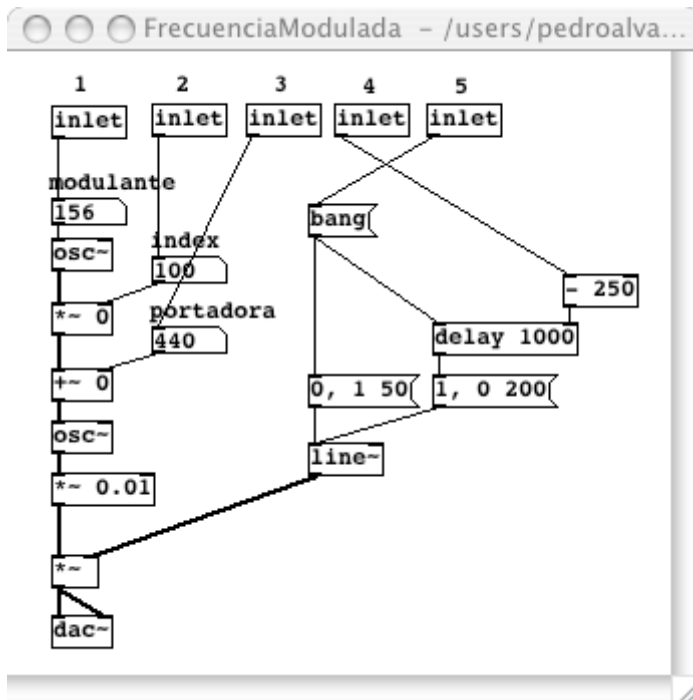
A cualquiera de estos ejemplos podemos agregarle una envolvente como vimos en el ejemplo 5.



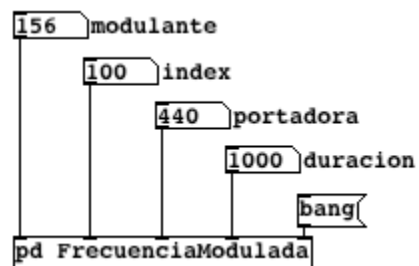
Encapsulamiento

Para una mejor optimización del espacio visual, así como para mantener los patches más ordenados podemos crear subpatches dentro de un patch que contengan parte de los procesos que diseñemos. Para crear un subpatch debemos crear un objeto llamado “pd” con un argumento arbitrario que servirá para identificar ese subpatch.

pd FrecuenciaModulada Pongamos en nuestro patch un objeto llamado pd con el argumento “FrecuenciaModulada”. Automáticamente se abre la ventana del subpatch creado con este nombre. Dentro de este subpatch podemos copiar el ejemplo anterior.

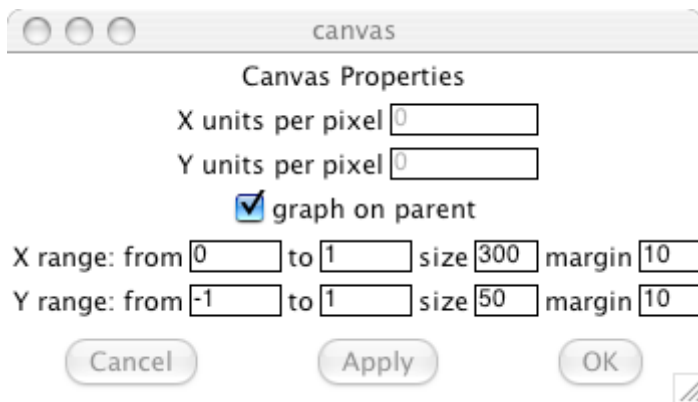


Para cada una de las variables creamos un objeto “inlet” cuya función es crear entradas en el objeto “pd FrecuenciaModulada” que se comunican con la salida del respectivo objeto “inlet”.

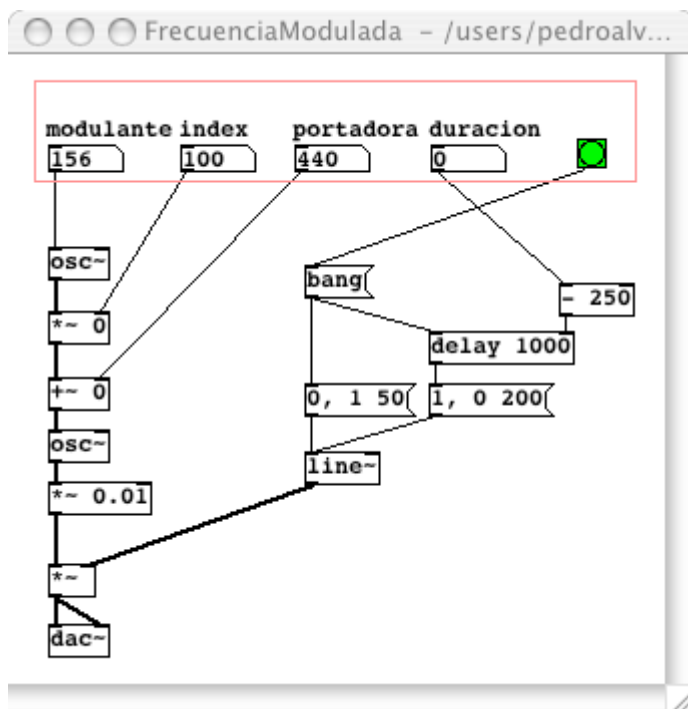


De este modo podemos tener a la vista, en el patch principal sólo lo estrictamente necesario para controlar las características del sonido que queramos; el resto estará dentro del subpatch. Al guardar el patch principal éste incluirá todos los subpatches que contenga.

Otra opción es permitir al subpatch mostrar los objetos gráficos en el patch principal; para esto debemos hacer click con el botón derecho en cualquier espacio en blanco del subpatch y seleccionar “graph on parent”.

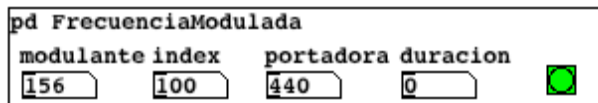


Al hacer esto se creará en el subpatch un cuadrado rojo dentro del cual podemos situar los objetos gráficos que se verán en el patch principal. En el diálogo de propiedades podemos definir también el tamaño y la posición (en píxeles, contando desde la esquina superior-izquierda de la ventana) de este cuadrado rojo.



Los elementos que podemos ver desde el patch principal, situándolos dentro del cuadrado rojo del subpatch son: número, símbolo, comentario y todos los objetos gráficos GUI. Cualquier otro elemento, aún cuando esté dentro del área roja, no se verá. Por tanto para controlar el mensaje bang tendremos que utilizar su versión gráfica, el objeto gráfico Bang (ctrl-shift-B o un objeto llamado “bng”)

De este modo la apariencia del objeto “pd FrecuenciaModulada” situado en el patch principal es la siguiente, permitiendo la modificación de cada uno de sus elementos:



Podemos hacer cuantas copias queramos de este objeto en el mismo patch, asignandoles parámetros diferentes para obtener una paleta más amplia de sonidos.