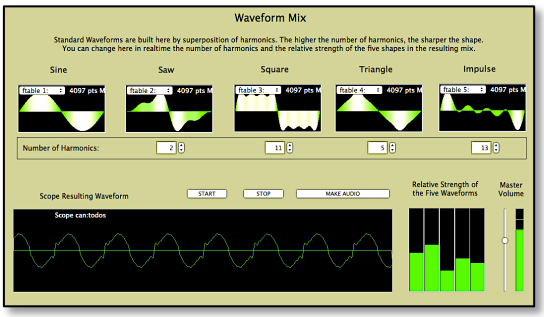
Práctica 9: Síntesis digital del sonido

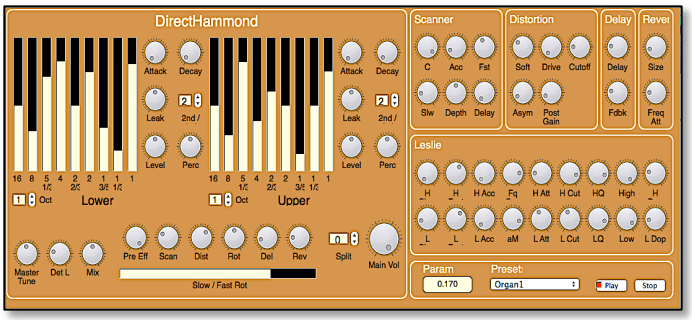
Esta tiene dos sesiones bien diferenciadas: en la primera trabajaremos conceptos generales relacionados con la síntesis, como el control en tiempo real de la misma, el control mediante secuencias almacenadas en ficheros MIDI estándar y la generación de envolventes. La segunda sesión se dedicará a poner en práctica el funcionamiento de algunas técnicas de síntesis vistas en teoría. Por esta razón, este material previo se da también dividido en dos partes, una para cada sesión.

PRIMERA SESIÓN

Control y programación de *widgets*

Los *widgets* permiten el control en tiempo real de los programas en CsoundQt. Usando el panel de *widgets*, se pueden diseñar interfaces gráficos que permiten un control interactivo de los instrumentos de Csound, así como una realimentación visual de su comportamiento. A continuación se muestran algunos ejemplos de estos interfaces.





La comunicación entre los *widgets* y CsoundQt se realiza a través de los opcodes **invalue** y **outvalue**. El operador **invalue** permite enviar un valor por un canal de transmisión desde un *widget* a Csound y **outvalue** envía valores desde Csound a los widgets. La estructura de ambos se describe a continuación:

```
kValor  invalue  "nombre_canal"
```

donde `kValor` – variable que contiene el valor enviado desde el *widget* (puede ser SValor si es una cadena).
`"n_c"` – nombre dado al canal de transmisión de la información desde el widget en sus propiedades.

```
        outvalue  "nombre_canal", kValor
```

donde `"n_c"` – nombre dado al canal de transmisión de la información hacia el widget en sus propiedades.
`kValor` – variable que contiene el valor que se envía al *widget* (también puede ser una cadena).

Por ejemplo, para leer un valor desde un widget:

```
instr 1
  kFrec invalue "frecuencia"
  aSal  poscil   iAmplitud, kFrec
        out      aSal
endin
```

Para escribir un valor en un widget:

```
instr 2
  kVal line     1000, 1, 2000
       outvalue "valor_salida", kVal

endin
```

Cuando usamos *widgets*, el control ya no lo hace necesariamente la partitura, sino que todo puede ser controlado desde el panel de *widgets*. En este caso, en la partitura basta con poner una orden que active el sistema durante determinado tiempo, especificado en segundos (salvo que una orden tempo, t, cambie el valor temporal de los tiempos). Las dos órdenes siguientes pueden hacer esa activación:

```
f 0 60    ; activa Csound durante 60 s sin una acción determinada
e 3600    ; activa Csound sin hacer nada y termina 1h después
```

Otra posibilidad es activar un determinado instrumento (con parámetros o no) y que luego el *widget* sea el que asuma el control de los valores de sus parámetros:

```
i 1 0 60 ; activa el instrumento 1 desde el principio, durante 60 s
```

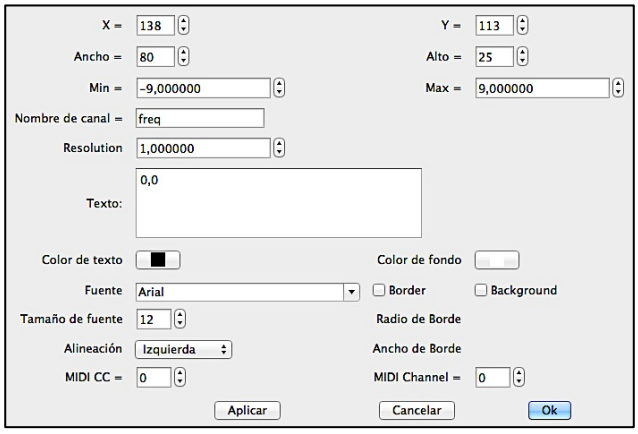
El control también puede ser mixto, en el sentido de que los instrumentos pueden ser activados con notas desde la partitura, pero sus parámetros de funcionamiento pueden ser controlados mediante *widgets*.

Tipos de *widgets* y su función

Un **panel** contiene *widgets* que pueden interactuar con Csound. Tiene dos modos: modo de acción y un modo de edición. En el *modo de acción*, los widgets se utilizan para enviar y recibir valores de Csound a través de canales a los que se da nombre en sus propiedades y pueden modificarse utilizando el ratón o el teclado. En el *modo de edición*, los widgets se pueden mover, redimensionar, copiar, pegar o duplicar.

Los modos se pueden alternar utilizando Ctrl–E o mediante el menú de CsoundQt **Editar → Modo de edición**. Cuando estamos en edición, aparecen marcos rojos alrededor de cada *widget*. Estos marcos sirven para mover y cambiar de tamaño los *widgets*.

Propiedades de un *widget*: pueden accederse haciendo clic derecho sobre el *widget* y seleccionando "Propiedades". Esto funciona tanto en modo edición como en acción. Las propiedades son diferentes para cada tipo de *widget*. Si hace clic derecho en donde hay no hay *widgets*, se puede establecer el color de fondo del panel de *widgets*. En la siguiente figura se muestra un ejemplo de la ventana de propiedades, correspondiente a un **Spin box**.

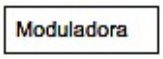


Algunas de las propiedades que se ven para este tipo de controlador son:

- Posición (X,Y) y tamaño (Ancho,Alto).
- Rango de valores (Min,Max).
- Nombre de canal (para que Csound se comunique con el widget mediante los operadores **invalue** y **outvalue**).
- Resolución (incremento o decremento).
- Texto: lo que aparece en la ventana.
- Color de texto, fondo y borde del widget.
- Propiedades del texto (alineación, tamaño y tipo).
- Canal MIDI (si se transmite MIDI).

Tipos de *widgets*:

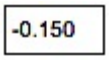
- **Labels:** su función es mostrar texto con formato. Se puede elegir la alineación, el tamaño, tipo de fuente, borde y colores de texto y fondo. Las etiquetas no envían ni reciben datos desde Csound.



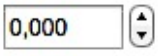
- **Displays:** son etiquetas cuyo valor puede establecerse desde Csound a través de un canal de salida. Este tipo de *widget* puede mostrar números y texto.

- **Sliders:** los controles deslizantes se utilizan para enviar y recibir datos desde el programa Csound utilizando canales. Si el ancho de un deslizador es mayor que su altura se convierte en un deslizador horizontal. En sus propiedades se puede establecer el rango de valores que puede emitir (mínimo y máximo).

- **Scroll numbers:** son como los **displays**, pero sus valores pueden cambiarse arrastrando el ratón sobre la ventana. El número de cifras decimales utilizado y el tamaño del paso para cada pixel del ratón se pueden fijar mediante la resolución en las propiedades.

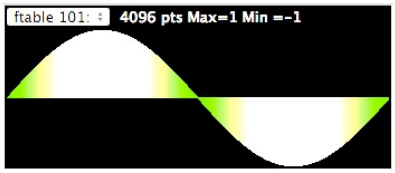


- **Spin Boxes:** permiten escribir números con el teclado y el ratón, para transmitir esos valores al programa por un canal de entrada. Funcionan de manera similar a otros *widgets* de texto o valor.



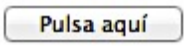
- **Knobs:** permiten dar valores números con el ratón al deslizarlo sobre un modelo de potenciómetro, para transmitir esos valores al programa por un canal de entrada. En sus propiedades se puede establecer el rango de valores que puede emitir (mínimo y máximo). Funciona como un **slider**, pero en forma de mando redondo.

- **Graphs:** Permite mostrar tablas de Csound. La tabla visible puede cambiarse enviando su valor por el canal del *widget*. Valores positivos cambian la tabla por índice y los valores negativos cambian la tabla por número de f-tabla. El segundo canal puede utilizarse para visualizar las tablas según el número de f-tabla en lugar del índice. La tabla mostrada también puede cambiarse con el



ratón mediante el menú en la esquina superior izquierda. Esto producirá una salida en ambos canales. También pueden mostrar el espectro de señales usando el opcode **dispfft**, o señales y variables (tipo a– y k–) utilizando el opcode **display**.

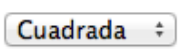
- **Buttons:** Estos *widgets* pueden tener dos propósitos. Pueden transmitir un valor cuando se pulsan y pueden generar eventos de partitura en tiempo real al ser pulsados. Su función exacta depende del tipo de botón:

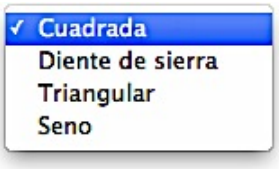


  - Los botones de tipo *value* siempre transmiten 0 por su canal cuando no están pulsados. Al ser pulsados transmiten el valor definido en sus preferencias.
  - Los botones de tipo *event*, aparte de transmitir un valor determinado mientras estén pulsados, también pueden enviar eventos de partitura a Csound. Cualquier evento de partitura válido puede ser activado por un botón (salvo caracteres especiales, como **.** + o <). El texto de la línea de partitura que se quiera enviar se establece en las propiedades del widget.
  - Hay algunos canales reservados que pueden enviar información a CsoundQt. El canal "`_Play`" puede iniciar y detener Csound. El canal "`_Stop`" puede iniciar y detener Csound. También hay un canal "`_Pause`". El botón debe ser de tipo *value* o *pictvalue*.
  - Hay otros tipos de botones con imágenes. *Pictevent* y *Pictvalue* son equivalentes a los botones de *event* y *value* pero muestran una imagen. El botón tipo *Pict* es simplemente una decoración.

- **Checkbox:** es un sencillo widget que envía un valor de 1 por su canal cuando se pulsa y 0 cuando no. Su valor pulsado también se puede definir mediante un canal.

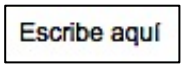
- **Menus:** pueden mostrar menús con opciones definidas por el usuario. Las opciones se establecen como una lista separada por comas de las propiedades. El widget menú transmite el índice del elemento seleccionado, contando desde 0 para el primero. También se puede definir el índice actual desde Csound.





- **Controllers:** son *widgets* que pueden ser utilizados para transmitir los datos de los movimientos del ratón. Algunos controladores pueden enviar un solo valor, pero otros pueden enviar los valores horizontales y verticales. El rango de valores transmitidos por el controlador puede establecerse en sus preferencias. También sirven para representar valores.

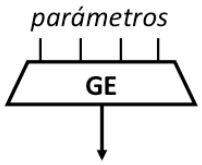
- **Line Edit**: estos pueden utilizarse para pasar cadenas desde el panel de *widgets* al programa Csound. También pueden utilizarse para recibir cadenas desde Csound.



- **Scope:** este es un osciloscopio que puede mostrar la salida de Csound. El osciloscopio puede mostrar canales individuales o una suma de todos los canales de salida. Haciendo doble click en él se congela la señal mostrada.

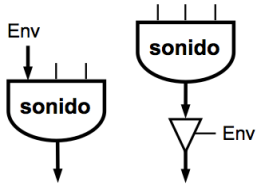
Si quieres practicar con los *widgets* antes de entrar al laboratorio puedes intentar abrir los ejemplos que hay en CsoundQt, en: **Ejemplos → Widgets**

Generación y uso de envolventes



Las envolventes permiten la evolución temporal del valor de un parámetro de control. Puede ser para la evolución de una amplitud, de una frecuencia, del panorama, etc.

Hay dos formas de aplicar una envolvente: una (*a priori*) controlando la amplitud cuando se está generando el sonido (sólo posible en síntesis) y otra (*a posteriori*) controlando la ganancia de un amplificador por el que pasa el sonido.

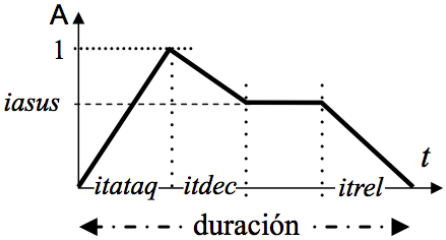


Para añadir una envolvente de amplitud a un sonido que se sintetiza tenemos que cambiar la amplitud constante (de tipo **i–**) por otra que evolucione a lo largo de la duración de cada nota (de tipo **k–**), por lo que la amplitud dada será ahora sólo la amplitud máxima.

Para ello se pueden utilizar distintos generadores de curvas unipolares en Csound, de forma que los sonidos sintetizados tengan distintos tiempos de ataques y caídas. Veremos los modelos de envolventes más habituales: ADS, ASR y AR.

Envolvente temporal tipo ADSR

Las envolventes ADSR son las más generales. Sus cuatro parámetros son los ya conocidos: t_A (en seg.), t_D (s), A_S (como fracción de la amplitud máxima) y t_R (s). Estas señales de control se generan con el opcode **adsr** que crea una curva normalizada en amplitud que luego se multiplicará por la amplitud (máxima) de la nota.




```
kCurva  adsr  itataq, itdec, iasus, itrel
```

Tiempos en la envolvente absolutos o relativos a la duración de la nota

Los operadores generadores de envolvente manejan los tiempos en segundos. Si los especificamos así, decimos que son *absolutos*, pero esto es peligroso porque podrían ser más largos que las notas más cortas. Es decir, si por ejemplo, $t_A = t_D = t_R = 0{,}1$ s, entre los tres tiempos ya suman 0,3 s, por lo que en notas cortas podemos tener problemas (si una dura menos de 0,3 s).

Es más práctico y seguro que sean *relativos* a la duración de la nota (no es realista, porque el ataque de un determinado timbre no tiene por qué depender de su duración, pero evita problemas). Para ello, las duraciones de las fases se especifican en proporciones (0,1 sería un 10%) y los tiempos que pasan a los opcodes se obtendrán multiplicando estas proporciones por la duración de cada nota. De esta forma, conseguimos que las duraciones de las fases de ataque y caída sean proporcionales a la duración y evitamos ese problema.

por ejemplo, si enviamos desde la partitura (con tempo 60 BPM) los parámetros

```
; . . . p3  p4  p5  p6
;----------------------
; . . .  6 0.1 0.2 0.3
```

y en el instrumento asignamos

```
iDur = p3
itA  = p4*iDur
itD  = p5*iDur
itR  = p6*iDur
```

tendremos las siguientes duraciones en segundos

```
iDur = 6
itA  = 0.1 × 6 = 0.6
itD  = 0.2 × 6 = 1.2
itR  = 0.3 × 6 = 1.8
```

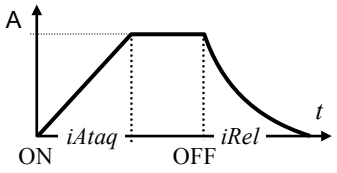
quedando para la fase estable (de sostenimiento) el resto de la duración = 6 – 0.6 – 1.2 – 1.8 = 2.4 s. También se podría hacer que t_A se haga de duración fija en segundos y las demás fases proporcionales a la duración. Basta con no multiplicar **p4** por la duración en este ejemplo.

Extensión del tiempo de funcionamiento por el tiempo de relajación

Si un instrumento se controla vía MIDI, cuando se activa (por un *note on*) no se puede saber cuánto tiempo va a permanecer activo (hasta que llegue el correspondiente *note off*), por lo tanto la duración no puede ser usada (no tenemos **p3**) para calcular los segmentos de la envolvente. Por lo tanto, todos los *opcodes* que usan la duración como uno de sus argumentos no podrán usarse. Deben ser sustituidos por otros similares que no necesiten la duración.

Por ejemplo, el generador de envolvente ASR **linen** tiene como parámetros: *Amp, tA, dur, tR*. Ahora, no sabemos la duración de la nota cuando esta se activa. La envolvente ASR debe comenzar a caer cuando se reciba el nota OFF. Esto lo lleva a cabo el opcode **linenr**,

linenr Genera una curva como la mostrada en la figura.

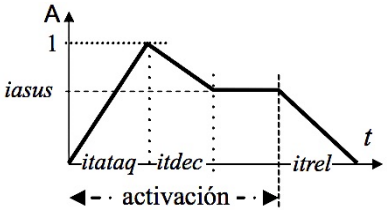



```
kCurva   linenr   iAmp, iAtaq, iRec, iCae
```

donde
- `iAmp` – valor de amplitud
- `iAtaq` – tiempo de ataque en segundos (si = 0 no existirá fase de ataque).
- `iRel` – tiempo de caída de la envolvente.
- `iCae` – coeficiente de la función de caída exponencial. Debe ser > 0.

El tiempo *iRel* resulta en una extensión de la duración de la nota a partir de la llegada del nota OFF MIDI. *iCae* debe ser positivo y valores habituales son del orden de 0.01.

En el caso de una envolvente de tipo ADSR, hay que usar el operador **madsr**, que tiene los mismos parámetros y sintaxis que el **adsr** (t_A, t_D, A_S, t_R), pero inicia la fase de relajación cuando llega el nota OFF (ver la figura). Así, el tiempo de funcionamiento del instrumento se extenderá *itrel* segundos.



Control mediante MIDI

Los instrumentos de Csound pueden ser controlados externamente por los mensajes MIDI que entren por el puerto MIDI IN o desde las pistas de un fichero MIDI. En el laboratorio no podemos hacer lo primero, pero sí podemos hacer lo segundo. Para ello, hay que saber hacer algunas cosas en Csound.

Control MIDI desde…

1. Un controlador MIDI externo: debes verificar que el sistema ha detectado tu controlador. Eso se hace desde la opción de CsoundQt **Configurar → Correr.** Entre todas las opciones, debes fijarte en estas:
2. Un fichero MIDI estándar: en este caso, hay que indicar que la entrada MIDI se hace desde un fichero. Para ello se usa la opción de compilación (en <CsOptions>) –F, seguida por el nombre del fichero. Por ejemplo, como en `-F ejercicio.mid`
3. Desde el teclado virtual de CsoundQt: en este caso, en las opciones de compilación debes escribir `-+rtmidi=virtual` y activar la visualización del teclado virtual, en **Ver → Show Virtual Keyboard**. Si hay problemas con su visualización, debes ir a la pantalla de configuración (ver caso 1.) y en Módula MIDI RT seleccionar "Virtual" (asegúrate que, en esa misma pantalla, está activada la opción "Use Csound MIDI modules".
4. Una última posibilidad es convertir un fichero MIDI en una partitura de Csound. Hemos creado un programa Csound llamado `midi2cs.csd`. Ese programa toma un fichero MIDI estándar y crea una partitura de nombre `canalN.inc` por cada pista (N) del fichero. Estas partituras parciales pueden ser leídas desde un proyecto CSD mediante órdenes del tipo `#include canalN.inc` escritas en la partitura del proyecto.

Preparación de la orquesta

Por defecto, Csound asigna cada canal MIDI a un instrumento con el mismo nombre: el canal 1 al instrumento 1, el canal 2 al instrumento 2, etc. Esto puede modificarse con la orden **massign** *canal*, *instr* en la cabecera de la orquesta. Por ejemplo,

```
massign  2,6
```

Asignará los mensajes que entren por el canal 2 al instrumento 6. Si queremos que todos los canales se dirijan a un determinado instrumento, usaremos un 0 para el primer argumento:

```
massign  0,6
```

Esto re-direccionará todos los mensajes de canal al instrumento 6.

Cuando un instrumento recibe el control desde un fichero MIDI, no desde la partitura de Csound, ya no usará p3, p4 o p5, sino la altura y velocidad MIDI de cada nota. Para leer estos valores, se deben usar los siguientes conversores MIDI al principio del instrumento para dar valor a variables de tipo **i–**:

- `iAmp` **`ampmidi`** `iEscal` : lee la velocidad de un nota ON y la asigna a *iAmp* en el rango [1 , *iEscal*].
- `iVel` **`veloc`** : lee la velocidad de un nota ON y la asigna a *iVel* en el rango [1 , 127].
- `iFrec` **`cpsmidi`** : lee el valor de la altura de un mensaje nota ON y lo convierte en frecuencia (en Hz) según la afinación bien temperada. No tiene parámetros de entrada.
- `iAlt` **`notnum`** : lee el valor de la altura MIDI (0–127) de un mensaje nota ON. Sin entradas.

Compatibilidad partitura de Csound - MIDI

Existe una forma de que los instrumentos de una orquesta de Csound reciban los valores de parámetros que llegan desde un dispositivo o fichero MIDI de la misma forma que los leen cuando son activados por una partitura. Para ello hay que definir unas opciones de compilación como las siguientes:

`--midi-velocity=4` : hace que el instrumento reciba en **p4** el valor de la velocidad MIDI (entre 0 y 127)
`--midi-key=5` : hace que el instrumento reciba en **p5** el número de nota MIDI (entre 0 y 127)

Hay más, cuyo aplicación depende del uso que queramos dar a los valores recibidos.

SEGUNDA SESIÓN

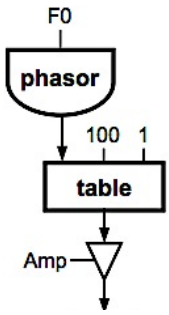
El objetivo de esta segunda sesión es hacer un repaso de algunas técnicas de síntesis digital del sonido, como los sintetizadores por tablas de onda (*wavetables*), sustractivos, *samplers* y FM. Utilizaremos las herramientas habituales en prácticas anteriores como CsoundQt y Audacity.

Síntesis por tablas de onda (*wavetables*)

Las dos piezas básicas de este tipo de síntesis son:

1. un generador de fase (**phasor**) que calcula los valores que debe tomar el índice de fase en función de la frecuencia deseada (F0) y
2. una tabla, para producir la onda al leer cíclicamente los valores almacenados en ella.

Este esquema es el de la figura. El operador **table** lee una tabla según el índice que se le indique en su primer parámetro. El valor 100 es el número de la tabla (en Csound las tablas se refieren por un número entero). El valor 1 es para indicar que los valores del índice están normalizados (sea cual sea el tamaño de la tabla, se lee entre 0 y 1).



La señal así generada entre –1 y +1 se amplifica para que oscile entre –Amp y +Amp mediante el multiplicador que hay a la salida del lector de la tabla.

Construcción de tablas en Csound

Csound puede crear tablas con datos mediante las numerosas rutinas GEN. Se pueden crear en la partitura mediante la orden "**f**":

```
f  NumTabla  0  size  numGEN  parámetros
```

donde el cero es el tiempo a partir del cual estará disponible (0 = desde el principio), `size` es el tamaño de la tabla en número de valores (normalmente potencia de 2 o 2^n+1) y `numGEN` es el número de rutina a utilizar. Luego aparecerán diversos parámetros que estarán en función de la rutina GEN que se use. Son tablas *globales*, por lo que el número de tabla *NumTabla* podrá ser usado por cualquier instrumento de la orquesta.

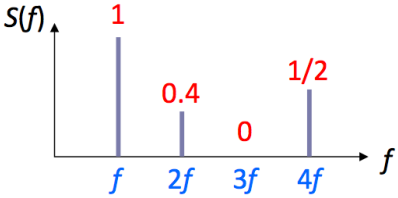
Lo mismo se puede hacer en cualquier instrumento de la orquesta (tabla *local*, asignada a una variable de tipo **i**–) o entre la cabecera y el primer instrumento (asignada a una variable global de tipo **gi**–) como:

```
giNumTabla  ftgen  numero, tini, long, numGEN, parámetros
```

A partir de donde esté esta orden se podrá usar la tabla, con la variable *giNumTabla*, que contiene el número de la misma. Como es *global*, cualquier instrumento podrá usarla. Si fuera *local* sólo en el que esté creada.

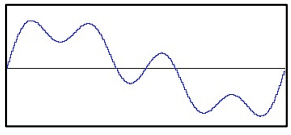
Construcción de tablas mediante especificación del espectro:

Las rutinas GEN 9, 10 y 11, por ejemplo, permiten construir un ciclo de una onda que sea la suma de parciales especificados mediante sus amplitudes relativas y frecuencias (9 con parciales arbitrarios, 10 con armónicos seno y 11 con armónicos coseno). Vamos a ver el funcionamiento de GEN 10:



Supongamos que queremos construir una tabla (la 100) con 256 valores de un ciclo de onda que tenga el espectro de la figura. Son 4 armónicos, por lo que con GEN 10 basta con indicar sus amplitudes (normalizadas):

```
f  100  0  256  10  1  0.4  0  [1/2]
```



El contenido de la tabla será el resultado de hacer esa combinación lineal de ondas sinusoidales a lo largo de un ciclo entre 0 y 2π:

Una consecuencia es que podemos crear una tabla con un ciclo sinusoidal usando sólo un armónico.

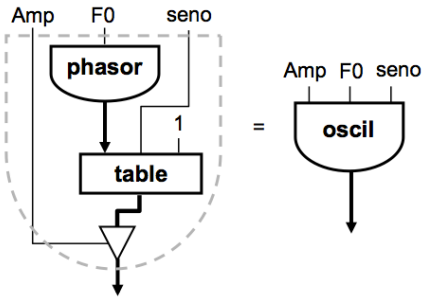
Construcción de tablas mediante especificación de una forma:

Para especificar la forma se utilizan rutinas GEN como la 2 (escribiendo sus valores), la 23 (leyéndolos de un fichero) o la 8 (creando una *spline* cúbica a partir de los puntos por los que ha de pasar). Vamos a ver cómo funciona la 2. Supongamos que queremos crear una envolvente adsr en una tabla de 5 puntos:

```
f  101  0  5  2  0  1  0.5  0.5  0
```

El resultado es que la tabla 101 almacenará lo que vemos en la figura de la izquierda, aunque si leemos la tabla interpolando entre puntos sucesivos (con **tablei**) obtendremos lo de la derecha, que es lo que se pretendía:

Implementación del *wavetable* con oscil



Csound encapsula los operadores anteriores (**phasor** + **table** + amplificador) en un operador único (**oscil**) que hace todo el trabajo si se le indica la frecuencia, la tabla a leer y la amplitud, comportándose como un sintetizador *wavetable*. Tiene dos variantes: una **oscili**, que interpola linealmente al leer valores de la tabla (equivalente a **phasor** + **tablei** + amplificador) y el que usa interpolación cúbica **oscil3** (equivalente a **phasor** + **table3** + amplificador).

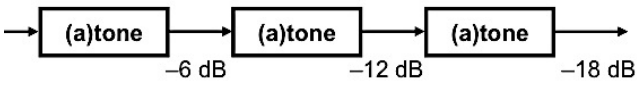
Filtros lineales en síntesis

Csound tiene multitud de operadores para procesar señales. Vamos a explorar algunas de sus posibilidades, usando algunos filtros, que usaremos en algunas técnicas de síntesis. Para ello, utilizaremos sonidos digitalizados como entrada y generadores de curvas para controlar dinámicamente los parámetros de los filtros.

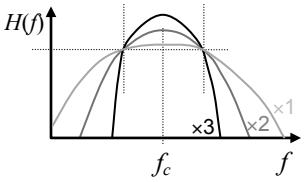
Entre las familias de filtros lineales que dispone Csound, los más sencillos son los cuatro típicos: pasa-baja de un polo (**tone**), pasa-alta de un polo (**atone**), pasa-banda de dos polos (**reson**) y elimina-banda de dos polos (**areson**).

Filtros más restrictivos

Estos operadores implementan filtros muy suaves en su banda de transición (–6 dB/oct los **tone**). La forma de aumentar esta pendiente y conseguir filtros más restrictivos es conectar varios filtros en cascada. De esta forma, mediante pasos sucesivos por un pasa-baja de –6 dB/oct de pendiente, se va aumentando la pendiente en otros –6 dB/oct de caída cada vez que se pasa, construyendo así un filtro más abrupto, como indica la figura.



Lo mismo se puede hacer con **reson**, que es de –12 dB/oct. Para incrementar esta pendiente y hacer un filtrado más selectivo, se conectan en cascada estos filtros. De esta forma, mediante dos pasos por un pasa-banda de –12 dB/oct de pendiente, se consigue una pendiente de –24 dB/oct.

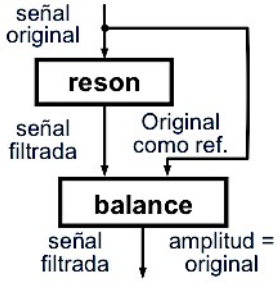


Corrección de cambios de volumen

El uso de filtros puede alterar la amplitud de la onda. Para corregir este efecto indeseado, el operador **balance** se usa para ajustar la amplitud RMS de la señal filtrada al valor de la amplitud de una señal de referencia:

```
aSalida  balance  aFiltrada, aReferencia
```

Si la señal de referencia es la original antes de ser filtrada, nos aseguraremos de corregir los cambios de volumen que provocan los filtros. El esquema de la figura, que corresponde a esta idea, se implementaría en Csound de la siguiente forma:

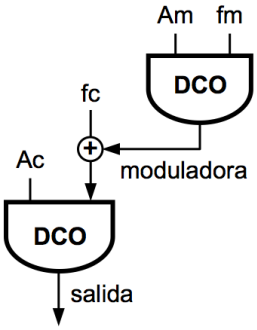



```
aFiltro  reson    aEntrada, iFcentral, iAnchoBanda
aSalida  balance  aFiltro, aEntrada
```

Síntesis sustractiva

Este tipo de síntesis se basa en el uso de filtros y amplificadores para adecuar el espectro de la señal de un oscilador al objetivo que se quiere conseguir. En esta práctica diseñaremos un sencillo sustractivo basándonos en el concepto de la síntesis modular: la interconexión de componentes simples (osciladores, filtros, amplificadores y envolventes), basados en las ideas vistas anteriormente.

Síntesis FM simple



Csound dispone de operadores específicos para síntesis FM. Sin embargo, la implementación explícita de esta técnica requiere únicamente 2 osciladores conectados como se indica en la figura. Asumimos que ambos osciladores serán sinusoidales, por lo que podremos implementarlos con un **oscili** (con interpolación para tener buena precisión) que lea una tabla creada con GEN 10 con un único armónico.

La repercusión que los valores de los 4 parámetros indicados (A_m , f_m , A_c , f_c) tiene en el sonido, es la que se resume a continuación:

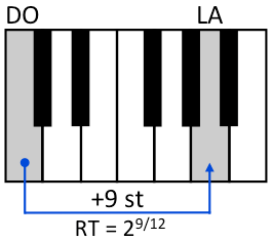
- La **amplitud de la portadora**, A_c, controla la amplitud del sonido sintetizado.
- La **relación** f_c / f_m, que controla su armonicidad [1], se especifica como dos números enteros que representan el numerador y el denominador de la relación entre ambas frecuencias N_c / N_m.
    - Por ejemplo, si f_c = 400 Hz y f_m = 200 Hz, la relación N_c / N_m = 2:1 = 2. Si f_c = 250 Hz y f_m = 750 Hz, la relación N_c / N_m = 1:3.

  Si N_c / N_m es un entero, será armónico, si es un número racional, normalmente será armónico pero con f_0 ausente (será percibido como inarmónico), y si es un número irracional será totalmente inarmónico.
- El **índice de modulación**, $I = A_m / f_m$, determina el número de bandas laterales y, con ello, su brillo.

Sampler con keymap

Haremos un sampler de piano muy sencillo. Se utilizarán tablas creadas con la función GEN 1, que transfiere el contenido de un fichero de sonido a una tabla que puede leerse luego (1 vez, sin oscilar) con el oscilador **loscil**. Se propondrá establecer un *keymap* basado en octavas: cada octava usada tendrá como referencia para la síntesis una nota de piano muestreada: el DO. De esta manera, sólo cada DO se utilizará sin transponer y el resto de alturas de su octava se generarán a partir de ella con la transposición adecuada.

Transposición:



La transposición T es la relación de frecuencias que se aplicará al sonido digitalizado para obtener cada nota. Para cada octava N se proporciona el sonido muestreado de su DO_N. Por lo tanto, para obtener cualquier nota de esa octava usando la afinación bien temperada bastará con saber el intervalo i desde DO en semitonos y calcularemos la relación de transposición como $T = 2^{(i/12)}$.

La selección de la muestra a transponer se hará por octavas. Para ello hay que usar el número de octava N para que la orquesta use la tabla 100+N en la que se habrá cargado la muestra del DO de esa octava.

Situación:

A veces los sintetizadores hacen uso del estéreo emitiendo diferentes notas en diferentes posiciones del panorama. Lo haremos por octavas. Calcularemos *pan* en función de la altura y usaremos este valor con un operador **pan2** que distribuye una señal en el panorama estéreo. La más grave saldrá por un canal y la más aguda por el otro, interpolando las demás en el panorama.

[1] Si N_c / N_m es un entero, será armónico, si es un número racional, normalmente será armónico pero con f_0 ausente (será percibido como inarmónico), y si es un número irracional será totalmente inarmónico.

APÉNDICE: SINTAXIS DE GENERADORES DE SEÑALES Y CONVERSORES

Además de los operadores detallados en el material de la práctica tutorial de Csound hay operadores específicos de síntesis cuya sintaxis simplificada se muestra a continuación. Recuerda que los detalles completos puedes encontrarlos en la ayuda del entorno CsoundQt. También se incluyen los conversores de altura y sonoridad.

Generadores de señales:

phasor Genera una fase normalizada ($0 \leq kfase < 1$) según la frecuencia *kfrec* que se le indica. Su sintaxis es:

```
kfase  phasor  kfrec
```

loscil Lee un sonido muestreado almacenado en una tabla, con bucles opcionales en el sostenimiento.

```
ares  loscil  xamp, kcps, ifn [, itrans [, parametros de bucle ]]
```

donde `xamp` – valor de amplitud
`kcps` – determina la transposición de la frecuencia fundamental de la señal de la tabla
`ifn` – número de la tabla de onda que contiene una nota digitalizada
`itrans` – relación de transposición de la nota. Si se desconoce úsese 1 aquí y para `kcps`

oscil / oscili / oscil3 / poscil Produce señales repitiendo el ciclo de una forma de onda de una tabla. **oscili** y **oscil3** usan interpolación lineal o cúbica, respectivamente, con la misma sintaxis. **poscil** usa por defecto una tabla seno.

```
ares  oscil  xamp, xcps, ifn
```

donde `xamp` – amplitud
`xcps` – frecuencia fundamental de la señal generada
`ifn` – número de la tabla que contiene un ciclo de la onda a generar

table / tablei / table3 Devuelve un valor almacenado en una tabla, según un índice dado como entrada. **tablei** es como **table** pero si el índice no es entero interpola el valor entre sus vecinos (**table3** con cúbica). Su sintaxis es:

```
xres  table  xindice, itabla, imodo, ioffset
```

donde `xindice` – entrada con el valor del índice a buscar en la tabla (tipo i-, k- o a-).
`itabla` – número de la tabla a leer.
`imodo` – 0 significa un índice sin modificar y 1 un índice normalizado (entre 0 y 1).
`ioffset` – (opcional) elemento de la tabla en el que se sitúa el cero del índice de lectura. Para una tabla con el cero en el centro, debe valer tamaño/2 (para índices sin modificar) o 0.5 (para índices normalizados). El valor por defecto es 0.

vco / vco2 Genera ondas mediante un simulador de un oscilador controlado por tensión:

```
ares  vco  kamp, kfrec, itipo, ipw, itabla
```

donde `kamp` – amplitud
`kfrec` – frecuencia fundamental
`itipo` – tipo de onda: = 1: diente de sierra, = 2: cuadrada o modulada en anchura de pulso (PWM), = 3: diente de sierra / triangular / en rampa
`ipw` – entre 0 y 1. Sin efecto si *itipo* = 1; cuando *itipo* = 2, es la anchura de pulso (si *ipw* = 0.5 será cuadrada); si *itipo* = 3 controla si es una sierra (*ipw* = 0), triangular (*ipw* = 0.5) o rampa (*ipw* = 1).
`itabla` – tabla que contendrá un ciclo de onda sinusoidal.

Conversores de unidades: **cpspch(.)**: de altura (octava.nota) a Hz ; **cpsmidinn(.)**: de altura MIDI a Hz

Equivalencia entre las notas de la cuarta octava, la codificación octava.nota de Csound y su frecuencia bien temperada

Nota	DO4	DO#4 REb4	RE4	RE#4 MIb4	MI4	FA4	FA#4 SOLb4	SOL4	SOL#4 LAb4	LA4	LA#4 Sib4	SI4	DO5
Hz	261,6	277,2	293,7	311,1	329,6	349,2	370,0	391,5	415,3	440,0	466,2	493,9	523,3
cpspch	8.00	8.01	8.02	8.03	8.04	8.05	8.06	8.07	8.08	8.09	8.10	8.11	9.00
MIDI	60	61	62	63	64	65	66	67	68	69	70	71	72

ampdbfs(.): de dBFS a PCM (16 bits) ; **dbfsamp(.)**: de dBFS (16 bits enteros con signo) a PCM

Tabla de conversión entre los valores en dB que usa Csound, dBFS para 16 bits con signo y valores de amplitud PCM

dBFS	–54	–48	–42	–36	–30	–27	–24	–21	–18	–15	–12	–9	–6	–3	0
amplitud PCM	65	130	260	519	1036	1464	2068	2920	4125	5827	8230	11627	16422	23197	32768