Comenzado el	miércoles, 20 de mayo de 2020, 11:05
Estado	Finalizado
Finalizado en	miércoles, 20 de mayo de 2020, 13:18
Tiempo empleado	2 horas 13 minutos

Información

Síntesis digital del sonido

En esta segunda sesión está enfocada a practicar diferentes técnicas de síntesis, como la síntesis por tablas de ondas (wavetables), la síntesis sustractiva, la síntesis FM y los samplers.

Al final de la sesión deberás entregar los CSD en un fichero zip.

Esta práctica no tiene sesión no presencial.

Información

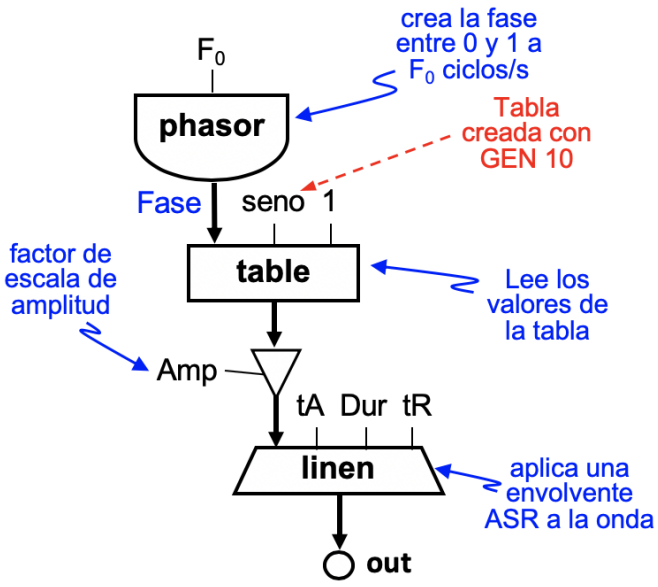
Síntesis por tablas de onda (*wavetables*)

Las claves de esta técnica de síntesis son: la generación de una fase que se repita f_0 veces por segundo y la lectura con ella de una tabla que contenga un ciclo con la forma de la onda. Eso es lo que se va a hacer en este ejercicio.

Ejercicio 9.6: creación de la fase y lectura de una tabla

Partes de la implementación de este sintetizador (sigue estas explicaciones en la figura):

- Crearemos una tabla muy pequeña con **GEN 10** con **ftgen**.
- Crearemos una fase que se repita (entre 0 y 1) con el operador **phasor**.
- Usaremos esta fase para leer con **table** la tabla de onda creada.
- La señal creada, entre –1 y +1, será multiplicada por la amplitud deseada (Amp).
- Finalmente le aplicaremos por una envolvente ASR mediante **linen** como indica la figura.



- La estructura se proporciona en esta PLANTILLA (botón derecho → guarar como).

Orquesta:

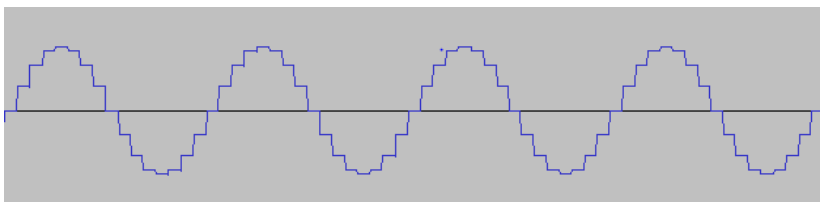
- Antes del instrumento se crea la tabla de onda mediante **ftgen**.
  - Será un ciclo de onda sinusoidal de 16 muestras, creada con la **GEN 10**(*)
  - Se le asignará el número 100 y este valor se almacenará en la variable *giSeno* (debe ser global, empieza por gi-, para que se vea desde dentro del instrumento).

```
giSeno ftgen 100, 0, 16, 10, 1   ; usa GEN 10 con un parcial de amplitud 1
```

- Dentro del instrumento:
  - La amplitud le será enviada en **p4** en dBFS (**ampdbfs(.)** la convierte a PCM) y
  - La frecuencia en **p5** como altura (**cpspch(.)** la convierte a Hz).
- Completa el instrumento con **phasor**, **table** y **linen** como se indica en la figura anterior y enviando la señal generada a la salida.
- Envolvente: haz que t_A = 0.01 × *iDur* y t_R = 0.2 × *iDur*.

Partitura:

- Ya está preparada para controlar el instrumento haciendo sonar un SOL_3 durante 4.5 tiempos (de 0,6 segundos).
- Compila y pruébalo. Despliega la ventana de los *widgets* para ver la onda.
- Comprobarás los problemas derivados del uso de incrementos fraccionarios con tablas muy pequeñas.



El sonido presenta mucha distorsión y la onda está "escalonada".

Mejoras:

- Cambia **table** por **tablei** y comprueba cómo cambia el sonido al leer la tabla con interpolación lineal.
- Luego cambia **tablei** por **table3** para usar interpolación cúbica en la lectura de la tabla. Pruébalo de nuevo.

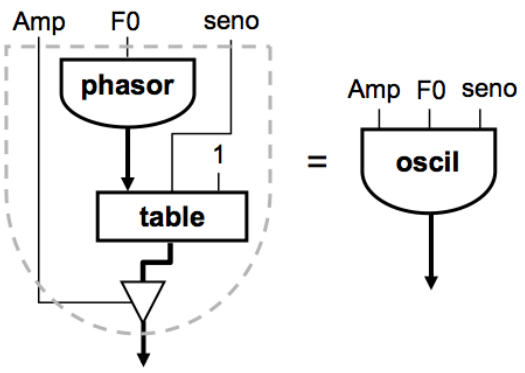
```
(*) Recuerda: la función GEN 10 permite construir un ciclo de una onda que sea la suma
de parciales armónicos. Para ello se especifican las amplitudes relativas de los armónicos
que intervienen en su construcción.
Si sólo hay una amplitud (= 1) será una onda con una única frecuencia, es decir, una sinusoidal.
```

Información

Síntesis por tablas de onda (*wavetables*)

Csound dispone de osciladores por tablas de onda, como **oscil**, que incorporan en un sólo operador, la generación de la fase, la lectura de la tabla y el escalado de amplitud.

Vamos a usar ahora el operador **oscil**, junto con otras funciones GEN.



Ejercicio 9.7: implementación con oscil

- Guarda el proyecto anterior como uno nuevo `ej9.7.csd`.
- Modifica en el nuevo proyecto el fichero de salida (`ej9.7.wav`).

Modificaciones en el instrumento:

- Como ahora vamos a tener varias tablas, le vamos a decir al instrumento desde la partitura (en **p6**) qué tabla usar: `iTabla = p6`
- Sustituye la generación de la fase, la lectura de la tabla y la multiplicación por la amplitud, por el operador **oscil** que hace todo eso en un único operador (como se indica en la figura superior). La amplitud será `iAmp`, la frecuencia será `iFrec` y la tabla será `iTabla`. El resto será igual que antes.

Partitura:

- A la nota existente hay que añadirle un sexto parámetro con el número de la tabla a utilizar. Si te fijas en la creación de la tabla, verás que se le asignó el número 100. Ese es el valor que debes indicar en esta primera prueba.
- Compila. El instrumento **wavetable**, tal y como está lee la tabla con la muestra más cercana y por eso distorsiona como en el ejercicio anterior. Cambia **oscil** por **oscil3** y deben sonar igual de bien que sonaba al final del ejercicio anterior.

Segunda parte: especificación de la onda por su espectro

La idea ahora es cambiar la onda sinusoidal especificada por otra diferente y hacer que cada instrumento use una distinta.

En la orquesta:

- Justo después de la definición de *giSeno*, crea *giSierra* (con **ftgen**), como una tabla (número 101) de 128 valores con una aproximación a una onda diente de sierra con 8 armónicos. Se creará con la rutina GEN10, pero con las amplitudes para los armónicos: 1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7 y 1/8. Como antes:

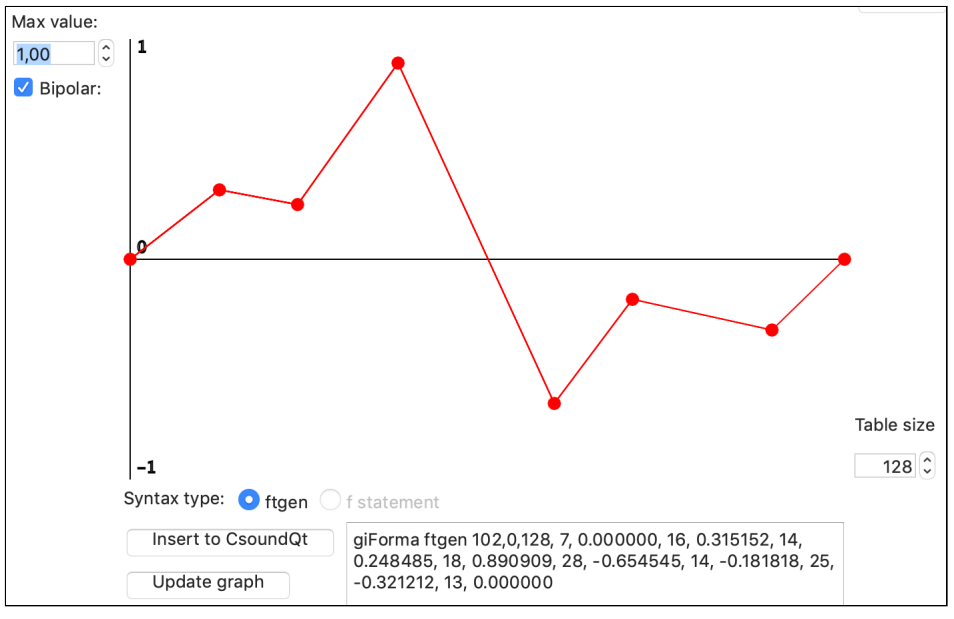
```
giSierra ftgen 101, 0, 128, 10, (y aquí se especifican las amplitudes de los armónicos)
```

- En la partitura:
  - Duplica la nota existente para que ahora active otra nota cuando acabe la primera (de la misma duración, amplitud –9 dBFS y altura MI_3 = 7.04), pero que use la tabla 101.
- Compila para comprobar que entre la primera y la segunda nota se produce un cambio de timbre, motivado por la lectura de la nueva tabla, con diferente forma de onda (puedes comprobar que, en efecto, es una aproximación a diente de sierra con el osciloscopio de los *widgets*).

Tercera parte: especificación de la onda por su forma

En la orquesta:

- Importante. Coloca el cursor en una línea vacía justo debajo de la definición de *giSierra*.
- Vamos a usar el editor de tablas de CsoundQt y el generador de tablas **GEN 7** (por segmentos rectos), siguiendo los pasos que se indican abajo.
- El aspecto puede ser como el de esta figura (¡ojo! sólo como referencia; no queremos que sea como esta).



- Despliega el editor de tablas con Ctrl + Mays + T (si es un Mac, con cmd + Mays + T). En cualquier caso, hay un menú **View → Show Table Editor**.
- En la ventana que aparece, crea la forma de onda (bipolar) que quieras, haciendo click sobre la línea y moviendo los puntos donde quieras. Sólo debes cumplir las siguientes condiciones: debe tener 7 segmentos, el primer y último valores deben ser 0 y que la mitad de los puntos estén en la parte > 0 y la mitad en la < 0.
- Indica Table size = 128 en la ventanita de la derecha.
- En la ventana inferior verás el código que creará la tabla. En ella cambia el nombre de la variable: en vez de giTable, llámala giForma.
- En esa misma ventan, verás que el primer valor después de ftgen es un 0. Cámbialo por 102.
- Cuando esté lista, pulsa [Insert to CsoundQt] y cierra el editor de tablas.

En la partitura:

- Duplica la nota existente para que ahora active otra nota cuando acabe la segunda (de la misma duración, amplitud –9 dBFS y altura DO_3 = 7.00), pero que use la tabla 102.
- Compila para comprobar que entre la segunda y la tercera nota se produce un cambio de timbre, motivado por la lectura de la nueva tabla, con diferente forma de onda (puedes comprobar que, en efecto, la tercera onda es la que dibujaste en el editor de tablas con el osciloscopio de los *widgets*).

Información

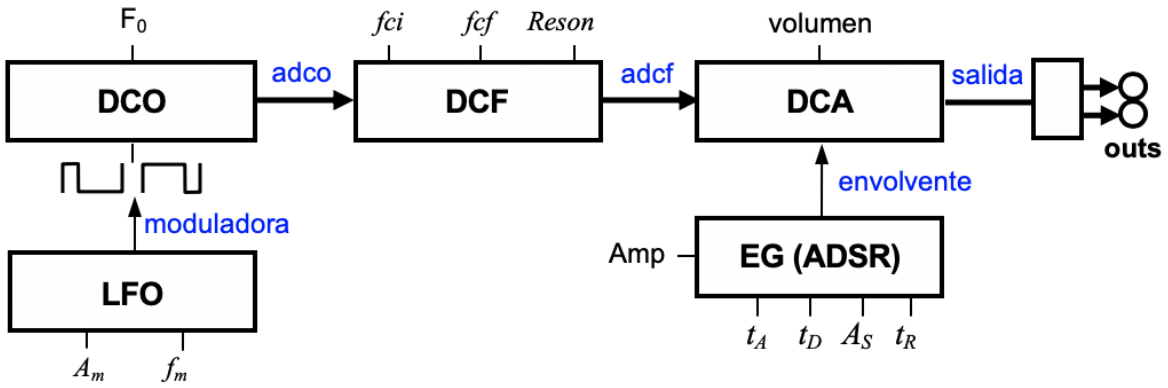
Ejercicio 9.8: Síntesis sustractiva

Se usará el concepto de la síntesis sustractiva para construir un sintetizador mediante componentes simples: osciladores, filtros, amplificadores y envolventes.

El sintetizador planteado tiene algunos de los módulos básicos de un sintetizador sustractivo digital:

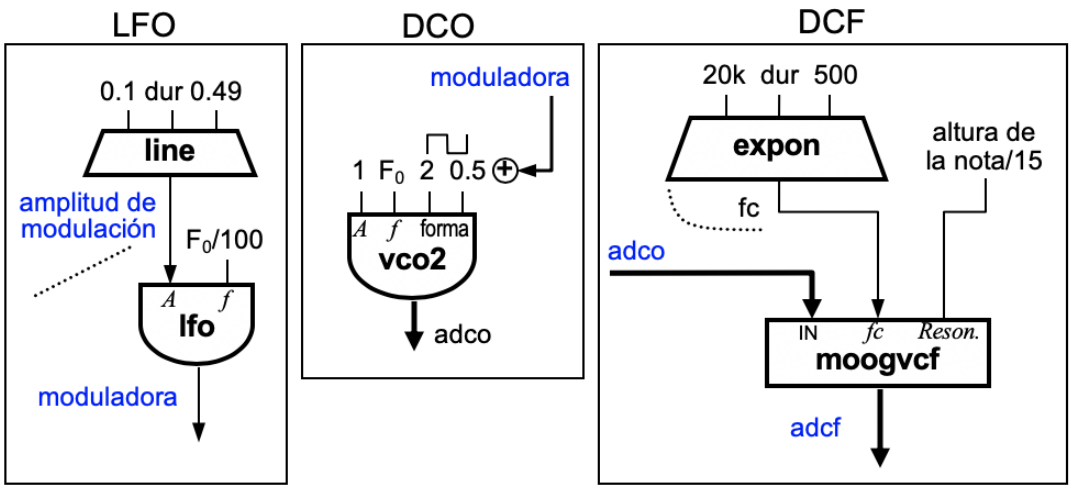
- un oscilador (DCO) que emitirá una onda cuadrada rica en armónicos
- un filtro digital (DCF) que será un pasa-baja resonante,
- un amplificador (DCA) controlado por un generador de envolvente (GE de tipo ADSR).
- un oscilador de baja frecuencia (LFO) que modificará periódicamente la forma de onda (y, por tanto, el timbre).

Su esquema de bloques es el siguiente:

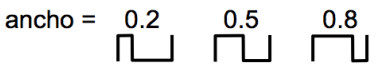


Descarga la PLANTILLA en la que se suministran las inicializaciones de las variables de nota en el instrumento y la partitura completa. Al instrumento se le suministrará la amplitud (**p4**, en dBFS) y la f_0 (**p5**, como altura de Csound). La partitura ha sido generada por un programa de conversión desde un fichero MIDI.

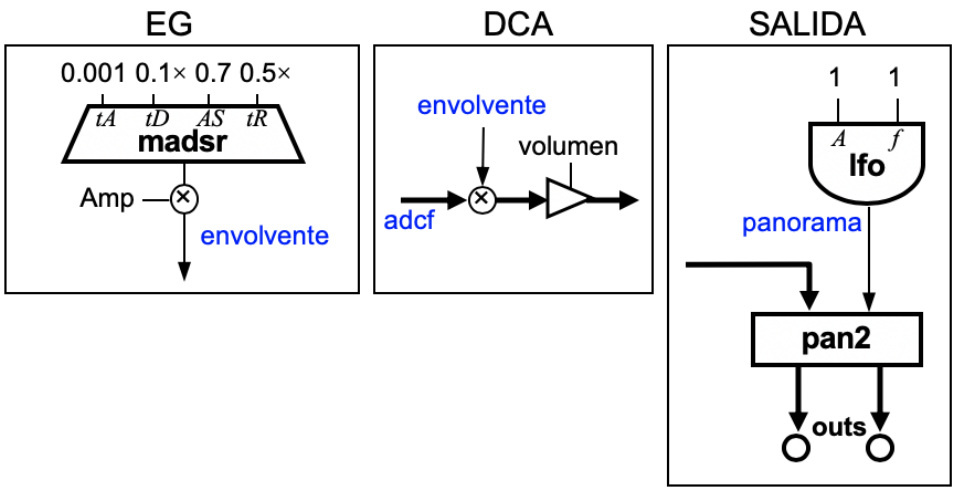
Los distintos módulos se describen en detalle en los siguientes diagramas:



- **LFO:** implementa una moduladora sinusoidal de amplitud creciente para realizar una modulación de anchura de pulso (PWM) sobre la onda cuadrada. Esta modulación se sumará al cuarto parámetro del oscilador, que es el que controla la anchura de pulso (tendrá una amplitud al final de 0.49 respecto a la onda cuadrada, cuya anchura es = 0.5) y una frecuencia = fundamental/100 Hz. La siguiente figura ilustra lo que ocurre periódicamente con la forma de la onda con una anchura del pulso de 0.5 ± 0.3:



- **DCO:** El *opcode* **vco2** es un simulador de oscilador analógico. Generará una onda cuadrada (tipo 2, ciclo activo 0,5 pero modulado como se indica anteriormente) de amplitud 1.
- **DCF:** Es un pasa baja resonante de frecuencia de corte decreciente, implementado con el operador **moogvcf**. El tercer parámetro indica la cantidad de resonancia (entre 0 y 1). Haremos que las notas más altas tengan más resonancia que las graves, usando un valor dependiente de la altura recibida (**p5**/15).



- **EG:** La envolvente se crea con **madsr**. El tiempo t_A debe ser independiente de la duración, pero t_D y t_R serán proporcionales a la duración de la nota (con × en la figura) multiplicándolos por *iDur*, **madsr** crea una curva ADSR normalizada en amplitud que luego se debe multiplicar por la amplitud deseada para cada nota.
- **DCA:** La señal se multiplica la envolvente y el volumen global se establece mediante *iVol*, inicialmente = 1.0.

- **SALIDA**: Para crear un efecto estéreo implementa la figura de ariba. El operador **pan2** distribuye una señal entre los canáles estéreo según el valor (entre 0 y 1) de su segundo parámetro.

- Implementa este sintetizador y pruébalo con la partitura suministrada.
- Ajusta el valor de *iVol* (coon 3 decimales) para que la salida de mayor amplitud (overall amps) se acerque pero no supere 20000 PCM (en el canal de la izquierda). Si tienes que ir haciiendo pruebas hasta ajustar, es recomendable usar *Render*.

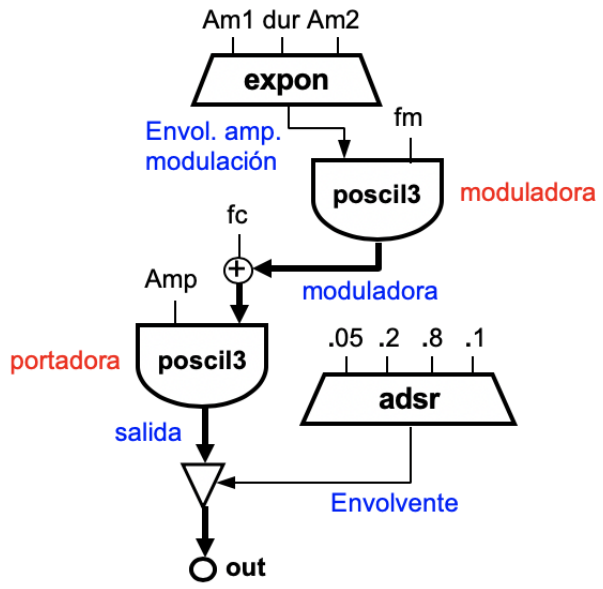
Información

Ejercicio 9.9: síntesis FM

Vamos a practicar con el algoritmo de síntesis FM mediante su implementación con dos osciladores sinusoidales y observar cómo cambia el timbre en función de los valores de las frecuencias involucradas, del índice de modulación ($I = A_m/f_m$) y su evolución temporal.

El instrumento se implementará según las instrucciones a partir de esta PLANTILLA (botón derecho → guardar como). Antes del instrumento hay instrucciones para comunicarse con el panel de *widgets*. Eso no hay que tocarlo.

La figura representa un sintetizador FM básico.



Instrumento FM

- Ambos osciladores usan el operador **poscil3** (un oscilador tabla de onda de precisión para mejorar su síntesis). Al no especificar la forma de la onda, crea por defecto una onda sinusoidal). Tanto la moduladora como la portadora mantendrán sus frecuencias constantes durante cada nota.
- La curva exponencial creada con **expon** controla la amplitud de la moduladora, A_m, responsable de la evolución del índice de modulación, que provoca la riqueza espectral. La curva recibirá sus valores inicial (A_{m1} = **p7**) y final (A_{m2} = **p8**) desde la partitura.
- El sonido sintetizado será multiplicado por la envolvente de amplitud de tipo ADSR generada por un operador **adsr**. Sus parámetros serán:
  - Tiempo de ataque t_A = 0.05 s,
  - Tiempo de caída t_D = 0.2 s,
  - Amplitud de la fase estable A_S = 80%
  - Tiempo de relajación t_R = 0.1 s.
- La amplitud de la portadora (en **p4**) será siempre 25.000 PCM y los valores en Hz de las frecuencias portadora (f_c = 400 Hz siempre) y la moduladora (f_m) llegarán desde la partitura en **p5** y **p6**, respectivamente.

Con las especificaciones que se indican a continuación veremos que las diferentes relaciones entre los valores de f_c, f_m y A_m hacen sonar este instrumento de maneras muy diferentes.

Partitura

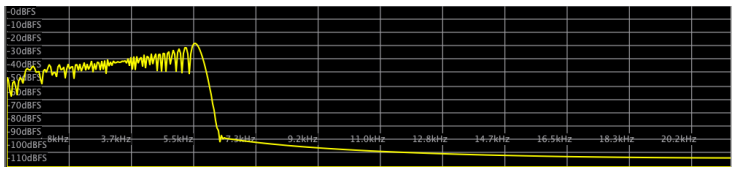
- El tempo será t = 30 BPM desde el principio.
- Deberás completar las notas para cada una de las siguientes especificaciones. Todas las notas tienen una duración de redonda y están separadas entre sí por silencios de corchea. La frecuencia de la portadora es siempre f_c = 400 Hz. Para cada nota deberás ajustar la frecuencia de la moduladora, f_m, y las amplitudes inicial y final de la modulación, A_{m1} y A_{m2}.
- Cuando estén listas las notas, compila y escucha el resultado de cada una.
- Despliega el panel de *widgets*, donde se ha preparado un visualizador del espectro generado y una explicación de lo que está ocurriendo en cada nota.

Primera nota: un vibrato

- Si la modulación (modificación periódica) de la f_c con una frecuencia f_m < 20 Hz se produce un vibrato.
- La amplitud de esta modulación establécela constante en A_{m1} = A_{m2} = 15 Hz. Estas amplitudes de modulación están en Hz porque son desviaciones de frecuencia.
- ¿Qué voy a ver en el *widget* del espectro? Básicamente el parcial de la portadora sinusoidal "temblando".

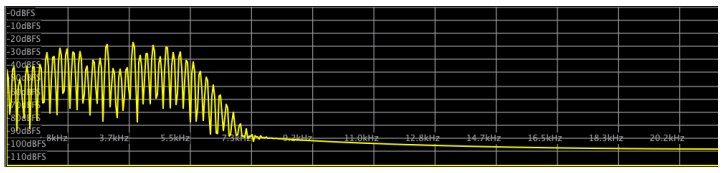
Segunda nota: vibrato creciente que termina siendo FM

- La frecuencia de modulación (**p6**) será ahora de f_m = 25 Hz.
- Amplitud de modulación: empiezará siendo un valor (**p7**) A_{m1} < 10 Hz y la final siendo (**p8**) A_{m2} > 6000 Hz.
- ¿Qué voy a ver en el *widget* del espectro? El parcial único de la portadora se empieza a agitar y se va a ir ensanchando hasta hacerse algo parecido a este:



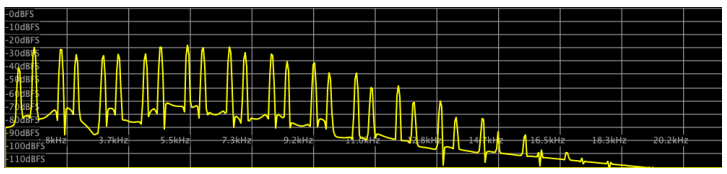
Tercera nota: síntesis FM armónica con índice de modulación decreciente

- Frecuencia de la modulación. Para que el sonido FM generado sea armónico, la fracción f_c / f_m debe ser un número entero o fracción sencilla. Establece f_m para que la f_c / f_m = 2; es decir, f_m = f_c / 2 = 200 Hz.
- Amplitud de modulación. Para que decrezca el índice de modulación ($I = A_m/f_m$) el valor de la amplitud de modulación debe ser variable. Que empiece siendo I > 30 (es decir, A_{m1} > 30 f_m = 30 × 200 Hz) y que acabe siendo I = 1 (A_{m2} = f_m) .
- ¿Qué voy a ver en el widget del espectro? Al principio veremos muchos parciales muy juntos (sonido más "brillante"), pero separados por la misma distancia entre ellos, como en la figura. Luego irán desapareciendo al ir disminuyendo el índice de modulación hasta quedarse en 4 o 5.



Cuarta nota: síntesis FM inarmónica con índice de modulación creciente

- Frecuencia de la modulación. Para un sonido inarmónico, la fracción f_c / f_m debe ser irracional. Establece f_m para que sea π; es decir, f_m = π f_c = 3.1416 × 400 Hz.
- Para que crezca el índice de modulación, que empiece siendo I = 0 (es decir, A_{m1} = 0.5 para que no dé error) y que acabe siendo I = 10 (es decir, A_{m2} = 20 f_m).
- ¿Qué voy a ver en el *widget* del espectro? Al principio será sinusoidal (1 único parcial). Poco a poco aparecerán nuevos parciales con separaciones diferentes (inarmonicidad). Algo parecido a esto:

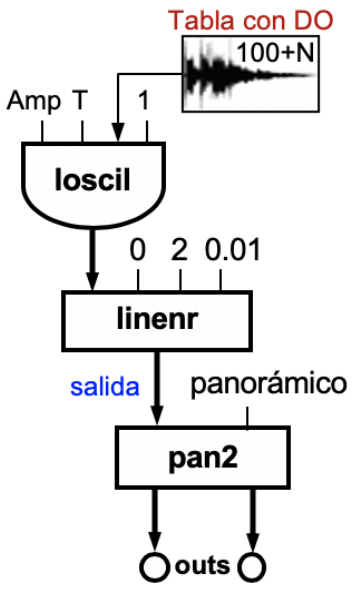


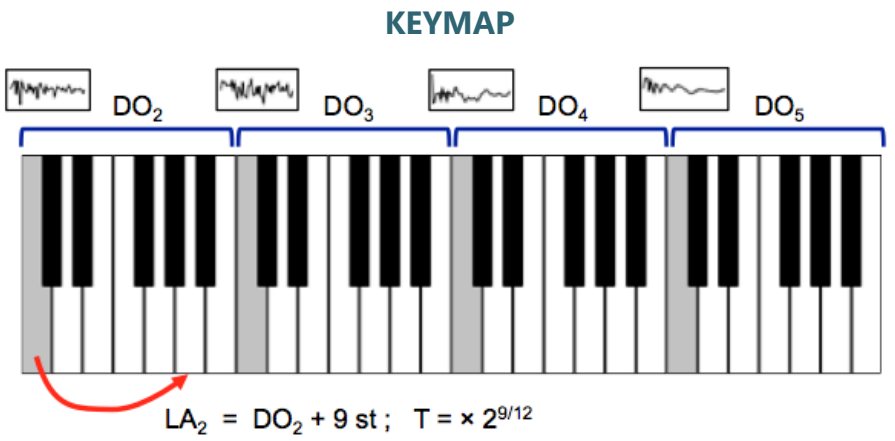
Información

Ejercicio 9.10: sampler con keymap

La estructura del instrumento y la partitura completa vienen con esta PLANTILLA (botón derecho → guardar como).

En la orquesta debes diseñar el instrumento de la figura. El *opcode* **loscil** lee sonidos muestreados desde una tabla, la cual recorrerá una sola vez, porque trabajaremos con sonidos percusivos (notas de piano) que no tienen fase estable y, por tanto, no se establecen puntos de bucle.





El keymap de este sampler establece que las notas de cada octava se obtendrán transponiendo el DO de la misma octava, usando la afinación bien temperada; es decir multiplicando cada frecuencia x $2^{st/12}$, donde st es la diferencia en semitonos desde el DO a la nota requerida.

Programa el instrumento según el esquema y estas instrucciones:

- La amplitud se suministra en dBFS en **p4** y se transformarla en amplitud PCM con **ampdbfs(·)**.
- La altura se envía en **p5** como octava.semitono. Por lo tanto, en el instrumento se deberá calcular:
    1. La octava, N, como la parte entera (con **floor(·)**) de **p5** – 4 (porque 8.00 = DO_4). Así, el instrumento usará la tabla 100+N en la que se habrá cargado la muestra del DO de esa octava. En esa tabla se habrá cargado (desde la partitura) la muestra del DO_N.
    2. El número de tabla a usar en cada activación del instrumento será, por lo tanto, 100+Número de octava.
    3. La cantidad de semitonos desde DO, st, como la parte fraccionaria de la altura (con **frac(·)**). Como son 2 decimales, habrá que multiplicar: **frac**(p5)*100 (de esta manera, por ejemplo, 0.07 → 7).
- La transposición T es la relación de frecuencias que se aplicará al sonido almacenado. 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 usar el intervalo st desde su DO en semitonos. En el instrumento calcularemos la relación de transposición como T = 2^(st/12).
- Como se ha dicho arriba, las fuciones básicas del sampler las lleva a cabo **loscil**.
- Función del operador **linenr**: crea una envolvente ASR en el que la parte de relajación se añade a la duración de la nota. Los samplers no necesitan envolventes porque los sonidos fuente ya tienen la suya propia. Pero si la nota es muy corta puede que la nota almacenada no pueda sonar entera, así que lo usamos para extender el sonido. Significado de sus entradas:
    - La primera es la señal a extender.
    - t_A = 0 es para no modificar el ataque original.
    - t_R = 2 es para alargar la duración en 2 s desde que se desactiva el instrumento cada vez.
    - El 0.01 establece la rapidez con la que decae la fase de relajación.
- A veces los sintetizadores hacen uso del panorama estéreo emitiendo diferentes notas en diferentes posiciones. Lo haremos por octavas. Calcularemos el panorámico = (**p5**−6)/3 y usaremos este valor como indica la figura. Así la octava 2 (6 de Csound) saldrá sólo por un canal y la 5 (9 de Csound) sólo por el otro, interpolando las demás en el panorama.

La partitura se ha diseñado para que se carguen las tablas de la 102 (para el DO_2) a la 105 (para el DO_5) con GEN 1, desde 4 ficheros denominados, respectivamente, `piano_doN.wav`:

```
f 102 0 0 1 "piano_do2.wav" 0 0 0   ; para el caso del DO2
```

Para el tamaño, se indica 0 para cargarlo entero (son sonidos de entre 1.2 y 2.0 segundos), dimensionando la tabla a ese tamaño (unas 90.000 muestras). Los tres ceros del final son 1º: incluir el fichero desde su inicio, 2º: detectar automáticamente su formato y 3º: leer todos los canales del fichero (es mono). Estos son los ficheros con las muestras:

- 0:00 / 0:00
- 0:00 / 0:00
- 0:00 / 0:00
- 0:00 / 0:00

La partitura suministrada en la plantilla codifica un fragmento de piano de 3 compases en 4/4.

- Compílalo y escucha el resultado. Debe de sonar sin cortes y bien panoramizado: las notas graves hacia un canal, las agudas hacia el otro y las medias por el centro.
- A continuación tenéis el sonido original del disco del que está sacada esta partitura. No sonará exactamente igual, porque hay adaptaciones, pero sirve para saber qué se puede esperar:

0:00 / 0:00

◄ Uso básico de los widgets en CsoundQt (~8 min.)

Ir a...

Entrega de la segunda sesión de la práctica 9 (P) ►