Comenzado el	martes, 7 de abril de 2020, 17:11
Estado	Finalizado
Finalizado en	martes, 7 de abril de 2020, 20:07
Tiempo empleado	2 horas 56 minutos

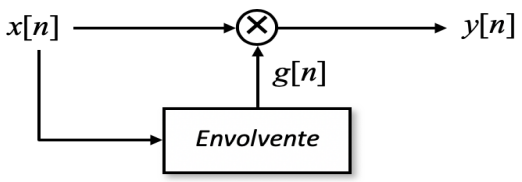
Información

Procesadores de la dinámica

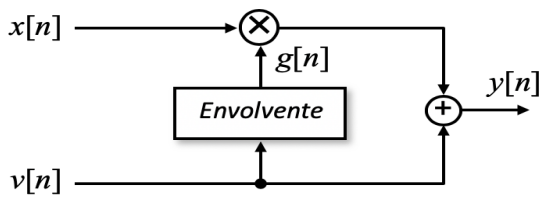
Información

Ducking

La diferencia entre la compresión normal y el *ducking* es que, en este caso, la compresión sobre una señal $x[n]$ es controlada, no por su propia amplitud, sino por la amplitud de otra señal, habitualmente una voz, $v[n]$, para que cuando ésta suene, se comprima la otra "para hacerle hueco" y que se oigan bien ambas (ver diagrama). Es un efecto usado habitualmente en la radio o en la sonorización de contenidos multimedia de todo tipo.



COMPRESIÓN



DUCKING

Ejercicio 7.11:

- Descarga la siguiente PLANTILLA (botón derecho → guardar como)
- Ten en cuenta que, en realidad, el instrumento es un compresor, por lo que si tienes a mano tu código del ejercicio 7.2 puedes usar aquel instrumento para implementar este. **No el fichero ej7.2.csd**, sino sólo el código de **tu** instrumento.
- La idea ahora es aplicar una tasa de compresión R_C grande a una señal *cuando la envolvente RMS de una voz aparte* supere el umbral.

En el instrumento:

- El instrumento se llamará ahora **ducking**
- Añade la lectura con **soundin** en una nueva señal de audio del sonido de la voz que hay en este fichero:

0:00 / 0:00

  El sonido musical a comprimir (cuyo nombre se enviará desde la partitura en **p6**) será ahora esta:

0:00 / 0:00

- Ambas señales (la voz y la música comprimida) deberán sumarse en la salida.
- Hay que hacer cambios importantes en el cálculo de la amplitud: debe calcularse como el valor RMS con el operador **rms** pero aplicado a la voz. Así, lo que condicionará el valor de la ganancia será la voz: comprimiremos la música cuando la amplitud de la voz supere el umbral.

En la partitura:

- Lo primero es oír qué pasa si no procesamos el sonido. Para ello programa esta activación en la partitura:

```
i "ducking"  0  15.0  0  [1/1]  "base.wav"  ; no copies y pegues porque puede dar problemas.
```

  para ver que sin compresión (R_C = 1:1) hay muestras fuera de rango y no se oye adecuadamente la voz.
- Para probar el procesamiento del sonido, cambia los parámetros de la activación para aplicar ahora un umbral de –30 dBFS y una tasa 15:1. Comprobarás la bajada de volumen de la música en presencia de la voz y que todo va bien.

Información

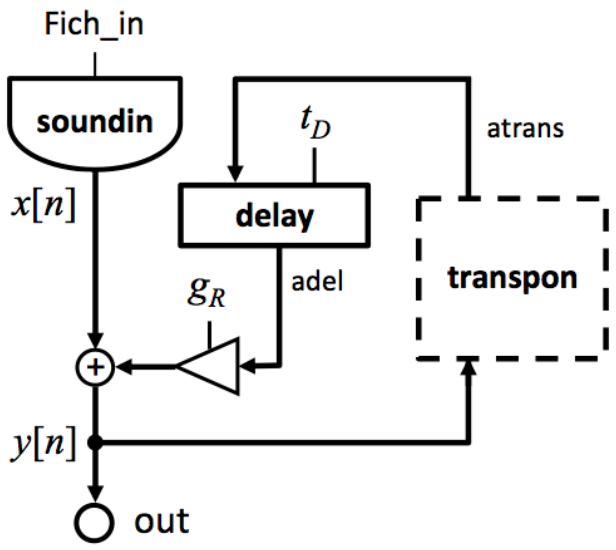
Procesadores del tiempo

Información

Eco múltiple incontable con transposiciones

Ejercicio 7.12

La idea es la misma que la del ejercicio del eco múltiple incontable, pero insertando un transpositor en el lazo de realimentación. Ver figura:



- La transposición se da ya implementada como un UDO (user-defined opcode), listo para ser usado desde otros instrumentos, como:
  - asalida **transpon** aentrada, iRelacion_de_transposicion *; observa los tipos porque deben ser respetados.*
- Utiliza la siguiente PLANTILLA (botón derecho → guardar como)

Instrumento:

- Recibirá desde la partitura:
  - El tiempo de retardo, t_D, en segundos (**p4**),
  - La ganancia de la realimentación, g_R (**p5**), y
  - La relación de transposición, T, en **p6**.
- El funcionamiento del instrumento es el mismo que el del eco múltiple de la parte presencial, pero cada vez que el sonido vuelve por el lazo de realimentación sufrirá una transposición.

Partitura:

- Descarga el sonido a procesar:

  0:00 / 0:00

  que debes cargar con **soundin** en el instrumento.
- Activa el instrumento desde el principio para que lo procese completamente (12 segundos), con t_D = 0.2 s, g_R = 0.85 y transposición de segunda mayor [9/8].

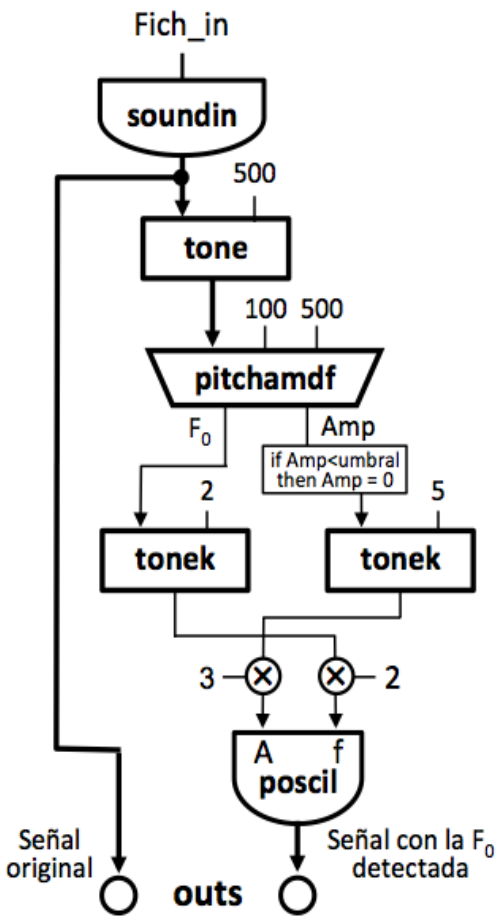
Información

Procesadores de la altura

Información

Ejercicio 7.13: Detección de frecuencia fundamental

- Un importante procesador de la altura es el detector de la f_0, pues esta es una información valiosa tanto para construir analizadores del sonido como para transformar la altura de los sonidos. Vamos a usar una de las herramientas que tiene Csound para hacer uno de estos procesadores.
- Para empezar el proyecto, descarga la PLANTILLA (botón derecho → descargar como).
- El sistema que hay que implementar es el de la siguiente figura:



La idea general es la siguiente: cargamos un sonido musical desde el siguiente fichero:

0:00 / 0:00

El proyecto es estéreo. Por uno de los canales sacaremos la señal original y por el otro, una onda sinusoidal generada según el análisis de frecuencia y amplitud de la señal cargada.

Comentarios e instrucciones para su implementación:

- **tone** es un filtro pasa-baja. Como vamos a buscar la f_0 hasta 500 Hz, con este filtro atenuamos lo que hay por encima, por no ser de interés.
- El operador que realiza el análisis es **pitchamdf**. Necesita saber en qué rango de frecuencias busca la f_0, que será entre 100 y 500 Hz. Este operador saca 2 valores variables (tipo k-, por tanto): la frecuencia fundamental y la amplitud del sonido que analiza.
- Fíjate que la amplitud pasa por un condicional antes de entrar al siguiente operador:

  if Amplitud < Umbral **then** Ampltud = 0

  De esta manera, si la amplitud no es significativa (menor que el umbral) se hace cero.
- Los valores calculados para f_0 y la amplitud pasan por 2 operadores **tonek** que son filtros pasa-baja de variables tipo k-, que consiguen suavizar su evolución. El segundo parámetro de cada uno de ellos son el grado de suavizado en cada caso.
- Finalmente, **poscil** es un oscilador de precisión que genera una onda sinusoidal controlada por los valores de amplitud y frecuencia que le llegan. Así "oiremos" la evolución en el tiempo de la f_0 y podremos compararla con la onda analizada.

Partitura:

- La partitura ya está preparada para activar el instrumento durante la duración del sonido completo.
- Compila y escucha el resultado.

Información

Procesadores del espacio

Información

Ladear un sonido

Este sencillo procesador es capaz de modificar la situación percibida de un sonido. La teoría es la siguiente:

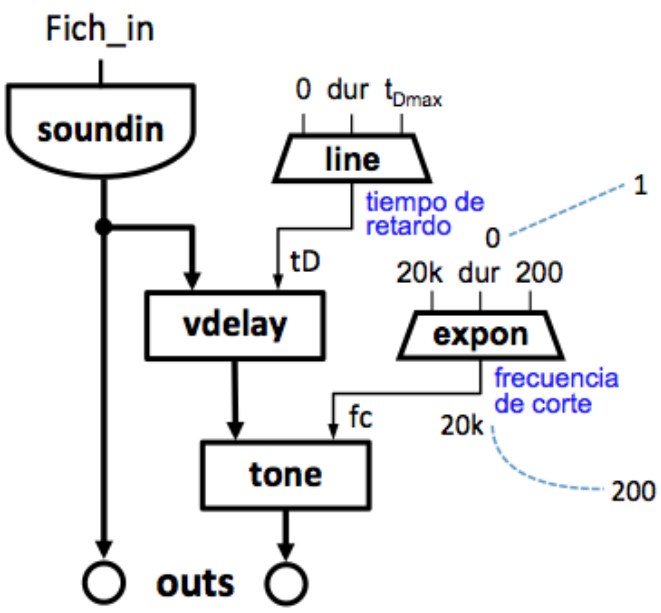
- Un sonido que nos llega desde un lateral llega a un oído antes que al otro y al segundo oído llega además, afectado en sus altas frecuencias por la *sombra acústica* de la cabeza. Por lo tanto, si a uno de los dos canales estéreo le aplicamos un pequeño retraso y un ligero filtro pasa-baja, estaremos simulando las condiciones en las que oímos un sonido que nos llegue de un lateral y, por tanto, deberíamos percibir esa lateralidad.

Ejercicio 7.14:

- Para implementar el instrumento **ladea**, descarga la PLANTILLA (botón derecho → guardar como)
- Esta plantilla contiene la estructura básica del proyecto (que será estéreo) y la partitura preparada para hacer sonar el instrumento durante 7 segundos.

Diseño del instrumento

- Implementa el instrumento, según las especificaciones del esquema.
- Por el canal izquierdo saldrá directamente el sonido leído por soundin. Por el derecho saldrá la señal retardada y filtrada para simular el efecto de la cabeza sobre el sonido para el oído que queda más lejos de la fuente.



- El retardo lo creará el operador **vdelay**. El tiempo de retardo, t_D, será variable, evolucionando desde 0 hasta 1 según una recta creada por **line**. El tercer parámetro de **vdelay** (que no aparece en el gráfico) será t_{Dmax}.
- El filtrado pasa-baja lo hará **tone**, con una frecuencia de corte, f_c, decreciente desde 20 kHz hasta 200 Hz. La curva será exponencial, creada por **expon**.

Prueba

- Descarga este fichero de audio para probar tu sistema:

0:00 / 0:00

- Al probarlo deberás percibir que el sonido inicialmente está centrado y que empieza a desplazarse progresivamente hacia uno de los canales, como si se alejara en la dirección de ese oído. Si lo pruebas con auriculares malos será más dificil apreciar el efecto.

Información

Procesadores del timbre

Información

Phaser

Csound dispone de un operador llamado **phaser1**, que implementa un filtro pasa-toda de orden 1 para construir los procesadores del timbre *phasers*. Este operador tiene los siguientes parámetros de entrada (en este orden):

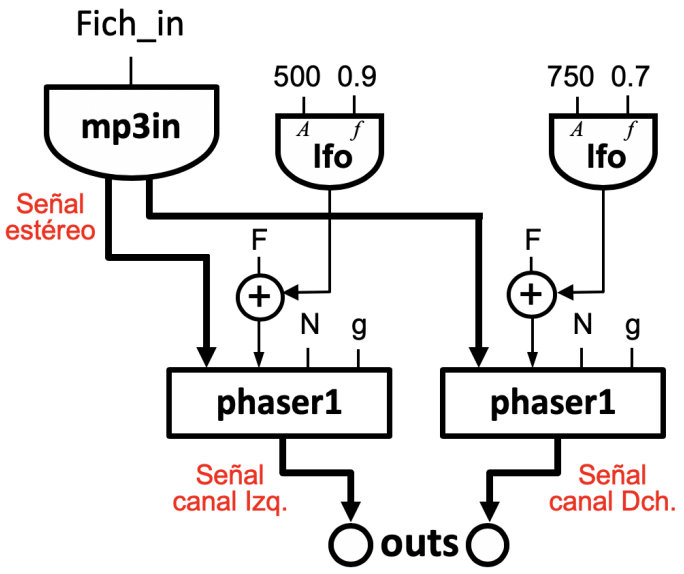
1. El sonido a procesar,
2. La frecuencia característica del filtro, fc, en Hz. En este filtro es la frecuencia a la que la fase se desplaza 90º.
3. El número de filtros iguales, N, que se añaden en cascada. Los phasers suelen utilizar casacadas de pasa-todas para hacer el efecto (que es muy sutil) más acentuado.
4. La ganancia g a aplicar en los lazos directo y de realimentación del pasa-toda, entre 0 y 1.

Ejercicio 7.15:

Desde la partitura se suministrarán algunos de estos parámetros, según se indica en la siguiente: PLANTILLA (botón derecho → descargar como). Es una orquesta estéreo (**nchnls** = 2).

En el instrumento:

- El instrumento deberá llamarse **faser**.
- Recibirá desde la partitura los siguientes parámetros (además de los habituales 3 primeros):
  - **p4**: El nombre del fichero con el sonido a procesar,
  - **p5**: La frecuencia característica del filtro en Hz,
  - **p6**: El número de filtros a aplicar en casacada,
  - **p7**: La ganancia de los filtros.
- La implementación es la que se especifica en la siguiente figura:



- Como se observa, el fichero que se va a procesar es un MP3 estéreo que leerá el operador **mp3in**.
- Cada canal va a pasar por un operador **phaser1**.
- Cada uno de estos operadores tendrá una frecuencia característica modulada por la señal generada por un oscilador de baja frecuencia (**lfo**). Esto hará que esa frecuencia oscile alrededor del valor proporcionado, F. Estas señales serán diferentes para cada canal para incrementar la sensación de estéreo.
- Cada una de las señales procesadas se enviará a cada uno de los canales de la salida estéreo **outs**.

Para la partitura:

- Descarga el sonido a ecualizar en el fichero "guitrim.mp3"

  0:00 / 0:00

  (10.0 s de duración).
- Se harán 3 activaciones del instrumento **faser**, consecutivas de 10 segundos:
  1. Phaser muy suave (un único filtro pasa-toda; N = 1) con: f_c = 1000 Hz y g = 0.3
  2. Phaser intermedio (con N = 5 filtros pasa-toda en casacada) con: f_c = 1000 Hz y g = 0.5
  3. Phaser intenso (con N = 10 filtros pasa-toda en casacada) con: f_c = 1000 Hz y g = 0.7

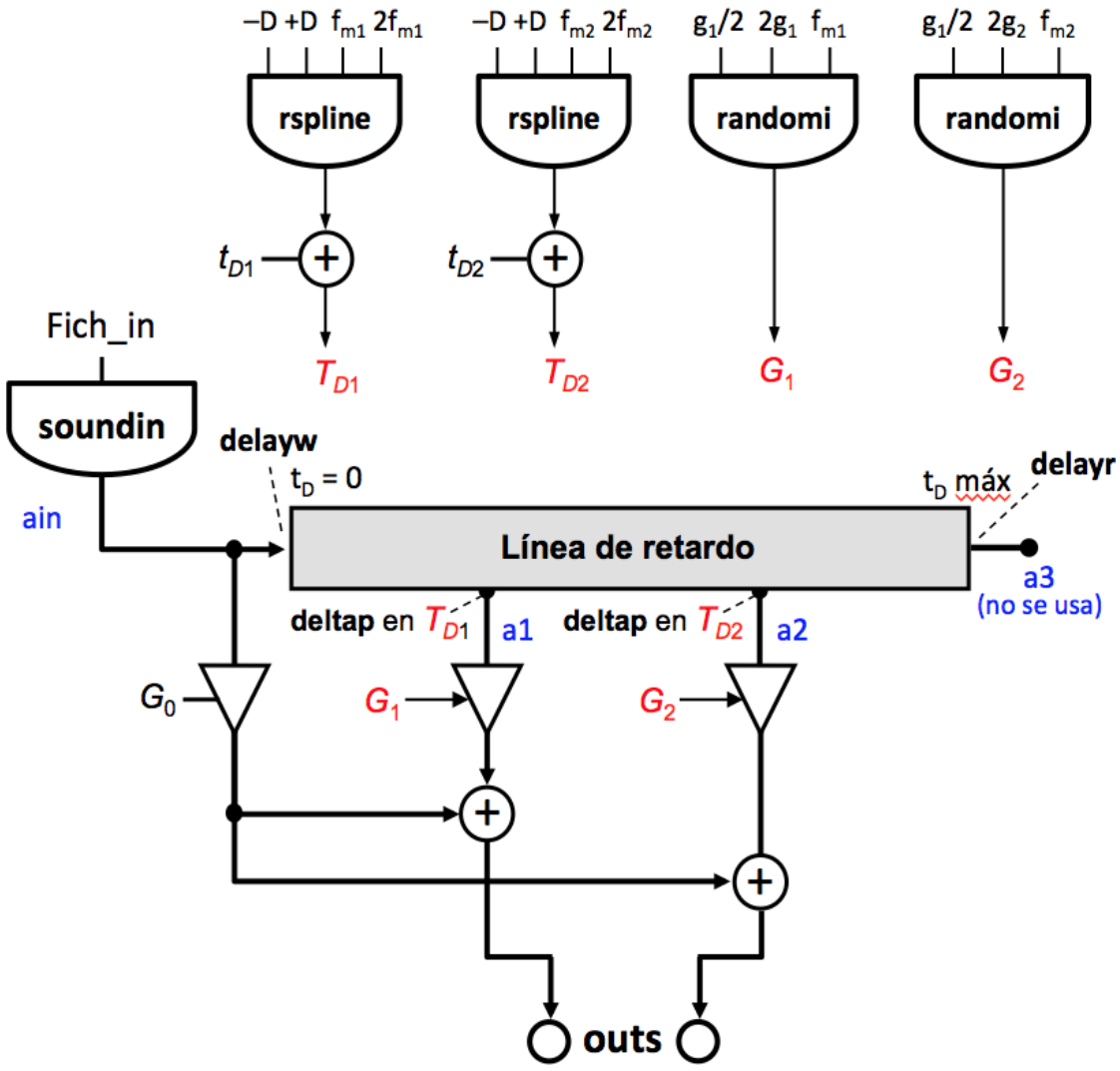
Información

El trabajo de la sesión no presencial ha terminado. Este ejercicio extra se valora aparte.
Si no quieres o no puedes hacerlo, termina la sesión en este momento, sin ninguna repercusión en la nota final.

Chorus (ejercicio adicional extra)

Ejercicio 7.16:

En este ejercicio se trata de implementar un chorus. La idea es simular que un sonido musical de entrada en realidad es tocado por un conjunto de instrumentos iguales. Para ello se generan copias contables, con ligeras desafinaciones y amplitudes aleatorias. El esquema para su implementación en Csound es el siguiente:



Para evitar un exceso de flechas en el diagrama, los parámetros de tiempos de retardo y ganancias variables, marcadas en rojo en la figura, conectan con las que tienen el mismo nombre. Por ejemplo, la G_1 de arriba conecta con la G_1 de abajo.

Los cuatro operadores de la parte superior generan valores de control variables:

- Los dos **rspline** generan curvas suaves aleatorias (*random splines*) que, sumadas a los valores de retardo t_{D1} y t_{D2}, crean los tiempos de retardo variables T_{D1} y T_{D2} (en segundos) en los que se va a leer la línea de retardo para crear las 2 copias. Los valores de estas curvas variarán entre $-D$ y $+D$ (valores de desviación máxima del tiempo de retardo) con frecuencias que también variarán aleatoriamente entre las frecuencias f_m que se indican. La variación aleatoria de estos tiempos creará vibratos aleatorios.
- Los dos **randomi** generan números aleatorios con interpolación lineal entre valores sucesivos, lo que crea líneas continuas entre valores. Se usan para generar ganancias variables aleatorias G_1 y G_2 entre los valores que se indican en las entradas.
- Implementación de la línea de retardo: en el material previo de esta práctica encontrarás la información para implementar, escribir y leer la línea de retardo necesaria (*multi-tap*).

Valores propuestos para los parámetros:

- Tiempos de retardo centrales: t_{D1} = 0.039 s y t_{D2} = 0.027 s
- Tiempo de retardo máximo para dimensionar la línea de retardo: t_D máx = 0.1 s
- Desviación máxima para el tiempo de retardo: D = 0.005 s
- Ganancias:
  - Ganancia para la señal de entrada: G_0 = 1/2
  - Ganancias centrales para las copias retardadas: g_1 = 0.5 y g_2 = 0.4
- Frecuencias de modulación comunes a ambos parámetros: f_{m1} = 0.4 Hz y f_{m2} = 0.7 Hz

Desde la partitura se suministrarán algunos de estos parámetros, según se indica en la siguiente: PLANTILLA (botón derecho → descargar como).

Implementa el programa y pruébalo con el sonido

0:00 / 0:00

La única diferencia entre las 2 activaciones es que la primera tendrá las 2 ganancias = 0.0, así sólo se oirá el sonido sin procesar, mientras que la segunda usará los valores de ganancias propuestos.

◄ Entrega de la segunda sesión P7.2 (P)

Ir a...

Entrega de la parte no presencial de P7 (NP) ►