Práctica 5: audio digital, lectura, descripción y escritura

El objetivo de esta práctica es familiarizarse con las muestras de audio digital, a través del manejo de su almacenamiento en ficheros con formato no comprimido, como el WAV. Haremos programas en **Csound** para leer los parámetros de los ficheros de este tipo y cómo determinan las señales que contienen. Manejaremos las señales al más bajo nivel posible: leyendo y generando las muestras una a una.

PRIMERA PARTE: EL FORMATO DE FICHEROS WAV

El formato WAV es un formato desarrollado por Microsoft como un tipo particular de un formato multimedia más general denominado RIFF (*Resource Interchange Resource Format*). Un fichero WAV tiene un bloque de formato que define los parámetros que caracterizan la señal digital que contiene (ver tabla siguiente).

En este cuadro se puede ver la codificación de los parámetros del bloque de formato:

Descripción	Bytes	Valores de los datos
Firma	4	**'fmt '**
Longitud	4	16 (del resto de bloque)
Tipo de Codificación	2	1 = PCM *
Número canales	2	1 = mono 2 = estéreo
Frec. de muestreo	4	(en Hz)
Flujo (Bytes/s)	4	Frec.muestreo × Bytes/ muestra
Bytes / muestra	2	1 = 8b mono 2 = 8b estéreo/16b mono 4 = 16b estéreo
Resolución en bits	2	8 ó 16

Si es PCM (valor = 1) → los valores se representan por números enteros.

Todos nuestros ficheros serán mono o estéreo

representa el número de bytes necesarios para codificar 1 segundo de la señal

El número de Bytes por muestra depende de la resolución en bits y de si es mono o estéreo

La resolución será de 8 o 16 bits en todos nuestros ficheros

Se analizará este bloque de formato para ver los valores de esos parámetros. Csound proporciona un conjunto de operadores, cuyos nombres comienzan con **file**–, que permiten leer estos parámetros del bloque de formato. Estos operadores tienen todos la misma sintaxis:

```
iValor  operador  iFichero
```

iValor recogerá el valor que el operador extraiga del *iFichero* que se le pase.

Los operadores de Csound que vamos a utilizar son:

- **filebit** — Devuelve el número de bits que codifica las muestras del fichero.
- **filelen** — Devuelve la longitud, en *segundos*, de la señal almacenada.
- **filenchnls** — Devuelve el número de canales del fichero.
- **filesr** — Devuelve la frecuencia de muestreo de la señal contenida en el fichero.

Algunos de los parámetros que caracterizan la señal no se leen directamente del fichero, sino que hay que calcularlos de esta manera:

- Número de Bytes/muestra = NCan × Nbits/8
- Flujo (en Bytes/segundo) = f_s × Bytes/muestra
- Tamaño (en número de muestras) = duración × flujo / (Bytes/muestra)

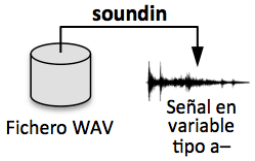
LECTURA Y ESCRITURA DE SEÑALES DIGITALES: BLOQUE DE DATOS

El bloque de datos contiene las muestras codificadas mediante PCM (enteros redondeados del valor del muestreo). Se codifican en bytes mediante el criterio *little-endian* (el primer byte es el menos significativo y los siguientes van multiplicando su peso por 256) en complemento a 2 (el primer bit se reserva para el signo).

Precisión	Bytes	0 dBFS	Rango de valores
8 bits	1 byte sin signo	128	Entre 0 y 255
16 bits	2 bytes con signo	32.768	Entre –32768 y +32767
24 bits	3 bytes con signo	8.388.608	Entre –8388608 y +8388607

En los ficheros en **mono** los valores de las muestras se codifican uno tras otro en un vector de bytes y en los **estéreo** las muestras se codifican alternando los valores de uno y otro canal. En este caso se considera que una muestra está formada por los valores de ambos canales.

Csound permite leer estas señales de diferentes maneras, como por ejemplo mediante el operador **soundin**, cuya salida será una variable de audio (tipo a–).



LEER VALORES DE SEÑALES Y ESCRIBIR VALORES EN SEÑALES

Si usamos *ksmps* = 1 en la cabecera, las muestras de las señales usadas se procesan una a una.

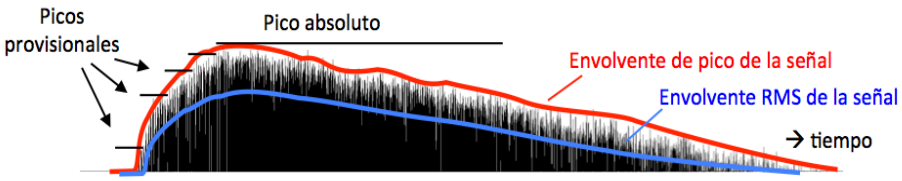
Al hacer una asignación de una variable k– a una de tipo a–, lo que hacemos es "rellenar" una única muestra del vector señal con el valor de la variable. Por ejemplo, `aSalida = kValor` dará valor a una única muestra de la señal de salida cada vez (cada *n*). Si *ksmps* fuera, por ejemplo, 10, esa asignación daría el mismo valor a 10 muestras de la salida.

Esto también funciona al revés: `kValor = aEntrada` da a la variable *kValor* el valor de una única muestra de la señal de entrada cada vez, si *ksmps* = 1. Esto permite leer las señales muestra a muestra.

CÁLCULO DE DESCRIPTORES DE LA SEÑAL

AMPLITUD DE PICO: A_{pico} es el máximo valor de amplitud de la onda en el tiempo. Es un parámetro de sonoridad, útil porque el valor A_{pico} nos permite saber qué margen nos queda hasta llegar al tope de amplitud digital posible. El operador **peak** de Csound calcula este valor en las unidades de amplitud definidas por **0dbfs**.

En la siguiente figura se ilustra cómo el sistema irá detectando picos cada vez que se encuentre un nuevo máximo, actualizando el valor del encontrado anteriormente, hasta llegar a un valor que ya no se supera y que será el valor de la amplitud de pico total para esa onda.



ENVOLVENTE RMS: Una posibilidad es calcular la envolvente como la raíz cuadrada del valor medio de los cuadrados de la señal (RMS) en una ventana de tiempo.

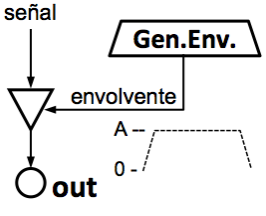
Por ejemplo, si utilizamos ventanas de duración d = 100 ms, la longitud de cada una de ellas en número de muestras será $N = f_s \times d$ = 44100 × 0,1 = 4410 muestras. Por lo tanto, en este caso, el valor de la envolvente RMS en esa ventana que empieza en la muestra n y llega hasta la n+4410 será:

$$e(t)_{n,4410} = \sqrt{\frac{1}{4410}\sum_{m=0}^{4410} s^2[n+m]}$$

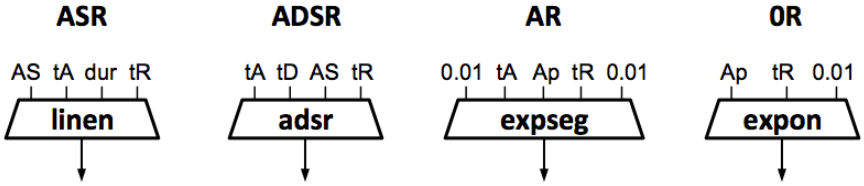
La curva de la envolvente RMS será la sucesión de estos valores para las sucesivas ventanas. El operador **rms** de Csound hace este cálculo.

APLICAR ENVOLVENTES: SONIDOS SIN *CLICKS*

Los *clicks* que aparecen al calcular y escribir señales ocurren porque los sonidos empiezan y terminan bruscamente. Este problema se resuelve aplicando una envolvente a cada nota. Este tipo de envolventes se puede crear en Csound con diferentes operadores, dependiendo del tipo de envolvente deseada. El valor de la envolvente en cada momento se almacena en una variable de tipo control (**k**–) que multiplica la onda antes de enviarla a la salida (ver figura, en la que el triángulo es un multiplicador). La forma de un ejemplo de envolvente ASR se muestra en la curva de puntos (entre 0 y A de amplitud).



El generador de envolventes (Gen.Env.) de la figura, es un operador en Csound a elegir uno de los siguientes, en función del tipo de envolvente (los parámetros que necesita cada operador quedan descritos por los nombres que aparecen en la figura, de izquierda a derecha en el orden en el que los necesita cada uno):



donde tA es el tiempo de ataque, tD el tiempo de caída, AS la amplitud de sostenimiento, tR el tiempo de relajación, dur es la duración de la activación del instrumento (en Csound en **p3**) y Ap es la amplitud de pico. **0R** es una envolvente AR con tiempo de ataque tA = 0

Los 2 primeros operadores generan envolventes con tramos lineales y los 2 segundos con tramos exponenciales (por tanto, lineales en sonoridad). Como los tramos exponenciales no pueden empezar o terminar en un valor cero, por eso se propone el uso del valor 0.01 para principios y finales en estos casos. Los detalles sobre estos operadores están en el manual de ayuda de Csound.

CÁLCULO Y ESCRITURA DE UNA ONDA SONORA

Ser capaces de dar valor a cada una de las muestras individuales, $s[n]$, de una señal nos permite calcularlas mediante funciones periódicas, tipo seno, en función del valor del "tiempo digital", n, que va avanzando paso a paso y que toma sentido físico a través de la frecuencia de muestreo y su inversa, el período de muestreo, de manera que el tiempo real será $t = n/f_s = nT_s$.

Mediante el valor de n podemos calcular una sinusoide digital: $\text{sen}(2\pi fn/f_s)$, siendo la frecuencia de muestreo ***sr*** en Csound. Todo son constantes menos n, que hay que crear como una variable de control (*kn*) que se incrementa en 1 en cada ciclo de Csound y después escribir en la señal el valor calculado por la función seno: **sin**(•). Una vez generada la onda sinusoidal se envía fuera del instrumento mediante **out**.

El proceso de dar valor, usar e incrementar en 1 el contador *kn* que representa el tiempo digital (la n de $s[n]$), se puede hacer de dos maneras en Csound:

Cuenta explícita	Usando operador

`kn  init  0 ; se inicializa`
`. . .      ; se usa`
`kn = kn + 1 ; se incrementa`
	`kn  timek  ; toma valor`
	`. . .      ; se usa`

El operador **timek** cuenta el número de ciclos de control (según *kr*) desde el inicio (**timeinstk** desde cada inicio de la ejecución del instrumento). Si *kr* = *sr* (*ksmps* = 1), entonces el número de ciclos de control coincide con el de muestras y el valor que devuelve ese operador es directamente el contador de muestras, n.

Cálculo y escritura de acordes y escalas

Una vez que podemos calcular una onda sencilla, podemos hacer combinaciones de ondas sinusoidales para crear acordes o ir variando la frecuencia para generar melodías.

Acordes: son varias notas simultáneas, por tanto basta con activar el instrumento generador de ondas sinusoidales simultáneamente, tantas veces como notas tenga el acorde. Cada una de las activaciones debe tener la frecuencia correspondiente a cada una de las notas. Para cada nota, la función seno irá calculando las muestras en función de su frecuencia y las ondas de cada nota se sumarán al sonar juntas, formando el acorde.

Escalas: para generar escalas, las activaciones del instrumento deben ser consecutivas, para crear una sucesión de ondas sinusoidales que constituyan una escala musical.

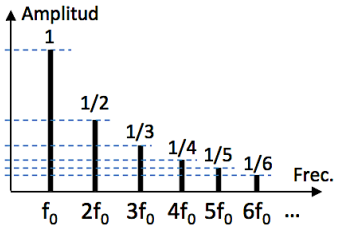
En ambos casos anteriores, lo importante son las relaciones entre la nota raíz o tónica y el resto de notas involucradas. Para ello se pueden usar diferente afinaciones como la justa, pitagórica o bien temperada.

	I	ii	II	iii	III	IV	IV+	V	vi	VI	vii	VII	VIII
Justa	1	16 : 15	9:8	6:5	5:4	4:3	45:32	3:2	8:5	5:3	7:4	15:8	2
Pitagórica	1	256:243	9:8	32:27	81:64	4:3	729:512	3:2	128:81	27:16	16:9	243:128	2
B. T.	1	$2^{1/12}$	$2^{2/12}$	$2^{3/12}$	$2^{4/12}$	$2^{5/12}$	$2^{6/12}$	$2^{7/12}$	$2^{8/12}$	$2^{9/12}$	$2^{10/12}$	$2^{11/12}$	2

CÁLCULO DE ONDAS COMPUESTAS

Las ondas geométricas son las formas de onda que se definen en términos de figuras (cuadrada, triangular, diente de sierra, pulso, etc.). Se pueden generar aproximaciones sumando señales sinusoidales de amplitudes concretas. Por ejemplo, para generar una aproximación a la diente de sierra se usan los *P* primeros armónicos, cada uno de ellos con amplitud igual a la de la fundamental ($p = 1$) dividida por el número (p) del armónico:

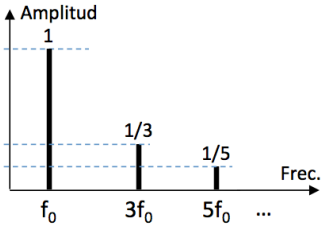
$s[n] = A\sum_{p=1}^{P}\left\{\frac{1}{p}\text{sen}(2\pi(pf_0)n/f_s)\right\}$; así, las amplitudes de los armónicos serían 1, 1/2, 1/3, 1/4, 1/5…



Es mejor generarla normalizada (entre –1 y +1 usando la expresión que hay dentro del sumatorio) y luego multiplicar el resultado por la amplitud *A* deseada, antes de enviarla a la salida.

Para una aproximación a una onda cuadrada con *P* armónicos, hay que usar sólo los armónicos impares, cada uno de ellos con amplitud igual a la de la fundamental dividida por el número del armónico:

$s[n] = A\sum_{p=1}^{P}\left\{\frac{1}{2p-1}\text{sen}(2\pi((2p-1)f_0)n/f_s)\right\}$; por lo tanto, las amplitudes serían 1, 1/3, 1/5, 1/7, 1/9…



ALIASING

Este fenómeno también se puede observar muy fácilmente usando estas señales sinusoidales sintéticas. Cualquier onda digital cuya frecuencia *f* esté por encima de la frecuencia de Nyquist del sistema ($f_s/2$) sufrirá el efecto de plegamiento de las frecuencias o *aliasing*, de forma que, en lugar de la frecuencia original, *f*, se verá representada por una frecuencia f_a cuyo valor será:

$$f_a = |f_s - f|$$

CONTROL DEL FLUJO DE EJECUCIÓN EN CSOUND

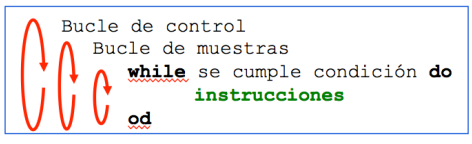
Csound tiene controladores del flujo de ejecución, como bifurcaciones condicionales (**if**), incondicionales (**goto**) y bucles (tipo **while**). Algunas de estas estructuras serán de utilidad en esta práctica.

USO DE CONDICIONALES: Los condicionales (**if**) en Csound se escriben así:

```
if  condición  then          ; la condición se escribe como en C
  sentencias si se cumple la condición
else                         ; o bien puede ser otro:  < elseif condición then >
  sentencias si no se cumple la condición
endif
```

La parte del "**else**" puede no ser necesaria. Para escribir la condición podemos asumir que la sintaxis del lenguaje C (C++ o Java) para expresiones lógicas y comparadores es válida.

USO DE BUCLES: Recuérdese que Csound ya realiza un bucle implícito (el de control), que se realiza *kr* veces por segundo y que, en cada iteración, calcula *ksmps* muestras. Por lo tanto, la definición de un bucle en un instrumento supone "anidar" un nuevo bucle dentro del bucle de control, como se ilustra en la siguiente figura:



El primer bucle es el citado bucle de control, el segundo es el que calcula las muestras correspondientes a cada iteración del bucle de control y el tercero es el que puedas implementar tú en tu código.

Los bucles en Csound se pueden escribir de maneras muy diferentes, por lo que aquí vamos a comentar sólo una de ellas:

```
while  condición  do          ; la condición se escribe como en C y puede ir o no entre paréntesis.
  sentencias mientras que se cumpla la condición
od
```

El funcionamiento es como cualquier bucle de este tipo en C (C++ o Java), recordando que debemos preocuparnos de que la condición deje de cumplirse en algún momento para no entrar en un bucle infinito.