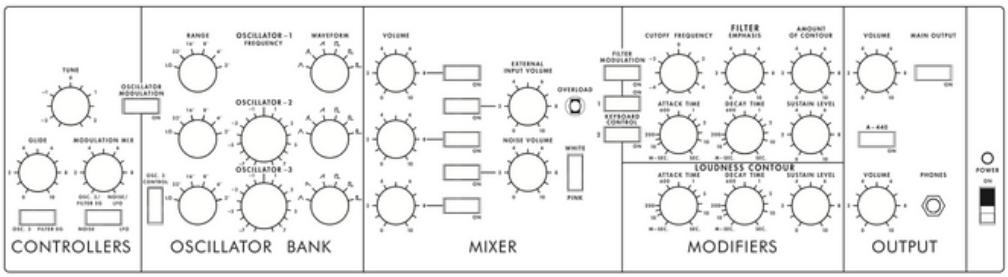
MINI-MINIMOOG

UNA IMPLEMENTAZIONE SEMPLIFICATA ISPIRATA DA UNO STORICO SINTETIZZATORE MEDIANTE CSOUND

di Luca Bimbi

Il Minimoog: lo strumento da studiare e modellare

Il Minimoog, strumento introdotto dalla Moog Music nel 1970, costituì una rivoluzione nell'ambito dei sintetizzatori: era infatti uno strumento portatile, dal costo relativamente contenuto, e facile da capire ed utilizzare.
La tipologia di sintesi implementata rientra nell'ambito della cosiddetta sintesi sottrattiva; tuttavia pregievolissima letteratura preferisce fare riferimento ad essa indicandola come sintesi generatore-modificatore.



Pannello principale del Minimoog

Osservando il pannello primario dei controlli del Minimoog, individuiamo quattro zone principali:

1. OSCILLATOR BANK - tre oscillatori intonabili, per i quali è possibile scegliere una varietà di forme d'onda. Quella che implementeremo nel nostro esempio, per tutti e tre gli oscillatori, è l'onda a dente di sega.
2. MIXER - dove è possibile miscelare sorgenti sonore. La nostra implementazione vedrà un controllo di volume di uscita per ogni oscillatore, con tutti e tre gli oscillatori aperti verso l'uscita stessa.
3. MODIFIERS - Questa sezione comprende un filtro di tipologia Moog Ladder e controlli per la definizione delle caratteristiche di due inviluppi: uno per il filtro, ed uno per l'ampiezza dell'onda in uscita, di tipo ADS (Attack, Decay, Sustain).
4. OUTPUT - per lo stadio di uscita del segnale.

CSOUND

Csound è un linguaggio di programmazione open-source, sviluppato a partire dal 1986; ha una vastissima base utenti mondiale e contiene un alto numero di funzioni e generatori utili alle finalità del sound design e della composizione (sintesi, campionamento, elaborazione digitale, analisi).
È liberamente disponibile per una pluralità di piattaforme; ai fini dei nostri esempi non è strettamente necessario procedere con installazioni di software sui propri dispositivi. Sarà sufficiente ricorrere, con un browser web, alle risorse del sito internet https://ide.csound.com
Un listato Csound include, in un file con estensione csd, diverse sezioni, la cui ossatura è definita mediante markup.

```
; questo è un commento - introdotto mediante l'uso del punto e virgola

<CsoundSyntesizer>  ; inizio del codice
<CsOptions>         ; definizione di opzioni di Csound da eseguire
</CsOptions>        ; fine definizione opzioni
<CsInstruments>     ; definizione strumenti-uno strumento è un insieme di
; generatori – la mentalità alla base del concetto è la modularità
</CsInstruments>    ; fine definizione strumenti
<CsScore>           ; sezione della partitura, dove inserire eventi da
associare a strumenti e funzioni
</CsScore>          ; fine sezione della partitura
</CsoundSyntesizer> ; fine del codice Csound
```

Il template di base utile a cominciare a scrivere codice in Csound, è che costituirà anche il punto di partenza per i nostri esempi è quello che segue:

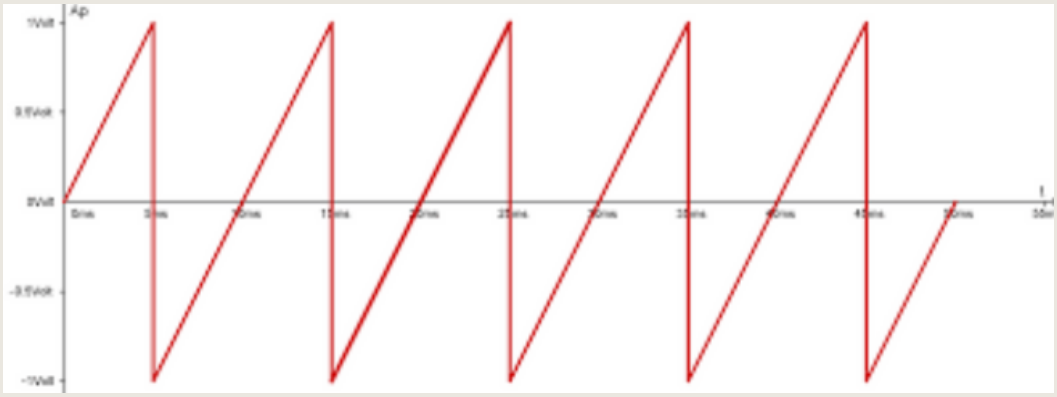
```
<CsoundSyntesizer>
<CsOptions>
-odac     ; istruisce Csound di utilizzare
; il convertitore di uscita della propria interfaccia audio.
; Ciò comporta che l'esecuzione del listato sia udibile in tempo reale.

</CsOptions>
<CsInstruments>
sr = 44100 ; Determina la frequenza di campionamento da usare.

ksmps = 64 ; La dimensione del vettore audio è formata da 64 samples
; Definisce Il numero di campioni in un periodo di controllo.
; La computazione dei segnali audio è eseguita in blocchi della
dimensione
; definita per mezzo di questa variabile.
; ksmps deve essere *sempre* un numero intero.
; ksmps = sr / kr (sample rate diviso control rate) - in questo caso
; il control rate è uguale a 689,0625.
; Storicamente, il control rate è usato per variabili il cui valore
; non necessiti di essere aggiornato ad elevata frequenza.
; Niente toglie che sussistano circostanze dove il control rate
; ed il sample rate addirittura possano coincidere,
; determinando un “vettore audio” di dimensioni pari ad un sample.
; Si pensi alle ipotesi di implementazione diretta di filtri digitali.

nchnls = 2     ; Uso di due canali in uscita sull'interfaccia audio.

0dbfs  = 1     ; Valore massimo di ampiezza nella scala digitale

</CsInstruments>
<CsScore>
</CsScore>
; Non faremo uso della sezione score, prevedendo lo scheduling degli
; eventi all'interno della sezione della definizione degli strumenti
</CsoundSyntesizer>
```

Csound fa differenza fra maiuscole e minuscole e fra tre tipi di variabili principali, che si distinguono per la differente lettera con cui inizia il nome della variabile stessa: variabili i, variabili k, variabili a.
Le variabili i sono definite al tempo di inizializzazione del codice, le variabili k sono aggiornate alla frequenza del control time ed infine le variabili a sono calcolate ed aggiornate alla frequenza di campionamento. Di fatto, Csound processa l’input ed output a control rate.
Le variabili a sono vettori, che contengono un gruppo sequenziale di campioni come impostato dalla variabile globale ksmps. Se kr equivale sr, entrambi i valori sono di natura scalare. Queste tipologie di variabili possono essere precedute dalla lettera g, che le rende globali. Ciò significa che lo scope standard di una variabile è legato al suo ambito, salvo che sia definita come variabile di tipo globale e quindi accessibile a tutta la sezione CsInstruments.

Queste tipologie di variabili possono essere precedute dalla lettera g, che le rende globali. Ciò significa che lo scope standard di una variabile è legato al suo ambito, salvo che sia definita come variabile di tipo globale e quindi accessibile a tutta la sezione CsInstruments.

Infine, preliminarmente, specifichiamo che Csound è composto da funzioni ed opcode. Le funzioni restituiscono un valore su chiamata, mentre gli opcode una volta instanziati hanno uno stato di esecuzione; ciò avvicina gli opcode stessi al concetto di metodo nella programmazione ad oggetti. Un esempio di funzione è ampdbfs(), che converte il valore ricompreso fra parentesi espresso in deciBel Full Scale in un valore lineare ricompreso fra zero ed il valore massimo di 0dbfs (che definiamo come pari ad 1 nei nostri listati); un esempio di opcode, è oscili, che legge una tabella descrivente una funzione ad una determinata frequenza in Hertz, per un determinato valore di ampiezza e ne salva il contenuto in una variabile di tipo a.

LA FORMA D'ONDA A DENTE DI SEGA



Come noto, la forma d'onda a dente di sega costituisce una delle possibili forme d'onda elementari. La sua caratteristica è data dalla presenza di armoniche pari e dispari, come multipli interi della fondamentale, la cui ampiezza decade come il reciproco del numero della parziale. Si avranno, quindi, per le prime dieci armoniche, i seguenti valori di ampiezza:

1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, 1/9, 1/10

Questa forma d'onda può essere implementata in Csound ricorrendo al sistema del table-lookup, mediante una funzione chiamata GEN10.
Il metodo del table-lookup è un sistema efficiente che non richiede una computazione in tempo reale delle forme d'onda necessarie. Una Table è composta di un certo numero di punti (in Csound, salvo che per l'uso dell'opcode poscil, tale numero è potenza di due o potenza di due più uno). La Table è determinata da una serie di indici a valore intero, a cui corrispondono i valori della funzione definita.

Usare una Table al fine di ottenere un segnale, comporta che si debba variare l'incremento di lettura dei valori al variare della frequenza. La formula dell'incremento è la seguente:

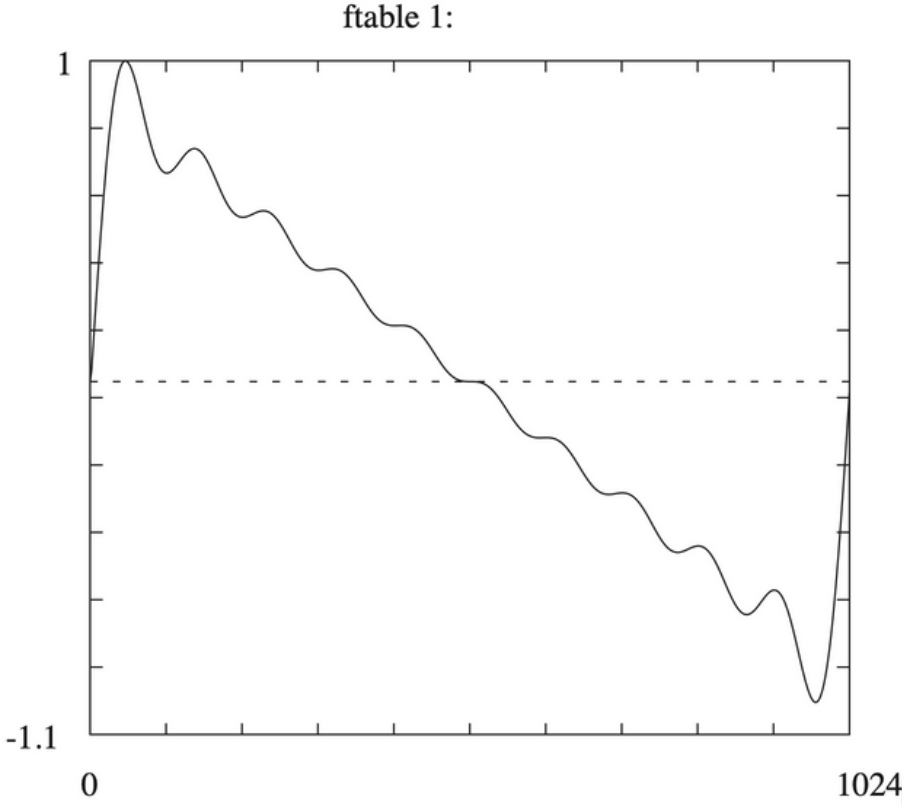
$$Incremento = \frac{Npunti}{sr} \times frequenza$$

Dove Npunti rappresenta il numero di punti della Table, sr la frequenza di campionamento e frequenza la frequenza richiesta – che nel nostro esempio determinerà l'altezza della fondamentale.
E' facile notare come, per quanto di regola gli indici siano rappresentati da numeri interi, il valore di incremento di lettura può essere frequentemente un numero non intero. Si ricorre quindi, nella pratica di utilizzo a varie tipologie di interpolazione dei valori. Nel nostro esempio utilizzeremo l'opcode oscili che prevede la lettura della Table con interpolazione lineare.
Vediamo quindi, in maniera legata al concetto della sintesi additiva, come generare un'onda a dente di sega con Csound, utilizzando la funzione GEN10, mediante una Table di piccole dimensioni (1024 punti). Tale Table verrà generata tramite ftgen, che accetta come argomenti il numero di funzione da assegnare, il tempo in cui deve essere generata dall'inizio dell'esecuzione del codice, la dimensione della tabella stessa, il numero di GEN da utilizzare ed a seguire i parametri richiesti per il GEN utilizzato. GEN10 richiede che si riportino i valori di ampiezza delle parziali. Per migliorare la leggibilità del codice alla funzione generata assegneremo una variabile globale ad init-time, chiamata giSawtooth.

Ricorreremo ad un inviluppo di uscita di forma trapezioidale, attraverso l'opocode linen e ricaveremo il valore di ampiezza massimo mediante la funzione ampdbfs per un valore di -10 dBFS. L'altezza da eseguire, 440 Hz, verrà passata come parametro (p4) di scheduling per l'esecuzione dell'instrument PlaySaw.

```
<CsoundSynthesizer>
<CsOptions>
-odac
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 64
nchnls = 2
0dbfs  = 1

; FORMA D'ONDA A DENTE DI SEGA COMPOSTA DA 10 ARMONICHE usando GEN10
; nomefn         n  t  dim   GEN a1 a2   a3   .....
giSawtooth ftgen 1, 0, 1024, 10, 1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8,
1/9, 1/10

instr PlaySaw

; variabili i - definite a tempo di inizializzazione

  iAmpiezza = ampdbfs(-10)
  iDurata  = p3
  iAltezza = p4

; Inviluppo di ampiezza trapezioidale (opcode linen):
; Il 10% della durata totale del suono è il tempo di attacco e
; di rilascio.
; La durata viene passata come terzo valore nello scheduling (p3).
; L'altezza da generare, in questo caso 440 Hz, come quarto parametro
; dello scheduling (p4).
; L'ampiezza massima iAmpiezza per il nostro esempio è definita nella
; scala digitale con un segnale pari a -10 dBFS.
; Tale inviluppo ha valori che
; vengono aggiornati a control time (variabile k)

  kInviluppo linen iAmpiezza, iDurata*0.1, iDurata, iDurata*0.1

; Nella variabile aUscita, i valori generati dall'oscillatore,
; per l'ampiezza kInviluppo, a iAltezza, utilizzando la table giSawtooth
; che descrive l'onda a dente di sega

  aUscita   oscili kInviluppo, iAltezza, giSawtooth

  out aUscita, aUscita ; output del contenuto di aUscita sui due canali
endin

; p1 definisce lo strumento da eseguire
; p2 da quale tempo eseguirlo dall'avvio del codice
; p3 il tempo di esecuzione (durata)
; ed in questo caso il parametro opzionale p4, l'altezza
; che viene passata all'oscillatore.

;           p1     p2  p3  p4
schedule("PlaySaw", 0, 5, 440)
</CsInstruments>
</CsoundSynthesizer>
```

Questo codice genera quindi un'onda a dente di sega e la manda in uscita su due canali di uscita della scheda audio, per la durata di 5 secondi ed una altezza della fondamentale di 440 Hz.



Approssimazione di onda a dente di sega con dieci armoniche, usando GEN10

Per la nostra implementazione ricorreremo all'opcode vco2, che ci evita di incorrere in problemi quali il Foldover (forma di distorsione del segnale nello spettro udibile per effetto del superamento della cosiddetta Frequenza di Nyquist, pari alla metà della frequenza di campionamento). In base all'altezza della fondamentale, infatti, almeno una armonica, in base al numero di armoniche definite nella funzione, potrebbe eccedere la Frequenza di Nyquist e costituire, quindi, un problema concreto.

GLI OSCILLATORI ED IL MIXER

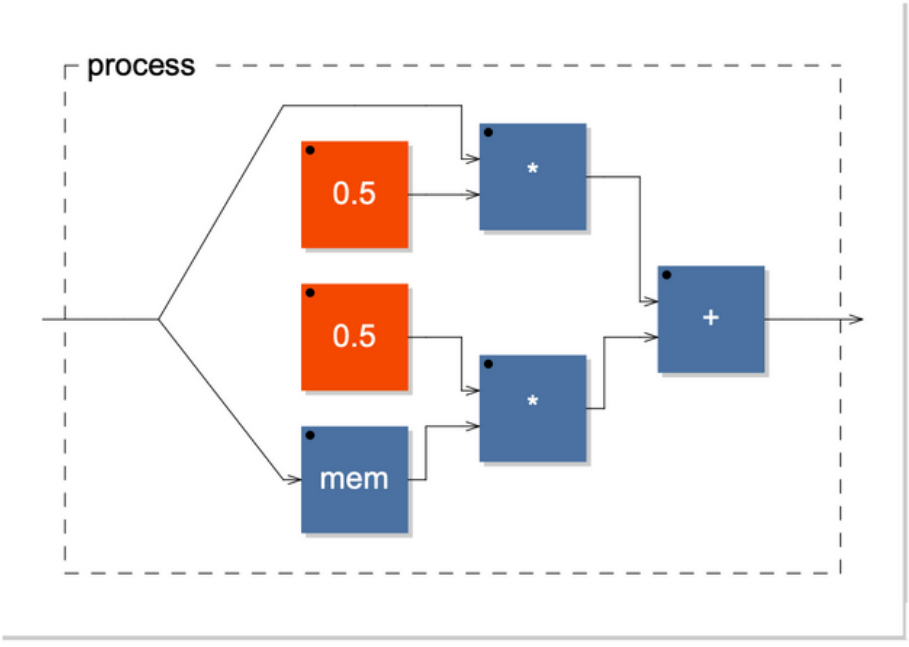
Scriviamo del codice che implementi una versione semplificata dell'OSCILLATOR BANK del Minimoog. Assumiamo di utilizzare solo l'onda a dente di sega per tre oscillatori; ammettiamo la possibilità di "scordarli" (tecnica del detuning) per rendere meno statico il suono risultante dalla somma degli stessi. Provvediamo quindi alla somma dei tre segnali risultanti dall'opcode vco2 e mandiamola in uscita sui canali 1 e 2 della scheda audio. Assumiamo al momento un valore di ampiezza per il livello di uscita pari a -10 dBFS. Ricaveremo il detuning come percentuale di semitono, avvelendonci della funzione semitone(), che ritorna per un certo numero di semitoni un fattore da applicare ad un valore in Hertz. Moltiplicando il valore di un semitono per il numero decimale corrispondente alla percentuale desiderata, otterremo la differenza necessaria ad effettuare la scordatura. Nel nostro esempio, ammettiamo per il VCO2 un detuning del 3.59% negativo, e per il VCO3 del 7.13% positivo.

```
<CsoundSynthesizer>
<CsOptions>
-odac
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 64
nchnls = 2
0dbfs  = 1

instr MiniMinimoog

  iDurata = p3
  iAltezza = p4

; VCOs BANK
;----------------------------------

  iDetuningVCO2 = semitone(1*0.0359)
  iDetuningVCO3 = semitone(1*0.0713)

; a-time     livello   altezza
  aVCO1 vco2 0dbfs,   iAltezza

;                              scordatura
  aVCO2 vco2 0dbfs, iAltezza-iDetuningVCO2
  aVCO3 vco2 0dbfs, iAltezza+iDetuningVCO3

; MIXER
;------------------
; somma dei tre VCO

  aMix = (aVCO1+aVCO2+aVCO3)/3

; OUTPUT SCALATO A -10 dBFS
;--------------------------
  out aMix*ampdbfs(-10), aMix*ampdbfs(-10)
endin
; Scheduling               p3   p4
schedule("MiniMinimoog", 0, 5, 440)
</CsInstruments>
<CsScore>
</CsScore>
</CsoundSynthesizer>
```

Il presente codice provvede quindi ad emulare i tre VCO accordati sulla stessa ottava, con applicato detuning, emularne il missaggio (somma del 100% dei tre segnali), scalare il segnale risultante a -10 dbFS (aMix*ampdbfs(-10)), eseguendo un LA 440 Hz per la durata di 5 secondi dal tempo di esecuzione 0 (tempo di avvio).

IL FILTRO MOOG LADDER

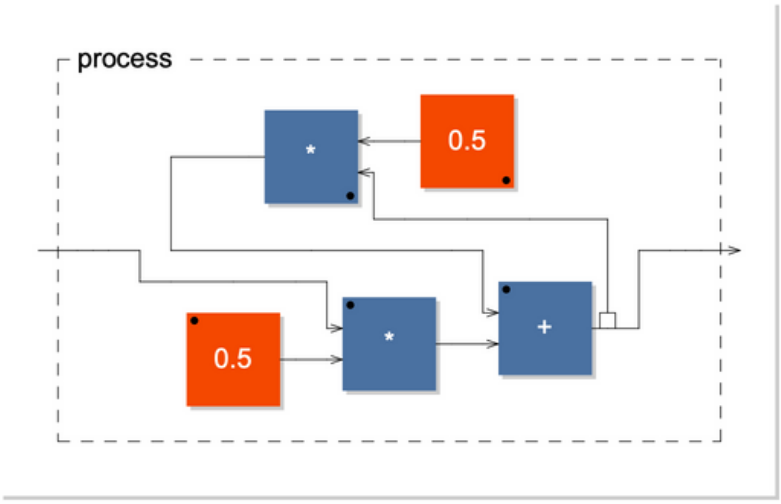
Dopo aver scritto il codice del "generatore", vediamo le caratteristiche del "modificatore" del Minimoog: il filtro Moog Ladder, brevettato dal Dr. Robert Moog nel 1966. Tale tipologia di filtro è caratterizzata, nella sua essenza e tralasciando le non-linearità indotte dalla possibile distorsione, da 4 filtri passa-basso IIR (Infinite Impulse Response) in serie del primo ordine con ripidezza di 6 dB/ottava (determinando quindi una ripidezza complessiva di 24 dB/ottava), con percorso di feedback dall'uscita verso l'ingresso per implementare il controllo di enfasi o risonanza. Tutto ciò per i neofiti può sembrare estremamente complesso.
Definiamo i filtri in base alle loro caratteristiche.
Un filtro passa-basso preserva le frequenze al di sotto di una frequenza di taglio, ed attenuerà secondo la sua ripidezza il segnale al di sopra di tale frequenza. In tal caso, un filtro passa-basso con ripidezza di 6 db/ottava settato su un segnale di rumore bianco alla frequenza di 1 Khz, attenuerà progressivamente il segnale al di sopra di tale frequenza, e nello specifico di 6 dB il segnale a 2 Khz. Un filtro con attenuazione di 6 dB/ottava è detto del primo ordine. I filtri si distinguono inoltre in FIR (Finite Impulse Response) ed IIR (Infinite Impulse Response), a seconda se includano un flusso di feedback (IIR), oppure no (FIR). Digitalmente, i filtri vengono implementati mediante l'utilizzo di ritardi (delay).
Vediamo quindi un esempio di filtro FIR, dove nel diagramma mem rappresenta il delay di 1 campione e il segnale originale e ritardato si sommano in uscita



Filtro FIR

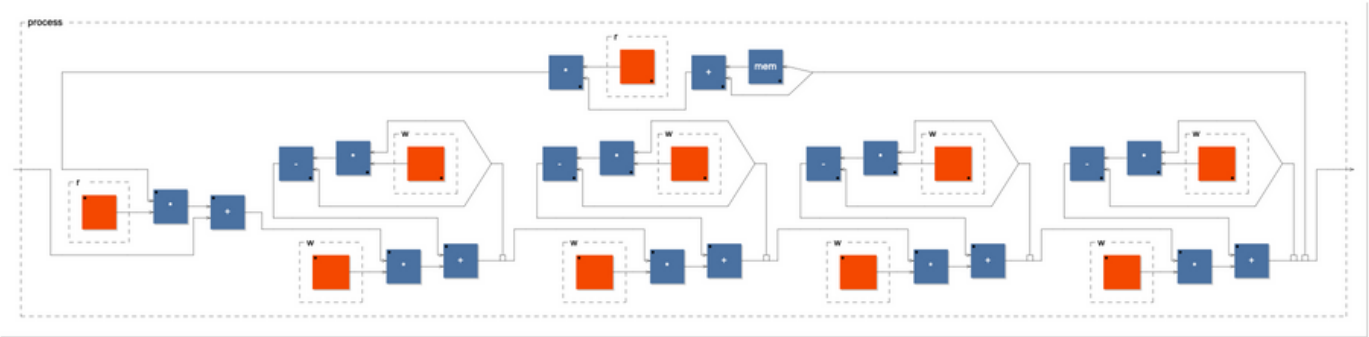
IL FILTRO MOOG LADDER

E adesso un esempio di filtro IIR, dove mediante un meccanismo di feedback, il delay di 1 sample (indicato dal piccolo rettangolo nel percorso di feedback) viene sommato in ingresso.



Filtro IIR

Il filtro Moog Ladder, come dicevamo, sarà quindi del seguente tipo:



Moog Ladder: 4 Filtri IIR in serie con Feedback dal'uscita all'ingresso per resonance

IMPLEMENTAZIONE FINALE DEL MINI-MINIMOOG, CON INVILUPPI DEL FILTRO E DI USCITA

Non ci resta quindi che inserire il codice per il filtro, ed in Csound abbiamo a disposizione un opcode che emula il filtro Moog Ladder chiamato moogladder2, che accetta un segnale in ingresso, una frequenza di taglio ed un valore di risonanza.
Definiamo poi gli inviluppi di uscita e del filtro, di tipo ADSR (Attack, Decay, Sustain, Release). L'opcode adsr serve proprio a questo, e ci consente di determinare il tempo di attacco dell'inviluppo, il tempo di decadimento, il livello di sustain ed il tempo di rilascio. Nel nostro esempio, definiamo il tempo di attacco per l'inviluppo di uscita come il 10% del tempo di esecuzione totale, un decadimento pari a zero, un livello di sustain pari al massimo possibile nella scala digitale (0dbfs), ed un tempo di rilascio dell'1%. Per l'inviluppo del filtro, un tempo di attacco del 50% del tempo di esecuzione, un decadimento pari a zero, un livello di sustain di 0 dBFS ed un tempo di rilascio pari a zero.
L'inviluppo di uscita kInviluppoUscita sarà moltiplicato all'ampiezza dei singoli VCO (questo potenzialmente ci consente di definire fino a 3 inviluppi di uscita, uno per ciascun VCO, a differenza del Minimoog che ne ha uno solo) determinando l'evoluzione dell'ampiezza degli stessi nel tempo di esecuzione; l'inviluppo del filtro kInviluppoFiltro sarà invece moltiplicato alla frequenza di taglio del filtro nell'opcode moogladder2, determinando la variazione del filtro nel tempo secondo i parametri definiti nel rispettivo adsr.

Questo modello elementare, oltre che esemplificare i meccanismi di funzionamento di base della sintesi generatore-modificatore e nello specifico ispirandosi alla forma presentata della stessa dal Minimoog, può essere espanso in molte direzioni. Si lascia spazio alla creatività e volontà del lettore per integrare il codice e farne gli usi che più lo aggradino, partendo dalla modifica dei parametri definiti nell'esempio in relazione alle altezze, le durate, le forme d'onda, il detuning, l'ADSR, la frequenza di taglio del filtro, il valore di resonance. Si ricorda infine la possibilità di effettuare scheduling multipli per lo stesso tempo, che pure si sovrappongano.

Si suggerisce, in tal caso, la lettura della seguente pagina del reference manual di Csound:
http://www.csounds.com/manual/html/vco2.html

Codice completo

```
;============================================================================
; Mini-Minimoog - esemplificazione di "Sintesi Sottrattiva"
; (Generatore/Modificatore) attraverso un modello ridotto di Moog
Minimoog
; in Csound 6.
; ---------------------------------------------------------------------
; Luca Bimbi, 2023
; Classi di Informatica Musicale, Conservatorio S. Giacomantonio, Cosenza
<CsoundSynthesizer>
<CsOptions>
-odac
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 64
nchnls = 2
0dbfs  = 1

instr MiniMinimoog

  iDurata = p3
  iAltezza = p4

; DEFINIZIONE INVILUPPI
;----------------------------------------------------------------------


; k time           adsr  t-attack       decay  livello sustain
release
  kInviluppoUscita adsr   iDurata*0.1,     0,      0dbfs,
iDurata*0.01
; k time           adsr t-attack        decay     livello
release
  kInviluppoFiltro adsr iDurata*0.5,       0  ,   0dbfs,
0

; VCOs BANK
;-----------------------------------

  iDetuningVCO2 = semitone(1*0.0359)
  iDetuningVCO3 = semitone(1*0.0713)

; a-time      inviluppo uscita   altezza
  aVCO1 vco2 kInviluppoUscita, iAltezza
;                                      scordatura
  aVCO2 vco2 kInviluppoUscita, iAltezza-iDetuningVCO2
  aVCO3 vco2 kInviluppoUscita, iAltezza+iDetuningVCO3

; MIXER
;-------------------------------------
; somma dei tre VCO -> verso il filtro

  aMix = (aVCO1+aVCO2+aVCO3)/3

; FILTRO
;----------------------------------------------------------------------
;                segnale in ingresso    cutoff con inviluppo  resonance
  aOut moogladder2    aMix,            2100*kInviluppoFiltro,       0.6

; OUTPUT
;---------
  out aOut, aOut
endin
schedule("MiniMinimoog", 0, 5, 440)
</CsInstruments>
<CsScore>
</CsScore>
</CsoundSynthesizer>
```