GSC4 - SINTESI GRANULARE PER CSOUND

Eugenio Giordani

Laboratorio Elettronico per la Musica Sperimentale
Conservatorio di Musica G. Rossini
Pesaro

Introduzione

L'idea di scrivere un programma per la Sintesi Granulare in ambiente Csound è nata quasi come una sfida verso una tipica idiosincrasia dei linguaggi Music-n relativa alla propria impostazione basata sul dualismo orchestra-partitura. In generale questa filosofia di fondo riproduce, sebbene con evidenti differenze, un concetto di costruzione musicale legato alle note (note-oriented) e per questo lineare. Nei linguaggi Music-n, sebbene una nota possa sottendere algoritmi molto elaborati, viene impiegata generalmente una relazione paritetica tra *orchestra* e *score* : una linea di partitura (nota) produce un oggetto sonoro, complesso a piacere, ma un solo oggetto. Se l'algoritmo è molto complesso, le linee di partitura possono essere poche e molto rarefatte, anche se ciò non rappresenta di certo una regola.
Nella Sintesi Granulare, che come è noto necessita di una grandissima quantità di frammenti sonori (grani o quanti acustici) per unità di tempo, sarebbe impensabile demandare allo *score* la generazione dei singoli grani, sia per evidenti questioni di praticità sia perché non sarebbe agevole governare i parametri della sintesi ad alto livello. L'obiettivo è stato quindi quello di automatizzare il più possibile la generazione dei singoli grani, e di lasciare quindi al musicista la possibilità di controllare globalmente gli andamenti di tutti i parametri della sintesi. Per fare questo è stato necessario pensare ad una struttura algoritmica dell'*orchestra* in grado di generare all'interno di una singola nota, un numero molto grande di "*sub-note"*, intendendo con questo termine microeventi acustici di durata generalmente molto limitata. Lasciando al lettore la facoltà di approfondire l'organizzazione progettuale e i dettagli di programmazione relativi all'*orchestra*, viene data di seguito una descrizione della sola utilizzazione del programma in termini di programmazione della *score* così che sia possibile la sperimentazione immediata dell'algoritmo.

Struttura generale dell'algoritmo e descrizione dei parametri della sintesi

GSC4 è in grado di generare 4 flussi indipendenti (voci) di grani sonori su un fronte stereofonico, ma è possibile per l'utente l'estensione dell'algoritmo ad un numero maggiore di voci e di canali. La ragione per cui la versione base è a 4 voci dipende dal fatto che questa configurazione consente allo *score* di essere eseguita in tempo reale con le versioni recenti di Csound per Windows 95.
Seguendo questa configurazione di base, l'*orchestra* è costituita da 4 strumenti di generazione, uno strumento per le funzioni di controllo (comune a tutti gli strumenti di generazione) e infine uno strumento per il mix e l'uscita e cioè precisamente:

instr 1, 2, 3, 4 : strumenti di generazione
instr 11 : strumento per il controllo
instr 21 : strumento per il mixing e l'uscita

Data questo schema, per generare un evento completo occorre quindi attivare contemporaneamente 6 strumenti ed attribuire ad essi lo stesso tempo di start (p2) e di durata (p3).
Il maggior numero di parametri (fino a p13) è concentrato nello strumento per il controllo mentre gli altri strumenti contengono solo 4 parametri (da p1 a p4). Il controllo del processo di granulazione avviene attraverso la specificazione da parte dell'utente di un certo numero di funzioni che descrivono l'andamento nel tempo dei parametri di sintesi così definiti ed associati allo strumento 11 (controllo):

1) durata media dei grani in ms p4
2) variazione random della durata dei grani in ms p5
3) delay medio dei grani in ms p6
4) variazione random del delay dei grani in ms p7
5) proporzione di rampa in unità adimensionali p8
6) frequenza centrale della forma d'onda audio in hz p9
7) variazione random della frequenza della forma d'onda in hz p10
8) fase centrale della forma d'onda (normalizzata) p11
9) variazione random della fase della forma d'onda (normalizzata) p12
10) ampiezza complessiva (normalizzata) p13

Ciascuno di questi valori indica il numero di funzione assegnata per il controllo del parametro associato per cui, durante l'attivazione dello strumento, devono esistere all'interno dello *score* 10 funzioni tabulate con uno dei metodi di generazione disponibili (GEN i) .
Se ci riferisce al listato relativo allo *score* riportato in Appendice, possiamo osservare le funzioni di controllo dei vari parametri della sintesi.
f11 :
la durata media dei grani è definita da una funzione lineare (GEN 7) con valore iniziale di 10 ms che dopo 256/512 di p3 (durata complessiva dell'evento) raggiunge il valore di 20 ms, rimane costante per 128/512 e tende al valore finale di 16 ms dopo 128/512.

f12:
il valore massimo della variazione random della durata dei grani è definito da una funzione lineare (GEN 7) con valore iniziale di 4 ms che dopo 256/512 di p3 si riduce a 1 ms e dopo 256/512 tende al valore finale di 0 ms (nessuna variazione random).

f13:
il delay dei grani è definito da una funzione lineare (GEN 7) con valore iniziale di 10 ms che dopo 256/512 di p3 aumenta a 20 ms e dopo 256/512 tende al valore finale di 5 ms.

f14:
il valore massimo della variazione random del delay dei grani è definito da una funzione lineare (GEN 7) con valore iniziale di 0 ms (nessuna variazione random) che dopo 128/512 di p3 rimane costante a 0 ms; dopo 256/512 sale a valore a 2 ms e dopo 128/512 tende al valore finale 0 ms (nessuna variazione random).

f15:
la proporzione di rampa dei grani è definita da una funzione lineare (GEN 7) con valore iniziale di 2 che dopo 256/512 di p3 aumenta a 4 e dopo 256/512 tende al valore finale di 2. In pratica, la forma dell'inviluppo dei grani si trasforma gradualmente da un triangolo iniziale verso un trapezio e di nuovo un triangolo. Nel triangolo, la rampa di salita è esattamente pari alla metà della durata dell'inviluppo mentre la seconda metà dell'inviluppo è pari alla rampa di discesa (la fase di sustain è nulla). Nel trapezio si arriva ad una rampa di salita (e discesa) pari ad 1/4 della durata dell'inviluppo (la fase di sustain non è nulla e vale 2/4 la durata dell'inviluppo).

f16:
la frequenza della funzione audio è definita da una funzione lineare (GEN 7) con valore iniziale di 220 hz che rimane costante per tutto l'intero evento. Nello score è riportata una linea (commentata) relativa all'andamento di valori di controllo per la frequenza (da 1.345 a 3.345) quando venga impiegato come materiale audio il file esterno sample.wav (vedi f1) di lunghezza 32768 campioni. In tal caso il valore iniziale 1.345 deriva proprio dal rapporto 44100/32768 e rappresenta la frequenza originale del campione."

f17:
il valore massimo della variazione random della frequenza della funzione audio è definito da una funzione lineare (GEN 7) con valore iniziale 0 hz (nessuna variazione) e che tende dopo p3 al valore finale 110 hz (equivale al 50 % di modulazione)

f18:
la fase della funzione audio è definita da una funzione lineare (GEN 7) con valore iniziale di 0 che rimane costante per tutto l'intero evento.

f19:
il valore massimo della variazione random della fase della funzione audio è definito da una funzione lineare (GEN 7) con valore iniziale 0 (nessuna variazione) e che rimane costante per tutto l'intero evento.

f20:
l'ampieza complessiva è definita da una funzione lineare (GEN 7) con valore iniziale di 0 che dopo 128/512 di p3 sale a 1, rimane costante per 256/512 e tende a 0 dopo 128/512.

f1:
la forma d'onda audio è definita da una funzione somma di seni (GEN 10) con un contenuto spettrale definito dai coefficienti di ampiezza delle singole componenti

E' opportuno notare che, ad eccezione delle funzioni f1 ed f20, il parametro p4 è sempre negativo (-7) poichè i breakpoints devono esprimere i valori dei vari parametri in una scala assoluta.
Per i 4 strumenti di generazione a cui si riferiscono i1,i2,i3,i4, è sufficiente specificare oltre ai 3 parametri obbligatori p1,p2 e p3, il numero della funzione audio (f1 in questo esempio) e per lo strumento 21 il fattore di scala all'uscita.
La funzione audio può essere indifferentemente un prototipo di forma d'onda periodica o un segnale campionato importato con la funzione GEN 1. Nel primo caso il valore del parametro p4 dello strumento 21 deve contenere l'ampiezza massima all'uscita (da 0 a 32767) , mentre nel secondo il valore è definito nel range 0:1. Ciò dipende dal fatto che generalmente il segnale audio campionato non viene rinormalizzato in sede di lettura (GEN -1).
Come già accennato, è opportuno evidenziare che il segnale audio può essere sia uno ciclo di una funzione periodica o un vero e proprio segnale campionato. Benché non vi sia nessuna differenza funzionale nell'uno o nell'altro caso, occorre fare attenzione nella specifica degli ambiti frequenziali.
Nel primo caso, il valore della frequenza e le eventuali variazioni random sono esplicitamente quelli desiderati. Nel secondo caso, il valore nominale della frequenza naturale dell'oscillatore (ovvero quel valore per cui il segnale audio viene riprodotto alla sua frequenza originale) si può calcolare dal rapporto tra la frequenza di campionamento (*sr*) e la lunghezza della tabella che contiene la forma d'onda (Fn = sr / lunghezza della tabella).
Ad esempio se il segnale audio campionato a 44.1 khz ha una lunghezza di 64k campioni (circa 1.486 secondi) , la frequenza naturale sarà pari a 44100 / 65536 = 0.672 hz.
Dal momento che quasi mai si verifica che la durata del segnale audio corrisponde esattamente alla lunghezza di tabella potenza di 2 è sufficiente prevedere una lunghezza di tabella che approssima per eccesso tale valore. Rispetto all'esempio precedente, se la durata del segnale audio a 44.1 khz fosse di 1.2 secondi l'effettiva lunghezza di tabella dovrebbe essere 44.1 x 1.2 = 52920 campioni e quindi approssimabile per eccesso con una lunghezza di 64 k. La frequenza naturale sarebbe comunque sempre 0.672 hz poichè gli oscillatori di Csound sono pensati per indirizzare aree di memoria potenza di 2. La differenza sostanziale è che l'indice di fase in questo caso deve essere compreso tra 0 e 0.807, valore quest'ultimo che si ottiene dal rapporto 52920/65536.
Anche le variazioni random di frequenza dovranno essere di ordine di grandezza congruente con i valori deterministici impostati. Sempre riferendoci all'esempio precedente, con un valore di frequenza naturale pari a 0.672 hz, una variazione pari al 10 % vale circa 0.06 hz.
Dal momento che la variazione è assoluta e non percentuale, l'ambito di variabilità non è simmetrico rispetto al valore centrale (o medio) per cui per grandi variazioni sarebbe più opportuno che i valori fossero espressi percentualmente o convertiti in unità pch.
Scendendo nei dettagli pratici, immaginando che l'uso più frequente della Sintesi Granulare sia quello di applicare tale processo a forme d'onda campionate, le considerazioni seguenti dovrebbero chiarire il concetto di *frequenza naturale* e di come si relazionano ad essa i relativi controlli.
Dal punto di vista dell'utente, come si è accennato sopra, le cose assumono una prospettiva diversa a seconda che la forma d'onda da granulare sia un solo ciclo di segnale o un intero sound-file. Vale la pena di ricordare che, in generale, un oscillatore "non ha modo di sapere" quando si verifica un caso piuttosto che l'altro: la generazione del segnale da parte di un oscillatore è legata meramente ad un processo di lettura ciclica dei campioni con un determinato passo di incremento (intero o frazionario) all'interno della tabella che li contiene; quando l'icremento è unitario allora il segnale viene riprodotto alla frequenza naturale. In ogni caso vale sempre la relazione :

Fn = I x **SR/ L**

dove :

Fn = frequenza naturale dell'oscillatore
SR = frequenza di campionamento
I = passo di lettura in tabella
L = lunghezza della tabella

Riferendoci all'esempio già citato, se si vuole granulare un file di durata 1.2 s campionato a 44.1 khz e lo si vuole riprodurre alla sua frequenza naturale, occorrerà specificare una funzione per il controllo della frequenza del tipo :

```
f16 0 512 -7 0.672 512 0.672
```

dove 0.672 è il valore in hz della frequenza naturale del segnale che si ottiene dal rapporto sr/lunghezza tabella ovvero 44100/65536 (essendo 65536 la lunghezza che approssima per eccesso il numero di campioni corrispondenti ad 1.2 s di suono, cioè 44100 x 1.2 = 52920). Quando si prevede di utilizzare sempre e solo la granulazione dei segnali campionati , il valore della variabile *ifreq* va modificato nell'orchestra e cioè sempre moltiplicato per l'espressione sr/ftlen(ifun). In questo modo la linea di score che controlla la frequenza si trasforma nel modo seguente:

```
f16 0 512 -7 1 512 1
```

In questo modo la frequenza viene gestita come rapporto rispetto alla frequenza naturale, evitando così di dover calcolare volta per volta i valori reali della frequenza. Se ad esempio si vuole eseguire un glissato continuo del segnale granulato dalla frequenza naturale ad una quinta naturale sopra (rapporto intervallare 3:2=1.5) la linea di controllo diviene:

```
f16 0 512 -7 1 512 1.5
```

Un discorso analogo può essere fatto per la variazione randomica.
Se la funzione audio è un prototipo di funzione periodica, il valore della fase dell'oscillatore non ha praticamente influenza sul risultato acustico finale mentre diventa di vitale importanza se la funzione audio è un segnale campionato. In questo caso infatti, la fase dell'oscillatore assume la funzione di *puntatore alla tabella* e permette di "granulare" punti ben precisi all'interno del segnale audio.
L'impiego più semplice ed immediato di questo parametro si realizza attraverso la definizione di una funzione lineare (GEN 7) con valore iniziale 0 e valore finale 0.999 :

```
f18 0 512 -7  0  512  0.999
```

In questo caso la forma d'onda viene letta mantenendo la freccia temporale originale. Se i valori 0 e 0.999 vengono scambiati di posto, la forma d'onda viene letta in modo retrogrado. Seguendo questo principio si possono ottenere risultati molto diversi ed interessanti.
Per esempio la seguente funzione realizza, nella prima metà dell'evento la retrogradazione della prima metà della forma d'onda mentre nella seconda metà la compressione relativa temporale dell'intera forma d'onda, questa volta ripercorsa in modo diretto:

```
f18 0 512 -7 0.5 256 0 256 0.999
```

Naturalmente, la durata dell'evento (p3) può essere mantenuta identica alla durata reale della forma d'onda o viceversa aumentata o ridotta . In questi due ultimi casi si avrà rispettivamente una dilatazione o una compressione assoluta del tempo.
Altre possibilità si aprono impiegando funzioni di controllo non lineari e di forma più complessa, eventualmente aggiungendo una variazione random della fase attraverso la funzione f19.

Descrizione dell'algoritmo di sintesi

L'algoritmo implementato per la sintesi granulare segue l'approccio indicato da Barry Truax e benché quest'ultimo sia stato realizzato in tempo reale attraverso il processore DMX-1000 controllato da un computer host (PDP - 11 Digital) si è tentato di riprodurne la sua filosofia di fondo. In sostanza si trattava di realizzare un banco di generatori di inviluppi controllati da un insieme di parametri che nel caso originale venivano aggiornati ad un rate di 1 khz (1 ms) dall' host (fig.1).

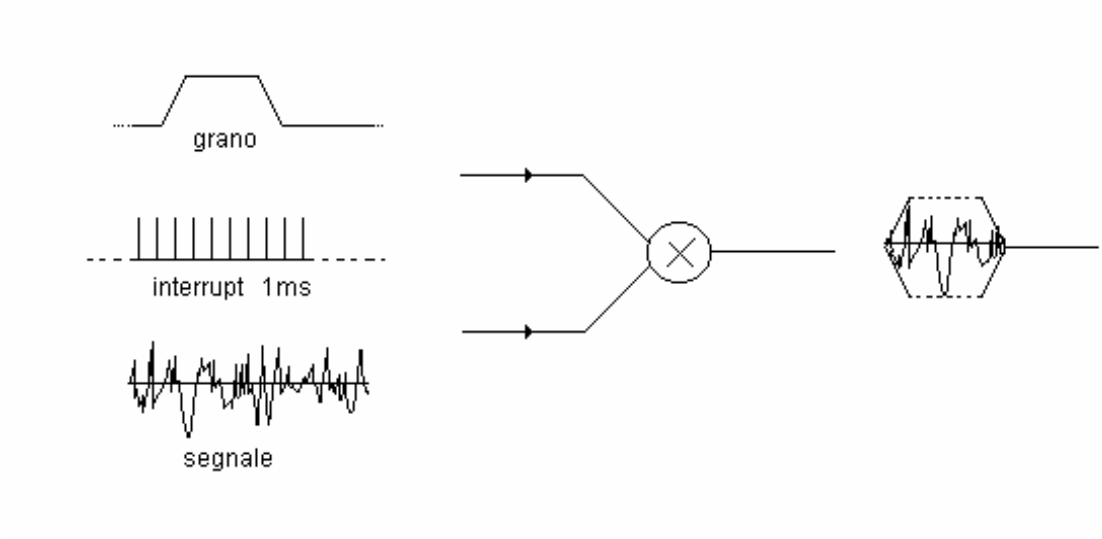

Fig. 1

Questa condizione nell'implementazione di Truax, veniva raggiunta attraverso due programmi concorrenti, uno sul DSP e l'altro sul'host. Mentre il primo aveva il compito di generare gli inviluppi, le relative letture del materiale audio (registrato o attraverso sintesi diretta FM o a look-up table) e le moltiplicazioni e scalature necessarie al processo, il secondo doveva provvedere all'aggiornamento dei parametri della sintesi con un meccanismo, probabilmente basato su una procedura ad interrupt.
La genesi di questa implementazione Csound risale al 1989 e quindi in un periodo in cui la sintesi real-time era esclusivo appannaggio dei DSP . D'altra parte, essendo i programmi della famiglia Music-n (Music V- Music 11, Cmusic, Csound etc.) lo strumento più diffuso e più economico impiegato dai compositori, poteva essere utile realizzare uno strumento di sintesi granulare con tali mezzi, anche in tempo differito.
Data la necessità di generare una grande quantità di micro-eventi per unità di tempo, non era ragionevole seguire un approccio un evento-una nota. In tal modo il problema sarebbe stato spostato ad un eventuale programma di front-end (interfaccia di utilizzazione) per la preparazione della partitura indipendente dalla sintassi di Csound. A tale scopo sarebbe stato sufficiente utilizzare Cscore o altri pre-processors.

L'implementazione Csound

GSC4 è basato sull'idea di generare sequenze di grani all'interno di macro-eventi di durata definita che nei linguaggi Music-n corrispondono ad uno *statement nota* sulla partitura (score). La difficoltà maggiore era rappresentata dalla necessità di produrre una attività di microlivello (generazione degli inviluppi ed elaborazione) e al tempo stesso di controllo all'interno di un unico contesto di programmazione. Occorreva quindi un meccanismo che in maniera *asincrona* potesse gestire questi due processi contemporanei. Tra i moduli primitivi disponibili nella libreria di Csound , si è ricorso al modulo ***timout*** come generatore asincrono e attorno ad esso si è sviluppato l'intero algoritmo.
Il modulo, inserito nella classe dei ***program control statement***. la cui sintassi, lo ricordiamo, è *timout istrt, idur, label ; istr* è il tempo di attivazione di un timer precaricato con un valore definito dall'utente, *idur* tale valore in secondi e *label* un'etichetta di programma che indica la direzione del flusso di programma, dall'istante iniziale *istrt* e durante tutto il periodo di tempo *idur*. Alla fine del conteggio è possibile, all'interno della stessa nota (macro-evento) reinizializzare il timer ed in generale aggiornare, se necessario e opportuno, un insieme di variabili di tipo *i*.
Il nucleo del programma è quindi costituito da tale struttura di controllo che consente da un lato di generare l'inviluppo corrente (grano) parametrizzato e dall'altro di controllare al macrolivello l'andamento di tali parametri.

```
;---------- Sezione di simulazione dell'interrupt (interruzione) ----------
;
;Il modulo timout lavora come un generatore di interrupt. Timout è caricato
;con la durata corrente del grano e decrementato in modo automatico fino a
;zero

timout 0,igrain,cont   ;se il valore del contatore è diverso da zero vai a cont
```

```
reinit loop            ;altrimenti salta alla reinizializzazione
```

La variabile *igrain* contiene, ad ogni reinizializzazione, il valore corrente di durata del nuovo grano. Fino a che tale durata non è esaurita, il flusso del programma non viene interrotto e passa sempre per la label *cont* . Quando il conteggio è terminato, il flusso viene interrotto e portato, attraverso lo statement ***reinit*** ad una sezione di reinizializzazione individuata dalla etichetta *loop*.
La struttura base dell'algoritmo (e quindi dell'orchestra) è costituita da tre blocchi fondamentali:
1) il *generatore dei grani* è uno strumento che contiene, oltre ovviamente alla procedura timout-reinit, anche la parte di inizializzazione e aggiornamento dei parametri dell'inviluppo e la parte di generazione; 2) il *controllo dei grani* è un secondo strumento che provvede alla generazione di tutte le funzioni di controllo dei parametri della sintesi esclusa la somma, il mixing e l'uscita che sono demandati ad un terzo strumento 3) *riscale,mix & out* così come mostrato in figura 2.
I tre blocchi fondamentali costituiscono la base dell'algoritmo che può così essere espanso ad un numero di voci indipendenti decise dall'utente.

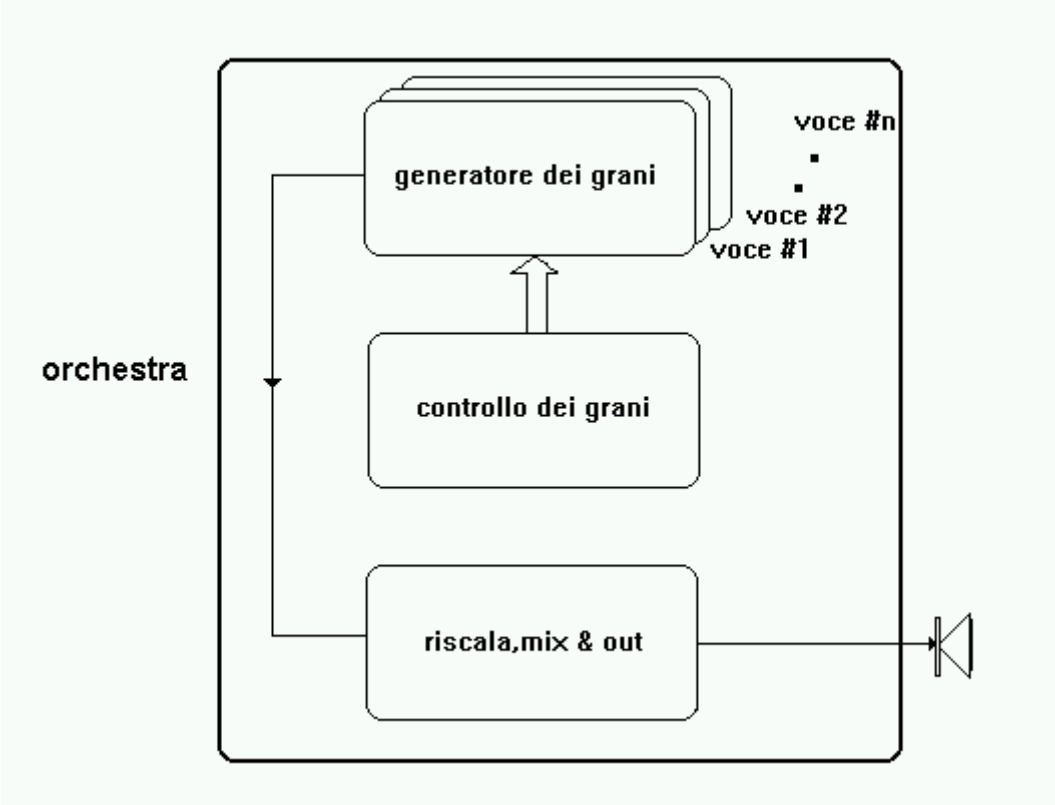

Fig. 2

Seguendo il diagramma di flusso della figura 2 si possono notare due collegamenti fra tre i blocchi che indicano un flusso di variabili globali che devono essere poi trasmesse tra i vari strumenti.
I parametri di ogni grano vengono aggiornati ad ogni fine conteggio del modulo *timout* e per tale ragione le variabili che contengono tali parametri sono del tipo *i-rate* (Aggiornamento dei parametri dei grani).

```
instr 1
;====================== GENERATORE DEI GRANI (VOCE #1)=========================
ifun   =  p4      ;funzione audio
;

;
;       Aggiornamento dei parametri dei grani (ri-inizializzazione)
;       ------------------------------------------------------
loop:
idur = i(gkdur)        ;il valore corrente di gkdur è campionato
```

```
                        ;dal generatore corrispondente nell' instr 11
                        ;e convertito in variabile di tipo i

idurr = i(gkrnd1)       ;il valore corrente di gkrnd1 è ......

itrpz = abs(0.001* (idur + idurr))  ;calcola la durata del trapezoide

iramp = i(gkramp)+ 0.1  ;il valore corrente di gkramp (più un numero magico)è ..

idel  = i(gkdel)         ;il valore corrente di gkdel....
idelr = i(gkrnd1y)       ;il valore corrente di gkrnd1y....
idely = abs(0.001 * (idel + idelr)) ;calcola il delay complessivo

ifreq   = i(gkfreq)       ;il valore corrente di gkfreq è ....
ifreqr  = i(gkran)        ;il valore corrente di gkran è ....
iphase  = i(gkphase)      ;il valore corrente di gkphase è ....
iphaser = i(gkrnd1p)      ;il valore corrente di gkrnd1p è ....
iamp    = i(gkamp)        ;il valore corrente di gkamp è ....
```

Ogni grano è costituito da un inviluppo a trapezoide e da un ritardo. Il trapezoide a sua volta comprende una rampa di salita, una fase stazionaria e una rampa di discesa come mostrato in figura 3.

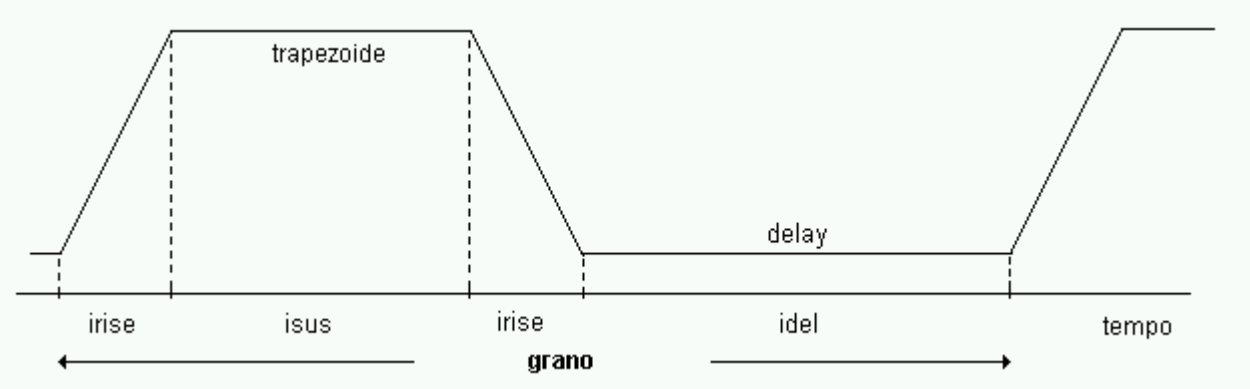

Fig. 3

La somma di tali valori costituisce la durata complessiva del grano e tale valore è correntemente contenuto nella variabile *igrain.* Le due rampe (simmetriche) sono calcolate come frazione della durata del trapezoide attraverso la variabile *iramp.* Se ad esempio iramp = 5 significa che il tempo di salita (discesa) delle due rampe sarà pari ad 1/5 della durata del trapezoide. Quando iramp = 2 il trapezoide degenera in un triangolo che rappresenta il limite minimo di tale variabile. Ciò implica che la durata della fase stazionaria (*isus*) è pari alla differenza tra il valore corrente della durata del trapezoide meno la durata delle due rampe. L'effettiva durata di ogni grano viene poi calcolata tenendo conto delle variazioni statistiche introdotte da un modulo *rand* incluso nel modulo *grain control* attraverso la variabile *idurr.*

```
irise   = itrpz/iramp            ;calcola il tempo di salita del trapezoide
isus    = itrpz - (2 * irise)    ;calcola la durata di sustain del trapezoide
igrain  = itrpz + idely          ;calcola la durata trapezoide+delay
iph     = abs(iphase + iphaser)  ;calcola la fase complessiva
ifq     = ifreq + ifreqr         ;calcola la frequenza complessiva
```

In sostanza, durata, delay, fase e frequenza istantanei vengono calcolati come somma di due contributi: uno deterministico ed uno aleatorio.
Il collegamento tra il modulo di controllo e quello di effettiva generazione è stato realizzato impiegando la funzione *i(x)* che permette ad un valore K-time di essere letto ad init-time (o reinit-time) quando è consentito.
I parametri di controllo del grano, sia quelli deterministici che quelli aleatori, sono aggiornati continuamente (a K-time) nello strumento di controllo e quindi per così dire *campionati* dallo strumento di generazione ad ogni fine conteggio di *timout* . I due processi, generazione e controllo avvengono in modo asincrono e consentono all'utilizzatore di specificare

ad alto livello (macro-level) gli andamenti delle variabili di controllo, in maniera direttamente correlata al rispettivo significato acustico.

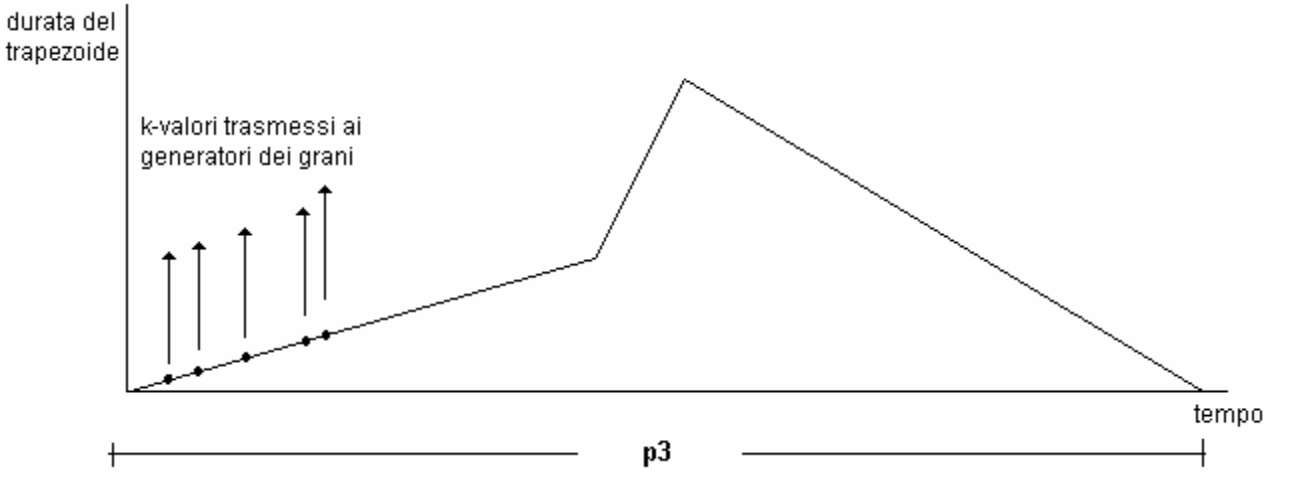

Fig. 4

In figura 4 è riportato un possibile andamento di una funzione di controllo per la durata del trapezoide distribuita sulla durata complessiva del macro-evento p3.
Nello strumento 11 (controllo) sono concentrati il maggior numero di moduli *oscil1* e *rand* rispettivamente per la generazione delle funzioni di controllo dei parametri deterministici e aleatori (randomici) della sintesi.

```
;============================== CONTROLLO DEI GRANI ========================
instr 11
;NOTE: tutte le variabili globali sono trasmesse agli instr 1,2,3,4
;
gkdur    oscil1  0,1,p3,p4       ;generatore di controllo per    idur
gkdurr   oscil1  0,1,p3,p5       ;                               idurr
gkdel    oscil1  0,1,p3,p6       ;                               idel
gkdelr   oscil1  0,1,p3,p7       ;                               idelr
gkramp   oscil1  0,1,p3,p8       ;                               iramp
gkfreq   oscil1  0,1,p3,p9       ;                               ifreq
gkfreqr  oscil1  0,1,p3,p10      ;                               ifreqr
gkphase  oscil1  0,1,p3,p11      ;                               iphase
gkphaser oscil1  0,1,p3,p12      ;                               iphaser
gkamp    oscil1  0,1,p3,p13      ;                               iamp

krnd1  rand 1 ,0.1         ;generatore random (VOCE #1)
krnd2  rand 1 ,0.9         ;                  (VOCE #2)
krnd3  rand 1 ,0.5         ;                  (VOCE #3)
krnd4  rand 1 ,0.3         ;                  (VOCE #4)

;I valori istantanei dei generatori random sono riscalati per ottenere i valori
;opportuni di frequenza, durata, delay e fase

gkran  = krnd1 * gkfreqr/2 ;riscalatura per la frequenza random (VOCE #1,2,3,4)

gkrnd1 = krnd1 * gkdurr/2  ;riscalatura per la variaz.random di durata(VOCE #1)
gkrnd2 = krnd2 * gkdurr/2  ;                                          (VOCE #2)
gkrnd3 = krnd3 * gkdurr/2  ;                                          (VOCE #3)
gkrnd4 = krnd4 * gkdurr/2  ;                                          (VOCE #4)

gkrnd1y = krnd1 * (0.05 + gkdelr /2) ;riscalatura variaz. random delay(VOCE #1)
gkrnd2y = krnd2 * (0.05 + gkdelr /2) ;                                (VOCE #2)
gkrnd3y = krnd3 * (0.05 + gkdelr /2) ;                                (VOCE #3)
gkrnd4y = krnd4 * (0.05 + gkdelr /2) ;                                (VOCE #4)
```

```
gkrnd1p = krnd1 * gkphaser/2   ;riscalatura variaz. random fase(VOCE #1)
gkrnd2p = krnd2 * gkphaser/2   ;                              (VOCE #2)
gkrnd3p = krnd3 * gkphaser/2   ;                              (VOCE #3)
gkrnd4p = krnd4 * gkphaser/2   ;                              (VOCE #4)

endin
```

Facendo riferimento al listato soprastante, ad esempio la variabile globale *gkdur* è generata da un oscillatore "one-shot" *oscil1* il quale legge una funzione (p4) definita all'interno del file score .
Sono stati utilizzati 4 generatori random indipendenti dai quali, tramite opportune scalature sono stati ricavati tutti i valori random per i diversi parametri.
I parametri che prevedono sia una componente deterministica che aleatoria (random) producono delle funzioni di controllo complesse che possono a tutti gli effetti essere assimilabili a *maschere di tendenza* (fig.5b), anche se a rigore questo termine viene quasi sempre associato ad un tipo di rappresentazione grafica dell'andamento di una certa grandezza.
In questo caso, non essendo previsto alcun input grafico, le maschere di tendenza vengono a formarsi unicamente per come sono definiti i singoli andamenti numerici dei parametri che le compongono.
Se, in riferimento alla figura 4, l'andamento della variazione random della durata del trapezoide è rappresentata dalla funzione di figura 5a, il valore corrente della sua durata è situato entro una fascia di valori determinati dalla combinazione delle due funzioni di controllo (fig 5b).
Il programma prevede la definizione di maschere tendenziali per le seguenti variabili:

a) durata dei grani
b) delay dei grani
c) frequenza dei grani
d) fase dei grani

tenendo conto che le variazioni random per la frequenza sono uniche per tutti e quattro gli strumenti di generazione.

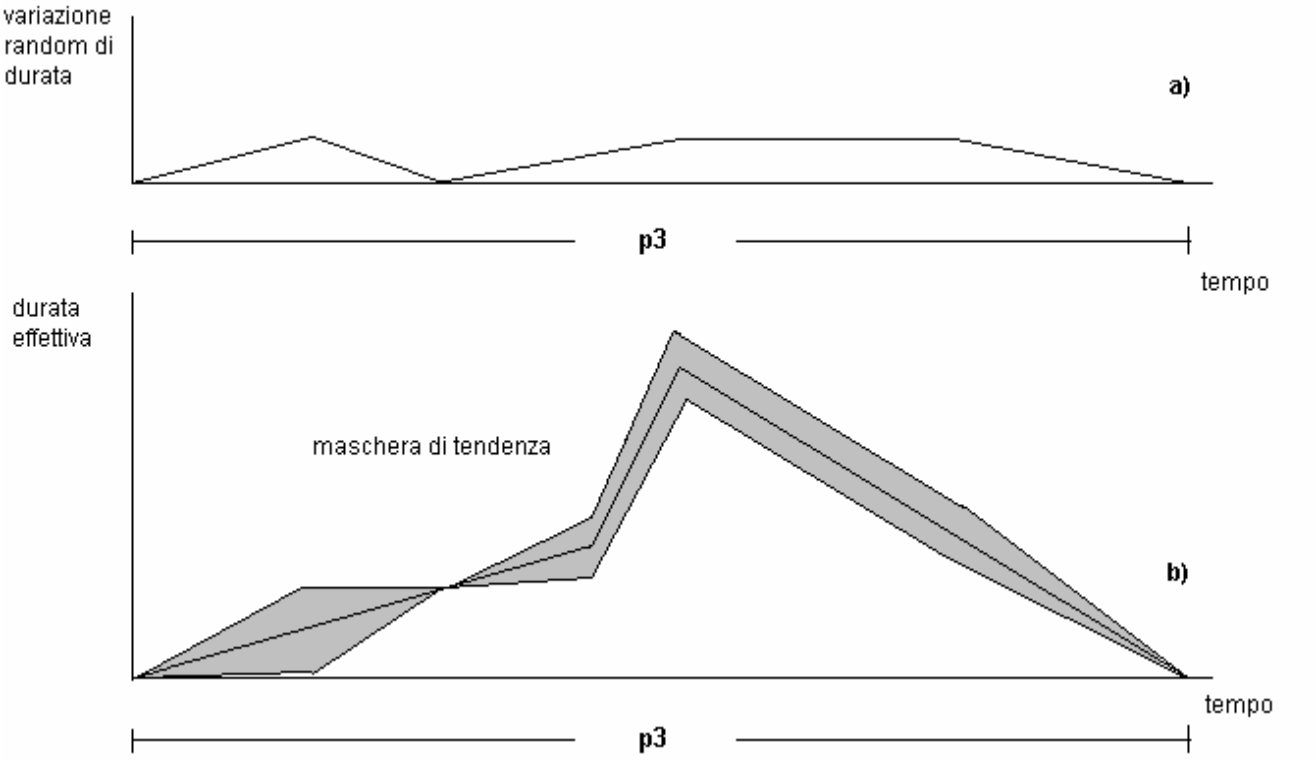

Fig.5

Una volta aggiornate tutte le variabili del grano corrente, viene generato l'effettivo inviluppo a trapezoide attraverso un modulo *linseg* , i cui parametri sono appunto *irise,isus, idel,iamp* (fig. 3*)* :

```
cont:
k1 linseg 0,irise,iamp,isus,iamp,irise,0,idel,0 ;genera l'inviluppo dei grani

ga1 oscili k1,ifq,ifun,iph  ;genera voce 1

endin
```

Successivamente lo stesso inviluppo viene utilizzato per modulare in ampiezza (*k1*) un oscillatore interpolato (*oscili*). La frequenza di tale oscillatore è controllata dalla variabile *ifq* che ingloba al suo interno le rispettive componenti deterministica e aleatoria. Come già detto in precedenza, la fase è controllata dalla variabile *iph* (anch'essa combinazione lineare di una parte deterministica ed una aleatoria) mentre *ifun* indirizza la funzione che contiene il segnale audio.
A questo proposito è opportuno evidenziare che il segnale audio può essere sia un prototipo di funzione periodica (1 ciclo di una forma d'onda qualunque) o un vero e proprio segnale campionato. Benché non vi sia nessuna differenza funzionale nell'uno o nell'altro caso, occorre fare attenzione nella specifica degli ambiti frequenziali.
Nel primo caso, il valore della frequenza e gli eventuali scostamenti random sono esplicitamente quelli desiderati. Nel secondo caso, il valore nominale della frequenza naturale dell'oscillatore (ovvero quel valore per cui il segnale audio viene riprodotto alla sua frequenza originale) si può calcolare dal rapporto tra la frequenza di campionamento (*sr*) e la lunghezza della funzione che contiene la forma d'onda.
L'algoritmo completo di GSC4 comprende infine l'aggiunta di altri 3 strumenti (instr 2,3,4) che provvedono alla generazione di voci addizionali al fine di creare una tessitura minima che possa conferire al suono un maggiore spessore timbrico e "volumetrico". I tre strumenti aggiuntivi sono praticamente identici allo strumento 1 ma si differenziano l'uno dall'altro per il contributo della componente aleatoria. Essi leggono dallo strumento 11 (*grain control*) tutti i parametri comuni, mentre ciascuno legge un valore differente di variazione aleatoria della durata, delay, frequenza e fase, tanti quanti sono i diversi generatori random contenuti in tale strumento:

```
krnd1  rand 1 ,0.1          ;generatore random (VOCE #1)
krnd2  rand 1 ,0.9          ;                  (VOCE #2)
krnd3  rand 1 ,0.5          ;                  (VOCE #3)
krnd4  rand 1 ,0.3          ;                  (VOCE #4)
```

Come si vede, l'unica evidente differenza consiste nel diverso valore dell'innesco (*seed*) dei generatori random (nell'esempio 0.1,0.9,0.5,0.3). Questo accorgimento consente di decorrelare temporalmente ogni singola voce e di produrre così un suono corale più ricco e interessante. Quando la variabile che controlla la quantità di variazione aleatoria della durata o del delay è nulla, le quattro voci sono sincronizzate. Si è comunque osservato che è conveniente lasciare un residuo di decorrelazione per mantenere un livello minimo di complessità del segnale risultante che può essere rimossa eliminando la costante 0.05 nei prodotti di riscalatura dei generatori random. Per ciò che riguarda la componente aleatoria della frequenza si è optato arbitrariamente per la condivisione di tale valore per tutti i 4 gli strumenti al fine di garantire la massima sincronizzazione laddove richiesto dalla partitura.

Conclusioni e Sviluppi

Si è descritta un' implementazione della Sintesi Granulare GSC4 con il linguaggio Csound seguendo il modello proposto da Truax . La struttura del programma è demandata ad un gruppo base di 6 strumenti controllati da una partitura nella quale vengono specificati gli andamenti di tutti i parametri della sintesi. Il materiale audio di granulazione può essere sia un prototipo di funzione periodica che un segnale campionato. A partire da questo nucleo è relativamente semplice espandere l'algoritmo ad un numero qualsiasi di voci indipendenti, ad un numero superiore di canali e prevedere la possibilità di eseguire granulazioni da diverse forme d'onda, includendo anche la possibilità di utilizzare inviluppi per i grani di forma gaussiana e con simmetrie variabili Gli ulteriori sviluppi , inclusi quelli appena accennati, sono in fase di sperimentazione e potranno essere richiesti direttamente all'autore.

Appendice (GSC4 - orchestra)

```
;gsc4.orc
;
;
;                            SINTESI GRANULARE
;                           -------------------
;                                Ver 2.1
;                              Eugenio Giordani
;
;
;-------------------------------------------------------------------------
;Questa orchestra implementa la Sintesi Granulare secondo il modello
;proposto da B. Truax.
;
;--------------   ORCHESTRA HEADER   ----------
;NOTE: per una miglioe qualità audio ksmps = 1
;
sr = 44100
kr = 22050
ksmps = 2
nchnls = 2
;---------- Inizializzazione delle variabili globali di controllo ---------
;
gkdur    init 0  ;durata media dei grani
gkdurr   init 0  ;variazione random della durata dei grani
gkdel    init 0  ;delay medio dei grani
gkdelr   init 0  ;variazione random del delay dei grani
gkramp   init 0  ;rapporto di rampa
gkfreq   init 0  ;frequenza media del segnale audio
gkfreqr  init 0  ;variazione random della frequenza del segnale audio
gkphase  init 0  ;fase del segnale audio
gkphaser init 0  ;variazione random della fase del segnale audio
gkamp    init 0  ;ampiezza globale

gkran  init 0    ;frequenza random istantanea

gkrnd1 init 0    ;durata random istantanea dei grani (VOCE #1)
gkrnd2 init 0    ;                                   (VOCE #2)
gkrnd3 init 0    ;                                   (VOCE #3)
gkrnd4 init 0    ;                                   (VOCE #4)

gkrnd1y init 0   ;delay random istantaneo dei grani  (VOCE #1)
gkrnd2y init 0   ;                                   (VOCE #2)
gkrnd3y init 0   ;                                   (VOCE #3)
gkrnd4y init 0   ;                                   (VOCE #4)

gkrnd1p init 0   ;fase random istantanea             (VOCE #1)
gkrnd2p init 0   ;                                   (VOCE #2)
gkrnd3p init 0   ;                                   (VOCE #3)
gkrnd4p init 0   ;                                   (VOCE #4)

;-------------------------------------------------------------------------
instr 1
;====================== GENERATORE DEI GRANI (VOCE #1)=========================
ifun   =  p4      ;funzione audio
;

;
;       Aggiornamento dei parametri dei grani (ri-inizializzazione)
;       -----------------------------------------------------------
loop:
```

```
idur = i(gkdur)        ;il valore corrente di gkdur è campionato
                       ;dal generatore corrispondente nell' instr 11
                       ;e convertito in variabile di tipo i

idurr = i(gkrnd1)      ;il valore corrente di gkrnd1 è ......

itrpz = abs(0.001* (idur + idurr))  ;calcola la durata del trapezoide

iramp = i(gkramp)+ 0.1  ;il valore corrente di gkramp (più un numero magico)è ..

idel  = i(gkdel)        ;il valore corrente di gkdel....
idelr = i(gkrnd1y)      ;il valore corrente di gkrnd1y....
idely = abs(0.001 * (idel + idelr)) ;calcola il delay complessivo

ifreq   = i(gkfreq)      ;il valore corrente di gkfreq è ....
ifreqr  = i(gkran)       ;il valore corrente di gkran è ....
iphase  = i(gkphase)     ;il valore corrente di gkphase è ....
iphaser = i(gkrnd1p)     ;il valore corrente di gkrn1p è ....
iamp    = i(gkamp)       ;il valore corrente di gkamp è ....

irise   = itrpz/iramp            ;calcola il tempo di salita del trapezoide
isus    = itrpz - (2 * irise)    ;calcola la durata di sustain del trapezoide
igrain  = itrpz + idely          ;calcola la durata trapezoide+delay
iph     = abs(iphase + iphaser)  ;calcola la fase complessiva
ifq     = ifreq + ifreqr         ;calcola la frequenza complessiva



;---------- Sezione di simulazione dell' interrupt (interruzione) ----------
;
;Il modulo timout lavora come un generatore di interrupt. Timout è caricato
;con la durata corrente del grano and decrementato in modo automatico fino a
;zero


timout 0,igrain,cont   ;se il valore del contatore è diverso da zero vai a cont

reinit loop            ;altrimenti salta alla reinizializzazione

cont:
k1 linseg 0,irise,iamp,isus,iamp,irise,0,idel,0 ;genera l'inviluppo dei grani
                                                ;
ga1 oscili k1,ifq,ifun,iph  ;genera voce 1


endin

;------------------------------------------------------------------------
instr 2
;====================== GENERATORE DEI GRANI (VOCE #2)=========================
ifun   =  p4      ;funzione audio
;

;
;       Aggiornamento dei parametri dei grani (ri-inizializzazione)
;       -----------------------------------------------------------
loop:
idur = i(gkdur)        ;il valore corrente di gkdur è campionato
                       ;dal generatore corrispondente nell' instr 11
                       ;e convertito in variabile di tipo i

idurr = i(gkrnd2)      ;il valore corrente di gkrnd2 è ......

itrpz = abs(0.001* (idur + idurr))  ;calcola la durata del trapezoide
```

```
iramp = i(gkramp)+ 0.1  ;il valore corrente di gkramp (più un numero magico)è ..

idel  = i(gkdel)        ;il valore corrente di gkdel....
idelr = i(gkrnd2y)      ;il valore corrente di gkrnd2y....
idely = abs(0.001 * (idel + idelr)) ;calcola il delay complessivo

ifreq   = i(gkfreq)      ;il valore corrente di gkfreq è ....
ifreqr  = i(gkran)       ;il valore corrente di gkran è ....
iphase  = i(gkphase)     ;il valore corrente di gkphase è ....
iphaser = i(gkrnd2p)     ;il valore corrente di gkrn2p è ....
iamp    = i(gkamp)       ;il valore corrente di gkamp è ....

irise   = itrpz/iramp            ;calcola il tempo di salita del trapezoide
isus    = itrpz - (2 * irise)    ;calcola la durata di sustain del trapezoide
igrain  = itrpz + idely          ;calcola la durata trapezoide+delay
iph     = abs(iphase + iphaser)  ;calcola la fase complessiva
ifq     = ifreq + ifreqr         ;calcola la frequenza complessiva



;---------- Sezione di simulazione dell' interrupt (interruzione) ----------
;
;Il modulo timout lavora come un generatore di interrupt. Timout è caricato
;con la durata corrente del grano and decrementato in modo automatico fino a
;zero


timout 0,igrain,cont   ;se il valore del contatore è diverso da zero vai a cont

reinit loop            ;altrimenti salta alla reinizializzazione

cont:
k1 linseg 0,irise,iamp,isus,iamp,irise,0,idel,0 ;genera l'inviluppo dei grani
                                                ;
ga2 oscili k1,ifq,ifun,iph  ;genera voce 2


endin
;------------------------------------------------------------------------
instr 3
;====================== GENERATORE DEI GRANI (VOCE #3)=========================
ifun   =  p4      ;funzione audio
;

;
;       Aggiornamento dei parametri dei grani (ri-inizializzazione)
;       -----------------------------------------------------------
loop:
idur = i(gkdur)         ;il valore corrente di gkdur è campionato
                        ;dal generatore corrispondente nell' instr 11
                        ;e convertito in variabile di tipo i

idurr = i(gkrnd3)       ;il valore corrente di gkrnd3 è ......

itrpz = abs(0.001* (idur + idurr))  ;calcola la durata del trapezoide

iramp = i(gkramp)+ 0.1  ;il valore corrente di gkramp (più un numero magico)è ..

idel  = i(gkdel)        ;il valore corrente di gkdel....
idelr = i(gkrnd3y)      ;il valore corrente di gkrnd3y....
idely = abs(0.001 * (idel + idelr)) ;calcola il delay complessivo

ifreq   = i(gkfreq)      ;il valore corrente di gkfreq è ....
```

```
ifreqr  = i(gkran)       ;il valore corrente di gkran è ....
iphase  = i(gkphase)     ;il valore corrente di gkphase è ....
iphaser = i(gkrnd3p)     ;il valore corrente di gkrnd3p è ....
iamp    = i(gkamp)       ;il valore corrente di gkamp è ....

irise   = itrpz/iramp            ;calcola il tempo di salita del trapezoide
isus    = itrpz - (2 * irise)    ;calcola la durata di sustain del trapezoide
igrain  = itrpz + idely          ;calcola la durata trapezoide+delay
iph     = abs(iphase + iphaser)  ;calcola la fase complessiva
ifq     = ifreq + ifreqr         ;calcola la frequenza complessiva



;---------- Sezione di simulazione dell' interrupt (interruzione) ----------
;
;Il modulo timout lavora come un generatore di interrupt. Timout è caricato
;con la durata corrente del grano and decrementato in modo automatico fino a
;zero


timout 0,igrain,cont   ;se il valore del contatore è diverso da zero vai a cont

reinit loop            ;altrimenti salta alla reinizializzazione

cont:
k1 linseg 0,irise,iamp,isus,iamp,irise,0,idel,0 ;genera l'inviluppo dei grani
                                                ;
ga3 oscili k1,ifq,ifun,iph  ;genera voce 3


endin
;------------------------------------------------------------------------
instr 4
;====================== GENERATORE DEI GRANI (VOCE #4)=========================
ifun   =  p4      ;funzione audio
;

;
;       Aggiornamento dei parametri dei grani (ri-inizializzazione)
;       -----------------------------------------------------------
loop:
idur = i(gkdur)        ;il valore corrente di gkdur è campionato
                       ;dal generatore corrispondente nell' instr 11
                       ;e convertito in variabile di tipo i

idurr = i(gkrnd4)      ;il valore corrente di gkrnd4 è ......

itrpz = abs(0.001* (idur + idurr))  ;calcola la durata del trapezoide

iramp = i(gkramp)+ 0.1  ;il valore corrente di gkramp (più un numero magico)è ..

idel  = i(gkdel)        ;il valore corrente di gkdel....
idelr = i(gkrnd4y)      ;il valore corrente di gkrnd4y....
idely = abs(0.001 * (idel + idelr)) ;calcola il delay complessivo

ifreq   = i(gkfreq)      ;il valore corrente di gkfreq è ....
ifreqr  = i(gkran)       ;il valore corrente di gkran è ....
iphase  = i(gkphase)     ;il valore corrente di gkphase è ....
iphaser = i(gkrnd4p)     ;il valore corrente di gkrnd4p è ....
iamp    = i(gkamp)       ;il valore corrente di gkamp è ....

irise   = itrpz/iramp            ;calcola il tempo di salita del trapezoide
isus    = itrpz - (2 * irise)    ;calcola la durata di sustain del trapezoide
igrain  = itrpz + idely          ;calcola la durata trapezoide+delay
```

```
iph     = abs(iphase + iphaser)  ;calcola la fase complessiva
ifq     = ifreq + ifreqr         ;calcola la frequenza complessiva


;---------- Sezione di simulazione dell' interrupt (interruzione) ----------
;
;Il modulo timout lavora come un generatore di interrupt. Timout è caricato
;con la durata corrente del grano and decrementato in modo automatico fino a
;zero


timout 0,igrain,cont   ;se il valore del contatore è diverso da zero vai a cont

reinit loop            ;altrimenti salta alla reinizializzazione

cont:
k1 linseg 0,irise,iamp,isus,iamp,irise,0,idel,0 ;genera l'inviluppo dei grani
                                                ;
ga4 oscili k1,ifq,ifun,iph  ;genera la voce 4


endin

;============================== CONTROLLO DEI GRANI ========================
instr 11
;NOTE: tutte le variabili globali sono trasmesse agli instr 1,2,3,4
;
gkdur    oscil1  0,1,p3,p4       ;generatore di controllo per    idur
gkdurr   oscil1  0,1,p3,p5       ;                               idurr
gkdel    oscil1  0,1,p3,p6       ;                               idel
gkdelr   oscil1  0,1,p3,p7       ;                               idelr
gkramp   oscil1  0,1,p3,p8       ;                               iramp
gkfreq   oscil1  0,1,p3,p9       ;                               ifreq
gkfreqr  oscil1  0,1,p3,p10      ;                               ifreqr
gkphase  oscil1  0,1,p3,p11      ;                               iphase
gkphaser oscil1  0,1,p3,p12      ;                               iphaser
gkamp    oscil1  0,1,p3,p13      ;                               iamp
;
krnd1  rand 1 ,0.1         ;generatore random (VOCE #1)
krnd2  rand 1 ,0.9         ;                  (VOCE #2)
krnd3  rand 1 ,0.5         ;                  (VOCE #3)
krnd4  rand 1 ,0.3         ;                  (VOCE #4)

;I valori istantanei dei generatori random sono riscalati per ottenere i valori
;opportuni di frequenza, durata, delay e fase

gkran  = krnd1 * gkfreqr/2 ;riscalatura per la frequenza random (VOCE #1,2,3,4)

gkrnd1 = krnd1 * gkdurr/2  ;riscalatura per la variaz.random di durata(VOCE #1)
gkrnd2 = krnd2 * gkdurr/2  ;                                          (VOCE #2)
gkrnd3 = krnd3 * gkdurr/2  ;                                          (VOCE #3)
gkrnd4 = krnd4 * gkdurr/2  ;                                          (VOCE #4)

gkrnd1y = krnd1 * (0.05 + gkdelr /2) ;riscalatura variaz. random delay(VOCE #1)
gkrnd2y = krnd2 * (0.05 + gkdelr /2) ;                                (VOCE #2)
gkrnd3y = krnd3 * (0.05 + gkdelr /2) ;                                (VOCE #3)
gkrnd4y = krnd4 * (0.05 + gkdelr /2) ;                                (VOCE #4)

gkrnd1p = krnd1 * gkphaser/2   ;riscalatura variaz. random fase(VOCE #1)
gkrnd2p = krnd2 * gkphaser/2   ;                              (VOCE #2)
gkrnd3p = krnd3 * gkphaser/2   ;                              (VOCE #3)
gkrnd4p = krnd4 * gkphaser/2   ;                              (VOCE #4)

endin
```

```
;======================== RISCALATURA, MIX & OUT ========================

instr 21
iscale = p4                          ;legge dallo score il fattore di scala
outs1 (ga1/2 + ga2/2 ) * iscale     ; manda in uscita left le voci 1 e 2
outs2 (ga3/2 + ga4/2 ) * iscale     ; manda in uscita right le voci 3 e 4

endin
```

(GSC4 - score)

```
;gsc4.sco
;
;-------- Funzione di controllo per la durata dei grani --------
;
f11 0  512 -7 10 256 20  128 20 128 16

;-------- Funzione di controllo per la variazione random di durata ---------
;
f12 0 512 -7 4 256 1 256 0

;-------- Funzione di controllo per il delay dei grani ------------
;
f13 0 512 -7 10 256 20 256 5

;-------- Funzione di controllo per la variazione random del delay ---------
;
f14 0 512  -7 0 128  0 256 2 128 0

;-------- Funzione di controllo per la proporzione di rampa --------------
;
f15 0 512 -7 2 256 4 256 2

;-------    Funzione di controllo per la frequenza ---------
;
;f16 0 512 -7 1.345 512 3.345
f16  0 512 -7 220 512 220

;--------  Funzione di controllo per la variazione random di frequenza ---------
;
f17 0 512 -7  0  512  110

;-------   Funzione di controllo per la fase  (o Puntatore al file audio)  -----
;
f18 0 512 -7 0 512 0

;-------   Funzione di controllo per la variazione random di fase -----------
f19 0 512 -7 0 128 0 256 0 128  0

;-------  Funzione di controllo per ampiezza complessiva  -----------
;
f20 0 512  7 0 128 1 256 1 128 0
;======================     Funzioni Audio     ==============================;
;f1 0 32768 -1 "sample.wav" 0 0 0
f1 0 1024 10 0.6 0.8  1   0.5  0.3  0.5  0.7
;=============================================================================
;p1  p2  p3   p4   p5     p6   p7     p8     p9     p10     p11     p12
;-----------------------------------------------------------------------------;
;            ifun
i1   0   40   1
i2   0   40   1
i3   0   40   1
i4   0   40   1
;-----------------------------------------------------------------------------
;            dur   durr   del    delr   ramp   freq  freqr   phase  phaser amp
i11  0   40  11     12     13     14     15     16     17     18     19     20
;-----------------------------------------------------------------------------
;            scale
i21  0   40  20000
e
```

Bibliografia

Gabor,D. "Acoustical Quanta and the Theory of Hearing." *Nature* May 3,1947
Truax,B. "Real-Time Granular Synthesis with a Digital Signal Processor." *Computer Music Journal* 12(2)
Truax,B. "Real-Time Granular Synthesis with the DMX-1000" *ICMC 86 Proceedings*
Roads,C. "Granular Synthesis" from *The Computer Music Tutorial* (MIT Press)
Roads,C. "Granular Synthesis of Sound." *Computer Music Journal* 2(2)
Roads,C. "The Realization of *nscor"* from *"The Computer Music and Digital Audio Series"* C.Roads Editor
Truax,B. "Handbook fo Acoustic Ecology" B.Truax Editor
Vercoe,B. "Csound User Manual" Media Lab MIT