Le macchine virtuali

Massimiliano Salfi

salfi@dmi.unict.it

Nozioni generali

- Uno degli aspetti più affascinanti della Musica Elettronica è sempre stato quello relativo alla possibilità di controllare tutte le fasi del processo compositivo: sia la microstruttura del suono che la sua macrostruttura.
- Gli enormi gradi di libertà offerti dall'utilizzo dell'elaboratore in quest'ambito, però, costituiscono paradossalmente un grave ostacolo alla realizzazione di una composizione, dato l'enorme numero di dati e parametri di controllo che risultano necessari alla sua implementazione su computer.

Nozioni generali

In altre parole, sin dagli albori della Computer Music ci si era subito resi conto che il problema principale diventava quello di poter disporre di un adeguato linguaggio con il quale comunicare all'elaboratore le istruzioni relative alla generazione ed al controllo della micro e macro-struttura della composizione.

Nozioni generali

Nel corso degli anni sono state implementate varie filosofie di approccio al problema, le quali si riferiscono alle due modalità operative esistenti:

- la sintesi in tempo reale (software-oriented o hardware-oriented);
- la sintesi in tempo differito.

Sintesi in tempo reale

L’approccio hardware-oriented, comportava l'assunzione del fatto che il calcolatore fosse solo un sistema di controllo e che l'hardware necessario alla sintesi e all’analisi del suono costituisse un sistema interfacciato all’*host computer*.

Si restava così vincolati ad una macchina che, per quanto espandibile e modulare, dimostrava prima o poi dei limiti fisici di utilizzo (la limitata velocità computazionale del sistema imponeva dei limiti nel numero massimo di moduli da poter implementare).

Sintesi in tempo reale

L'enorme progresso tecnologico ha portato ad un notevole aumento nella velocità intrinseca di un calcolatore e, pur esistendo ancora sistemi di sintesi basati su schede o rack esterni all'unità di controllo, sui computer attuali (ad alte prestazioni) è possibile implementare (tramite un approccio software-oriented) la sintesi ed il controllo delle unità virtuali in tempo reale.

Si tenga comunque presente che ogni sistema ha i suoi limiti e pertanto anche quest'approccio, pur se molto più economico di quello hardware, risente del fatto che lo strumento (o gli strumenti) non potranno essere molto complicati come architettura, pena la crisi (parziale o totale) del sistema, che incomincerà a produrre suoni in cui saranno avvertibili delle discontinuità.

Sintesi in tempo reale

Vantaggi

Possibilità di interagire in tempo reale con lo strumento realizzato.

Svantaggi

Costi elevati, necessità di hardware con grande potenza di calcolo, limite nel numero di moduli coinvolti nella progettazione degli strumenti virtuali.

Sintesi in tempo differito

I sistemi di sintesi in tempo differito richiedono oggi come hardware unicamente un calcolatore, anche relativamente lento (e quindi molto economico) e una scheda di conversione digitale-analogica, lasciando invece totalmente a carico del software la possibilità di generare suoni mediante dei linguaggi che possono essere definiti una sorta di *compilatori acustici*: questo modo di procedere comporta che sia il programma a generare, in base ad opportune istruzioni, un file di campioni digitali che possono essere convertiti in suono.

Sintesi in tempo differito

Vantaggi

Possibilità di avvicinarsi alla disciplina senza spendere cifre astronomiche.

Svantaggi

il file sorgente può essere modificato solo in tempo differito e va quindi poi successivamente ricompilato, non esiste alcun feedback immediato tra il musicista e lo strumento 'virtuale' realizzato.

Linguaggi per macchine virtuali

Fu alla luce di queste ed altre riflessioni che, già negli anni '60, Max Mathews incominciò la stesura del capostipite di tutti i “linguaggi per macchine virtuali”, il leggendario Music V.

Da tale linguaggio sono scaturiti, nel tempo, una lunga serie di successori, basati tutti sostanzialmente sulla stessa idea di base (e spesso anche con molti criteri di programmazione ed operatività simili), fino ad arrivare ai relativamente recenti Csound (Barry Vercoe, MIT) e Cmusic (F.Richard Moore, UCSD).

Linguaggi per macchine virtuali

La logica di questo tipo di linguaggi consiste nell'utilizzo del calcolatore per sviluppare degli algoritmi che siano in grado, sotto forma di istruzioni (cioé di codici operativi con una precisa sintassi e relativi parametri), di generare uno 'strumento virtuale' in grado di accettare la 'partitura' per esso concepita.

Un apposito compilatore farà si che lo strumento, le relative funzioni d'onda ed i parametri, vengano calcolati matematicamente dal computer e trasformati in un file di campioni numerici (cioé un'informazione digitale) tali che, una volta inviati ad un convertitore digitale-analogico siano trasformati in un segnale elettrico analogico da inviare ad un sistema di amplificazione sonora per essere ascoltati.

Il concetto di modulo

Un **modulo** di un linguaggio per la sintesi del suono (come Csound, o Audio FX) è un algoritmo in grado di accettare dei parametri esterni di controllo ed i dati (provenienti da un altro modulo o da una tabella) sui quali compiere l'elaborazione per la quale è stato progettato.

Il concetto di strumento

Uno **strumento** (realizzato ad esempio per mezzo del linguaggio Csound) non è altro che un *insieme* di **moduli** (cioè di algoritmi) strutturati per la definizione di uno o più strumenti virtuali che corrispondono ad uno o più modelli di generazione e/o elaborazione del segnale (un modello di sintesi, un filtraggio o altro tipo di operazione su di un file audio o di dati preesistente, etc.)

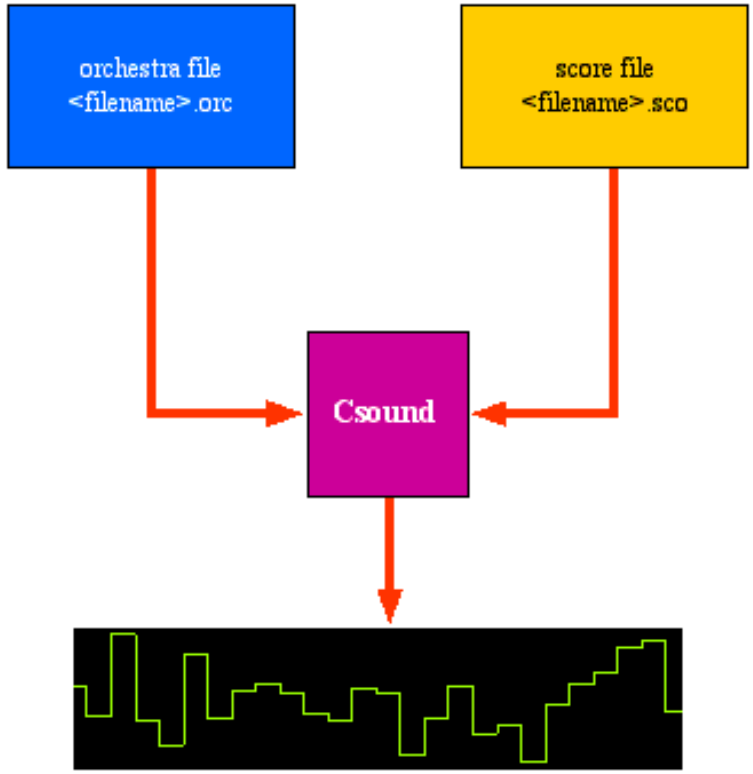
Approccio alla programmazione

Al fine di eseguire una corretta progettazione di uno strumento, è essenziale che si abbiano ben chiare le modalità di stesura dello stesso, e cioé:

- scomporre lo strumento da realizzare in vari sottoproblemi, (ognuno dei quali deve necessariamente trasformarsi in un modulo)
- disegnare sempre un diagramma grafico del tipo top-down, rappresentante lo schema completo dei moduli che formano lo strumento con i relativi collegamenti;
- testare se ciascun modulo svolge effettivamente le funzioni desiderate nell'ambito del progetto, (attraverso la visualizzazione dei dati prodotti da detto modulo in forma numerica e/o grafica);
- compilare lo strumento così progettato e testare se effettivamente, nell'ambito delle gamme stabilite, lo stesso risponde in maniera corretta.

Il linguaggio CSound

Gli strumenti 'virtuali' di Csound vengono creati per mezzo di due file di testo: ognuno di essi contiene una serie di istruzioni e di dati alfanumerici che consentiranno poi al compilatore di realizzare lo strumento (o gli strumenti) richiesti, con annessa partitura. Uno schema del sistema è illustrato nella figura seguente:



Il linguaggio CSound

Il primo file di testo, denominato 'orchestra file', (<filename>.orc), è quello che ci consente di 'costruire' il nostro strumento mediante una serie di **codici operativi** ed **operandi** (i **moduli** di cui si è parlato precedentemente): i valori agli operandi potranno essere forniti o direttamente dall'interno dell'orchestra file, ma è preferibile assegnarli tramite lo 'score file' (<filename>.sco), grazie al quale sarà definita la nostra 'partitura', relativamente allo strumento implementato.

Il linguaggio CSound

Analizziamo, senza entrare troppo nei dettagli, il contenuto di un semplice *orchestra file*. Esso è formato da due sezioni ben distinte:

- la *header section* (o sezione intestazione), nella quale vengono definiti la frequenza di campionamento, la frequenza di controllo, il numero di campioni in ciascun periodo di controllo, ed il numero di canali di uscita;
- l'*instrument section* (o sezione strumento) nella quale vengono creati e stabiliti tutti i moduli relativi alla progettazione del nostro strumento.

Il linguaggio CSound

Consideriamo un semplice esempio di sezione intestazione:

sr = 20000
kr = 1000
ksmps = 20
nchnls = 1

In esso, indichiamo al compilatore di CSound di voler fissare una frequenza di campionamento pari a 20000 Hz, una di controllo di 1000 Hz, che calcoleremo il numero di campioni in ciascun periodo di controllo, mediante la formula ksmps = sr / kr, ed infine assegneremo il numero di canali di uscita pari ad 1.

Il linguaggio CSound

Consideriamo, poi, un esempio di sezione strumento. In genere, in esso, vengono numerati tutti gli strumenti cui si fa riferimento nello *score file*. Come detto, ciascuno strumento consiste di un insieme di “unità” o “moduli” software, che vengono collegati tra loro per mezzo di blocchi di I/O.

Inoltre, ciascuna istruzione per l'*orchestra file* occupa una singola linea ed ha sempre lo stesso formato di base, ovvero:

<etichetta> risultato azione argomenti

Il linguaggio CSound

L'etichetta, (o label), è opzionale, ed è necessaria solo quando si deve poter far uso di istruzioni di salto condizionato o incondizionato, per andare da una parte all'altra del codice che implementa il nostro strumento.

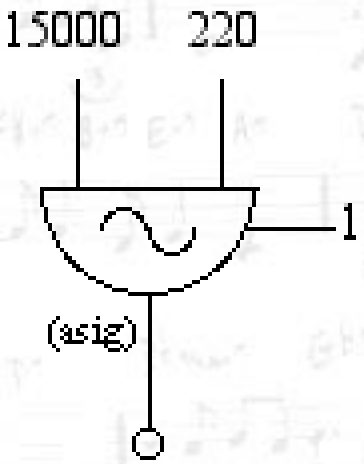
Per risultato s'intende uno dei tipi di variabili gestiti da CSound nella quale verrà memorizzato il risultato dell'azione prodotta da uno dei tanti moduli software, ed i cui argomenti riceveranno i valori che vengono stabiliti nei due seguenti modi:

- nell' *orchestra file*, assegnando direttamente i valori degli argomenti relativi a quell'unità (parametro azione);
- tramite l'attribuzione dei campi parametrici, sempre nell'*orchestra file*: a ciascuno di essi verrà attribuito un valore mediante i dati scritti nello score file.

Il linguaggio CSound: esempio

Proviamo a creare un semplicissimo strumento, il cui scopo è, ad esempio, quello di generare una nota che supponiamo di frequenza 220 Hz (un LA), di ampiezza 15000 (in valori lineari), di durata 3 secondi, e la cui forma d'onda è costituita da una semplice sinusoide.

In base a quanto visto, ci sono quindi due listati da realizzare, uno relativo allo 'strumento', ed uno relativo alla sua 'partitura'. La sua schematizzazione grafica è rappresentata dalla seguente figura:



Il linguaggio CSound: esempio

Il listato relativo allo strumento appena indicato sarà:

```
instr 1
asig oscil 15000,220,1
out asig
endin
```

il cui significato è il seguente:

- definisci lo strumento n° 1;
- crea un modulo **oscil** (un oscillatore, cioé un'unità generatrice) con segnale di ampiezza 15000, frequenza 220 Hz e forma d'onda 1, assegnandolo alla variabile a frequenza audio **asig**;
- mandala in uscita, operazione ottenuta tramite l'assegnazione della variabile **asig** come argomento del modulo **out**;
- fine strumento, segnalata dal modulo **endin**.

Il linguaggio CSound: esempio

A questo punto non ci resta che definire la 'partitura' relativa a questo strumento, vale a dire:

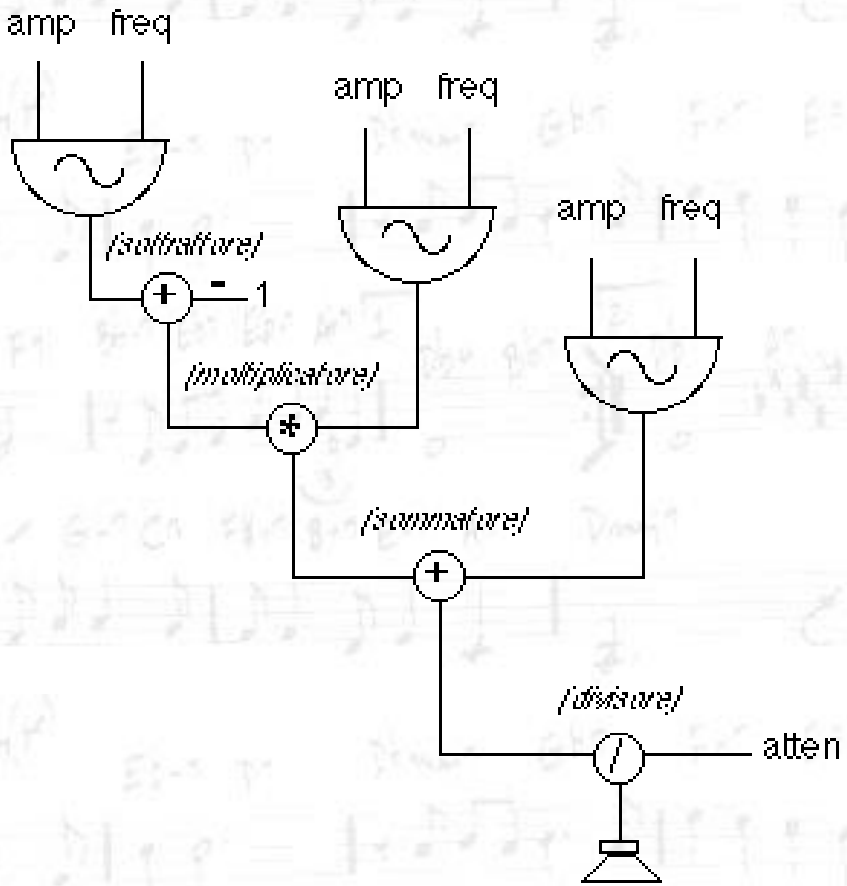
f1 0 4096 10 1
i1 0 4
e

il cui significato è il seguente:

- usa la funzione n°1 per generare all' istante 0, mediante la routine generatrice 10, una tabella di 4096 campioni allo scopo di creare una forma d'onda composta da somme pesate di semplici sinusoidi: la posizione dell' ultimo valore inserito qui stabilisce che solo la prima armonica (la fondamentale) ha ampiezza normalizzata ad 1, e ci restituisce quindi una singola onda sinusoidale;
- Infine, lo strumento 1 genererà, alla frequenza relativa alla produzione di una nota (**i1**), un segnale che partirà all'istante 0 e terminerà all'istante 4 (durata 3 secondi).
- Il simbolo '**e**' indica la fine della partitura.

Esempio di diagramma di flusso

Un diagramma di flusso costituisce la rappresentazione grafica del modo in cui i vari moduli sono interconnessi per formare uno strumento. Un esempio è illustrato nella figura seguente:



Regole fondamentali

Aldilà del significato di ciascun modulo (che non è oggetto di questa trattazione), esistono due regole fondamentali da applicare alla fase di interconnessione dei moduli:

- l'uscita di un modulo può essere collegata ad uno o più ingressi di uno o più moduli: quindi, un'uscita può pilotare uno o più ingressi;
- Due o più uscite non possono mai essere connesse insieme direttamente, poiché ciò potrebbe dar luogo a situazioni ambigue, se i moduli forniscono valori numerici che possono entrare in conflitto tra loro.

Regole fondamentali

Altre regole:

- Le uscite di due o più moduli possono, comunque, essere combinate tra loro allo scopo di produrre operazioni matematiche: somma, sottrazione, moltiplicazione, divisione.
- I parametri che vengono assegnati ad uno strumento, di solito, sono indicati nel diagramma per mezzo di mnemonici descrittivi: ad esempio, il parametro che controlla l'ampiezza di uno strumento è spesso chiamato AMP;
- Ogni strumento deve necessariamente avere almeno un'uscita. La presenza di più uscite, indicherà un sistema audio multicanale (stereofonico, quadrifonico, etc.).

Programmi orientati alle GUI

Audio FX 1.5

http://www.etnasoftware.it/im/download/audiofx/fx15.zip

Audio FX 2.08

http://www.etnasoftware.it/im/download/audiofx/fx2.zip

Clam

http://clam-project.org/