# Csound (autore: Vittorio Albertoni)

## Premessa

Questo documento non ha nessuna pretesa di costituire un riferimento per programmare nel linguaggio Csound, ci vuole ben altro; il suo obiettivo è semplicemente quello di mostrare che cos'è Csound, come funziona, come possiamo procurarcelo e come possiamo procurarci certi strumenti per utilizzarlo con maggiore facilità.

## Indice



## 1 L'interprete

Csound è un linguaggio di programmazione attraverso il quale diamo istruzioni al computer usando termini che, pur rigidamente catalogati e inquadrati in una particolare sintassi, sono comprensibili a noi umani.

Affinché il computer, che utilizza solo un linguaggio binario fatto di 0 e di 1, possa comprendere queste istruzioni è necessaria la presenza di un interprete.

Se andiamo sul sito *www.csounds.com*, nella sezione Download, ci procuriamo questo interprete che, nella ultima versione uscita il 17 novembre scorso, si chiama Csound6.06 ed è disponibile per Linux, Windows e OS X. Per questi ultimi sistemi operativi possiamo scaricare l'installatore, rispettivamente con estensione .exe e .dmg mentre per Linux possiamo scaricare il tarball: sicuramente, tuttavia, la nostra distro ci consente l'installazione attraverso il gestore di applicazioni.

L'interprete, una volta installato, essendo un programma a riga di comando, si fa lavorare digitando il comando csound nel terminale (quello che in Windows si apre eseguendo cmd).

Se digitiamo il solo comando csound, senza indicare nulla da interpretare, ci troviamo di fronte una serie di scritte, tra cui la versione del Csound che stiamo usando, istruzioni sull'uso di Csound e l'elenco dei flags sulle opzioni d'uso.

Di tutto questo le cose di base sono le seguenti:

- per testare il nostro file programma musicale il comando da usare è

  `csound` <nome del file da interpretare>

  con il quale vediamo se il programma funziona, nel qual caso viene generato un file test.wav che contiene i suoni prodotti (inserendo una certa opzione possiamo anche ottenere di udire il suono in tempo reale);

- per generare un file audio contenente i suoni prodotti dal nostro file programma musicale il comando da usare è

  `csound -W -o` <nome da dare al file audio> <nome del file da interpretare>

  `csound -A -o` <nome da dare al file audio> <nome del file da interpretare>

  a seconda se vogliamo generare un file .wav o un file .aiff.

## 2 Come si scrive il programma musicale

Il programma musicale da indicare all'interprete va scritto in un linguaggio appropriato, il linguaggio di Csound.

In rete, a partire dal sito *www.csounds.com* sul quale, nella sezione RESOURCES -> DOCUMENTATION, troviamo innanzi tutto il manuale completo di Barry Vercoe, possiamo trovare parecchi tutorial che ci introducono all'uso di questo linguaggio non propriamente semplice a causa delle numerosissime sfaccettature attraverso le quale ci dà modo di fare di tutto.

Per entrare nei segreti del linguaggio possiamo scaricare un utile documento in lingua italiana a questo indirizzo *www.mediaducks.info/pdf/corso_perfezionamento_informatica_musicale.pdf*.

Esiste inoltre, sempre in lingua italiana, il testo di Giorgio Zucco, Sintesi digitale del suono - Laboratorio pratico di Csound, edito da Giancarlo Zedde, molto ben fatto.

Fino a qualche tempo fa - e, per i nostalgici, ancora adesso - il programma musicale era composto da due file separati: un file con estensione .orc, contenente le istruzioni per la predisposizione degli strumenti con cui suonare (l'estensione .orc sta per orchestra) e un file con estensione .sco, contenente le note da suonare (l'estensione .sco sta per score, partitura). In presenza dei due file separati, all'interprete dovremmo indicarli uno seguito dall'altro, prima il file .orc e poi il file .sco.

Ormai da alcuni anni si usa un file solo, con estensione .csd, che ha la seguente struttura sintattica:

```
<CsoundSynthesizer>
<CsOptions>
    in questa sezione si indicano eventuali flag per lo svolgimento di operazioni particolari
</CsOptions>
<CsInstruments>
    è la sezione dedicata all'orchestra, con le indicazioni un tempo contenute nel file .orc
</CsInstruments>
<CsScore>
    è la sezione dedicata alla partitura, con le indicazioni un tempo contenute nel file .sco
</CsScore>
</CsoundSynthesizer>
```

Non disponendo di altri strumenti, di cui parlerò nel prossimo paragrafo, la scrittura del file di programma può essere effettuata con un qualsiasi elaboratore di testi che produca testo non formattato.

Il seguente file programma produce una scala di Do maggiore partendo dal Do centrale.

```
<CsoundSynthesizer>
<CsInstruments>
sr = 44100
kr = 4410
ksmps = 10
nchnls = 1
instr 1
a1 oscil 10000, p4, 1
out a1
endin
</CsInstruments>
<CsScore>
f1 0 4096 10 1
i1   0     1   261.626
i1   1.5   1   293.665
i1   3     1   329.628
i1   4.5   1   349.228
i1   6     1   391.995
i1   7.5   1   440
i1   9     1   493.883
i1   10.5  1   523.251
</CsScore>
</CsoundSynthesizer>
```

Nella sezione CsInstruments, quella dell'orchestra, abbiamo innanzi tutto la dichiarazione di quattro variabili fondamentali per la qualità del nostro lavoro di generazione del suono:

- sr, che sta per sample rate, cioè frequenza di campionamento, espresso in hertz, è il numero delle volte al secondo che un segnale analogico viene valutato e memorizzato in forma digitale; quella indicata in 44100 è la frequenza con la quale vengono prodotti i CD musicali;
- kr, che sta per frequenza del controllo, cioè la bassa frequenza per le funzioni di controllo (quali quelle per produrre effetti di vibrato, tremolo, ecc.);
- ksmps, che è il numero di campioni per ogni periodo di controllo e si ottiene come rapporto tra sr e kr;
- nchnls, che indica il numero dei canali su cui far viaggiare il segnale (1 sta per mono).

Quelli indicati nell'esempio sono i valori di default e, come tali, avremmo fatto a meno di indicarli, per cui queste quattro righe di programma, se ci vanno bene i valori di default, possiamo non scriverle.

Con la scritta instr 1 apriamo la costruzione dello strumento 1, che è poi l'unico del nostro programma ma potremmo averne decine e decine.

Per costruire lo strumento definiamo una variabile audio, nel nostro caso a1 (per dichiarare una variabile audio la cosa più importante è iniziarne il nome con una a, poi può esserci di tutto, nel nostro caso ho messo 1); definiamo la variabile come un oscillatore (oscil) che generi un suono di ampiezza 10000 (l'ampiezza è la forza con la quale il suono colpisce il timpano dell'orecchio), che, oltre ai tre parametri fissi che vedremo tra poco ne abbia un quarto, p4 e che si avvalga della forma d'onda che genereremo con la funzione 1, che vedremo tra poco.

Segue l'istruzione out a1 che collega il suono generato dallo strumento a1 al nostro sistema audio di ascolto (out gestisce un unico canale mono).

Con endin dichiariamo di aver finito di costruire lo strumento.

E così si chiude la sezione CsInstruments.

Veniamo ora alla sezione CsScore, quella della partitura.

Innanzi tutto richiamiamo la funzione 1, per generare la forma d'onda, con f1 seguito da quattro parametri: il primo (0) indica l'istante in cui viene generata la forma d'onda, il secondo (4096) indica il numero di punti con cui sarà rappresentata (si usa indicare con numeri che siano una potenza di due, da qui 4096 che è 2 alla dodicesima; se mettiamo 2048, che è 2 all'undicesima, non cambia nulla), il terzo (10) indica il metodo di generazione (la Gen10 crea sinusoidi), il quarto (1) indica che stiamo creando una sinusoide unica.

Poi arriva il vero e proprio score, cioè l'elencazione dei suoni (note musicali) da rendere udibili.

Le prime tre colonne dello score ospitano parametri fissi: nella prima indichiamo lo strumento da usare (i1), nella seconda indichiamo, in secondi, il tempo in cui deve avvenire l'esecuzione del suono (nel nostro caso ad intervalli di un secondo e mezzo), nella terza indichiamo la durata del suono (nel nostro caso un secondo per tutte le note) e arriviamo al parametro p4 che, quando abbiamo creato lo strumento oscillatore avevamo destinato alla frequenza dei suoni da generare: e qui abbiamo indicato l'altezza delle note indicandone la frequenza in hertz nel sistema temperato (261,626 per il Do centrale, 440 per il La dell'ottava del Do centrale, il La del diapason, ecc.).

Dopo di che chiudiamo lo score e chiudiamo il programma.

Memorizziamo il file dandogli un nome e l'estensione .csd.

Con i comandi visti nel paragrafo precedente generiamo il file audio e, passandolo in un lettore di media, sentiremo la nostra scala scandita con note della durata di un secondo ogni secondo e mezzo.

Come dire che dalla musica siamo passati alla fisica: del resto il software che stiamo esaminando si chiama Csound non Cmusic.

Il vantaggio di questa astrazione dagli schemi musicali cui siamo abituati è quello di poter creare qualsiasi gioco di suoni, suoni che non necessariamente si debbano chiamare note e che non necessariamente debbano coincidere con le note del sistema temperato occidentale.

Se nel nostro score, anziché inserire il valore 261,626 per produrre il suono che chiamiamo Do centrale inserissimo il valore 270 otterremmo un suono intermedio tra Do e Do# che nessuna tastiera è in grado di produrre (solo su uno strumento senza tasti e tacche, come il violino, potremmo farlo: ma non lo troviamo scritto in nessuna partitura di musica occidentale basata sul sistema temperato).

* * *

Ma per dimostrare la potenza e la versatilità di Csound faccio quest'altro esempio nel quale, invece di produrre il suono con un oscillatore, lo produciamo utilizzando un soundfont nel sistema MIDI.

```
<CsoundSynthesizer>
<CsOptions>
-odac -+rtmidi=virtual -M0
</CsOptions>
<CsInstruments>
nchnls = 2
0dbfs = 1
gimotoremidi fluidEngine
isfnum fluidLoad "/usr/share/sounds/sf2/FluidR3_GM.sf2", gimotoremidi
fluidProgramSelect gimotoremidi, 1, isfnum, 0, 56
instr 1
mididefault 60, p3
midinoteonkey p4, p5
ikey init p4
ivel init p5
fluidNote gimotoremidi, 1, ikey, ivel
endin
instr 2
imvol init 7 asinistra, adestra fluidOut gimotoremidi
outs asinistra*imvol, adestra*imvol
endin
</CsInstruments>
<CsScore>
i1 0 2 69 100
i2 0 120
</CsScore>
</CsoundSynthesizer>
```

Rispetto all'esempio precedente abbiamo una prima novità: è inserita un'opzione per fare in modo che siano in tempo reale l'audio e il MIDI in. Questo è necessario in quanto con questo programma creeremo una tastiera MIDI virtuale sulla quale potremo suonare pigiando i tasti con il mouse e, durante il funzionamento di questa tastiera virtuale, anche la tastiera del computer potrà essere utilizzata per suonare.

Passando alla zona dell'orchestra abbiamo innanzi tutto l'inizializzazione delle variabili.

Accettiamo per default sample rate e frequenza di controllo e non le indichiamo. Enunciamo invece il valore della variabile nchnls (il numero dei canali su cui far viaggiare il segnale), che per default sarebbe 1 (mono) ma noi vogliamo lavorare in stereo, perciò assegniamo a questa variabile il valore 2.

Altra variabile che enunciamo è quella dell'ampiezza del segnale: assegnando il valore 1 alla variabile 0dbfs stabiliamo un valore massimo per l'ampiezza del segnale prima che avvenga il clip e introduciamo una scala di indicazione dell'ampiezza che va da 0 a 1 (nell'esempio precedente avevamo stabilito l'ampiezza durante la costruzione dell'oscillatore, attribuendo il valore 10000).

Arriva ora la variabile chiave del nostro esempio, che ho chiamato gimotoremidi (le prime due lettere sono obbligatorie, g sta per variabile globale e i sta ad indicare che siamo in fase di inizializzazione, il seguito motoremidi è per capirci) in quanto questa variabile è destinata ad inglobare nel nostro programma il sistema MIDI: invece di costruire l'oscillatore per produrre il suono, come abbiamo fatto nell'esempio precedente, qui produciamo il suono attraverso il sistema MIDI, in particolare il sistema imperniato sui soundfonts fluid, quelli con estensione .sf2 (in Csound si chiama fluidEngine).

Nella riga successiva indichiamo l'indirizzo per arrivare alla raccolta dei soundfonts (quello indicato è relativo al mio computer con sistema operativo Linux; in Windows partiremo probabilmente da C:\ ecc. ecc. con la tipica barra rovesciata di Windows rispetto ai sistemi Unix like).

Segue poi la scelta dello strumento, che avviene con l'istruzione fluidProgramSelect che, nell'ordine, richiama la variabile motore cui ci riferiamo (gimotoremidi), il canale MIDI (1), la variabile nella quale sono stati caricati i soundfonts (isfnum inizializzata nella riga precedente), il banco MIDI (0) e il preset MIDI (56): il preset 56 del banco 0 produce il suono della tromba.

Finalmente arriviamo agli strumenti.

Lo strumento 1 si attrezza per accettare istruzioni dalla sezione dello score in linguaggio MIDI: dopo i primi tre parametri, che sono i soliti (strumento, tempo e durata), abbiamo il p4, nel quale indichiamo il tasto MIDI: cioè per indicare il suono non usiamo più la frequenza in hertz come abbiamo fatto nel precedente esempio, quando dovevamo far suonare un oscillatore, ma indichiamo il codice MIDI della nota musicale che vogliamo suonare; poi abbiamo il p5 che indica la velocity, cioè la forza con cui deve essere resa la nota.

Lo strumento 2 prevede la possibilità di immettere suoni generati dal nostro motore MIDI direttamente in variabili audio (le variabili cominciano per a perché sono variabili audio e sono due, asinistra e adestra, in quanto abbiamo scelto di lavorare in stereo): l'immissione dei suoni potrà avvenire grazie al richiamo sullo schermo di una tastiera MIDI virtuale e basterà pigiare un tasto di questa con il mouse per poter proseguire a suonare anche usando la testiera del computer.

Termina così la zona dedicata all'orchestra.

La zona dello score è semplicissima.

Nella prima riga indichiamo al nostro strumento 1 di suonare un bel La diapason (codice MIDI 69) con velocity 100.

Nella seconda riga indichiamo al nostro strumento 2 di darci la possibilità di suonare per 120 secondi, cioè per due minuti.

Memorizzato il file con estensione .csd, se lo passiamo all'interprete con il comando csound udiamo un bel La suonato con la tromba e troviamo sullo schermo una tastiera MIDI con la quale possiamo suonare per due minuti, sempre la tromba, scegliendo le note con i tasti della tastiera virtuale o del computer.

# 3 Per facilitarci il compito

## 3.1 Editor incorporato in Csound

L'installatore di Csound su Windows o su OS X automaticamente installa anche un editor molto utile per scrivere programmi in Csound: nelle vecchie versioni fino alla 5 compresa questo editor si chiamava Qutecsound; con la versione 6 si chiama CsoundQt.

Per chi usa il sistema Linux questo editor va installato a parte (sicuramente lo si trova nel gestore delle applicazioni).

La figura 1 mostra la finestra di lavoro di CsoundQt.

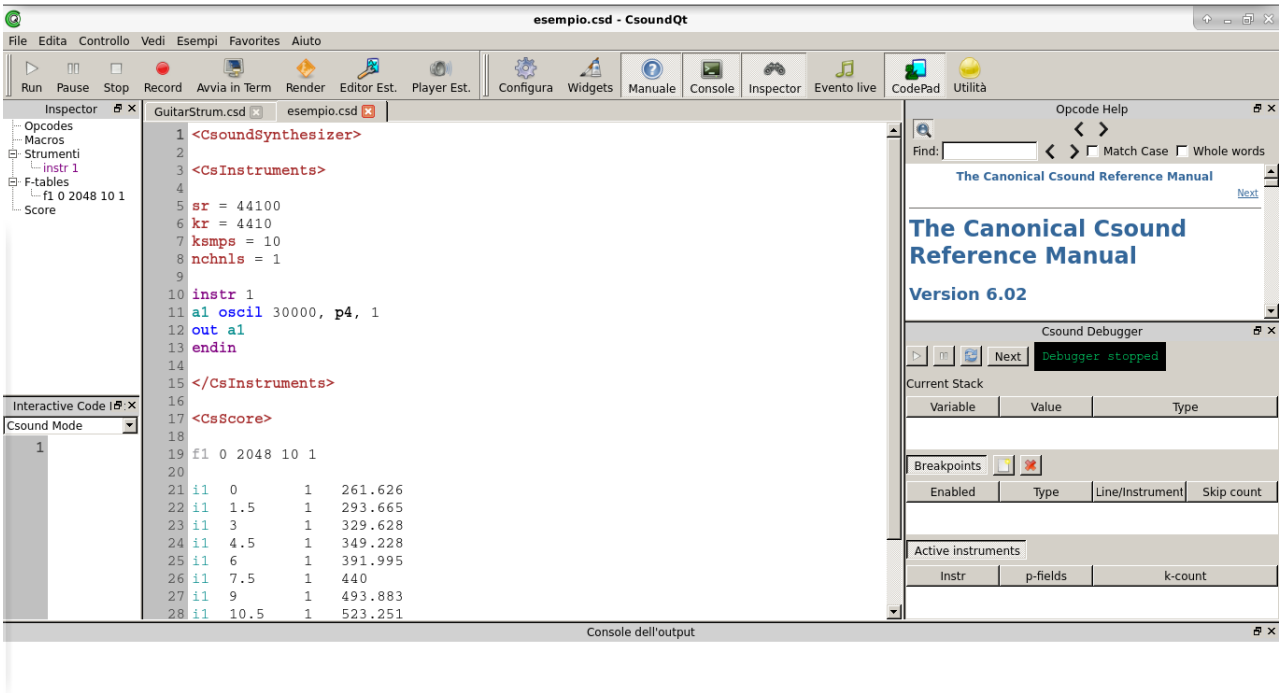


Figura 1: Finestra di lavoro di CsoundQt

Vi è caricato il programma del primo dei due esempi illustrati nel capitolo 2.

Il vantaggio di usare questo editor anziché un qualsiasi editor di testo sta innanzi tutto nella evidenziazione colorata delle parole chiave e nella funzione di completamento che aiuta la digitazione corretta delle istruzioni di programma.

Inoltre, come si vede nella fascia destra dalla figura, abbiamo a disposizione il manuale del linguaggio Csound e un debugger utile per la ricerca di eventuali errori nel programma.

Premendo il tasto Configurazione o utilizzando il menu EDITA -> CONFIGURAZIONE -> PROGRAMMI ESTERNI è utile verificare che i programmi scelti per default siano quelli che abbiamo installati sul nostro computer, eventualmente modificandoli, se avendone più di uno, ne preferiamo uno diverso da quello scelto per default. Nella configurazione possiamo anche scegliere l'italiano come lingua per i menu e il tipo di file audio da generare per default (all'installazione la scelta cade sul file formato wave, quello con estensione .wav).

Le più importanti funzioni che possiamo scegliere nella barra degli strumenti per un primo approccio sono

- RUN, che avvia l'interprete ed esegue il nostro programma (con questa funzione udiamo il suono anche se non abbiamo attivato l'opzione audio in tempo reale nel programma);
- RECORD, che, oltre a farci sentire il suono, lo registra in un file .wav o in un file nel formato scelto nella configurazione;
- EDITOR EST., che apre il file audio generato dal nostro programma in un editor audio (quello scelto nella configurazione).

## 3.2 Blue

La figura 2 mostra un altro editor di lusso, che si chiama Blue ed è opera di uno dei più attivi membri della Community Csound, Steven Yi.

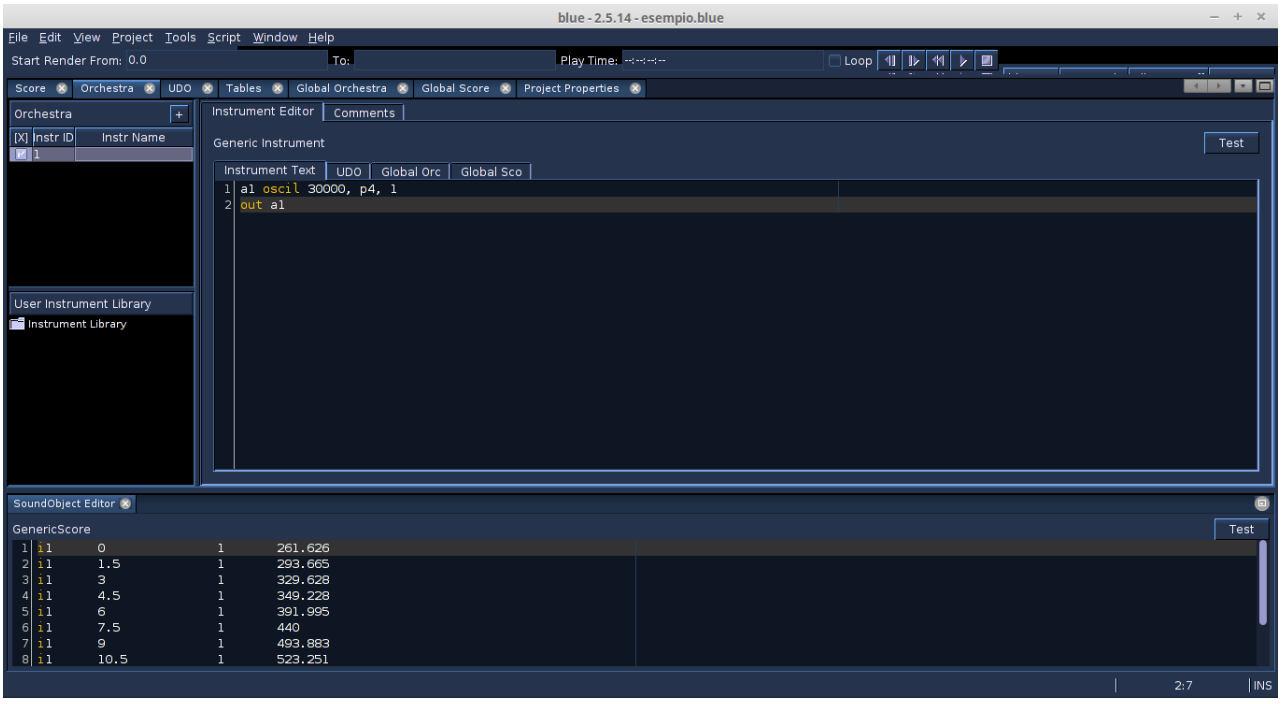


Figura 2: Finestra di lavoro di Blue

Anche in questo editor è caricato il programma del primo dei due esempi visti nel capitolo 2.

In questo caso siamo in presenza di un prodotto di alta professionalità, con molte funzioni in più rispetto all'editor che abbiamo visto prima, ma che ci porta fuori dai limiti di questo manualetto.

La citazione era comunque doverosa per un ossequio a Steven.

Possiamo procurarci Blue scaricandolo dal sito di Csound.