

AUDIO FACILE (18)

Sintesi Granulare (2)

di Alfredo Capozzi

Parlare di tecniche di sintesi non è facile, anche perché nel tempo, grazie anche ai processi informatici, si sono molto ampliate le possibilità di manipolazione del suono in tutte le sue caratteristiche. Ciò comporta ovviamente da parte del sottoscritto un'attenzione particolare a questo settore ed a come si evolve attraverso i plug-in ed i programmi sottesi a tale compito. Tuttavia, rincuorato da un nostro lettore, ho voluto dare spazio anche ad un programma verso cui, in tutta sincerità, non ripongo molta simpatia. Si tratta di Csound, spesso indicato su queste pagine come programma potentissimo nei risultati, ma molto ostico da gestire. Alessandro Petrolati, esperto dello stesso, ci darà un piccolo aiuto a capirne il potenziale.

Piccola premessa

Dopo l'articolo sulla sintesi granulare, Alessandro Petrolati, un nostro attento lettore e capace programmatore in ambiente Csound, ha richiesto di poter in-

serire all'interno di questa rubrica un suo applicativo che sta riscuotendo molti apprezzamenti in ambienti universitari e di ricerca, anche internazionali. Ligo al voler dare spazio ai nostri lettori, mi è sembrata l'occasione giusta

a leggere e ne scoprirete delle belle. Cedo, quindi, la parola ... pardon, il testo ad Alessandro Petrolati.

Un po' di storia: Csound e Density

per poter trattare, in maniera trasversale, questo tipo di linguaggio di programmazione musicale. Ciò che non sapevo, riguardo a Csound, era che questo programma consente, nelle versioni più recenti, anche la programmazione di una serie di comandi, con tanto di grafica, richiamabili da una serie di finestre in stile plug-in. Ciò che consente di fare **Density GSC**, trattato in questo numero dallo stesso A. Petrolati, è proprio questo: cioè applicare la modellazione granulare ai file audio o alla sintesi sonora mediante un'interfaccia grafica, mentre il vero elaborato sonoro viene svolto da Csound. Continuate

Csound è considerato tra i più potenti linguaggi per la "sintesi del suono". Per usarlo bisogna scrivere un codice articolato su due files sorgenti *orc/sco* (*Orchestra* e *Score*) e poi mandarli in esecuzione. L'approccio algoritmico lo rende versatile e potente (Fig.1), si possono realizzare illimitati **DSP** (*Digital Signal Processing*) e la potenza di calcolo dipende esclusivamente dalla velocità del processore. L'approccio però "a programmazione" lo rende ostile per l'utente media che preferisce applicazioni "accattivanti" che abbiano un'interfaccia grafica, come i plug-in.

Le origini del linguaggio risalgono a *Music1* (1957) di Max Mathews, dopo di che il programma si è sviluppato negli anni con *Music5* (1966), *Music360* (1968) e *Music11* (1973). Il "porting" del codice dall'assembler del PDP-11 al C di Unix (da cui prende il nome) originò *Csound* (1986) grazie a Barry Vercoe che lo sviluppò al MIT (*Massachusetts Institute of Technology*).

Il "compilatore" *Csound* gira attualmente sulle principali piattaforme operative (Windows, Mac OsX, Linux) e nell'ultima decade, attenendo allo stesso, si sono sviluppate moltissime distribuzioni "non ufficiali" cercando, in modi diversi-

```

<CsoundSynthesizer>
<CsdOptions>
-# eg -#0 -#sdac1 -#256 -#0256 -#0 temp-err temp-err
</CsdOptions>
<CsdInstrument>
sr = 44100.0
kmps = 1.0
nhkms = 2
nhkms2 = 2

gkch20 chowport "CMM20",2 ; 200
gkch20_1 chowport "CMM_1",2 ; Evolution 1
gkch20_2 chowport "CMM_2",2 ; Evolution 2
gkch20_3 chowport "CMM_3",2 ; Evolution 3

instr 30
gkch20_1 loopseg 0.0254,0 , 0.0,0.0, 0.9956,1.0, 0.4071,0.0278, 0.2372,0.0
gkch20_2 loopseg 0.0255,0 , 0.0,0.0, 0.5179,0.9874, 0.4821,0.0
gkch20_3 loopseg 0.0242,0 , 0.0,0.0, 0.3043,0.0782, \
0.6376,1.0, 0.1724,0.4232, 0.4855,0.0

knl random 10, 400, 10
knl oscil 1, knl,1
knl pkrick knl, 1, 10
kfl mwgklddr knl, [ gkch20_3 * 1000.0 + 1000.0 ] , .8

asig1 = afl
asig2 = afl

kvol = gkch20_1 * 1.0+0.0
kmp = kvol * 25000
kouts = 0.0125
kmpout = kmp * kouts
outs asig1 * kmpout, asig2 * kmpout
krms = 0.2725
kmpverrb = kmp * krms
sum asig1 * kmpverrb, 1
sum asig2 * kmpverrb, 2
krms = 5000.0
krms rms [ (asig1+asig2)* kmpout + (asig1+asig2)* kmpverrb ]
krms pwrth krms, 0.5
gkch20 = krms / kouts
endin

instr 100
asig1 arr 1
asig2 arr 2
asig1 revrb asig1, asig2, .93, 7680, sr, .2, 0
asig1d dblock asig1
asig1d clip asig1d, 2, 32000
kblock lineeg 0, .01, 1, p3-.02, 1, -.02, 0
outs asig1 * kblock, asig2 * kblock
asig1 0, 2
endin

```

Fig.1 - Una classica schermata di Csound.

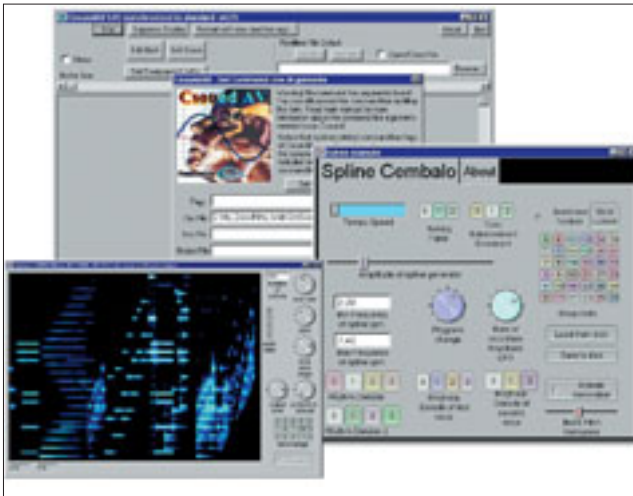


Fig.2 – Alcune videate presenti in CsoundAV.

ficati, di rispondere all'esigenza del moderno paradigma dell'audio. La versione italiana di *Gabriel Maldonado* (Fig.2 e 3) è stata la prima a sperimentare la potenza di *Csound* in tempo reale, iniziata nel '98 con *RTsound* e poi proseguita e migliorata fino all'attuale *CsoundAV* (2002), confluita di recente nella versione ufficiale di *Csound5* (ancora in versione beta).

Il progetto *Csound5* propone una versione unica del programma integrando gli sviluppi più importanti delle distribuzioni "non ufficiali". La stesura univoca delle API (*Application Programming Interface*), ossia dei codici operativi del linguaggio (*opcodes*), consolida ulteriormente la "portabilità" del codice. La versione di Maldonado ha, inoltre, introdotto il *FLTK* (*Fast Light Toolkit*), che consente lo sviluppo di interfacce gra-

fiche (*Grafic User Interface*), che agiscono direttamente sugli *script* di *Csound* (*widgets*). Grazie a questo tipo di programmazione il *FLTK* sarà presente anche in *Csound5*, assicurandone la compatibilità con il codice *CsoundAV*.

Se da un lato *Csound* è sinonimo di "codice", i *widgets* ora offrono la possibilità di sviluppare applicazioni **user-friendly** con interfaccia grafica,

questo è un argomento nuovo in via di sviluppo che presenta il vantaggio di allargare l'utenza anche ai non esperti, lasciando però il codice "aperto".

La prima versione di *Density* è stata pubblicata nel 2000: si basava su un programma di controllo scritto in Visual Basic da Flavio Tordini e tre file di codice *Csound*. I due *software* (*GUI* e *Csound*) comunicavano tramite protocollo MIDI, appoggiandosi sul *loop Midi* virtuale. L'introduzione dei *Widgets* (*FLTK*) nel 2001 ha permesso l'implementazione dell'interfaccia direttamente. Dalla versione *DensityWidgets* del 2001, il programma ha progredito insieme a *CsoundAV*. L'attuale *GSC* (*Granular Synthesis for Csound*) termina un ciclo di sviluppo pluriennale. Prendendo spunto da *GSC4* che, conosciuto sin dal 1988, è in asso-

luto il primo *script Csound* che implementa la *Sintesi Granulare*. L'autore del programma, Eugenio Giordani, è attualmente responsabile del Laboratorio Elettronico per la Musica Sperimentale (L.E.M.S.) del Conservatorio G. Rossini di Pesaro. Il progetto *Density* è nato al L.E.M.S. sia per lo **scheduling** dei grani,

sia per la gestione dei parametri di sintesi improntata sul meccanismo **deterministico/aleatorio**.

Presente, al momento solo in versione PC, in futuro *DensityGSC* girerà anche su altri sistemi operativi (Mac, Linux, Solaris, etc...), appoggiandosi a *Csound5*.

Come Installare Il programma

Il programma gira sotto *Windows*, 95/98 e 2000/XP. L'installazione è guidata da un *Wizard* che provvede alla creazione della cartella con i vari collegamenti (*Program*, *Desktop* and *Quick launch icon*). Alla fine del processo parte l'*installer* del *Driver LoopBe1*, necessario per simulare una serie di porte virtuali sui *computers* non dotati della porta MIDI fisica. Questo permette a *DensityGSC* di agganciarsi ad uno degli ingressi virtuali, in assenza di quello fisico.

Nota:



Nel caso si voglia disabilitare l'input MIDI, si dovrà rimuovere la *flag --K* dalle *<CsOptions>* nei *files* sorgenti (con estensione *.csd*), presenti dentro la cartella d'installazione e anche sul file "*Settings Flags*", raggiungibile dalla *shell*.

Cos'è DensityGSC

Si tratta di un software adatto per la "*Sintesi Granulare Asincrona*" (Fig.4). Un ambiente grafico, disposto su tre livelli di programmazione, ne controlla tutte le funzioni: la "Re-Sintesi" di materiali pre-esistenti (1), l'interazione *live* da una sorgente microfonica (2) e la generazione diretta di forme d'onda "prototipali" (3).

Tre file con estensione *csd* (file unificato *orc/sco*), contengono, in forma testuale, l'implementazione nel linguaggio *CsoundAV*:

1. **SoundFiles**
2. **Prototypes and Synth**
3. **Live**

La "*Density Shell*" permette di "istanziare" gli algoritmi di sintesi e di configurare il file "*settings flags*": frequenza



Fig.3 – CsoundAv consente anche la manipolazione di effetti grafici.



Fig.4 – DensityGSC (Granular Synthesis for Csound).

di campionamento, numero per il *device I/O* audio o MIDI e l'*editor* associato agli *script*.

Il primo programma “**SoundFiles**” processa files audio di tipo *Wav* o *Aiff*. Questa tecnica disintegra il segnale in ingresso e in tempo reale lo ricostruisce mediante particelle acustiche (grani). Manipolando i parametri di ricostruzione del segnale si possono ottenere sonorità straordinariamente complesse: nuvole granulari su bande frequenziali, dissociazione puntillistica della materia sonora, de-correlazione time/pitch, esplorazione micro-temporale.

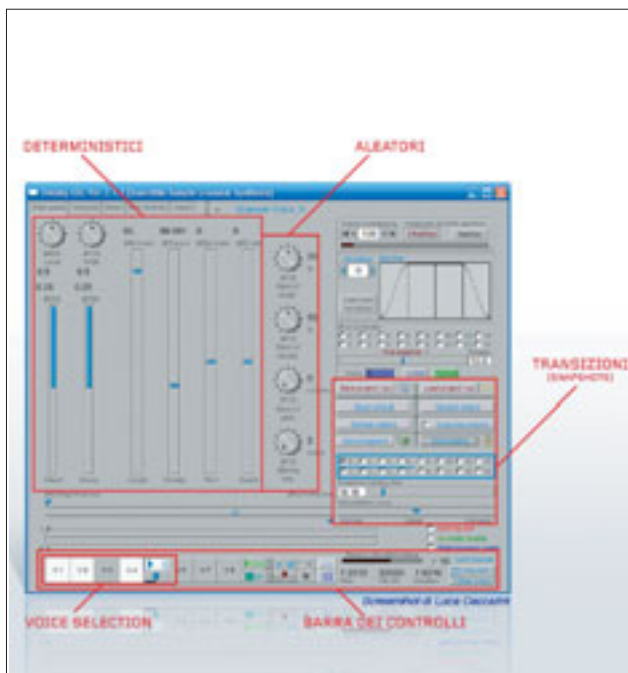


Fig.5 – DensityGSC: parametri della Main Page.

Il secondo “**Prototypes**” è generativo, a differenza di “**SoundFiles** o **Live**”. Non elabora alcuna sorgente, ma fornisce uno “*stream*” granulare di forme d’onda modellate matematicamente o importate da suoni campionati. I parametri principali sono gli stessi di “**SoundFiles**”, a cui però si aggiunge il “**Morphing**” che interpola due forme d’onda diverse, “**Grainlet**” per l’autocorrelazione frequenza/durata grano e un **Synth** polifonico suonabile da *Master Keyboard*.

Il “*Density Synth*” integra, inoltre, un generatore d’involuppo di tipo *ADSR* (**Attack**, **Decay**, **Sustain** and **Release**) e un banco di filtri. Sono preassegnati i controlli MIDI più generici: *Modulation Wheel*, *Pitch Bender*, *Note On/Off* e *Volume*. Allungando la durata del grano, è possibile sfruttare il motore granulare per la generazione di *patterns* ritmici, per esempio *groove/loop*.

Il terzo programma “**Live**” in linea di principio è simile a “**SoundFiles**”, ma agisce direttamente sul segnale “*bufferizzato*”, campionato dalla scheda audio (microfono).

Il terzo programma “**Live**” in linea di principio è simile a “**SoundFiles**”, ma agisce direttamente sul segnale “*bufferizzato*”, campionato dalla scheda audio (microfono).

Uno sguardo al programma

L’impostazione grafica dei tre programmi è coerente. Quattro schede (*tab*), poste in alto a destra della videata, consentono di accedere alle diverse funzioni: **Main**, **Vectorial**, **Mixer** e **MIDI Controls**.

Main (Fig.5), contiene i controlli dei parametri di sintesi, sono disponibili otto flussi

(*stream*) granulari chiamati *Granular_Voices*; ogni voce ha un colore diverso ed è configurabile individualmente. Ad esempio, in “**SoundFiles**” si possono caricare otto campioni audio, “*splitabili*” mediante i selettori “[#4] **Beginning loop point**” e “[#5] **Ending loop point**”.

La schermata si può dividere in quattro aree principali. Seguendo la **figura 5**, troviamo i parametri:

- **DETERMINISTICI**: [#0] **Level** controlla il valore d’ampiezza (volume) del segnale dei grani, mentre [#1] **Width** produce una pseudo – stereofonia. Se disabilitato (valore 0), i grani sono replicati sui canali Left/Right (mono); diversamente s’innesca proporzionalmente un processo casuale di distribuzione. [#2] **Attack** e [#3] **Decay** controllano la pendenza d’attacco e decadimento dell’involuppo del grano: valori di 0.5 non inducono nessuna distorsione, dell’involuppo. Valori di 0.25 o minori, invece, lo degenerano in un trapezoide, mentre valori prossimi allo zero in una finestra rettangolare. Controllando le pendenze si possono finemente scolpire timbri diversi.

[#4] **Beginning loop point** e [#5] **Ending loop point**, sono i selettori temporali del file (suono caricato in **soundFiles** oppure congelamento in **Live**). Funzionano come dei *markers* temporali semplicemente spostandoli a piacimento, un puntatore mobile traccia virtualmente il tempo. [#6] **Length** è un parametro importante perché gestisce la lunghezza del grano espressa in millesimi di secondo, non altera la morfologia del grano ma solo la durata. Sperimentando congiuntamente al parametro [#7] **Density**, si possono ottenere elongazioni temporali fedeli. [#7] **Density** gestisce il numero dei grani nel tempo, espresso in *gps* (grani per secondo). Il valore 0.5 produce un solo grano ogni 2 secondi (default) cento grani per secondo.

[#7] **Glissando**, presente solo in **Prototypes**, modula la frequenza all’interno del grano, provocando un glissato ascendente o discendente per tutta la durata del grano. [#8] **Pitch**, espresso

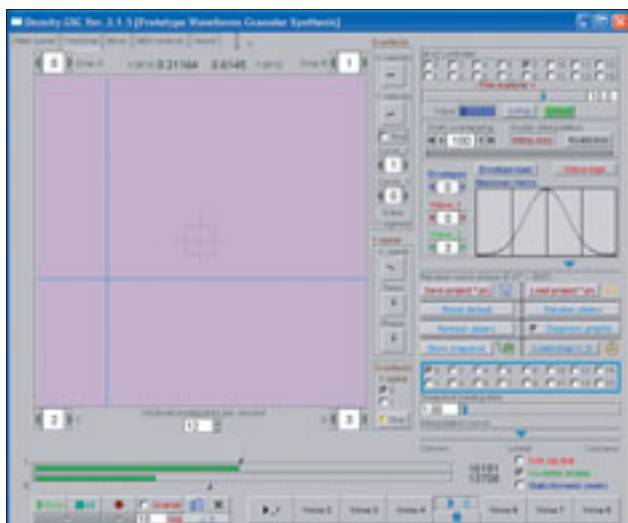


Fig.6 – Nella finestra Vectorial è possibile modulare i suoni in stile Wavestation.

in semitoni temperati, controlla l'altezza del suono. [#8] **Morphing** regola il punto d'interpolazione tra i prototipi *wave 1* e *wave 2*. Ad esempio, se *wave 1* è una sinusoide e *wave 2* un'onda triangolare, con *Morphing* a 0.5, ne consegue una forma intermedia. [#9] **Speed** manipola la velocità di lettura temporale del file, in altri termini la frequenza del tempo de-correlato dalla frequenza. Il valore zero provoca il congelamento del *time pointer*, valori negativi inducono la lettura all'inverso. Lavorando finemente sul parametro si realizzano esplorazioni micro-temporali del suono campione.

- **ALEATORI:** [#10] **Rnd % Length**,

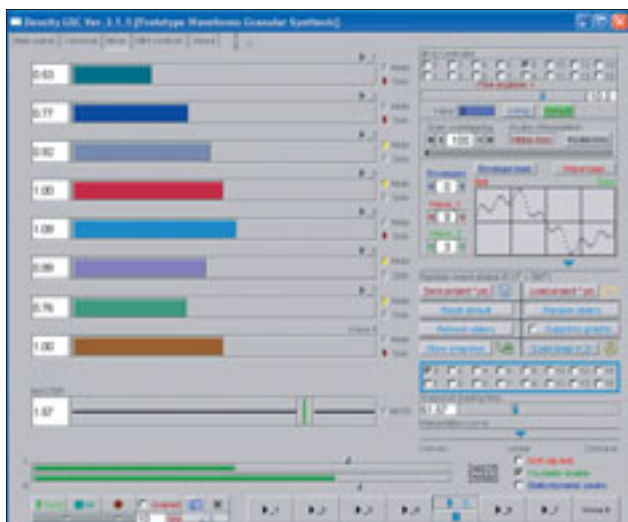


Fig.7 – La finestra Mixer relativa alle otto Voice.

[#11] **Rnd % Density**, [#12] **Rnd % Pitch** e [#13] **Rnd Jittering** determinano *randomizzazioni* sui diretti parametri deterministici. Dipendono dal valore del relativo parametro deterministico: quando azzerati non introducono alterazioni; aumentando i valori s'immette una deviazione casuale (+/-) rispetto al valore centrale deterministico.

Per esempio [#13] **Rnd**

Jittering agisce sul puntatore al file, provocando un movimento arbitrario. Valori elevati producono salti significativi del puntatore al file.

- **VOICE SELECTION:** selezionando una voce, compare il bottone *play*, questo la accende o la spegne, se è contrassegnata dal colore bianco significa che sulla voce è allocato un file. Inoltre, quando è in corso una avanzazione, viene mostrato lo stato d'avanzamento (*load snapshot*). La sessione di lavoro può essere salvata su un file di progetto (**Save project *.prj**).

Nel caso di "**SoundFile**" e "**Live**" si possono inoltre esportare i campioni audio caricati in memoria o congelati da *live-performance* salvandoli su un nuovo file associato al progetto. La barra dei controlli ospita comandi aggiuntivi: i bottoni con il simbolo *return arrow* accedono alle relative opzioni.

- **TRANSIZIONI:** le posizioni degli *sliders* dei parametri di sintesi, possono essere registrate mediante il comando **Store snapshot** nei 16 slot disponibili per ogni voce; il comando **Load**

snapshot le richiama. In *DensityGSC*, per "transizione" s'intende il passaggio (*morphing*) temporizzato tra due *snapshots*. Questo processo modella intrinsecamente "l'oggetto sonoro", potenziando il "**Sound Design**".

La transizione s'imposta con il sottostante comando di **Snapshot loading time**, mentre con **interpolation curve** se ne stabilisce la curva temporale che può essere concava o convessa. Con *curve* si fa in modo che pur rimanendo il tempo totale invariato si provoca un'accelerazione seguita da una decelerazione o viceversa del *morphing* dei suoni.

Nota:



Una transizione **comincia sempre dallo stato attuale degli sliders** e si muove verso lo *snapshot* richiamato. Nel caso in cui lo *slot* è vuoto (*empty*), assume i valori di *defaults*. Ogni voce gestisce indipendentemente i *presets* e le transizioni, tramite MIDI è possibile sincronizzare il **Load snapshot** delle otto voci associando il comando ad un "*program change*".

Vectorial, Mixer e MIDI: selezionando queste altre tre schede dal menu principale, si accede ai parametri di controllo dei parametri precedentemente descritti.

- **Vectorial pad** (Fig.6) è una superficie di controllo per il *morphing* complesso di quattro *snapshots* (posti agli angoli). Nella colonna di destra ci sono dei *tools* per automatizzare il *pointer pad*.

- **Mixer**, invece, controlla i livelli delle voci e del livello *Master*, permettendo l'ascolto isolato tramite i *solo/mute* (Fig.7).

- In **Midi**, infine, si associano i parametri di *DensityGSC* (*slider*, *knob* o bottoni) ai controlli MIDI in ingresso su uno dei 16 canali selezionato (Fig.8). Il dato ricevuto è automaticamente mappato sul valore del parametro, ma può essere riscaldato sui valori *min/max* in percentuale. Tutte le assegnazioni avvengono in tempo reale.

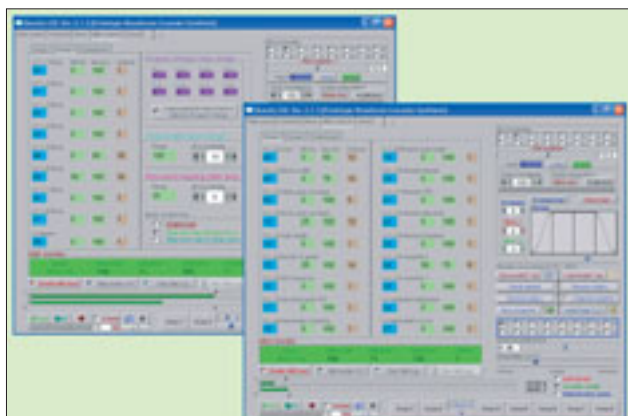


Fig.8 – In MIDI Control è possibile assegnare ai parametri di DensityGSC qualsiasi Control Change

Progettazione guidata di un suono

Dalla *Density Shell*, “lanciate” il primo programma “**SoundFiles**”, una finestra di dialogo aspetterà la selezione del numero corrispondente alla porta **MIDI input**. Se al computer non sono collegate altre periferiche, compariranno solo i *devices* virtuali. La schermata successiva elencherà le porte per l'**audio output**: *MME*, *DirectSound*, *ASIO*. Vi consiglio quest’ultimo per tempi bassi di latenza (dipende dalla vostra scheda audio).

Dato che per *default* non è allocato nessun campione, premete su *play*: ciò pro-

vocherà l’importazione di un file mediante la finestra **Soundfile load preferences**. Aumentate l’esponente (*default* 2^{19}) nel caso vi serva più memoria con i campioni più pesanti. Ripetete la stessa operazione nella seconda voce, caricando il medesimo file.

Portate il cursore **Speed** al valore zero (congelamento) su

ogni voce, in alternativa adoperate il pulsante **Freeze** sulla barra di controllo. Impostate il cursore dello *slider Pitch* della prima voce a +4 semitoni, mentre quello della seconda voce a -4. Con il selettore **Beginning loop** identificate un punto del file che ritenete interessante. Aggiustate gli altri parametri fino al conseguimento di un timbro piacevole.

Registrate gli *snapshots* sulla posizione 0 del banco premendo il pulsante **Store snapshot**, ripetere l’operazione sulla seconda voce. Selezionate, nel banco il *preset 1 (empty)* e impostate un tempo di 20 secondi di transizione su entrambe le voci. Mettete in *play* le

due voci e lanciate le transizioni. Questa *performance* di 20 secondi circa, muoverà i parametri dalla posizione corrente verso i valori originali. Chiaramente, è possibile registrare i campioni audio anche in *real-time*.

Conclusioni

Devo essere sincero. Se il futuro di Csound è anche questo (faccio riferimento alla gestione grafica), credo che il programma si ritaglierà una parte importante anche fra i semplici appassionati di Home Recording. *DensityGSC* ne rappresenta solo un’appendice dedicata alla sintesi granulare. Tenete inoltre presente che proprio questa tecnica di sintesi sta ricevendo l’attenzione di sempre più software house che iniziano ad implementarla all’interno dei loro strumenti: ultimo in ordine di tempo, lo stesso Virus TI nella recente versione 2.

Chi volesse avere ulteriori informazioni, invece, sul programma può collegarsi al sito di Alessandro Petrolati (www.alessandro-petrolati.com), dove, oltre ad indicazioni varie su Csound, potete scaricare il manuale d’istruzioni per approfondire la conoscenza con *DensityGSC*. Buon lavoro.

Per contatti:

a.capozzi@cm2magazine.it

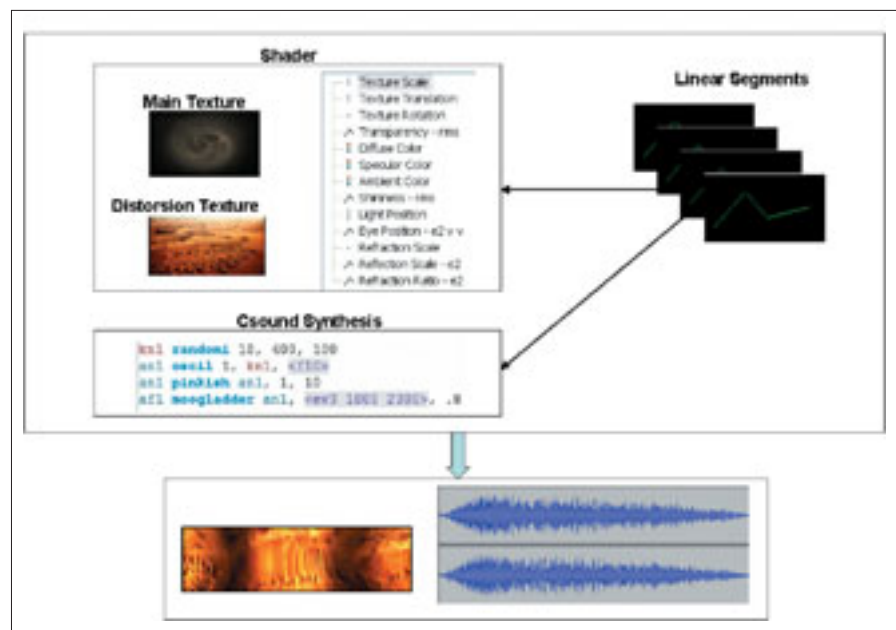


Fig.9 – Un altro screen relativo alle possibilità di gestione audio - video di Csound.obel, che ha gettato le basi concettuali sulla sintesi granulare.

ALTRI LINK CORRELATI

Accademia di Belle Arti di Urbino

<http://www.campivisivi.net>

<http://www.accademiadiurbino.it>

LEMS Eugenio Giordani

www.eugenio-giordani.it

CsoundAV by Gabriel Maldonado.

<http://csounds.com/maldonado>

Csound Home Page

<http://www.csounds.com/>