Conservatorio di Musica “Santa Cecilia" – Roma

Diploma Accademico di II Livello in
Musica Elettronica

TESI

“NORMAN BATES” e “MISTER C”:

due esempi di composizione algoritmica realizzati mediante
la formula di McAdams e la scelta casuale tra limiti

Relatore: Chiarissimo M° Giorgio Nottoli

Candidato: Luca Margoni

a.a. 2009/2010

A Giorgio Nottoli e Stefano Petrarca,
con grande riconoscenza

INDICE

"Chi diventa musicista è sfuggito
all'insegnante di Matematica, sarebbe
terribile se costui alla fine lo riacchiappasse."
Theodor W. Adorno, "Vers une musique informelle*", Darmstadt, 1961*

PREMESSA

Per definizione il termine 'algoritmo', ora assai usato, sta ad indicare una serie di procedimenti ordinati secondo un criterio-guida, spesso di natura logico-matematica, utili a risolvere un determinato problema.

La tendenza a scrivere musica secondo una logica di natura algoritmica risale quantomeno a centinaia di anni fa e ha spesso sfiorato, se non coinvolto fortemente, i più importanti compositori.

Senza voler troppo sistematizzare, né in senso cronologico, né in senso classificatorio, penso possa essere importante fare qualche riferimento relativo alla storia della musica occidentale.

Nel 1747 Bach si iscrisse ad una sorta di esclusivissimo sodalizio di musicisti, la *Corrispondierende Societät der musicalischen Wissenschaften.*
Quale compito annuale ciascuno degli iscritti aveva l'obbligo di presentare e illustrare di fronte a una severissima commissione un suo lavoro che impiegasse una logica di tipo algoritmico.
Questo spiegherebbe perché proprio le tarde composizioni del Maestro tedesco siano, oltre a dei capolavori assoluti, dei 'rompicapo' sul cui funzionamento logico si sono interrogate nei secoli miriadi di studiosi e di interpreti.
Ecco cosa dice Alberto Basso nel suo *Frau Musika* a tal riguardo:
"Il regolamento prevedeva che membri della Società potessero essere tanto i musicisti attivi nel campo della teoria quanto quelli impegnati nella pratica, ma che essi dovessero essere esperti di filosofia e di matematica, e come prova delle loro conoscenze scientifiche musicali dovevano presentare un lavoro teoretico o pratico"[1].

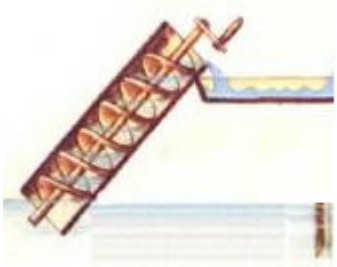
[1] A. Basso "*Frau Musika*", vol II, Torino, EDT, 1983, pp. 187-188.

Per la precisione Bach presentò una composizione dal titolo *Einige canoniche Veraenderungen über das Weinacht-Lied Vom Himmel hoch da komm ich her vor die Orgel mit 2 Clavieren und dem Pedal (* le famose variazioni sul corale *Vom Himmel hoch* BWV 769).

È utile, in questa sede, citare il famoso gioco *Musikalisches Wurfelspiel*, un tempo attribuito a Mozart, che permette a chiunque, lanciando dei dadi, di ottenere dei minuetti in perfetto stile mozartiano.

Gli stessi procedimenti canonici dei polifonisti antichi come l'inversione, il retrogrado, l'inversione del retrogrado sono evidentemente il frutto di una mentalità algoritmica, tendente ad applicare alla musica una logica decisamente molto vicina a quella logico-matematica.

Possiamo pertanto definire l'algoritmo come una specie di macchina logico-operativa in grado, tra l'altro, di compiere una parte del lavoro che spetterebbe al compositore, alla stessa stregua di una coclea egizia, che serviva per 'avvitare' l'acqua del Nilo fino a quattro metri d'altezza senza dover fare avanti e indietro con i secchi.



La coclea degli antichi egizi

Negli ultimi anni, soprattutto per quanto concerne la musica cosiddetta post-tonale, la mentalità *algorithm-oriented*, per così dire, si è fatta sempre più strada, e sempre più in una direzione matematica in senso stretto, basti pensare ai personalissimi procedimenti generativi che ciascun compositore è arrivato a crearsi.

L'esecutore stesso, di fronte a molte composizioni contemporanee, non può più esimersi dal comprenderne la struttura nonché la logica, spesso di natura algoritmica.

Naturalmente la storia della musica elettronica è caratterizzata ancor più da questa istanza. Non dimentichiamoci che proprio la possibilità di realizzare il serialismo integrale senza doversi scontrare con i limiti degli strumenti acustici ma soprattutto con quelli degli strumentisti fu una molla che spinse una pletora di compositori verso la musica elettronica.

La $\sqrt[25]{5}$ come principio intervallare per le misture di *Studie II* (1954) di Stockhausen, l'ampio uso della stocastica sia riguardo alle tecniche compositive come in *Polytope de Cluny* (1972) di Xenakis, per sette delle otto tracce di un registratore, con l'ultima traccia riservata a controlli elettromagnetici per luci, laser e specchi, sia riguardo alla sintesi stessa, come accade per la sintesi granulare in *Riverrun* (1986) di Truax, la scelta casuale tra limiti frequenziali per le *texture*[2] di *Antony* (1977) di Wessel, le pseudo-ottave create da Chowning in *Stria* (1977) con l'impiego della sezione aurea e della successione di Fibonacci sono solo alcuni esempi nel mare delle fantasiose invenzioni scientifiche o, talvolta, pseudo-scientifiche che hanno caratterizzato la produzione elettronica dagli albori ai nostri giorni.

L'ascolto e l'analisi di *Antony* sono stati la molla che mi ha spinto a lavorare con la scelta casuale tra limiti frequenziali: la mia ricerca è iniziata in questo modo. La notevole massa di oscillatori a dente di sega impiegata nel pezzo, con i suoi colori cangianti, la sonorità al tempo stesso statica e dinamica, ieratica e drammatica, questo senso di movimento talora di tipo tellurico, talora solare e ronzante, la perenne impressione di glissato verso l'acuto, uniti alla relativa semplicità nell'attuare su un moderno computer i procedimenti tecnici adottati da Wessel hanno acceso la mia curiosità; inoltre, dopo anni passati a provare tutti i tipi di sintesi, l'idea di riprendere a lavorare con la sintesi additiva mi affascinava non poco.
Il passo successivo, ispirato da alcune composizioni di Giorgio Nottoli e dalle lezioni da lui stesso svolte sull'argomento, è stato l'uso della formula di McAdams, un mezzo estremamente efficiente per generare fasci di sinusoidi (Per quanto concerne la descrizione tecnica degli anzidetti metodi generativi se ne veda la trattazione nel I e nel II capitolo).

Appare evidente come per un compositore contemporaneo sia praticamente impossibile prescindere dalla conoscenza della possibilità di avvalersi di tecniche algoritmiche nella costruzione delle proprie opere.
Tali tecniche possono servire a partire dalle microstrutture, ad esempio la generazione delle altezze dei suoni, fino alla macrostruttura, cioè la forma della composizione stessa. Ciascun musicista valuterà, in relazione alle proprie esigenze nonché alla propria personale natura, se, quando e quanto ragionare in tali termini.

Non posso infine fare a meno di citare, come nell'epigrafe del presente lavoro, Theodor W. Adorno:
"L'artista speculativo dovrebbe, più di ogni altro, preservare una porzione di buon senso che lo avverta di una cosa: ciò che non si capisce non deve essere necessariamente più

[2] D'ora in poi adopererò questo e tutti gli altri termini inglesi senza la "s" al plurale, come si fa per ogni parola inglese assunta nella lingua italiana.

evoluto, ma potrebbe essere semplicemente così primitivo e ottuso che una tale possibilità non viene neppure in mente"[3].

Anche alla luce di queste affermazioni, comporre musica oggi significa forse cercare, nei propri limiti umani, di mantenere viva la capacità di sapersi osservare dall'esterno, con il buon senso prescritto nella citazione di cui sopra unito a una dose, oggi più che mai rara, di una sana auto-ironia.

Questo scritto descrive sinteticamente il lavoro implementativo e musicale da me svolto nell'arco di due anni, sfociato nella composizione di due pezzi, "Norman Bates" e "Mister C", nonché nell'ideazione di un *plugin* VST per l'elaborazione audio in tempo reale.

[3] Theodor W. Adorno *Vers une musique in formelle, 1961* in *Immagini dialettiche*, a cura di G. Borio, Torino, Einaudi, 2004.

CAPITOLO I

La formula di McAdams e alcune sue possibili applicazioni nella musica elettroacustica

Stephen McAdams, professore di Teoria Musicale presso la Schulich School of Music di Montreal, si occupa prevalentemente di percezione dell'ascolto, in particolare, come si legge nella scheda esplicativa della McGill University, della quale la Schulich School of Music è un dipartimento:
"Prof. McAdams is interested in auditory perception and cognition in everyday and musical listening. Topics of particular interest include examining: 1) the mechanisms of auditory analysis of complex scenes with multiple sources of sound, 2) the perception of the timbre of musical instruments, 3) auditory psychomechanics or the perception, recognition and identification of vibrating objects in the environment, and 4) the perception of musical materials and forms, particularly in naturalistic conditions like sitting in a concert. The primary emphasis of the research is on psychophysical techniques capable of quantifying relations between the properties of vibrating objects, acoustic signals or complex messages and their perceptual results. A long-term goal is to provide empirical data that will allow the integration of lower- and higher-level auditory processes."

Da un'ulteriore indagine effettuata attraverso il *Canada Research Chairs*, l'ente che classifica tutti i professori universitari canadesi, risulta, nella scheda relativa a McAdams:
"*Sound waves arise from the interactions of objects and must be processed by the brain for us to understand what is happening in the world. Listeners can separate multiple sounds into distinct auditory images, store these images in memory, learn implicitly the way sounding objects behave, and learn the rules and patterns that govern the structuring of complex sound sequences, such as speech and music.*

Dr. Stephen McAdams explores the perception and understanding of this sonic realm, from the short sound of an impacted metal bar to a large-scale piece of orchestral music. As the Canada Research Chair in Music Perception and Cognition, he is extending his previous laboratory work on sound event and musical structure perception to more naturalistic settings that also involve listeners' other modalities, such as vision and action.

Dr. McAdams explores sound source perception in complex environments that are real or artificially created by digital technologies. One of his goals is to understand more fully how humans grasp the sense of what is happening in the physical world by judiciously combining information from hearing, vision, and haptic (vibratory) senses during active behaviour.

Dr. McAdams also studies the nature of the learning processes in auditory perception of sound sources and new musical grammars. In addition, he is interested in the dynamics of hearing in an ever-changing world. The human brain continually processes sensory information and interprets it on the fly, making use of knowledge that has been acquired through past experience. To study these dynamic processes, Dr. McAdams is developing new methods for measuring perception, emotional reaction and understanding in real time."

Nessuna notizia, come si può vedere, riguardo alla formula né alla paternità della stessa, per cui mi sono attivato in ogni modo per ottenere l'indirizzo *e-mail* di McAdams e, una volta ottenuto, gli ho scritto:

"*Esq. Professor McAdams,*

I'm writing from Italy this mail to you because I'd like to get some notices about the formula

"$\boldsymbol{f_i = f_0 * [1 + fdev * (Rand - 0.5) * 2] * i^{fexp}}$"[4]

In Italy this formula is called "Mc Adams formula" but in every research I've made on the Web I've never been able to find any relation between this formula and your name.
Would you kindly send me some notices about the birth of this formula, the purpose of its birth and the use you have been making during this years?
I'm sorry to bother you, Professor McAdams, but after a long period of unsuccessful researches I thought to ask notices to the source, that is you.
I'd be happy and proud to get a help from you.
Thank you very much and best wishes

Luca Margoni"

Ecco la risposta che ho ricevuto dall'interessato:
"*Dear Luca,*

This looks like a formula I invented for generating inharmonic spectra in the 1980s at IRCAM. I don't think I ever actually published anything on it, but probably presented it in conferences. It does two modifications to a harmonic series. the Rand operator generates a random number between 0 and 1 to create random perturbations around the harmonic

[4] La formula originaria dello studioso canadese prevedeva anche l'uso di una costante aggiunta $\boldsymbol{k}$, che rende lo spettro immediatamente inarmonico. Come scrivo in seguito, ho preferito fare riferimento a quella indicata da Giorgio Nottoli, cfr. G.Nottoli, *Ruota del tempo : composing the wheel of time*, Proc. MIPCM (Malta International Project of Computer Music), University of Malta-Mediterranean Institute, 1997 pp. 31-45.

positions. The amount of perturbation is controlled by fdev. The fexp exponent on the harmonic rank i stretches or compresses the harmonic series depending on whether fexp>1 or fexp<1, respectively. It has been used by several composers at IRCAM in the 1980s, but the only one I can remember exactly was a piece called Résonance by York Höller in which the random part was used in real time to create independent jitter on the harmonics to create a chorus effect on a violin-like sound.

I hope this helps.

With best wishes,

Prof. Stephen McAdams
Canada Research Chair in Music Perception and Cognition
CIRMMT - McGill University"

Da ciò ne possiamo dedurre che la formula è effettivamente da attribuirsi a McAdams ma anche che il suo ideatore non ha mai pubblicato nulla in proposito.

La formula di McAdams si inserisce in molti dei settori di ricerca dello studioso, in particolare quello sulla percezione del timbro.

Risulta, in campo musicale, di estrema utilità per agire sullo spettro sonoro e per creare delle *texture* o delle *gesture* timbricamente molto dense ed interessanti.

Essa si basa in realtà su pochi parametri e questa è forse una delle ragioni per cui risulta così efficace nelle sue varie applicazioni.

Eccone l'enunciazione:

$$\boldsymbol{f_i = f_0 * [1 + fdev * (Rand - 0.5) * 2] * i^{fexp}}$$

dove

$\boldsymbol{f_i}$ è la frequenza finale per l'ordine i della parziale

$\boldsymbol{i}$ è l'ordine della parziale (un numero naturale) , calcolato come segue

$$\boldsymbol{i = [h + (n - 1) * Rand\%(n)]}$$

$\boldsymbol{h}$ è la prima parziale che noi scegliamo nello spettro

$\boldsymbol{n}$ è il numero di parziali a partire da $\boldsymbol{h}$ che vogliamo impiegare, compreso $\boldsymbol{h}$ stesso; lo spettro sarà perciò compreso fra $\boldsymbol{h}$ e $\boldsymbol{h + (n - 1)}$

$\boldsymbol{f_0}$ è la frequenza fondamentale

$\boldsymbol{Rand}$ è un valore random compreso tra 0 e 1

$\boldsymbol{fdev}$ è la cosiddetta deviazione di frequenza, compresa fra 0 (cioè 0%) e 1 (cioè 100%);

poiché $\boldsymbol{Rand}$ è compreso tra 0 e 1 e viene poi sottoposto all'elaborazione $(\boldsymbol{Rand} - .\mathbf{5}) * \mathbf{2}$ il risultato della moltiplicazione $\boldsymbol{fdev} * (\boldsymbol{Rand} - .\mathbf{5}) * \mathbf{2}$ sarà perciò compreso fra $+\boldsymbol{fdev}$ e $-\boldsymbol{fdev}$.

Il tutto, sommato alla frequenza fondamentale $\boldsymbol{f_0}$, mi darà come risultato finale la frequenza

$$(\boldsymbol{f_0} - \boldsymbol{fdev}) * \boldsymbol{i}^{\boldsymbol{fexp}} < \boldsymbol{f_i} < (\boldsymbol{f_0} + \boldsymbol{fdev}) * \boldsymbol{i}^{\boldsymbol{fexp}}$$

Naturalmente, nel caso $\boldsymbol{fdev} = \mathbf{0}$, avremo $\boldsymbol{f_i} = \boldsymbol{f_0} * \boldsymbol{i}^{\boldsymbol{fexp}}$

Un discorso a parte merita $\boldsymbol{fexp}$, il vero "cuore" della formula di McAdams.

- Se $\boldsymbol{fexp} = \mathbf{1}$ abbiamo la successione canonica delle armoniche multipli interi della fondamentale; nel caso limite in cui $\boldsymbol{fdev} = \mathbf{0}$ avremo, pertanto, $\boldsymbol{f_i} = \boldsymbol{f_0} * \boldsymbol{i}$, laddove $\boldsymbol{i}$, ricordiamo, è l'ordine dell'armonico ed è un numero naturale.
- Se $\mathbf{0} < \boldsymbol{fexp} < \mathbf{1}$ abbiamo la serie contratta, multipla intera di $\boldsymbol{f_i} * \boldsymbol{fexp}$
- Se $\mathbf{1} < \boldsymbol{fexp} < 2$ abbiamo la serie espansa, anch'essa multipla intera di $\boldsymbol{f_i} * \boldsymbol{fexp}$

Un discorso a parte merita $\boldsymbol{fdev}$: questo parametro determina la deviazione rispetto alla frequenza prevista. Se esso è $\mathbf{0}$ non vi è alcuna deviazione ed il suono risulta un po' freddo. Se è intorno a $\mathbf{0.01 - 0.02}$ si ha una sonorità che assomiglia a quella di un coro. Aumentando sempre più ci si avvicina progressivamente ad un timbro simile ad uno sciame di api. Con il valore di $\mathbf{1}$ si ottiene una frequenza compresa tra $\mathbf{0}$ e il doppio di quella nominale.[5]

Nella formula originale di McAdams, così come usata all'IRCAM, vi era anche l'aggiunta di una costante $\boldsymbol{k}$ la quale, per ovvie ragioni matematiche, rende immediatamente lo spettro inarmonico.
Per controllare l'armonicità/inarmonicità dello spettro abbiamo visto che è sufficiente l'uso di $\boldsymbol{fdev}$.

[5] Tutta la spiegazione della formula è tratta dal materiale esplicativo del plugin "Texture" di Giorgio Nottoli, più precisamente dalla presentazione *Power Point* "A sound texture synthesizer based on algorithmic generation of micro-polyphonies", scritta dal Maestro nel 2007

Come vedremo più avanti con una serie di esempi pratici, le possibilità di utilizzo di questa formula nel campo della musica elettroacustica sono svariate.

Per ora è interessante osservare che, oltre alla compressione-espansione dello spettro armonico, fondamentalmente si possono ottenere due risultati:

- *Texture* (o tessitura o mistura, anche se probabilmente è preferibile il termine inglese, in quanto più completo);

- *Gesture* (o gesto, idem).

Il lavoro sulle *texture* consiste nel mettere in azione un alto numero di oscillatori, ciascuno dei quali oscillerà a una frequenza calcolata di volta in volta con la nostra formula, lasciandoli oscillare fino ad un successivo ricalcolo applicato ad ognuno. Tali oscillatori potranno partire (e di conseguenza cambiare) allo stesso istante oppure, come vedremo più avanti, potranno essere distanziati tra loro da un preciso istante di tempo.
In quest'ultimo caso si avrà una maggiore sensazione di colori cangianti secondo una scala graduale e impercettibile, senza soluzione di continuità. Scegliendo adeguatamente la fascia di parziali da utilizzare e la $\boldsymbol{fdev}$, si verranno a creare all'interno dello spettro delle combinazioni quasi armoniche (sappiamo, per fare un esempio, che sommando gli armonici dal 4° all'8° compresi si ottiene un accordo di 7ª di dominante F, 3ª, 5ª, 7ª, 8ª).

Per quanto riguarda le *gesture*, esse risultano più articolate nel loro uso.
Si possono azionare dei singoli oscillatori con dei suoni brevi, a distanze fisse o casuali, oppure si possono sovrapporre a piccoli gruppi, creando delle micro-*texture* momentanee, il tutto con delle combinazioni che sono quasi infinite.
Le *gesture* necessitano di un lavoro particolarmente approfondito, avendo ben presente l'obiettivo che si persegue, a causa di una loro caratterizzazione semantica molto più marcata che potrebbe renderne difficile un uso appropriato in ambito colto contemporaneo.

Non va dimenticata infine la possibilità di scrivere musica per gli strumenti tradizionali, calcolando le altezze dei suoni con la formula stessa e facendo muovere gradualmente $\boldsymbol{fexp}$ sino ad ottenere gli effetti spettrali che si desiderano, tenendo presente le proprietà della compagine strumentale , la quale possiede anch'essa delle caratteristiche timbriche a seconda degli strumenti che la compongono e dalle quali il compositore non potrà prescindere nei suoi 'calcoli'.
Quest'ultima tecnica è fra tutte sicuramente la più raffinata e richiede una maestria compositiva e un'abilità manipolativa che non molti posseggono, oltre a degli strumentisti dalla solidissima preparazione. Inoltre ci si deve scontrare con le pesanti limitazioni degli strumenti acustici, con la sola eccezione forse degli archi e dei fiati con *coulisse* che hanno la possibilità di glissare senza soluzione di continuità.

È uno dei procedimenti in uso tra i compositori cosiddetti spettralisti, con risultati talora (ma non sempre, per i sovraesposti motivi) interessanti.

La compositrice finlandese Kaija Saariaho, i francesi Gerard Grisey e Tristan Murail sono certamente tra i più importanti musicisti che si sono avvalsi della formula, essenzialmente per i calcoli relativi alle altezze da assegnare.
Essi hanno probabilmente utilizzato la formula originaria di McAdams, cioè quella con la costante aggiunta.

Ribadisco come la formula così come utilizzata da noi musicisti del gruppo di ricerca che fa capo al corso di Musica Elettronica del Conservatorio di Santa Cecilia diretti da Giorgio Nottoli, cioè $\boldsymbol{f_i = f_0 * [1 + fdev * (Rand - 0.5) * 2] * i^{fexp}}$, possa offrire delle possibilità di gran lunga superiori ai compositori che intendano sperimentarla, con una plasticità dello spettro maggiore, in quanto il prodotto $\boldsymbol{fdev * (Rand - .5) * 2}$ varia in continuazione, a differenza di qualunque costante.

CAPITOLO II

Generazione di partiture CSound mediante file eseguibili realizzati con il linguaggio di programmazione C: "Norman Bates", un pezzo creato con questi procedimenti, che attua la formula della scelta casuale tra limiti e quella di McAdams

La generazione di partiture per CSound mediante un *file* eseguibile è un esempio di composizione algoritmica di notevole efficacia per creare dei *file* audio in tempo differito. Nel *file* eseguibile verranno formulate delle domande relative a ciò che l'utente può liberamente decidere tra i parametri adoperati nell'algoritmo previsto, dopodiché, una volta generato il *file* ".sco", è possibile sintetizzarlo con CSound insieme ad un adeguato *file* ".orc". Risulta chiaro come questo metodo consenta un uso praticamente illimitato di oscillatori. Si deve solo pazientare per tutto il tempo in cui Csound sintetizzerà il tutto su di un *file* audio.

Il mio lavoro, in realtà, si è mosso verso due direzioni: l'una è consistita nell'implementazione della formula di McAdams mentre l'altra si è basata sulla scelta casuale tra limiti.

Quest'ultima consiste nella formula

$\boldsymbol{f = fmin + (fmax - fmin) * Rand}$ applicata al primo oscillatore e poi ai successivi

dove $\boldsymbol{f}$ è la frequenza finale, $\boldsymbol{fmin}$ e $\boldsymbol{fmax}$ sono rispettivamente il limite frequenziale minimo e quello massimo scelti dall'utente. Fra questi vengono calcolati dei limiti relativi intermedi $\boldsymbol{fminrel}$ e $\boldsymbol{fmaxrel}$, che tendono a salire verso $\boldsymbol{fmax}$ ed entro i quali verrà calcolato il valore di $\boldsymbol{f}$.
$\boldsymbol{Rand}$ è un valore casuale tra 0 e 1. Dopo il calcolo di $\boldsymbol{f}$ si pone $\boldsymbol{fmin = fminrel}$ e in questo modo il limite $\boldsymbol{fmin}$ si avvicina sempre più a $\boldsymbol{fmax}$, producendo così un graduale glissato ascendente, in base alla mia scelta estetica volta a elaborare in tal senso le altezze dei suoni. Va però precisato che tale glissando è tendenziale: facendo la media tra tutti gli oscillatori messi in azione in un ciclo di calcolo e messa a confronto la stessa con quella del calcolo successivo la seconda sarà una frequenza superiore. Ciò non toglie che, all'interno di uno stesso ciclo, possa tranquillamente accadere, per esempio, che la frequenza dell'ultimo oscillatore arrivi ad essere più bassa di quella del primo.

Se si fosse voluto invece produrre un glissato discendente, procedente nella direzione contraria (cioè da $\boldsymbol{fmax}$ a $\boldsymbol{fmin}$) allora la formula sarebbe dovuta essere

$\boldsymbol{f = fmax - (fmax - fmin) * Rand}$

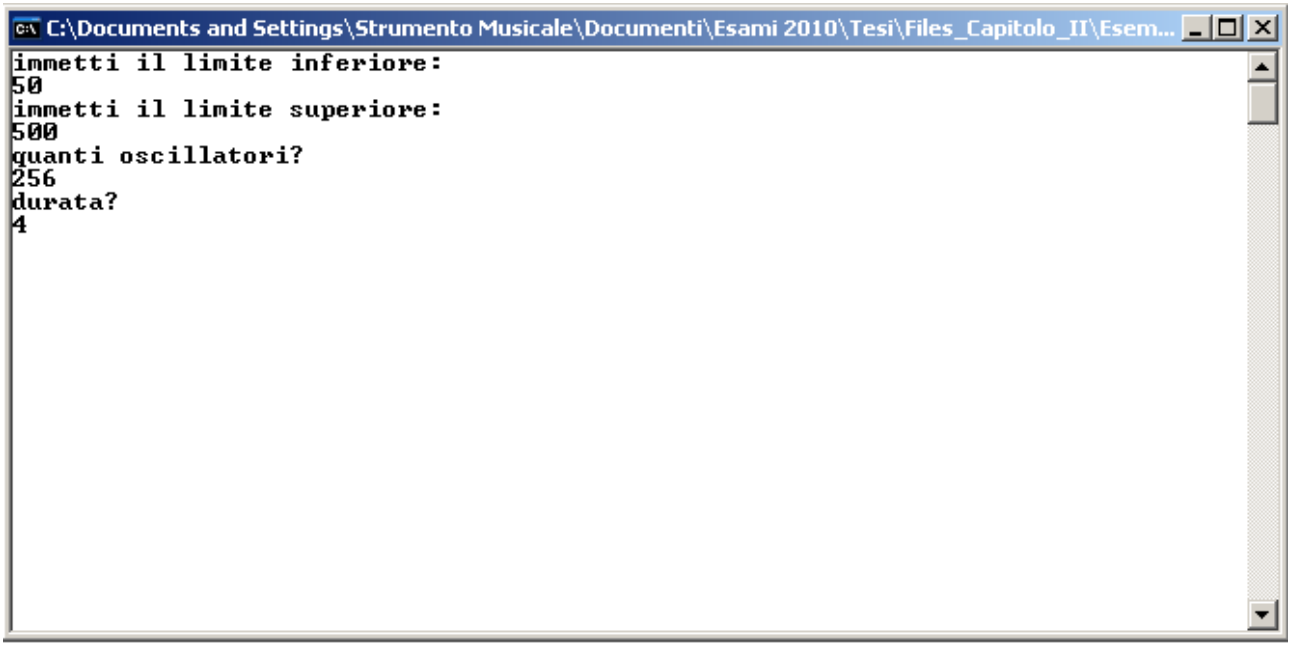
Assumendo poi $fmax = f$ prima del calcolo dell'altezza dell'oscillatore successivo. Questo avrebbe quindi prodotto il glissando desiderato verso il limite inferiore.

In alcuni casi ho invece fatto salire gradualmente secondo un passo preordinato ad ogni ciclo di calcolo sia $fminrel$ che $fmaxrel$, calcolando f all'interno di questi. Il risultato è stato un generale e più palese, oltre che regolare, glissando verso l'acuto.
Ecco un listato in C per implementare quest'ultima opzione (in rosso):

```
#include<stdio.h>
#include<time.h>
#include<math.h>
#include <stdlib.h>
#include <conio.h>
void main()
{
	srand( (unsigned)time( NULL ) );
	double ampiezza, k;
	float fmin, fmax, stereo, atk, iatk, dur, pitchrand, fa, fb, fosc;
	int oscillatori,i;
	printf("immetti il limite inferiore:\n");
	scanf("%f", &fmin);
	printf("immetti il limite superiore:\n");
	scanf("%f", &fmax);
	printf("quanti oscillatori?\n");
	scanf("%d", &oscillatori);
	printf("durata?\n");
	scanf("%f",&dur);
	k=log((double)oscillatori)/log((double)2);
	atk=dur/(float)oscillatori;
	ampiezza=1/k;
	printf("L'ampiezza di ciascun oscillatore sarà:\n%2f\015\012", ampiezza);
	pitchrand=((float)rand()+1)/(float)RAND_MAX;//genera un numero tra 1 e RAND_MAX e lo normalizza a 1
	fa = fmax/8;// ho fissato il primo limite superiore relativo
	printf("primo lim rel sup=%f\n",fa);//mi serve per controllare fa2 già nell'eseguibile
	fb = fmin;// ho fissato il primo limite inferiore relativo
	FILE	*fp;
	fp=fopen("Esempio_01.sco","wb");
	fprintf(fp,"; limite inferiore=%f, limite superiore=%f, limite rela=%f,
	limrelb=%f,oscillatori=%d\015\012",fmin,fmax,fa,fb,oscillatori);
	fprintf(fp,"f1 0 4096 10 1");
	fprintf (fp,"\015\012");
	fosc=fmin;
	for(i=1;i<=1;i++)
	{
		stereo=0;
		fprintf(fp,"i1 0 %f %f ",dur,fosc);
		fprintf(fp,"%f %f; %d\015\012",ampiezza, stereo,i);
		iatk=0;
	}
	for(i=2;i<=oscillatori;i++)
	{
		stereo=stereo+1/(float)oscillatori;
		iatk=iatk+atk;
		pitchrand=((float)rand()+1)/(float)RAND_MAX;
		fosc=(fb+(fa-fb)*pitchrand);//limite inferiore relativo che tende a salire verso il limite superiore relativo
		fprintf(fp,"i1 %f %f %f ",iatk, dur,fosc);
		fprintf(fp,"%f %f; %d\015\012",ampiezza, stereo,i);
```

```
	}
		while (fa<fmax)	{
			fa=fa+fmax/50;
			fb=fb+(fa-fb)/50;
for(i=1;i<=1;i++)
	{
		iatk=iatk+atk;
		stereo=0;
		fosc=fb;
		fprintf(fp,"i1 %f %f %f ",iatk,dur,fosc);
		fprintf(fp,"%f %f; %d\015\012",ampiezza, stereo,i);
	}
	for(i=2;i<=oscillatori;i++)
	{
		stereo=stereo+1/(float)oscillatori;
		iatk=iatk+atk;
		pitchrand=((float)rand()+1)/(float)RAND_MAX
		fosc=(fb+(fa-fb)*pitchrand);//limite inferiore relativo che tende a salire verso il limite superiore relativo
		fprintf(fp,"i1 %f %f %f ",iatk, dur,fosc);
		fprintf(fp,"%f %f; %d\015\012",ampiezza, stereo,i);
	}
			}
	fclose(fp);
}
```

Ecco la schermata di un *file* eseguibile per scrivere la *score*, alle cui domande ho risposto nel modo indicato:



Il risultato è stato un *file*

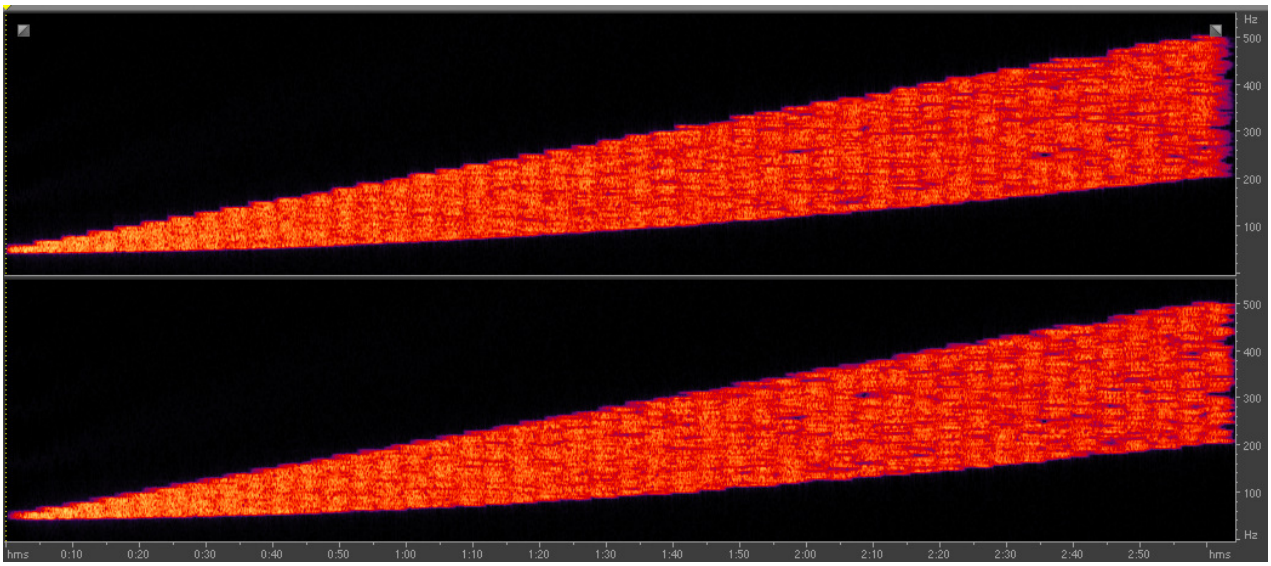
Files_Capitolo_II\Esempio_01\Release\Esempio_01.sco (Esempio01.sco)

Di ben 638 kb, che per un *file* di testo è una dimensione notevole, scritto dalla macchina in circa due decimi di secondo.

Il *file* audio ottenuto è

Files_Capitolo_II\Esempio_01\Release\ESEMPIO_01.WAV

Con la seguente immagine spettrale:



Ora illustrerò nel dettaglio le operazioni da me effettuate nello scrivere il programma in C:

- ho fatto in modo che l'utente (in questo caso me stesso) potesse scegliere il limite inferiore, quello superiore, il numero di oscillatori e la durata di ogni ciclo di calcolo;

- il ciclo di calcolo sta ad indicare quanto tempo passerà fra l'attivazione di un oscillatore e il cambio di frequenza successivo applicato allo stesso;

- ho diviso il ciclo di calcolo per il numero degli oscillatori, facendo così in modo che ogni oscillatore partisse sfalsato rispetto al precedente di un tempo pari al risultato ottenuto;

- a questo punto ho operato in due modi (poiché ho generato vari *file* eseguibili), creando sostanzialmente due tipi di partiture *CSound* :
  Tipologia 1) mantenendo la durata di ogni ciclo sempre uguale in tutta la partitura (sempre in base al numero immesso dall'utente): con questo sistema si possono ottenere delle *texture* senza soluzione di continuità tra un ciclo e l'altro, soprattutto se si impiega un elevato numero di oscillatori;
  Tipologia 2) ricalcolando automaticamente ad ogni ciclo in maniera casuale tale durata secondo un valore compreso tra 0 ed il valore scelto. Con tale sistema, lavorando prevalentemente con un piccolo numero di oscillatori e con suoni caratterizzati da un rilascio esponenziale, ho ottenuto, fra l'altro, delle *gesture* con dei cicli dalla durata variabile fra loro, ma omoritmici all'interno di ciascuno di essi;

- Durante ogni ciclo ho sempre spostato il fronte stereofonico da sinistra a destra, suddividendone la spazializzazone per il numero degli oscillatori.

Per quanto concerne invece l'uso della nostra formula di McAdams, essa è stata impiegata nel modo seguente, cominciando dall'eseguibile:

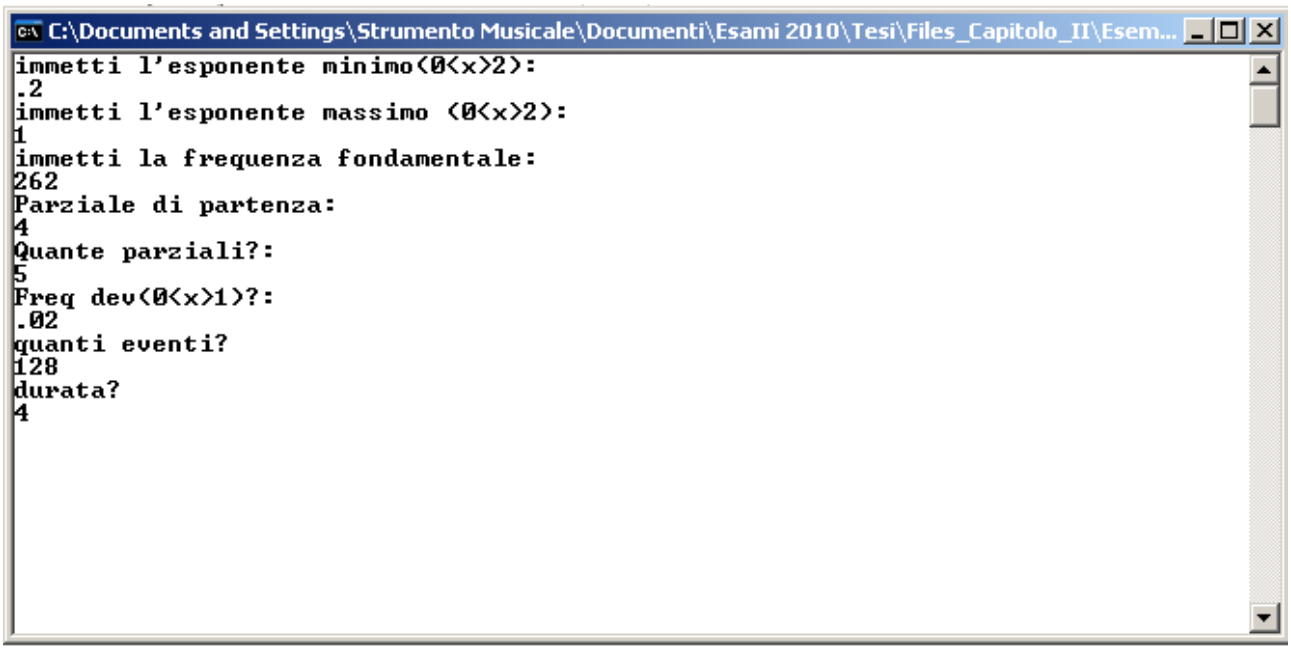
- all'utente viene chiesto il $\boldsymbol{fexp}$ iniziale ed uno finale, superiore al primo;
- poi la frequenza;
- a questo punto si richiede la parziale di partenza;
- poi il numero di parziali, contando anche quella di partenza;
- poi la $\boldsymbol{fdev}$;
- quindi il numero di oscillatori desiderato;
- infine la durata di ogni ciclo di calcolo.

Il programma fa in modo che $\boldsymbol{fexp}$ migri dal limite inferiore scelto dall'utente a quello superiore secondo uno *step* preordinato (0.02 ad ogni ciclo).

Un discorso a parte merita $\boldsymbol{fdev}$: poiché con la formula di McAdams si ottiene comunque una serie di suoni in qualche maniera ordinati tra loro, almeno all'interno di ciascun ciclo di calcolo, il che la rende meno casuale rispetto a quella della scelta casuale tra limiti, l'uso di $\boldsymbol{fdev}$, come chiaramente precisato nella mail inviatami da McAdams, può dare sicuramente una maggiore varietà allo spettro, con sfumature timbriche cangianti, a causa del calcolo casuale tra $\mathbf{0}$ e il valore scelto, ed anche con una notevole gamma coloristica, a seconda del succitato valore.

Il fronte stereofonico si muove sempre da sinistra a destra, equispaziato tra gli attacchi degli oscillatori.

Vediamo ora nel dettaglio la schermata del *file* eseguibile:



Questo è il listato in C relativo alla formula di McAdams (in rosso):

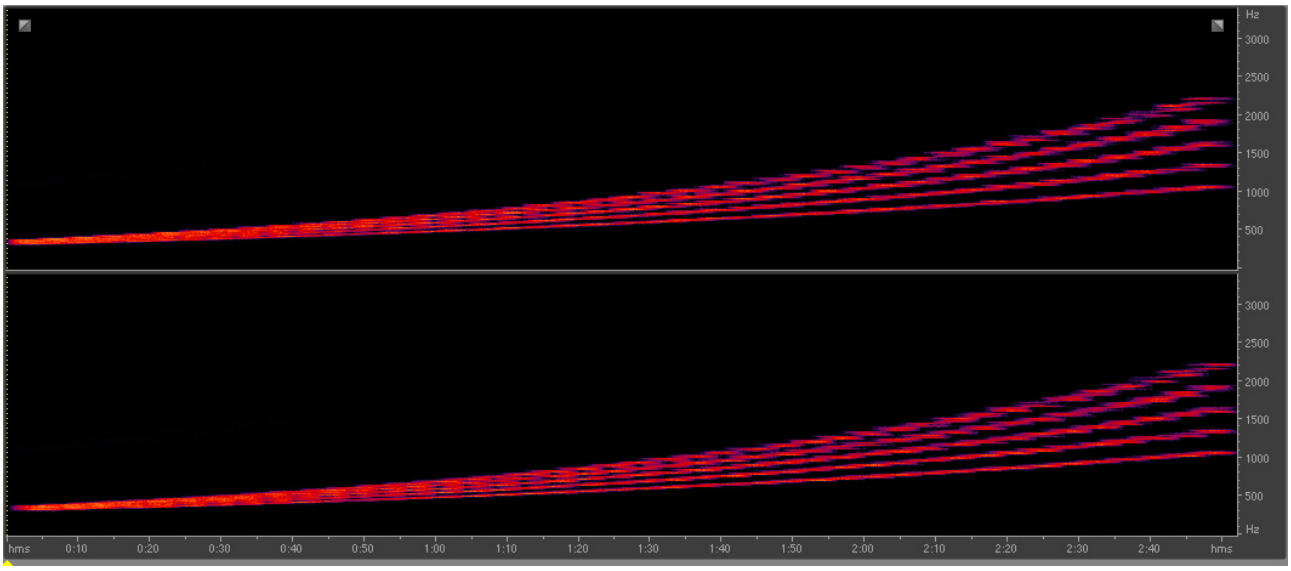
```
#include<stdio.h>
#include<time.h>//per il random
#include<math.h>
#include <stdlib.h>
#include <conio.h>
void main()
{
	srand( (unsigned)time( NULL ) );
	double ampiezza, k;
	float parziale, freq, esp,stereo,espmin,espmax,atk,iatk, dur, pitchrand,freqdev;
	int oscillatori,i, parzmin,numparz,j;
	printf("immetti l'esponente minimo(0<x>2):\n");
	scanf("%f", &esp);
	printf("immetti l'esponente massimo (0<x>2):\n");
	scanf("%f", &espmax);
	printf("immetti la frequenza fondamentale:\n");
	scanf("%f", &freq);
	printf("Parziale di partenza:\n");
	scanf("%d", &parzmin);
	printf("Quante parziali?:\n");
	scanf("%d", &numparz);
	printf("Freq dev(0<x>1)?:\n");
	scanf("%f", &freqdev);
	printf("quanti eventi?\n");
	scanf("%d", &oscillatori);
	printf("durata?\n");
	scanf("%f",&dur);
	espmin=esp;
	k=log((double)oscillatori)/log((double)2);
	atk=dur/(float)oscillatori;
	ampiezza=1/k;
	printf("L'ampiezza di ciascun oscillatore sarà:\n%2f\015\012", ampiezza);

	FILE	*fp;
	fp=fopen("Esempio_03.sco","wb");
```

```
	fprintf(fp,"; esponente=%f, esponente max=%f, Frequenza=%f,\015\012;percentuale di pitch shift=%f, Oscillatori=%d,
Parziale=%d, Numero di parziali=%d\015\012",esp,espmax,freq,freqdev,oscillatori,parzmin,numparz);
	fprintf(fp,"f1 0 4096 10 1/1 1/2 1/3 %f/4 1/5 1/6 1/7 1/8 1/9 1/10 1/11 1/12 1/13 1/14 1/15 1/16\015\012");
	fprintf (fp,"\015\012");
	for(i=1;i<=1;i++)
	{

		stereo=0;
		j=(rand()%numparz)+parzmin;
		pitchrand=(float)(rand()+1)/(float)RAND_MAX;
		parziale=(float)(freq*freqdev*(pitchrand -.5)*2.+freq)*(float)pow(j,esp);
		fprintf(fp,"i1 0  %f %f ",dur,parziale);
		fprintf(fp,"%f %f; %d esp=%f parz=%d\015\012",ampiezza,stereo, i, esp,j);
		iatk=0;
	}
		for(i=2;i<=oscillatori;i++)
	{
		iatk=iatk+atk;
		j=(rand()%numparz)+parzmin;
		pitchrand=(float)(rand()+1)/(float)RAND_MAX;
		parziale=(float)(freq*freqdev*(pitchrand -.5)*2.+freq)*(float)pow(j,esp);
		fprintf(fp,"i1 %f %f %f ",iatk, dur, parziale);
		fprintf(fp,"%f %f; %d esp=%f parz=%d\015\012",ampiezza, stereo,i, esp, j);
		stereo=stereo+1/(float)oscillatori;

	}
		while(esp<espmax){
			esp=espmin+ (float).02;
			stereo=0;
		for(i=1;i<=oscillatori;i++)
			{
		iatk=iatk+atk;
		j=(rand()%numparz)+parzmin;
		pitchrand=(float)(rand()+1)/(float)RAND_MAX;
		parziale=(float)(freq*freqdev*(pitchrand -.5)*2.+freq)*(float)pow(j,esp);
		fprintf(fp,"i1 %f %f %f ",iatk, dur, parziale);
		fprintf(fp,"%f %f; %d esp=%f parz=%d\015\012",ampiezza, stereo,i, esp, j);
		stereo=stereo+1/(float)oscillatori;
		espmin=esp;
			}
		}

	fclose(fp);
}
```

Infine il file audio ottenuto:

Files_Capitolo_II\Esempio_03\Release\ESEMPIO_03.WAV

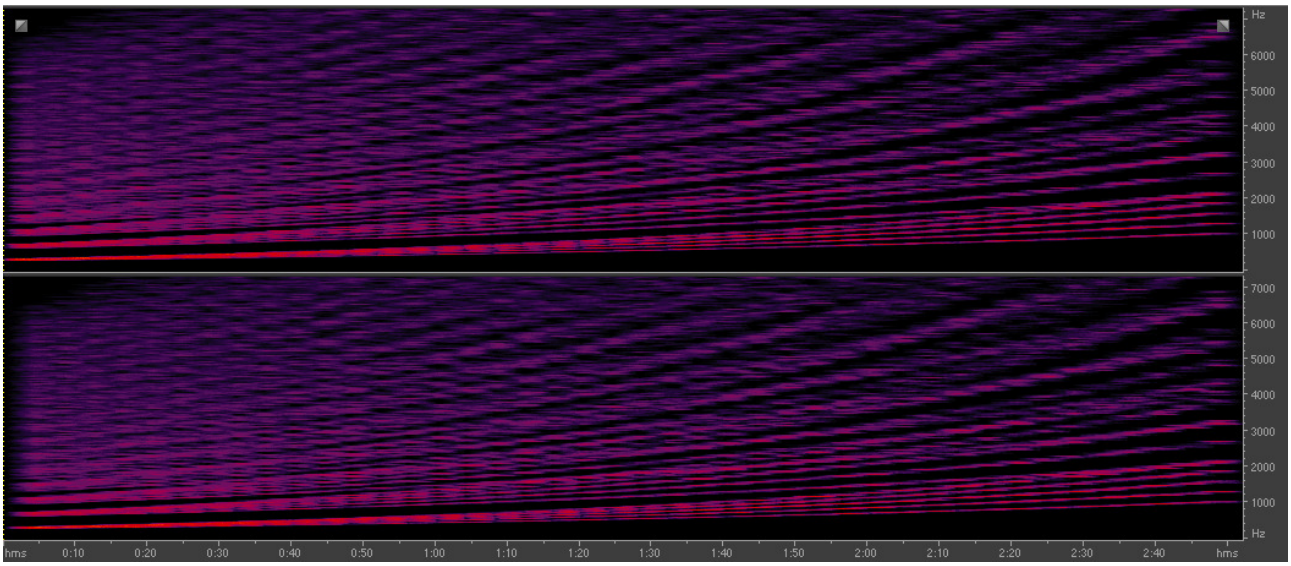
Con questa immagine spettrale:



Da cui si evincono chiaramente le cinque fasce sonore corrispondenti alle parziali dalla 4ª alla 8ª, scelte dall'utente nell'eseguibile.
La $\boldsymbol{fdev}$, pari a 0.02, dà allo spettro una timbrica quasi 'vocale'.
Anche nell'uso della formula di McAdams, come per la scelta casuale tra limiti, ho creato una **tipologia 1** e una **tipologia 2**, a seconda, ricordo, se la durata d'azione di ciascun oscillatore dovesse essere rispettivamente la stessa ad ogni ricalcolo oppure variare stocasticamente ogni volta, restando comunque fissa all'interno di ogni ciclo.
Rispetto all'immagine spettrale della *texture* ottenuta con la scelta casuale tra limiti, si nota in modo palese come gli oscillatori siano fondamentalmente "polarizzati" intorno alle parziali scelte, laddove invece nell'altro caso essi si trovano ad essere distribuiti in maniera equanime tra i limiti inferiore e superiore.
Per ottenere con la formula di McAdams qualcosa di simile è sufficiente aumentare la $\boldsymbol{fdev}$ in modo tale da "riempire" i vuoti presenti tra una parziale e l'altra, mentre è impossibile (o almeno estremamente ed inutilmente laborioso) con la scelta casuale tra limiti polarizzare gli oscillatori intorno a delle parziali prescelte.

Ho scelto di usare sempre degli oscillatori sinusoidali, essenzialmente per ragioni timbriche e anche perché con i moderni computer non ci sono più limiti alla sintesi additiva; pertanto l'uso di forme d'onda come la quadra, triangolare, ecc. non è più così importante, a meno che non avvenga per volontà precisa del compositore in senso estetico.
Nel mio caso, proprio per quest'ultima ragione, ho usato molto anche la sintesi FM, utile a creare delle atmosfere volutamente 'artificiali'.

Ecco qua sotto riportato il *file* Esempio_03 in cui nella score sono stati immessi i valori per ottenere degli oscillatori a dente di sega.
Files_Capitolo_II\Esempio_03\Release\ESEMPIO_03_SAW.WAV
Vediamo ora una parte dello spettro



Le originali cinque parziali sono rimaste, nel basso; ad esse se ne sono aggiunte altre $\mathbf{5 * 15}$, giacchè ho fatto in modo di avere un'onda quadra con sedici componenti.
Per fare un esempio, la parziale più alta della componente fondamentale arriva, alla fine del percorso, a circa 2200 Hz, pertanto, moltiplicata per sedici, la ritroviamo in alto a 35200 Hz (teoricamente, perché entra in ballo il *foldover* che la riflette in basso a 11600 Hz).
Questa tecnica, applicata a tutti i *file* che ho ottenuto, avrebbe prodotto, almeno a livello timbrico, un effetto non molto dissimile da ciò che accade in *Antony*, dal quale, come già detto, ho tratto essenzialmente la mia ispirazione riguardo ai procedimenti adottati.
In realtà la mia composizione non è un continuum e questa, unita a quella timbrica, è forse la differenza più importante riguardo alla tecnica compositiva adottata.

NORMAN BATES

Poiché, almeno nel mio caso, la ricerca tecnica è imprescindibile dalla ricerca, riuscita o meno che essa sia, di carattere musicale e dalla costante esigenza di forme espressive consone alla mia visione poetica mi sono messo al lavoro per la stesura di un pezzo. Anzi, possiamo dire che finanche le caratteristiche tecniche dei listati in C da me scritti siano state concepite in funzione del risultato musicale che volevo ottenere.

Dopo aver visto per l'ennesima volta uno dei film che mi hanno più colpito, cioè "*Psycho*" di Afred Hitchcock, è maturata in me progressivamente l'esigenza di realizzare un ritratto musicale del protagonista, Norman Bates.

L' intenzione non è stata però quella di ritrarre dall'esterno bensì di effettuare un viaggio dentro l'anima: un ritratto attraverso le sensazioni interne dell'uomo, certamente correlate agli avvenimenti esterni, ma soprattutto determinate, oltre che dalla natura stessa

dell'individuo, dalle reazioni che detti avvenimenti esterni provocavano nella fragile psiche di Bates.

Vorrei ora passare a descrivere il pezzo nelle sue varie fasi realizzative, anche e soprattutto in relazione al materiale di volta in volta usato.

Il pezzo, del 2009, dura 9'19" ed è stato scritto impiegando *file* audio generati con *CSound*. Tali *file* sono stati generati da partiture ".sco" a loro volta generate da *file* eseguibili come quelli segnalati sopra, indicando di volta in volta i parametri più idonei ad ottenere i risultati musicali desiderati. I *file* audio così generati sono infine stati assemblati con *Cubase* 5. Il materiale usato, di pura sintesi, non è stato sottoposto ad alcuna elaborazione anche se talvolta, come vedremo innanzi, se ne ha l'impressione.

Al tutto è stato aggiunto un video, una specie di "colonna visiva" della musica.

Dall'inizio del pezzo fino al minuto 0'32" c'è il tema iniziale: qui si è usata la formula della scelta casuale tra limiti, secondo la **tipologia 2**, quella, ricordiamo, del ricalcolo casuale della durata alla fine di ogni ciclo. Sono stati usati solo quattro oscillatori.

Il fatto che ciascun ciclo sia isoritmico al suo interno contribuisce notevolmente a tratteggiare il ritratto in questione, caratterizzato da una marcata tendenza all'ossessione. Il protagonista si affaccia alla vita ed esprime il suo stupore attraverso questo tema, nel quale, oltre a quanto detto prima, emerge un altro aspetto che ci tenevo ad evidenziare, cioè una costante ricerca di purezza infantile. Il movimento delle frequenze è abbastanza ondivago, ad indicare quella omnidirezionale ricerca di stimoli propria del bambino.
Files_Capitolo_II\Tema_iniziale\TEMA_INIZIALE.WAV

Dopo una pausa di circa 3" compare un altro *file* audio ALTRO_ANTONY_NO_MULTISTRUM_RIT_MERAVIGLIOSO02bis.WAV. In questo file ho creato una voluta mobilità dei limiti frequenziali, con il limite superiore che tende a salire anche parecchio sopra rispetto a quanto stabilito dall'eseguibile. Il tutto produce una specie di allucinata cantilena, resa ancora più evidente dalla scelta timbrica. Infatti il *file* ".sco" è stato sintetizzato con un *file* ".orc" che prevedeva la sintesi FM per ottenere un suono simile a una tromba. Gli oscillatori usati sono stati sedici e si è impiegata la formula della scelta casuale tra limiti . Questo è il tema della madre nel quale, grazie ad un'opportuna configurazione del calcolo, vi è una forte tendenza invece a reiterare le note vicine al limite superiore, in senso tendenzialmente ascendente. Questo, in congiunzione con la rotazione sul fronte stereofonico, produce un inevitabile effetto simile al Doppler, dato dal movimento e da un variare della frequenza che lo richiama. Attraverso un suono che può essere seduttivo ed opprimente al tempo stesso si fanno strada la coercizione e il desiderio di dominio, producendo subito un immediato effetto sul bambino.

In contemporanea si sente il tema del bambino, ottenuto non a caso con lo stesso file eseguibile servito per quello della madre messo però in relazione con un ".orc." di uno strumento simile a una chitarra. Anche qui abbiamo un effetto simil-Doppler.

Files_Capitolo_II\Figlio\ALTRO_ANTONY_NO_MULTISTRUM_RIT_MERAVIGLIOSO.WAV

Il tutto produce una specie di dialogo allucinato , di assimilazione e di mimesi unilaterale.

Da 2'51" a 3'31" la voce della madre è in a-solo, a rimuginare

3'31": la trasformazione è avvenuta e l'allievo ha superato il maestro. Qui si fanno strada l'ossessione e la follia sotto forma di micro-cicli su frequenze sempre uguali. Questo tema, ora in primo piano, fungerà variamente da sfondo fino a 8', una colonna sonora formata dai fantasmi della mente del protagonista.
Files_Capitolo_II\Ossessione\ALTRO_ANTONY_NO_MULTISTRUM_RIT_RANDMAGNIFICO04.WAV

3'31': inizia ora la fase più articolata. Ho qui inserito un Canone tra un *file* audio con il suono di chitarra e sé stesso con il suono di una stralunata marimba FM. Files_Capitolo_II\Canone\Canone.wav
In questo punto la vera personalità di Bates cerca di venire fuori, articolando dei pensieri e delle caotiche elaborazioni personali.

Purtroppo (5'58") l'incubo comincia sempre di più a farsi strada (Files_Capitolo_II\Incubo\Incubo.wav) sotto forma di una coppia di *file* audio correlati. Il delitto si sta compiendo, l'ossessione si fa sempre più strada e va avanti scivolando verso una calma spettrale.

Ritorna il tema iniziale (6'50"), immutato ed immutabile: il delitto è stato l'unico modo per ritornare bambino. Nel frattempo però la colonna sonora dell'Ossessione è sempre presente, nulla potrà più essere come prima.

Si fa strada ora, per la prima volta, una *texture* ottenuta con la formula di McAdams, valendosi della **tipologia 1**, una ricerca di pace della mente e dell'anima che procederà quasi sino alla fine. È questo il Tema dell'Epilogo Files_Capitolo_II\Epilogo\Epilogo.WAV.

Al minuto 8' sparisce definitivamente il tema dell'Ossessione e compare quello che può essere considerato un controcanto al tema dell'epilogo, un controcanto non oppositivo ma di accompagnamento, per così dire. In questo caso si tratta di una *gesture* ottenuta anch'essa con McAdams ma con la **tipologia 2**.

Alla fine del tema dell'epilogo un urlo acutissimo ed un progressivo scivolare verso il silenzio.

La catarsi finale.

Il video merita alcune considerazioni a parte.
Come già detto esso è una 'Colonna Visiva' della musica.
Ho cercato cioè di fare il lavoro inverso rispetto a quello del compositore di colonne sonore.

Durante l'esposizione del tema iniziale, un 'brodo' primordiale volto a descrivere la nascita e la progressiva acquisizione di coscienza, ci sono tanti colori sfocati e si intravedono vagamente delle figure in movimento. Alla fine del tema viene calato il sipario. Durante la pausa c'è il buio.

Grande ricchezza durante l'ingresso dei temi della madre e del bambino, gran dispiegamento di immagini ora vetrose, ora liquefatte o granulate ma via via sempre più tendenti alla fissità.

Durante l'assolo della madre tutto diventa bianco e nero, affascinante ma quasi senza via d'uscita.

L'ossessione si riflette in modo evidente anche sullo schermo, pur se è tornato il colore: le immagini si sdoppiano e si dondolano senza pausa, unico cambiamento un progressivo schiarimento dei colori sino alla sezione successiva.

La sezione del canone sviluppa il tema della vetrosità e della granulosità in modo nuovo ma tendente a ripiegarsi su sé stesso.

Il delitto è una potentissima allucinazione, una sensazione di cui il protagonista stesso non riesce ad afferrare la portata, uno sconvolgimento e uno sparigliamento necessari per la sopravvivenza e per ritrovare la propria identità.

L'esplosione coloristica serve a ricondurci nel 'brodo' primordiale dell'inizio, qui riproposto, *mutatis mutandis*, in occasione della ripresentazione del tema iniziale.
Anche qui c'è qualcosa di simile al calarsi di un sipario. Si tratta però dell'apertura di una porta, che ci conduce verso una nuova 'stanza dell'anima', la dirittura finale verso l'epilogo.

L'epilogo: immagini quasi televisive si fanno strada, come un guardarsi dall'esterno del protagonista, un progressivo spostarsi generale verso l'immobilità, infine un ultimo urlo di colore rosso (preannunciato pochi secondi prima) e nero, sino al nero finale.

CAPITOLO III

Implementazione della formula di McAdams mediante MaxMSP: Mister C, un pezzo in Real Time composto con questo programma

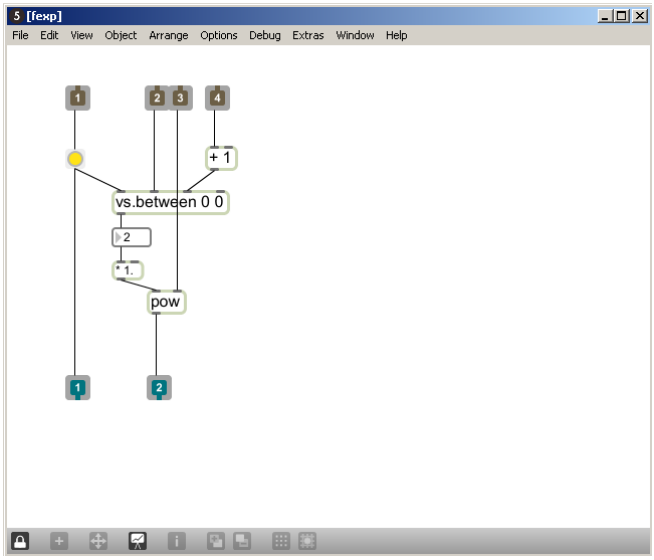
In MaxMSP si può facilmente realizzare una *patch* che permetta l'attuazione della nostra formula in ambito *Real Time*.

Cuore della *patch* che andrò ad illustrare sono i due oggetti *ioscbank~* e *counter*.
Counter, comandato da metro, ha come argomento il numero di oscillatori desiderati. La sua utilità è multipla: da un lato comanda tutti i calcoli che servono relativi alla frequenza di ogni oscillatore, dall'altro manda a *ioscbank~* l'indice dell'oscillatore stesso.
Questo determinerà un intervallo di ingresso di ciascun oscillatore pari al valore indicato da metro. Finito il ciclo di oscillatori ciascuno di essi si andrà ad agganciare con sé stesso senza soluzione di continuità, creando quindi quel glissando graduale che, come già detto, rende la *texture* particolarmente efficace.
Naturalmente si sarebbe anche potuto far agire tutti gli oscillatori simultaneamente. Sarebbe bastato utilizzare l'oggetto *uzi*, che avrebbe fatto partire tutti gli oscillatori in simultanea.

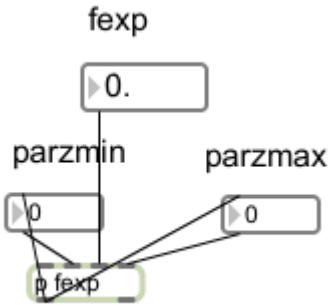
Passiamo ora ad osservare da vicino le varie parti componenti l'algoritmo.

La *patch* McAdams.maxpat ha al suo interno due *subpatch*.
La prima, *fexp* (prende il nome dall'esponente, *fexp*, appunto), ci dà come risultato l'elevazione a potenza fra una parziale scelta tra una minima e una massima fornite dall'utente e l'esponente $fexp$ (cioè $* \; \boldsymbol{i^{fexp}}$) ed è così fatta:

riceve la parziale minima e quella massima rispettivamente nel secondo e nel quarto *inlet*, mentre il primo è collegato all'oggetto *counter* che comanda il *button* all'interno della *subpatch*. Il terzo *inlet* riceve invece l'esponente *fexp*, un numero decimale compreso tra 0 e 2.
La parte del calcolo casuale è affidata a *vs.between*, che sceglie una parziale compresa tra la minima alla massima e la invia all'oggetto *pow*, che a sua volta la usa come base per il calcolo di cui *fexp*, proveniente dall'*inlet* 3, è l'esponente.

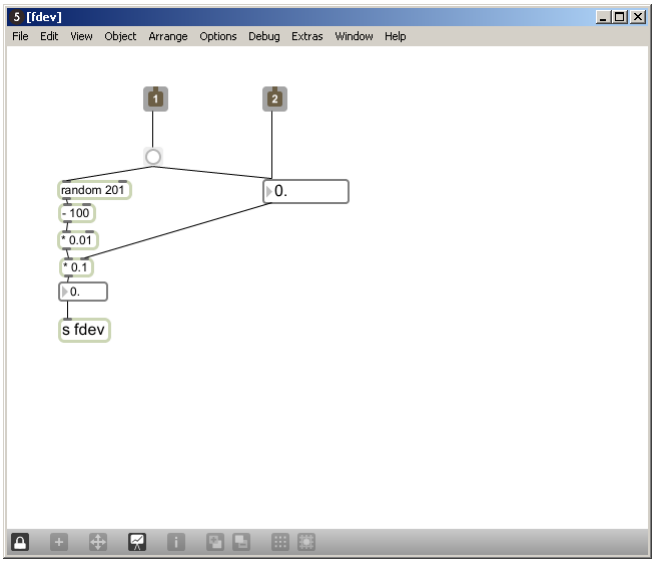


La *subpatch fexp*



La sezione di controllo esterna alla *subpatch fexp*

La seconda *subpatch* è chiamata *fdev*, dal nome dell'omonima variabile nella formula. In realtà all'interno della *subpatch* ciò che avviene è tutta la parte $\boldsymbol{fdev} * (\boldsymbol{Rand} - .\boldsymbol{5}) * \boldsymbol{2}$, che poi va moltiplicata per la frequenza e sommata alla stessa, per ottenerne un aumento o decremento percentuale in base alla effettiva *fdev* scelta. Ricordiamo che la variabile $fdev$, che viene immessa nell'*inlet* 2, è compresa tra 0 e 1, dove 1 è il 100%, in positivo o in negativo in base al calcolo casuale, della deviazione rispetto alla frequenza base. L'*inlet* 1, come per la *subpatch fexp*, è collegato con counter, che comanda il *button*.
Ecco nel dettaglio la *subpatch.*

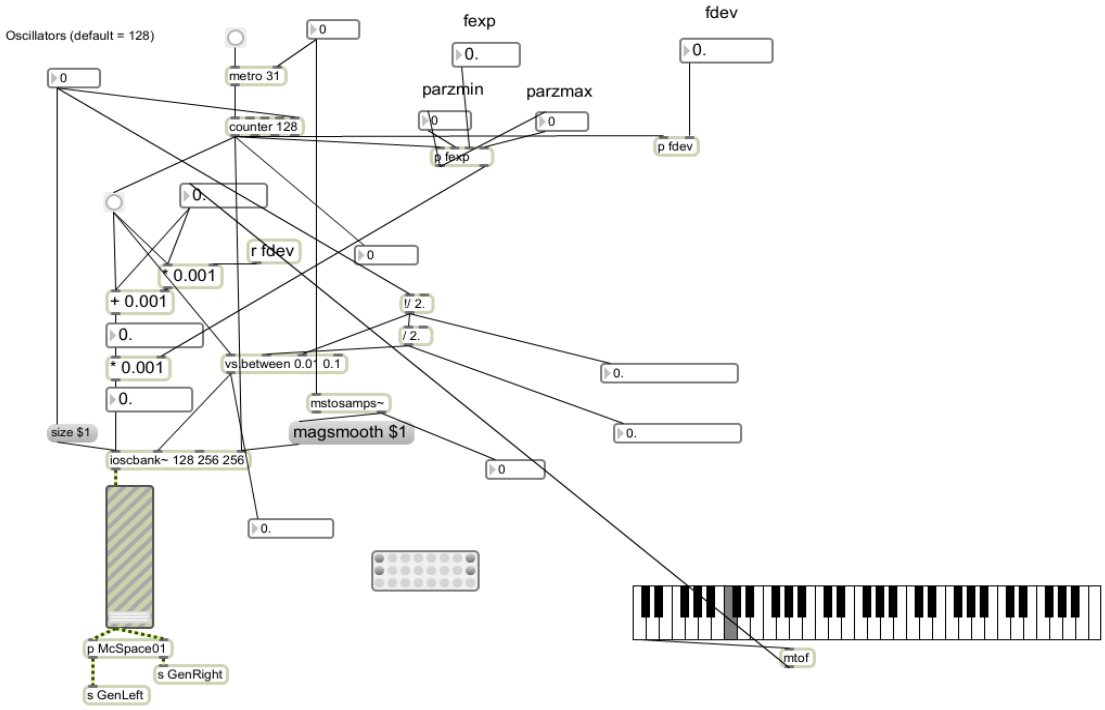


La *subpatch fdev*

Il risultato viene poi mandato nella *patch* principale, moltiplicato per la frequenza e sommato alla stessa.

A sua volta questo risultato (cioè $\boldsymbol{f_i = f_0 * [1 + fdev * (Rand - 0.5]}$) verrà poi moltiplicato per *fexp* (cioè $* \boldsymbol{i^{fexp}}$) e inviato al primo *inlet* di *ioscbank~*.

L'oggetto *counter*, in simultanea, comanda l'indice dell'*inlet* di destra che decide a quale numero di oscillatore debba essere applicato.
Ciascun oscillatore entra con un intervallo fissato dal numero che comanda *metro* e cambierà la sua frequenza quando *ioscbank~* riceverà il suo numero nel ciclo successivo.
La frequenza da usare può essere comandata sia all'interno del *box* numerico, come con gli altri parametri, che con una tastiera *kslider*.
Vi sono inoltre una *subpatch* per la spazializzazione stereofonica e un preset nel quale è possibile caricare le principali configurazioni desiderate.
Ecco così applicata la formula di McAdams, nella seguente *patch*:

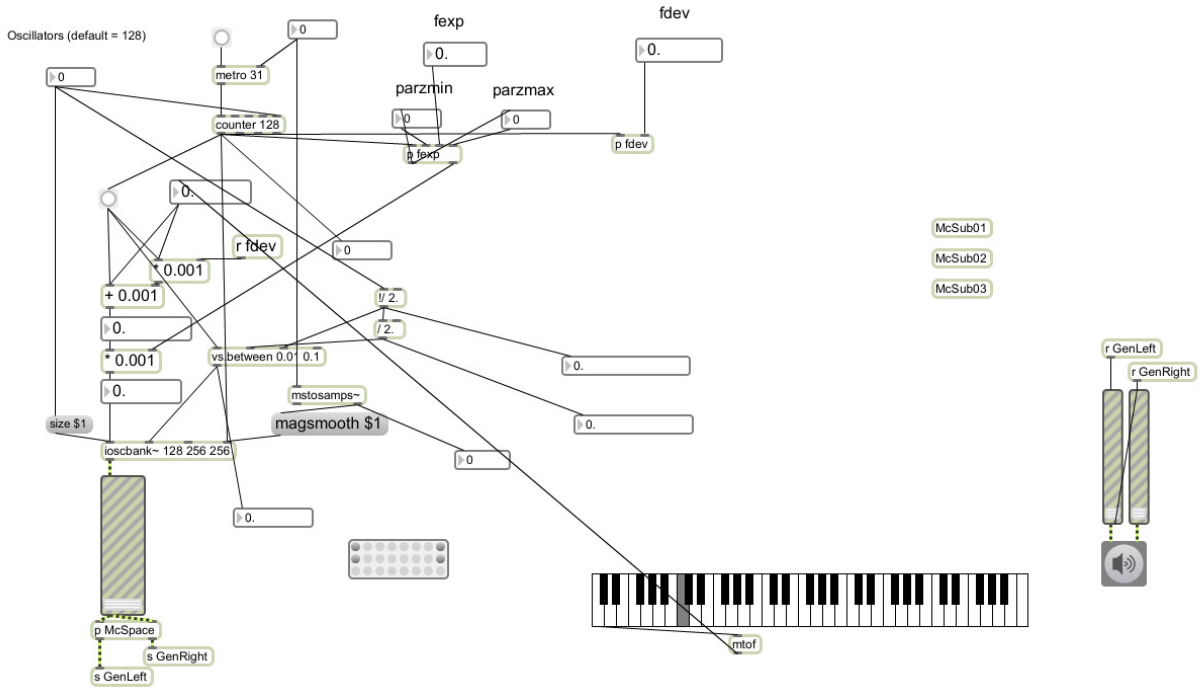


La patch su McAdams

MISTER C

Il pezzo, relativamente all'organico, prevede solamente la presenza di un esecutore al *live-electronics*.

Per quanto concerne il software ho usato la *patch* di cui sopra arricchita da pochi altri elementi, come si può vedere sotto.



La patch principale di Mister C

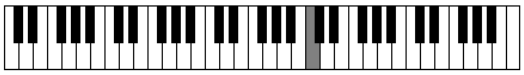
Come si può vedere, questa *patch*, la principale, ingloba al suo interno tre *abstraction*, identiche a quella della figura di pag. 23.
Inoltre essa ha, alla sua destra, un controllo generale del volume, al quale viene inviato l'audio delle quattro *patch*.
Le quattro *patch* ottenute vanno considerate, a tutti gli effetti, come quattro sezioni di una grande orchestra.
Ecco una breve descrizione del pezzo

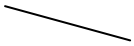
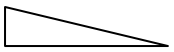
- È formato da cinque sezioni.
- Con l'eccezione della I sezione vi è una forte prevalenza delle *texture*.
- I sezione: *gesture*, 8 oscillatori per ogni *patch*, ogni oscillatore entra dopo 1200 msec nella *patch* principale e rispettivamente 1000, 800, 600 nelle altre; si agisce su fdev.
- II sezione: grande massa di oscillatori (2048), ogni oscillatore entra dopo 31 msec., si agisce sulla frequenza della terza *Subpatch* e alla fine, su fdev e fexp di tutte.
- III sezione: parziale gestualità, 32 oscillatori per *patch*, si agisce su fdev e sulla frequenza della terza *subpatch*.
- IV sezione: stesso *preset* della II sezione, ma la frequenza viene fatta glissare progressivamente da 48,99 (Sol 0) in tre delle *patch*, rispettivamente a 500, 1000 e 3000 Hz, mentre la quarta resta su di essa a mo' di bordone.
- V sezione: mentre nelle tre subpatch si continua a lavorare con l'ultimo preset, nella principale ritorna la *gesture* iniziale. Il tutto sfuma verso l'epilogo finale.

Nelle prossime quattro pagine riporto, rispettivamente, la legenda e la partitura del pezzo.

LEGENDA

McAdams, McSub01, McSub02, McSub01 = sono le quattro *patch*, va aperta prima la principale, McAdams, poi le altre tre, presenti come *abstraction* nella prima. Vanno lasciate aperte tutte, passando dall'una all'altra, ove richiesto, con i normali controlli Windows (ALT+TAB).
bang = è il *bang* iniziale che comanda il calcolo in ciascuna *patch*. Va usato solo all'inizio, dove prescritto dalla partitura.
p = preset, ciascuna *patch* prevede l'utilizzo di tre preset: p1, p9 e p16. p8 ha solo una funzione di controllo e non sarà mai usata in questa composizione. Nella patch McAdams c'è anche un p12, da usarsi a fine pezzo.
freq = frequenza, comandabile dal box numerico o dalla tastiera. Per quest'ultima si userà un normale pentagramma, tenendo presente che il Do centrale è posto come in figura



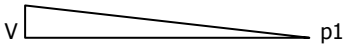
fdev = deviazione di frequenza, un numero compreso tra 0 e 1.
fexp = esponente, un numero compreso tra 0 e 2.
p1 = 512 oscillatori, metro 31, **fdev** .02, **fexp** 1, parziali da 1 a 9, **freq** differente in ogni *patch.*
p9 = 32 oscillatori, metro 31, **fdev** .1, **fexp** 1, parziali da 1 a 16, **freq** differente in ogni *patch.*
p16 = 8 oscillatori, metro tra 600 1 1200 a seconda della *patch*, **fdev** .02, **fexp** 1, parziali da 1 a 9, **freq** differente in ogni *patch.*
p12 = come la p16 ma con volume più alto: è presente solo nella *patch* principale.
 = un glissato discendente da effettuarsi su uno dei tre ultimi parametri.
 = un glissato ascendente da effettuarsi su uno dei tre ultimi parametri.
V = volume della singola *patch*. È l'oggetto gain~ in basso a sinistra.
GV = volume generale, presente solo nella *patch* McAdams. Sono i due oggetti gain~ abbinati, alla destra in basso: va azionato solo quello sinistro, che comanda entrambi.
 = diminuire il volume prescritto fino a zero: la rapidità del gesto è proporzionale all'estensione del triangolo in senso orizzontale.

Prima di iniziare:
caricare la *patch* principale e le tre *abstraction*;
attivare il convertitore ezdac~ a destra in basso;
entrare nella *subpatch* sotto vol di ciascuna delle tre *abstraction* e posizionare lo spazializzatore di McSub01, McSub02, McSub03 rispettivamente a metà tra centro e sinistra (circa .250 del *box* numerico), a metà tra centro e destra (circa .750) e a destra (1);
azionare in tutt'e quattro le *patch* il p16;

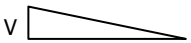
MISTER C

PARTITURA

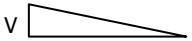
McAdams bang fdev .06



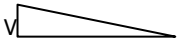
McSub01 bang fdev .06

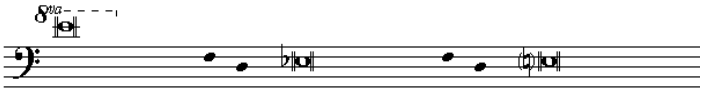


McSub02 bang fdev .06



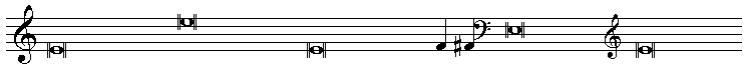
McSub03 bang fdev .06

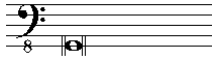
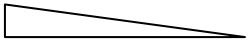
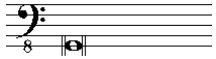
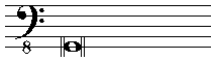
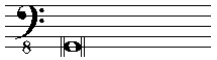
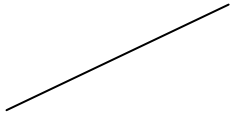




ca 1’30” - 2”

ca 3”

fexp .2 fdev 1	V p9 fdev .01	
fexp .4 fdev 1	V p9 fdev .01	
fexp .6 fdev 1	V p9 fdev .01	
fexp .8 fdev 1	V p9 fdev .01 	
		ca 2”

V p1   1000 Hz		V p12 GV 
V p1   500 Hz		
V p1 		
V p1   3000 Hz		
ca 3″		ca 1′

CAPITOLO IV

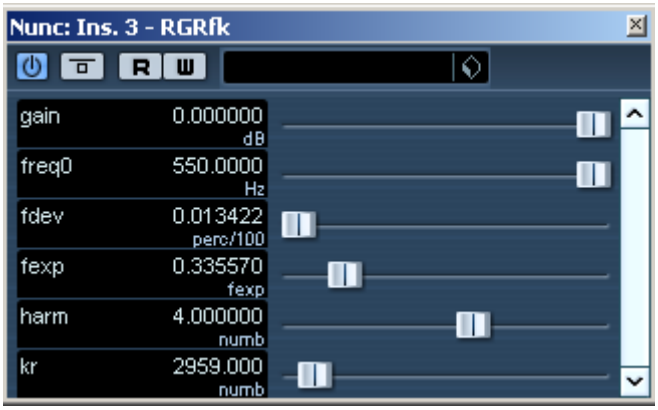
Un plugin VST creato in C++ che implementa la Formula di McAdams per la ringmodulazione con l'audio in Real Time

All'interno di *Cubase* esistono, come sappiamo, due tipi di *plugin VST*: gli *instruments*, da applicarsi alle tracce MIDI, e gli *effects*, da impiegarsi con le tracce audio.
Entrambi possono anche essere usati in *Real Time*, giacché tale *software* prevede tranquillamente, e con una notevole efficienza, questa possibilità.

Nel settore dei *VST instrument* un lavoro fondamentale è già stato fatto, probabilmente l'implementazione più complessa sinora realizzata a partire dalla formula di McAdams.
Si tratta del *plugin* "Texture", realizzato in anni di ricerca da Giorgio Nottoli, per parlare dettagliatamente del quale occorrerebbe una pubblicazione a parte, che permette di creare delle *texture* e delle *gesture* in *Real Time* a partire da note MIDI utilizzate come frequenze generatrici.
Proprio partendo dall'uso che io ho fatto come musicista di questo *plugin*, che è un modello di riferimento per chiunque voglia avvalersi di un *tool* per generare in *Real Time* dei fasci di sinusoidi e lavorare con lo spettro a fini timbrico-coloristici, è maturata in me l'idea di cominciare a cimentarmi con altre applicazioni della nostra formula.

In particolare ho voluto creare un *plugin* sviluppato all'interno dell'*SDK* della Steinberg, che prevede l'utilizzo del linguaggio di programmazione C++.

L'oggetto da me creato è invece un *effect*, in particolare un ringmodulatore, da me chiamato 'RGRfk', che permette di moltiplicare un segnale audio sia preregistrato sia in *Real Time* con una sola sinusoide, calcolata con la formula di McAdams.



Il plugin RGRfk

Il *plugin*, che si può vedere nell'illustrazione precedente, è composto da:

- **gain**, cioè il controllo di volume;

- **freq0**, cioè la frequenza della sinusoide;
- **fdev**;
- **fexp**;
- **harm**, cioè l'armonico massimo scelto dall'utente;
- **kr**, la frequenza di controllo;

La **fdev** è compresa tra 0 e 1; in ogni ciclo di calcolo, la cui velocità, come vedremo, è decisa da **kr**, verrà calcolato casualmente un numero compreso tra 0 e quello scelto dall'utente.
La **fexp**, come negli altri casi, è un numero compreso tra 0 e 2.
Lo **harm** è l'armonico scelto, compreso tra 0 e 8; se è 0 la ring modulazione non avviene, mentre se è un qualsiasi altro valore verrà scelto un numero casuale compreso tra 1 e il suddetto valore.
La **kr** è la frequenza con cui il calcolo viene effettuato, quindi è una *control rate*, ed è espressa in campioni; è compresa tra 0 e 44100, quindi se si sceglie 44100, alla frequenza di campionamento di 44100 Hz, tale calcolo avverrà ogni secondo, altrimenti in un tempo **x** dato da **kr** */*44100. Ad esempio, se **kr** è 2205 il calcolo avverrà ogni 2205/44100 secondi, cioè ogni 0.05 secondi (50 millisecondi).
Ecco di seguito i listati dei due *file* determinanti per ottenere il *plugin*, cioè AGain.hpp (in rosso) e AGain.cpp (in blu).
AGain.hpp:

```
#ifndef __AGAIN_H
#define __AGAIN_H

#include "audioeffectx.h"

//-------------------------------------------------------------------------------------------------------
class AGain : public AudioEffectX
{
public:
	AGain (audioMasterCallback audioMaster);
	~AGain ();

	// Processes
	virtual void process (float **inputs, float **outputs, long sampleFrames);//DEF funzione membro (tipo di dato e dato membro, tipo di dato2 e dato membro2, ecc.)
	virtual void processReplacing (float **inputs, float **outputs, long sampleFrames);//DEF i dati membro andranno poi inseriti nello stesso ordine in cui sono stati scritti!
		// Program
	virtual void setProgramName (char *name);
	virtual void getProgramName (char *name);

	// Parameters
	virtual void	setParameter (long index, float value);
	virtual float	getParameter (long index);
	virtual void	getParameterLabel (long index, char *label);
	virtual void	getParameterDisplay (long index, char *text);
	virtual void	getParameterName (long index, char *text);

	virtual bool	getEffectName (char* name);
```

```
	virtual bool	getVendorString (char* text);
	virtual bool	getProductString (char* text);
	virtual long	getVendorVersion () { return 1000; }

	virtual VstPlugCategory getPlugCategory () { return kPlugCategEffect; }

protected:
	float fGain, freq0, freq2, freq;//080910;//191010
	char programName[32];
	float seno[8192]; //roba nuova
	double x, fdev, fexp;//roba nuova
	int step, fase, i,step2, kr, contatore, k, kfinale, krfinale, deltastep;//roba nuova
};

#endif
```

AGain.cpp

```
#ifndef __AGAIN_H
#include "AGain.hpp"
#include "math.h"
#include<iostream>			//roba nuova per il random
#include<ctime>				//roba nuova per il random
#endif
#define tablen 8192			//roba nuova
#define pi 3.14159265			//roba nuova
//-----------------------------------------------------------------------------------------
AGain::AGain (audioMasterCallback audioMaster)
	: AudioEffectX (audioMaster, 1, 6)	// 1 program, 1 parameter only //090910 6 parametri
{
	kr=44100;				//080910 200 millisecondi (2205)
	contatore=0;				//080910
	fase=0;					//roba nuova
	fexp=2.;				//090910
	k = 8;					//090910
	kfinale=k;				//160910
	fGain = 1.;				// default to 0 dB
	setNumInputs (2);		// stereo in
	setNumOutputs (2);		// stereo out
	setUniqueID ('Gain');		// identify
	canMono ();				// makes sense to feed both inputs with the same signal
	canProcessReplacing ();		// supports both accumulating and replacing output
	strcpy (programName, "Default");	// default program name
	x=2*pi/tablen;				//roba nuova
	//randomizzazione della generazione di numeri casuali utilizzando il tempo: //roba nuova per il random
	srand(time(0));
	fdev=1.;//roba nuova per il random
	freq0=550.0;//roba nuova
	step=freq0*tablen/44100;// 16092010
	freq=(freq0*fdev*((rand()-0.5)/RAND_MAX)*2+freq0)*pow(((kfinale*(rand()%k))+1),fexp);//201010

for (i=0;i<tablen;i++)			//roba nuova
{
	seno[i]=(float)sin(x*i);
}

}

//-----------------------------------------------------------------------------------------
AGain::~AGain ()
{
```

```
	// nothing to do here
}

//------------------------------------------------------------------------------------------------
void AGain::setProgramName (char *name)
{
	strcpy (programName, name);
}

//------------------------------------------------------------------------------
void AGain::getProgramName (char *name)
{
	strcpy (name, programName);
}

//------------------------------------------------------------------------------
void AGain::setParameter (long index, float value)			//090910 tutto
{
	switch(index) //090910
	{
		case 0 : fGain =  value; break;
		case 1 : freq0 = value*550; break;
		case 2 : fdev = value; break;
		case 3 : fexp = value*2; break;
		case 4 : k = value*8; break;
		case 5 : kr = value*44100; break;


	}
}

//------------------------------------------------------------------------------
float AGain::getParameter (long index)
{
	return fGain;

}

//------------------------------------------------------------------------------
void AGain::getParameterName (long index, char *label)
{
//	strcpy (label, "Gain");
	switch(index) //090910
	{
		case 0:strcpy (label, "gain");break;
		case 1:strcpy (label, "freq0");break;
		case 2:strcpy (label, "fdev");break;
		case 3:strcpy (label, "fexp");break;
		case 4:strcpy (label, "harm");break;
		case 5:strcpy (label, "kr");break;


			}
}

//------------------------------------------------------------------------------
void AGain::getParameterDisplay (long index, char *text)
{
	//dB2string (fGain, text); istruzione originale
	switch(index)
	{
```

```
                case 0: dB2string (fGain, text);break;
                case 1: float2string (freq0, text);break;
                case 2: float2string (fdev,text);break;
                case 3: float2string (fexp, text);break;
                case 4: float2string (k, text);break;
                case 5: float2string (kr, text);break;



        }
}

//------------------------------------------------------------------------------------------
void AGain::getParameterLabel(long index, char *label)
{
        //strcpy (label, "dB"); istruzione originale
        switch(index) //090910
        {
                case 0:strcpy (label, "dB");break;
                case 1:strcpy (label, "Hz");break;
                case 2:strcpy (label, "perc/100");break;
                case 3:strcpy (label, "fexp");break;
                case 4:strcpy (label, "numb");break;
                case 5:strcpy (label, "numb");break;


                        }
}

//------------------------------------------------------------------------
bool AGain::getEffectName (char* name)
{
        strcpy (name, "Gain");
        return true;
}

//------------------------------------------------------------------------
bool AGain::getProductString (char* text)
{
        strcpy (text, "Gain");
        return true;
}

//------------------------------------------------------------------------
bool AGain::getVendorString (char* text)
{
        strcpy (text, "Steinberg Media Technologies");
        return true;
}

//------------------------------------------------------------------------------------------
void AGain::process (float **inputs, float **outputs, long sampleFrames)
{
   float *in1  =  inputs[0];
   float *in2  =  inputs[1];
   float *out1 = outputs[0];
   float *out2 = outputs[1];

   while (--sampleFrames >= 0)
   {
      (*out1++) += (*in1++) * fGain;    // accumulating
```

```
        (*out2++) += (*in2++) * fGain;
    }
}

//-----------------------------------------------------------------------------------------
void AGain::processReplacing (float **inputs, float **outputs, long sampleFrames)
{
    float *in1  =  inputs[0];
    float *in2  =  inputs[1];
    float *out1 = outputs[0];
    float *out2 = outputs[1];
    while (--sampleFrames >= 0)
    {
                        if (contatore<=krfinale)//160910 ho messo krfinale al posto di kr
                                contatore++;
                        else
                        {
                                contatore=0;
                                freq = freq2;//201010
                                freq2=(freq0*fdev*((rand()-0.5)/RAND_MAX)*2+freq0)*pow(((kfinale*(rand()%k))+1),fexp);
                                step2=freq2*tablen/44100;////191010
                                step=freq*tablen/44100;// spostata qui il 16092010
                                deltastep=(step2-step)/kr;//191010
                                krfinale=kr;//eliminato rand il 191010
                        }
                        (*out1++) =(*in1++) *fGain * seno[fase];        // replacing //roba nuova
                         (*out2++) =(*in2++) *fGain * seno[fase];       //roba nuova
                        step=step+deltastep;//191010
                        fase =(fase + step )% tablen; //roba nuova + SPOSTATO QUI DAL COSTRUTTORE E MODIFICATO (fase modulo
tablen)


    }
}
```

Con quest'ultimo *plugin* mi sono limitato, per ora, a fare dei semplici esperimenti.
In realtà si tratta solo del primo di una serie di *VST effect* con la formula di McAdams che ho intenzione di creare per scopi compositivi futuri.

Bibliografia generale

Theodor W. Adorno *Vers une musique informelle, (1961)*, in *Immagini dialettiche*, a cura di G. Borio, Torino, ed. Einaudi, 2004

L. Berio, *Intervista sulla musica,* a cura di R.Dalmonte, Bari, Laterza, 2007

R. Bianchini, A. Cipriani, *Il suono virtuale,* Roma, Contemponet, 2003

A. Cipriani, M. Giri, *Musica Elettronica e sound design,*Roma, Contemponet, 2009

C. Dodge, T. A. Jerse, *Computer music.Synthesis, composition and performance,* New York, 1985

G. Nierhaus: *Algorithmic Composition - Paradigms of Automated Music Generation.* Springer, 2008

S. Petrarca, *Matematica per la musica e il suono,* Roma, Aracne, 2010

M. Puckette, *Theory and techniques of electronic music,* San Diego, University of California, 2003

C. Road, *The computer music tutorial,* Cambridge, Massachusets, MIT Press, 1996

Bibliografia specifica

G. Nottoli, *Ruota del tempo : composing the wheel of time*, Proc. MIPCM (Malta International Project of Computer Music), University of Malta-Mediterranean Institute, 1997 pp. 31-45

G. Nottoli, *Generazione di microstrutture timbriche mediante processi stocastici applicati a modelli percettivi,* Proc. Capire e creare la musica, Seconda Università di Napoli, 2001

G. Nottoli, *Musical composition and parametric control of sound textures,* Proc. Capire e creare la musica, Seconda Università di Napoli, 2004