Classe	Instrumento	F. básicas
1	Diapasão	1,2
	Flauta	1,2,3,4
	Guitarra	1,2,3
	Marimba	1,2,3,5
	Violino	1,2,3,4
	Contrabaixo	1,2,3,5
2	Trompete	1,2,3
	Trompa Francesa	1,2
	Trombone	1,2
3	Clarinete	1,2,3
	Sax Tenor	1,2
	Sax Alto	1,2

Tabela 1: Ilustra um exemplo de classificação de instrumentos musicais e as respectivas funções básicas K-L.

Como podemos observar, a tabela 1, apresenta os instrumentos divididos em classes, as funções básicas utilizadas por cada instrumento. Um exemplo da eficiência da técnica é observado quando da síntese do clarinete figura 1: uma representação razoável utilizando Fourier exigiria em torno de 19 funções básicas [MOO 77], enquanto que por Karhunen-Loève, seriam necessárias apenas 3, como visto na tabela 1.

Este método de representação de sinais pode ser utilizado com sucesso em outras aplicações onde as informações componentes dos sinais variam rapidamente. Tal é o caso de tons não harmonicos.

Referências

[AHM 75] N. Ahmed, K.R. Rao, *Ortogonal Transforms for Digital Signal Processing*, Springer-Verlang, New York, 1.975.

[AND 70] H. C. Andrews, J. Kane, *Kronecker Matrices, Computer Implementation, and Generalized Spectra*, Journal of the Association for Computing Machinery, Vol 17, Nro 2, April 1.970, p 260-268.

[CAM 71] S.J. Campanella, G.S. Robinson, *A Comparison of Orthogonal Transformations for Digital Speech Processing*, IEEE Transactions on Communication Technology, Vol.Com-19, Nro 6, December 1.971, p 1045-1049.

[CHE 91] C.S. Chen, K.S. Huo, *Karhunen-Loève Method for Data Compression and Speech Synthesis*, IEE Proceedings-I, Vol. 138, Nro. 5, October 1.991, p 377-380.

[CHR 79] R.A. Christen, A.D. Hirschman, *Automatic Phase Alignment for the Karhunen - Loève Expansion*,IEEE Transactions on Biomedical Engineering, vol.bme-26,Nro 2, February 1.979, p 94-99.

[DeF 88] D.J. DeFatta, J.G. Lucas, W.S. Hodgkiss, *Digital Signal Processing: A System Design Approach*, John Wiley & Sons, New York, 1.988

[GRE 77] J.M. GREY, J.A. Moorer, *Perceptual evaluations of Synthesized Musical Instrument Tones*, J. Acoust. Soc. Am., Vol. 62, Nro 2, August 1.977, p 454-462.

[HUT 75] B.A. Hutchins, *Applications of a real-time Hadamard Transform Network to sound synthesis*, Journal of Audio Engineering Society,Vol 23, Nro 7, September 1975, p 558-562.

[MAR 92] J. F. Marar *Utilização da Transformada Karhunen-Loève em Síntese de Tons Musicais*, Dissertação de Mestrado -USP-São Carlos, 1.992.

[MAS 87] H. Massey, A. Noyes, D. Shklair, *A Synthesist's Guide to Acoustic Instuments*, Amsco Publications, New York, 1.987.

[MOO 77] J.A. Moorer, *Signal Processing Aspects of Computer Music : A Survey*, Proceedings of IEEE, Vol 65,Nro 8, August 1.977, p 1108-1137.

[PRO 88] J.G. Proakis, D.G. Manolakis, *Introduction to Digital Signal Processing*, Macmillan Publishing Company, New York, 1.988.

[SER 90] M.H. Serra, D. Rubine, R. Dannenberg, *Analysis and Synthesis of Tones by Spectral Interpolation*, J. Audio Eng. Soc., Vol. 38, Nro. 3, March 1.990, p 111-128.

[STA 88] J.C. Stapleton, S.C. Bass, *Synthesis of Musical Tones Based on the Karhunen-Loéve Transform*, IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 36, Nro. 3, March 1.988, p 305-319.

[WAW 89] J. Wawrzynek, *VLSI Models for Sound Synthesis* , Current Directions in Computer Music Research, The MIT Press, Massachusetts, 1.989, p 113-148

[WIN 78] S. Winograd, *On Computing the Discrete Fourier Transform*, Math. Comp. 32, 1.978.

[WOM 77] M.E. Womble, J.S. Halliday, S.K. Mitter, M.C. Lancaster, J.H. Triebwasser, *Data Compression for Storing and Transmitting*, Proceedings of the IEEE, Vol 65, Nro 5, May 1.977, p 702-706.

Agradecimentos

Ao Departamento de Computação da UNESP-Bauru, pelo afastamento concedido para a realização do programa de doutoramento no DI/UFPE.

À Capes-PICD e CNPq pelo apoio financeiro.

Ao Departamento de Informática da UFPE pela utilização dos recursos computacionais.

Ao Dr. Edson Costa de Barros Carvalho Filho (D.I./UFPE) pelo grande incentivo para realização deste artigo.

The Synthesis of Complex Sonic Events by Functional Iterations

Agostino Di Scipio & Ignazio Prignano

Laboratorio Musica & Sonologia
Dipartimento di Matematica Pura ed Applicata, Univ. di L'Aquila (Italy)
e-mail *lms@vxscaq.aquila.infn.it*

Abstract

This paper introduces a non-standard sound synthesis method that the authors, with terminology drawn from the literature on chaos theory, call *synthesis by functional iterations*. It describes the general formalism and the application of a difference equation model - the "sin map". Sounds generated with this method can show dynamical properties ranging from very "active" behavior (spectrally rich transient phenomena, turbulence and noise) to relatively "inactive" behavior (smooth, if not almost flat curves), and unpredictable transitions in between.

Motivations

In computer music research, sound synthesis is a major topic. In short, one can easily recognize two distinct approaches to the design and implementation of sound synthesis methods; here, we refer to them as the "standard" and the "non-standard" approach.

Most research work is usually undertaken in the first kind of approach. It entails the study of one or more acoustic models of some theoretical coherence and the implementation of algorithms capable of reproducing them on the computer. In general, this is the case with most well-known synthesis methods, whether based on linear strategies - additive and subtractive synthesis, plus the related analysis-resynthesis methods - or non-linear transformations of one or more signals - like waveshaping, FM, RM, AM (De Poli, 1981). This is the case for physical modelling, too - an area of major concern in current research (De Poli et al., 1991).

Less has been done in the non-standard approach, although many composers have developed and utilized a variety of methods (e.g. G.M.Koenig, I.Xenakis, and H.Brün). In this perspective, there is no pre-existing acoustic model, while the synthesis process is of the composer's own invention, aiming at the deepest integration between sound synthesis and the compositional process.

If appropriately understood, non-standard methods represent an approach of *microstructural time modelling of sound*, a perspective of sonic design also proper to instances of asynchronous granular synthesis/processing and other techniques based on microstructural representation of sound in the discrete-time domain. There is a bias towards the blurring of the neat distiction between *sound* and *structure* (Truax, 1990a), between the composer's *models of sonic materials* and his/her *models of large scale musical design* (Di Scipio, 1993).

A fresh perspective of non-standard synthesis

Any particular instance of microstructural time modelling can be seen as the operationalization of a definite music-theoretical problem: how the local organization of myriads of low-level elementary units may bring forth higher-level, global properties of sonic and musical structure, according to a personal strategy of the composer's own invention.

This point, however, has been largely underhestimated in previous work in non-standard synthesis. Indeed, the challenge lies precisely here: much like musical form can be understood as the epiphenomenon of a dynamical process captured in a model of musical design (i.e.: at some macro-temporal scale), in computer music the properties of the sonic structure - whose local Gestalt is usually called *timbre* - should themselves be understood as epiphenomena of micro-temporal *compositional* processes, unrelated to acoustic models but capable of modelling a phenomenon of morphological

emergence (Di Scipio, 1994). The problem raises about what may ever be the relevant features in such kind of models.

A possible answer comes from the mathematical modelling of complex systems. In chaos theory, it is shown that simple iterated difference equation systems can capture the details of rich, dynamical phenomena showing peculiar qualitative emergent properties of macrostructure (May, 1977; Collet & Eckmann, 1980). In the following we describe what can be called *synthesis by functional iteration* - drawing from terminology introduced by Mitchell Feigenbaum in an early article on nonlinear systems (1980). The effort consists in exploiting the notion of *iteration* as a source of self-organization in the dynamical behavior of sound.

Theoretical basis of sound synthesis by functional iterations

The formal frame of our mathematical model can be described as follows: we shall call

$A \subset \Re$	the set of "initial values" for our iterations;
$G \subset \Re^m$	the set of parameters of the particular map(s) considered;
$B \subset \Re$	the set of samples of the sound signal finally generated,

and consider the following cartesian product:

$$A \times G \subset \Re \times \Re^m.$$

Let F be a map defined as

$$\begin{array}{lll} F: & A \times G & \rightarrow \quad B \\ & (x, \{a_i\}) & \rightarrow \quad F(x; \{a_i\}) \qquad (\{a_i\} \equiv a_1, a_2, \ldots, a_m) \end{array}$$

which can be considered as a parameter-dependent function which maps from A to B with a_i as varying parameters. By fixing a set of m real parameters (a point of G) we have:

$$\begin{array}{lll} f: & A & \rightarrow \quad B \\ & x & \rightarrow \quad f(x) \end{array}$$

$$f(x) \equiv F(x; a_1 \ldots a_m).$$

By considering g_i a sequence of points from G, we get a sequence of maps f_i. Then, if

$$B \subset A$$

we can construct the iteration of the function f - thereby introducing the operation called *functional iteration* - by repeatedly applying f to itself n times:

$$f^n(x) \equiv f(f(\cdots f(x) \cdots)) \equiv (f \circ f \circ \cdots \circ f)(x)$$

Given a sequence of initial values $x_{0,i} \in A$ and a sequence of parameter sets $g_i \in G$ and correspoding functions f_i, a discrete time series can be computed where the i-th sample is the i-th n-iterate of $x_{0,i}$:

$$\begin{array}{ll} & \vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ \text{time} \downarrow & x_{n,i-1} = f^n_{i-1}(x_{0,i-1}) = f_{i-1}(f_{i-1}(\ldots (f_{i-1}(x_{0,i-1})) \ldots)) \\ & x_{n,i} = f^n_{i}(x_{0,i}) = f_{i}(f_{i}(\ldots (f_{i}(x_{0,i})) \ldots)) \\ & x_{n,i+1} = f^n_{i+1}(x_{0,i+1}) = f_{i+1}(f_{i+1}(\ldots (f_{i+1}(x_{0,i+1})) \ldots)) \\ & \vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \end{array}$$

In order to create time series with complex behavior, we must choose a nonlinear f. In the following section, we introduce and study the main properties of one such function. However, it is clear that the relevant point here is the *process* of functional iteration, more than the function we work with: "*Yet, precisely because the same operation is reapplied* [...] *self-consistent patterns might emerge where the consistency is determined by the key notion of iteration and* not *by the particular function performing the iterates*" (Feigenbaum, 1980).

The "sin map" model

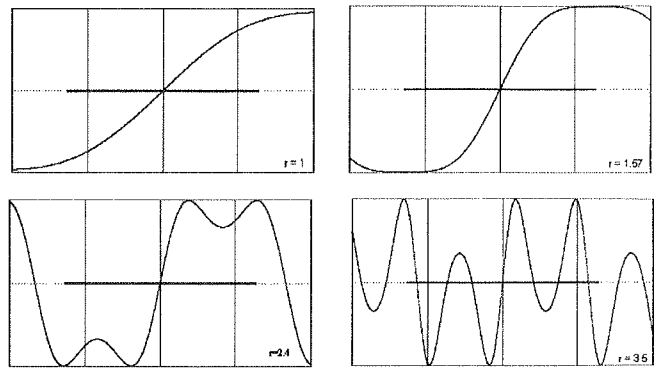
We have chosen to study the application of the "monoparametric" map ($m = 1$) defined by

$$\begin{array}{lll} F: [-\pi/2, \pi/2] \times [0,4] & \rightarrow (-1,1) \\ (x, r) & \rightarrow \sin(rx) \end{array}$$

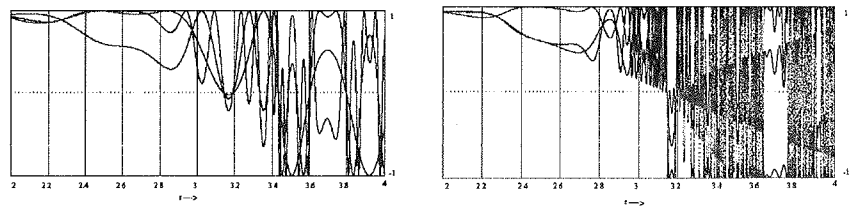
whose explicit iterated form is written as

$$x_{k,i} = \sin(r_i \, x_{k-1,i})$$

We set $A = [-\pi/2, \pi/2]$ because of the fact that, given the periodicity of the sin function, a larger space would only return trajectories already achievable starting from within $[-\pi/2, \pi/2]$, with the exception of x_0, of course. This is becasue the 1st iterate would anyway fall in the interval [-1,1], completely covered by $\sin(rx)$ for x_0 in $[-\pi/2, \pi/2]$ and $r \Rightarrow 1$. Moreover, we set $G = [0,4]$ because any larger value in r would only provide dynamical behaviors already achievable with r just below 4. Following are four examples showing the graphs of $\sin(rx)$ with increasing values of r.



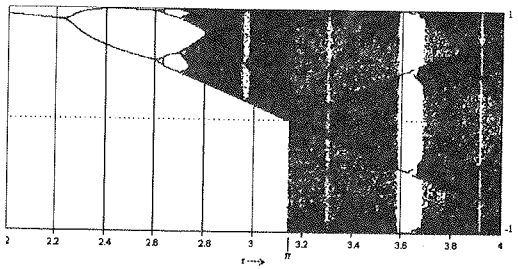
The behavior of the iterated process can be observed by calculating the n-iterate of some x_0 for linearly increasing r. In the leftmost figure, the 5th, 6th and 7th iterates are shown as r moves from 2 to 4; in the righmost, the 8th, 9th and 10th iterates are plotted. In all cases we set $x_0 = 0.1$.



Observe that the higher is the iterate and the more dynamical is the evolution of the signal traced by successive n-iterates. To our knowledge, the oscillating trajectories traced by such sequences of n-iterates have not been explicitly addressed by any scientifical study. This means that our applications is in need of further theoretical insight into mathematical details which may be unknown as yet. (This is the topic of a forthcoming study).

When we consider a larger number of iterations, the initial datum x_0 is *forgotten* as the process goes on (one cannot say, by any analytical means, where in the interval $[-\pi/2, \pi/2]$ the process started from). Moreover, transient phases to any attractor disappear; indeed, the calculation actually traces the

bifurcation diagram which characterizes the dynamical behavior of the model. Following is a plot of the 100th iterate of x_0= 0.1 for r going from 2 to 4.



A simple implementation

As typical to non-standard methods, the process of synthesis by functional iterations is only conceivable and explorable by implementing a specific algorithmic structure on the computer. The basic procedure is relatively simple and straightforward. However, notice that, since n is not defined before the synthesis starts (it's up to the user the decision of what iterate is to generate the sound signal) we have a loop of variable length within the computation of each single sample; therefore, for real-time applications we must be able to adjust the computation in order to mantain a stable sampling rate.

An other point worth of being mentioned is that for r in the interval $[0,\pi]$ the signal is only positive. We suggest that a conditional control be adopted in order to activate a normalization of the signal in the range [-1,1] as far as $r < \pi$ and to switch to no normalization when $r > \pi$. (In most cases this does not produce any noticeable discontinuity, given that $r = \pi$ is itself a somewhat chaotic area in the bifurcation diagram).

The following is a very simple C program implementing functional iteration synthesis with time-varying r and constant x_0. (The normalization problem is not taken into account, here).

```
#include <math.h>
#include <stdio.h>
int cot=32767; /* max amp */
int dur=22050; /* number of output samples */
double nlos(double x, double r, int n); /*prototype*/
void audio(void);
void main()
{
 int k, outsam, n, i_time=0;
 double icr, r_start, r_end, x0, r;
 printf("input start r, end r, start x, considered   iterate: \n");
 scanf("%lf %lf %lf %d ",&r_start,&r_end,&x0,&n);
 icr=(r_end-r_start)/dur;
 for(r=r_start;r < r_end; r += icr) {
        i_time++;
        outsam=nlos(x0, r, n)*cot;
        printf("sample %d time %d  current r %f \n", outsam, i_time, r);
        audio ();  /* send to audio output(file) */
 }/* end for */
}/* end main */

double nlos(double x, double r, int n) {
 int k;
 for(k=0; k<n; k++) {
        x=sin(r*x);
 } /* end for */
 return(x);

void audio(void)
{
        /* audio output code */
}
```

Small modifications in this short piece of code would make it work as an opcode of CSOUND. Actually, the algorithm can be itself rendered with existing CSOUND opcodes, but of course this would make the synthesis slower.

Exploring the phase space of the "sin map" model

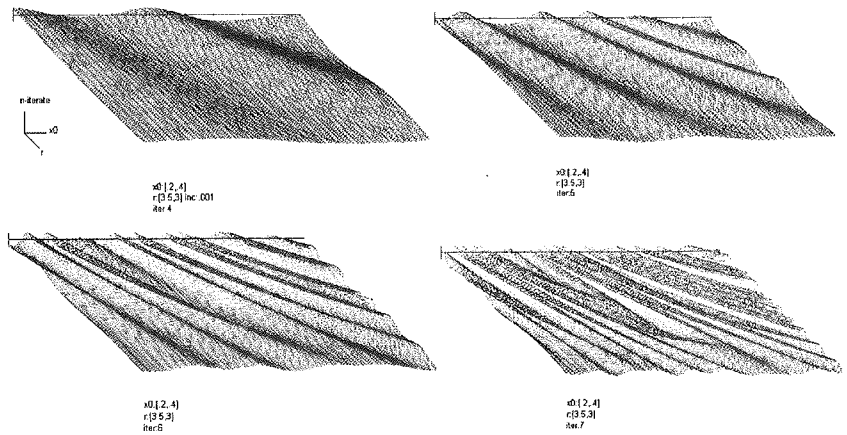
As observable in the bifurcation diagram, moving r in its space [0,4] determines the *kind* of global evolution of the generated signal, ranging from very smooth curves (e.g. r = 2) to a more acrive and complex behavior (e.g. r = 4), passing through many transitory phases corresponding to periodical cycles. A most important aspect of signals thus synthesized is an high dependency on the particular region of the phase space $[-\pi/2,\pi/2] \times [0,4]$ being visited. As suggested above, each orbit in the space corresponds to a signal of particular properties. In the synthesis process, we can

- change r while keeping x_0 constant; this is what the C program above does, but we could also
- change x_0 while keeping r constant; and
- change both r and x_0 according to some defined driving function.

Moreover, in all these cases we can choose an arbitrary n-iterate to generate the signal - although, in practice, useful values of n are bound up to few possibilities (see discussion below). Simply speaking, the three control strategies result respectively in

- rapid changes of global structural properties in the sound signal (highly dynamical spectra);
- innumerable signals of the very same properties (as captured by r);
- a complex mixture of these two.

Notice, finally, that the higher is the iterate and the more "active" the sound synthesized. Looking at the following 3D graphs, one may grasp some idea of how the sound signal changes as we move in the phase space. Here the n-iterates of $f(r,x)=\sin(rx)$ are plotted over a particular region in the phase space $[-\pi/2,\pi/2] \times [0,4]$, namely in the region $[3,3.5] \times [.2,.4]$. Each single graph relates to a different iterate, i.e. n = 4, n = 5, n = 6 and n = 7 (we used such small values not to complicate the graphical rendition too much).



Typically, the morphological properties of sounds generated with functional iteration synthesis are in a flux of continual variation through time, but include both local and global correlation; especially when very "active" ($r > 3$, $n > 8$), they are perceived as textural sonic phenomena, rich of sudden transitions, turbolence and, eventually, broad-band noise. If we use closed orbits in the phase space (by cycling or "modulating" either r or x_0) periodic signals can be obtained. Thus, the sounding results can range from pseudo-random (but locally and globally correlated) sound signals to sounds of harmonic spectra.

Before ending this section, we should recall that *sensitive dependency on the initial conditions* is the essential feature of chaotic systems. A measure that would enable us to characterize such feature in our model - hence the correlation degree within a single signal and among signals - can be computed in terms of the *Lyapunov coefficients* reflecting the exponentially growing distance of trajectories that start

with near but distinct x_0. In short, such a measure would provide a means to gain control over the predictabilty of the system behavior, thus making possible the user's meaningful choices of n. Indeed, in many cases higher iterates will certainly not result into sounds of more interesting structural properties - though they will certainly result in a longer computation time. Sometimes a large n may even destroy in the signal any observable relation with the signal's generating orbit in the phase space.

Some conclusions

Relavant theoretical details and the empirical use of synthesis by functional iteration must be both investigated in more depth. This is necessary, although a most peculiar aspect of such approach to sound synthesis lies exactly in the possibility of an explorative, nonlinear style of sonic design. Indeed, it is dubious that further analytical knowledge would make the model of better use in a linear and completely deterministic approach. Functional iteration synthesis provide *indeterministic* models of sonic material: the composer must learn his/her strategy by interacting with a source of structured information activated at the level of the microstructure of music, within and through the sound.

We think that the concept of *iteration*, being a concrete source of unpredictable but self-organized, consistent behavior, can capture large-scale design features which are particularly useful when one works at the microstructural level of sound. Methods of chaos theory have been already proposed and used as models of syntactical articulation of music (Pressing, 1988) and as powerful control structures of granular synthesis techniques (Di Scipio, 1990; Truax, 1990b). Our study proposes the extension of this approach to the level of the sample itself, by operationalizing a model of sonological emergence which projects the compositional process down to the micro-time scale in the musical structure. In so doing, it also implies a blurring of the conventional distinction between *sound* and *structure*, since with this kind of model the composer can generate entire fabrics of sound events and extended sonic textures that can hardly be perceived and conceived as separate partial components of the musical structure.

References

Collet, P. & Eckmann, J.P (1980) *Iterated Maps on the Interval as Dynamical Systems*. Boston: Birkhauser

De Poli, G., Piccialli, A. and Roads, C. eds (1991) *Representations of Musical Signals*. Cambridge Ma.: MIT Press

De Poli, G. (1981) Tecniche Numeriche di Sintesi della Musica. *Quaderni del LIMB*, v.1, 12-45

Di Scipio, A. (1990) Composition by Exploration of Nonlinear Dynamical Systems. *Proceedings ICMC90*, SanFrancisco: ICMA, 324-327

Di Scipio, A. (1994) Micro-time Sonic Design and the Formation of Timbre. *Contemporary Music Review* 10(2). In press.

Di Scipio, A. (1993) Models of Material and of Musical Design Become Inseparable. A Study in Composition-theory. *Proceedings International Conference on Cognitive Musicology*, Jyäskylä: Jyäskylän Yliopisto, 300-316

Feigenbaum, M (1980) Universal Behavior in Nonlinear Systems. *Los Alamos Science*, 1, 4-27

May, R. (1977) Simple Mathematical Models with Very Complicated Dynamics. *Nature*, 261, 459-67

Pressing, J. (1988) Nonlinear Maps as Generators of Musical Design. *Computer Music Journal*, 12(2), 35-46

Truax, B. (1990a) Composing with Real-time Granular Sound. *Perspectives of New Music*, 28(2), 120-134

Truax, B. (1990b) Chaotic Nonlinear Systems and Digital Synthesis. An Exploratory Study. *Proceedings ICMC90*, SanFrancisco: ICMA, 100-103

Modular programming of instruments in cmusic

CARLOS CERANA

Laboratorio de Investigación y Producción Musical (LIPM)

Junín 1930- 1113 Buenos Aires - República Argentina

tel/fax: 54-1-804-0877

email: rqcerana@arcriba.edu.ar

Abstract

This paper explains a system developed at the Laboratorio de Investigación y Producción Musical (LIPM), for modular programming of instruments in cmusic. The system allows users to build up their own patches using ready-made pieces of code, in a self-explanatory process. Every musician trained in hardware synthesizers programming can easily develop a complex instrument following simple rules, what turns it useful both for composition and for teaching software synthesis. The system allows coherent connections between envelope generators, devices for pitch and amplitude modulation, and for additive, subtractive, FM and waveshaping synthesis. As it is made exclusively of cmusic operators, users can add their own modules to follow their particular needs.

What it is

AMI (Aid of Modular Instruments) is an attempt to develop a user-friendly interface for instrument programming in cmusic.

The system allows the user to design complex instruments by adding simple modules. These modules are easily understood by every musician trained in hardware synthesizers programming, and their names are self-explanatory.

What it does

In its present state, the system allows coherent connections between envelope generators, devices for pitch and amplitude modulation, for additive, subtractive, FM and waveshaping synthesis, and for spatialization of sound.

User interface

Users build up their own patches by assembling modules, that are ready-made pieces of code. The system guarantees the coherence of connections between modules. By following certain simple rules it is easy to develop a complex instrument, because the constitutive parts are already debugged.

The resultant instrument is later invoked from the score by means of a macro.