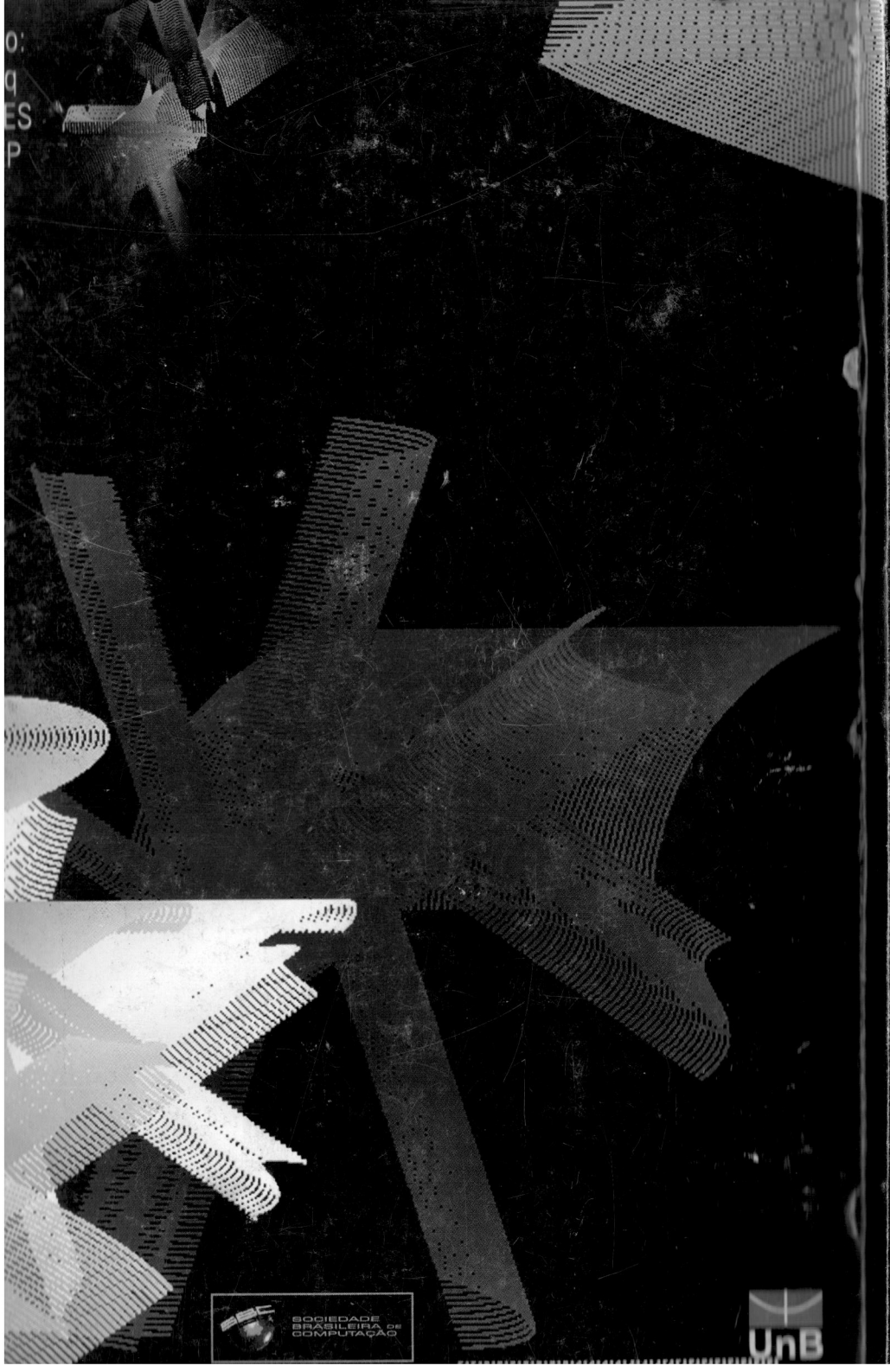


SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

UnB

IV SBCM

Simpósio Brasileiro de Computação e Música

3-7 AGO 1997

XVII Congresso da Sociedade Brasileira de Computação

IV SBCM

Simpósio Brasileiro de Computação e Música

Brasília, DF 3-7 Agosto 1997

IV Simpósio Brasileiro de Computação e Música

Brasília, 3-8 Agosto 1997

Anais

Editado por
Aluizio Arcela

Realização
Departamento de Ciência da Computação
Universidade de Brasília

Promoção
Sociedade Brasileira de Computação

Imagem de Capa: *A time-tree sound structure*, A. Arcela, 1994.

Apoio: CNPq
Unimix
Silicon Graphics
MSD

Site: http://www.cic.unb.br/sbc97/sbc97.html

SBCM'97 (4.: 1997 : Brasília, DF)

Anais do IV Simpósio Brasileiro de Computação e Música / editado por Aluizio Arcela. Brasília: CIC-Universidade de Brasília, 1997.

VIII, 217p.: il.

Inclui índice

1. COMPUTAÇÃO -- Congressos 2. MÚSICA -- Congressos 3. ARTE COMPUTACIONAL -- Congressos, Concertos

Catalogação na Fonte CIC-UnB, Biblioteca Imediata

Prefácio

Coube à Universidade de Brasília a tarefa de organizar o XVII Congresso da Sociedade Brasileira de Computação -- o encontro de maior representatividade da comunidade brasileira de Ciência da Computação. Fizemos o possível para manter a tradição e a qualidade já estabelecidas pela SBC, ao mesmo tempo em que buscamos construir uma versão condizente com os dias atuais.

O *SBC'97* apresenta uma programação diversificada, englobando mais de onze eventos, nos quais são abordados temas científicos, técnicos, políticos, educacionais e profissionais, todos eles oferecendo palestras proferidas por especialistas de renome internacional e por pesquisadores e professores da mais alta qualificação.

O *XXV SEMISH -- Seminário Integrado de Software e Hardware* -- conta com a apresentação de trinta e seis trabalhos selecionados dentre artigos submetidos em nível nacional e internacional. O *IV SBCM -- Simpósio Brasileiro de Computação e Música* --, que este ano tem por tema "Música e Tecnologia de Redes", promove os já habituais artigos, concertos e tutoriais, e recebe para uma seqüência de palestras a visita honrosa do pioneiro Max Mathews. Estão previstos onze cursos de curta duração no *XVI JAI -- Jornadas de Atualização em Informática*. O *X CTD -- Concurso de Teses e Dissertações* e o *XVI CTIC -- Concurso de Trabalhos de Iniciação Científica* -- apresentam trabalhos premiados de alunos de pós-graduação e graduação, respectivamente. O *V WEI -- Workshop de Educação em Informática* --, em conjunto com o *WIE -- Workshop de Informática nas Escolas* --, oferecem diversos painéis, grupos de trabalho, palestras convidadas e artigos que tratam de temas como a disseminação da informática no ensino fundamental além das estratégias de desenvolvimento curricular, avaliação nacional de cursos e formação profissional. Esses temas, aliados a questões de regulamentação profissional e mercado de trabalho, são também abordados no *XV ENECOMP -- Encontro Nacional de Estudantes de Computação*.

A "Sociedade de Informação" é o eixo central do tema para o *XXVII SECOMU -- Seminário de Computação na Universidade* -- contando com uma ampla participação de representantes das comunidades científica, técnica e política para reflexão e discussão de tendências e perspectivas futuras. Espera-se encontrar neste seminário formas alternativas e efetivas de integração entre a Universidade, o Governo e a Sociedade.

Contamos ainda com o *I WRI -- Workshop de Robótica Inteligente* --, que apresenta trabalhos em desenvolvimento da aplicação da informática no suporte a robôs inteligentes; do *I ENIA -- Encontro Nacional de Inteligência Artificial* --, o qual busca manter a comunidade de IA mais coesa em torno de seus propósitos de ensino/pesquisa de IA no Brasil, além do *I WCOMB -- Workshop de Computação e Biologia* --, o qual inclui conferência internacional e painéis relativos a pesquisas em andamento envolvendo Computação e Biologia.

A realização deste XVII Congresso da Sociedade Brasileira de Computação não seria possível sem a dedicação valiosa de professores, alunos e funcionários do Departamento de Ciência da Computação da UnB, bem como de colegas de outras instituições que se dedicaram à coordenação de eventos e revisão de artigos, além de outros, que direta ou indiretamente nos apoiaram nesse empreendimento, a quem agradecemos. Em especial, agradecemos o suporte financeiro do CNPq, FINEP, CAPES, FAPDF, ao apoio da UnB, CESPE, GDF, UNESCO, IFIP e Banco do Brasil.

Um abraço a todos,

Maria Elenita M. Nascimento
Coordenadora do XVII Congresso da SBC
Brasília, 11 de junho de 1997

Chairman's Note

Last year when we began to design the format for this fourth symposium, it was clear right from the start that a good theme to be explored should be something conecting music to the technology with networks. Certainly, current computer network technology is capable of promoting a significant evolution in music, not only in sound generation aspects but also in computing and performing whole compositions, or at least in calculating large structures as melodies and timbres in real time.

However, no matter what the theme to be focused is, a true evolution will only be possible if music itself can give in return tc computer science a significant and useful knowledge, so as to balance and conciliate the condition of a technology user with that of a technology developer. On the other hand, provided that besides logic, compilers, or databases, computing is art as well, the contribution of music to computer science has also to be an effort to correlate the musical intelligence to some kind of human computability. To write compositions or to write programs capable of writing compositions is a way of finding out how such musical intelligence works.

I am sure this is why the **Brazilian Computer Music Symposium** is going ahead along with the computer science community rather than alone. Four subsequent meetings under the partnership of the Brazilian Computer Society is a true demonstration of mutual confidence and commitment.

Aluizio Arcela
Brasilia, August 1997

Comitês de Seleção

Papers
Andrew Bentley (Finlândia)
Barry Truax (Canadá)
Carlos Cerana (Argentina)
Eduardo Reck Miranda (Brasil)
Fernando Lopez-Lezcano (Argentina)
Fúrio Damiani (Brasil)
Geber Ramalho (Brasil)
Jamary de Oliveira (Brasil)
Paul Berg (Holanda)

Coordenação
Jônatas Manzolli (UNICAMP)

Concertos
Eduardo Reck Miranda (UFSM)
Leo Kupper (SRSEA, Bruxelas)
Maurício Loureiro (UFMG)
Rodolfo Caesar (UFRJ)

Coordenação
Conrado Silva (UnB)

Tutoriais
Anselmo Guerra de Almeida (PUC/SP)
Kirk Corey (University of Iowa, Estados Unidos)
Pablo Di Liscia (Universidad Nacional de Quilmes, Argentina)
Xavier Serra Pompeu Fabra (University of Barcelona, Espanha)

Coordenação
Robert Willey (University of California, San Diego)

Coordenação Geral
Aluizio Arcela

Coordenação Geral

Maria Elenita Menezes Nascimento

Coordenação de Eventos

CTD - Maria Isabel Cavalcanti Cabral, UFPB
CTIC - Sérgio Augusto Sampaio de Moraes, EMBRAPA
ENECOMP - Guilherme Olivieri C. Borges, UnB
JAI - Cláudia Maria Bauzer Medeiros, UNICAMP
SBCM - Aluizio Arcela, UnB
SECOMU - Eratóstenes Edson Ramalho, CNPq
WEI - Maria de Fátima R. Brandão, UnB

Serviços

Secretaria - Rosa Amariles, UnB
Secretaria - Mônica Rosalino, UnB
Inscrições - Márcio Brandão, UnB
Infraestrutura - Renato Guadagnin, UnB
Administração de rede - João J.C. Gondim, UnB
Finanças - Wagner Teixeira da Silva, UnB
Transporte - Marcos Saldanha Ventura, UnB
Divulgação - Alba M.Magalhães, UnB
Social - Luciano Pina Góis, UnB

Diretoria

Presidente: Ricardo Augusto da Luz Reis, UFRGS
Vice-Presidente: Paulo Roberto Freire Cunha, UFPE
Vice-Presidente Adjunto: Edson C. B. Carvalho Filho, UFPE
Eventos e Comissões Especiais: Flávio Rech Wagner, UFRGS
Planejamento e Programas Especiais: Rosa Maria Viccari, UFRGS
Secretarias Regionais: Iara Terezinha Pereira Cláudio, PUC/RS
Publicações: Cláudia Maria Bauzer Medeiros, UNICAMP
Administração: Guilherme Horta Travassos, UFRJ
Educação: Roberto Silva Bigonha, UFMG
Finanças: Therezinha Souza da Costa, PUC/RIO
SBC Editora: Clarindo Isaias P.S.E.Pádua, UFMG

Conselho - Membros Titulares

Cláudio Leonardo Lucchesi, UNICAMP
Daltro José Nunes, UFRGS
Eratóstenes Edson de Araújo, CNPq
Nívio Ziviane, UFMG
Paulo César Masieiro, USP
Pedro Manoel da Silveira, UFRJ
Philippe Oliver Navaux, UFRGS
Sérgio Bampi, UFRGS
Sílvio Romero de Lemos Meira, UFPE

Conselho - Membros Suplentes

Daniel Schwabe, PUC/RJ
Júlio César S.do P. Leite, PUC/RJ
Miguel Jonathan, UFRJ
Sérgio de Mello Schneider, BRUFU

Secretarias Regionais

Minas Centro-Oeste : João Paulo Kitajima, UFMG
Norte-Nordeste: José dias dos Santos, UFPE
Rio de Janeiro: (vago)
São Paulo: Neucimar Jerônimo Leite, UNICAMP
Sul: Iara Pereira Cláudio, PUC/RS

Summary

Invited Talks

The Beginnings of Computer Music:1957 to the Early 1970's

Max Mathews
CCRMA, Stanford University, USA

Abstract. Computer performance of music was born in 1957 when an IBM 704 in New York City played a 17 second composition on the *Music I* program which I wrote. The timbres and notes were not inspiring, but the technical breakthrough is still reverberating. *Music I* led me to *Music II* through *V*. A host of others wrote Music 10, Music 360, Music 15, CSound, Cmix. Many exciting pieces are now performed digitally. The IBM 704 and its siblings were strictly studio machines--they were far too slow to synthesize music in real-time. Chowning's FM algorithms and the advent of fast, inexpensive, digital chips made real-time possible, and equally important, made it affordable.

The Radio-Baton and Conductor Program with Emphasis on Compositional Algorithms for Live Performance

Max Mathews
CCRMA, Stanford University, USA

Abstract. Starting with the Groove program in 1970, my interests have focused on live performance and what a computer can do to aid a performer. I made a controller, the radio-baton, plus a program, the conductor program, to provide new ways for interpreting and performing traditional scores. In addition to contemporary composers, these proved attractive to soloists as a way of playing orchestral accompaniments. Singers often prefer to play their own accompaniments. Recently I have added improvisational options which make it easy to write compositional algorithms. These can involve precomposed sequences, random functions, and live performance gestures. The algorithms are written in the "C" language. We have taught a course in this area to Stanford undergraduates for two years. To our happy surprise, the students liked learning and using "C". Primarily I believe it gives them a feeling of complete power to command the computer to do anything it is capable of doing.

NetSound, Extended Csound, and Structured Audio Processing

Barry Vercoe
MIT Media Lab, USA

Abstract. For several years *Csound* has been the language of choice for computer music synthesis. Three concurrent developments have now taken that in distinct new directions: onto the Internet, with client side rendering of server-defined orchestras and scores; into the PC sound-world, with DSP-based realtime processing of live input and of synchronized accompaniments; and into the broadcast community with the adoption of *Structured Audio* within the MPEG-4 audio standard. We present an overview of each of these, with live demonstrations of at least one, and describe how the global computer music community can become a developer force in all of them.

PAPERS

A Cellular Automata Based Music Algorithm: A Research Report

Kenneth B. McAlpine[1,2], Eduardo R. Miranda[2,3], Stuart G. Hoggar[1]

[1] Dept. of Mathematics, [2] Dept. of Music,
University of Glasgow, Glasgow, Scotland, G12 8QQ
and [3] Laboratorio de Musica Electroacustica, UFSM.
km@maths.gla.ac.uk, eduardo@music.gla.ac.uk, sgh@maths.gla.ac.uk

Abstract: *In this paper we report the current status of our research work involving the application of cellular automata to music composition. We introduce the fundamentals of our work and present CAMUS, an algorithmic composition system which uses cellular automata as the basis for its control system. We then discuss the limitations of CAMUS as it currently stands, and suggest techniques for improving the capabilities of the system.*

1. Introduction

Mathematics and music have long enjoyed a close working relationship: mathematicians have frequently taken an interest in the organisational principles used in music, while musicians often utilise mathematical formalisms and structures in their works. This relationship has thrived in recent years, particularly since the advent of the computer, which has allowed mathematicians and musicians alike to explore the creative aspects of various mathematical structures quickly and easily.

One class of structures of particular interest is the class of dynamical systems - those that change some feature with time. This includes fractal zooms, evolutionary computing techniques and cellular automata, each of which holds some potential as the basis of a composition algorithm. It is, however, cellular automata that we wish to discuss in the present paper. A cellular automaton is a mathematical model of a dynamical system over which space and time are discrete and all quantities take on discrete values. A cellular automaton is often viewed as an array of elements, referred to as cells, to which we apply some evolution rule which will dictate how the automaton develops in time.

As a compositional tool, cellular automata have a great deal to offer. Firstly, for the automata presented below, the processes involved are wholly deterministic. That is, for any given starting configuration, the resulting evolutionary states of the automaton will be identical each time the system is run. This is an exceptionally useful property, as it allows the composer who uses automata as a means of driving a composition algorithm to use the system in real-time performance, and be certain that the music will emerge exactly as intended. A deterministic system also enables the user to audition several different versions of a piece by, for example, comparing the results obtained from running the composition system with identical starting configurations, but different control data.

Secondly, music can essentially be thought of as a type of pattern propagation: we begin with initial themes and motifs (patterns) which, during the course of the composition, are subjected to certain transformations and developments according to the rules dictated by

the composer or the musical form. This is exactly analogous to the process which occurs within a cellular automaton: initial configurations of cells are transformed and developed according to a set of evolution rules.

The type and quality of the music which is produced in a cellular automata-driven composition system depends on a number of things, including the evolutionary rules employed, the initial configurations of the automata, and how the automata are mapped to musical output. These topics will be discussed further in a later section.

The paper consists of three sections. The first is a case study, and will present an outline of the techniques behind the CAMUS system developed by Eduardo Miranda ([Miranda 1993] [Miranda 1994]); the second discusses the limitations of the system as it stands, and presents extensions which address these shortcomings, and the final section presents methods for analysis of the techniques described in the first two sections.

2. CAMUS - a Case Study

The CAMUS (Cellular Automata MUSic) system uses the 'Game of Life' and 'Demon Cyclic Space' automata to generate compositions ([Wolfram 1984]).

The Game of Life automaton consists of an array of $(m \times n)$ cells, which can exist in two states, alive (i.e. **1**) or dead (i.e. **0**). The rule which determines the development of the automaton is: *A cell will be alive at timestep* $t + 1$ *if and only if it has precisely* **3** *live neighbours at timestep* t. The Demon Cyclic Space automaton is an array of $(m \times n)$ cells which can exist in k states. The evolution of the automaton is determined by: *A cell which is in state* j *at timestep* t *will dominate any neighbouring cells which are in state* $j - 1$, *so that they increase their state to* j *at timestep* $t + 1$.

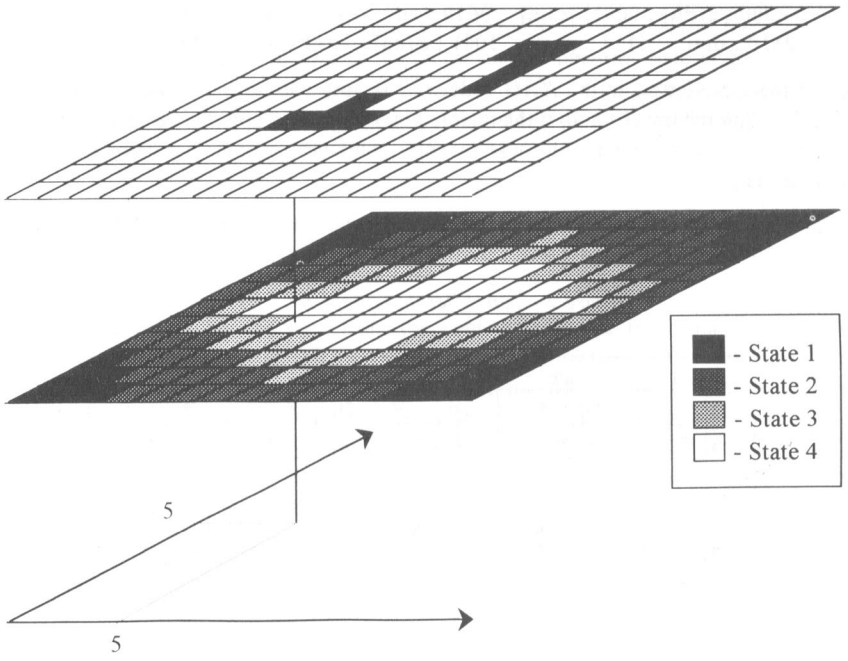
The CAMUS system also allows these rules to be tailored slightly, to allow for variations in the composition algorithm.

To begin the composition process, the Game of Life automaton is set up with a starting configuration, the Demon Cyclic Space automaton is intialised with random states, and both are set to run. At each time step, the co-ordinates of each live cell are analysed and used to determine a *triad*[1] which will be played at the corresponding time in the composition. The state of the corresponding cell of the Demon Cyclic Space automaton is used to determine the *orchestration*[2] of the piece. This configuration is demonstrated in figure 1. In this case, the cell in the Game of Life at **(5, 5)** is alive, and thus constitutes a sonic event. The corresponding cell in the Demon Cyclic Space is in state **4**, which means that the sonic event would be played by instrument number four (e.g., using MIDI channel 4). The co-ordinates **(5, 5)** describe the intervals in a triad: the fundamental pitch, the note five semitones above the fundamental, and the note ten semitones above the fundamental.

[1] In this paper, the term 'triad' is used to describe any set of three (not necessarily distinct) notes.

[2] Similarly, the term 'orchestration' is taken to mean which instrument 'plays' the cells.

Figure 1 - Configuration for each time step of the cellular automata music algorithm used in CAMUS



This configuration, of the points in the euclidean plane being used to determine musical intervals is known as the *von Neumann Music Space* ([Miranda 1993] [Miranda 1994]). Once the triad for each cell has been determined, the states of the neighbouring cells in the Game of Life are used to calculate the temporal position and duration of each note as follows:

Suppose we wish to determine the temporal positioning of the cell (i, j). We can construct a set of values from the states of the neighbouring cells - the value being 1 if the cell is alive and 0 if it is dead:

$$a = \text{cell}(i, j - 1) \qquad b = \text{cell}(i, j + 1)$$
$$c = \text{cell}(i + 1, j) \qquad d = \text{cell}(i - 1, j)$$
$$m = \text{cell}(i - 1, j - 1) \qquad n = \text{cell}(i + 1, j + 1)$$
$$o = \text{cell}(i + 1, j - 1) \qquad p = \text{cell}(i - 1, j + 1)$$

We then form the four four-bit words, ***abcd***, ***dcba***, ***mnop*** and ***ponm***. Next, we perform the bitwise inclusive OR operation, '|', to form two four-bit words, ***Tgg*** and ***Dur***:

$$Tgg = abcd \mid dcba$$
$$Dur = mnop \mid ponm$$

From ***Tgg***, we derive the note trigger information, and from ***Dur***, the note duration information. With each relevant four-bit word, we associate a code, known as an **AND**

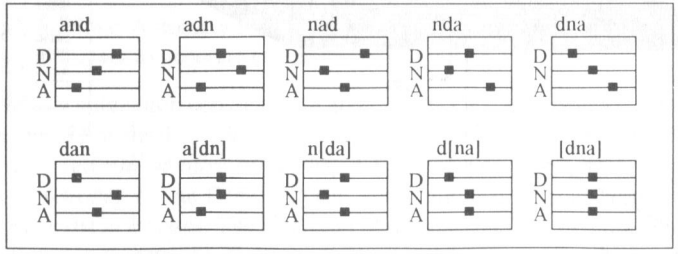
(cellulAr geNetic coDe). Note that the **AND** code is distinct from the bitwise AND operator, and is assigned as follows:

0000 : a[dn] 0001: [dna] 0010 : adn 0011 : dna 0101 : and
0110 : dan 0111 : nad 1001 : d[na] 1011 : nda 1111 : n[da]

Each of the three letters in the **AND** codeword is used to represent a note in the triad, with **a** denoting the lower pitch, **n** the middle pitch, and **d** the upper. The square brackets are used to indicate that the note events contained within that bracket occur simultaneously.

We associate each codeword with a particular temporal configuration, as shown in figure 2.

Figure 2 - Ten different temporal codes



These codes determine the *temporal shape* of each triad, the actual values for the trigger and duration parameters are calculated according to a user-defined equation.

The final stage in the CAMUS compositional process is the application of articulations. These are simply user-defined control parameters, which specify a speed, dynamic level, and 12-pitch sequence which is performed at each timestep. Up to 22 such articulations can be defined, giving a fairly wide scope for compositional development. CAMUS has been successfully used to compose a number of compositions, including a prize-winning piece for chamber orchestra, *Entre l'Absurde et le Mystère*, by Eduardo Miranda.

3. Developing the CAMUS system

As it stands, there are several inherent limitations of the CAMUS system. The first is that the use of a two-dimensional von Neumann Music Space to generate triads effectively limits the complexity of the system. There are two techniques which may be used to address this.

The first is an extension of the von Neumann Music Space to n-dimensions. This is a fairly logical progression of the system, and would allow the development of much more complex chordal and temporal structures within each composition. The work required to carry out this extension is feasible, involving only a corresponding extension of the arrays of cells in the Game of Life and Demon Cyclic Space automata, and an increase in the number of neighbours whose states must be checked at each timestep. However, two possible sources of difficulty arise.

The first is due to the increase in the number of calculations brought about by the higher dimensionality of the control automata. In general, an n-dimensional Game of Life requires 3^n calculations for each timestep, and an n-dimensional Demon Cyclic Space requires 3^n - 1. One can see that as the number of dimensions is increased, the number of calculations required simply to update the automata can become quite large. However, today's processors are more than capable of dealing with this number of calculations for arrays of cells which would be used in practical cases, and if a problem did arise, it would be perfectly feasible to compose off-line, saving the resulting file for editing at a later time.

The second problem, however, does not have such a neat solution. The problem is one of human-computer interface - an issue which has a very real effect on the usefulness of any particular system.

It is desirable to display the data from the control automata in graphical form. The reasons for this are twofold. Firstly, visual display of data is often an aid to the understanding of a system, by clicking on cells and making general parameter changes, and seeing the results implemented immediately onscreen, one can quickly develop an intuitive feel for how the system behaves. Secondly, seeing the evolution of the control automata helps to correlate what is happening in the system to the musical output which is generated.

Graphical displays are easily implemented for 2-dimensional automata, but a difficulty arises when one considers how to obtain and display the data from an automaton of higher dimensionality. The three-dimensional case has a partial solution in that it is possible to display an isometric view of a three-dimensional automaton on a two-dimensional computer screen, but this has its limitations in that some cells may be hidden from view by neighbouring cells, and that it is still not a viable method for the input of cell data using a two-dimensional input device, such as a mouse. If the dimensionality is increased still further, then we hit an intractable problem - the human brain can only perceive three spatial dimensions. Some sort of compromise must be decided upon, and to this end, we propose the following.

Rather than try to display a two dimensional (or isometric) representation of an automaton with dimensionality ≥ 3, the user is presented with a series of embeddings of the automaton (see figure 3). These embeddings will then offer the user feedback from the control system as to the current states of the automata.

Input from the user can be accomplished in two ways. The first, and most precise method is by the direct entry of coordinates for cells. However, this is not a particularly intuitive method, and does not in any way aid the user's visualisation of the current state of the automaton in question. The second method involves clicking cells in embeddings of the automaton, which is a much more visual method of input, but also much slower. The user first clicks on a cell in one embedding, fixing the two coordinates described by that embedding. Clicking on a cell in the next embedding fixes a further point and so on.

As an example of this input method, consider again the three-dimensional case, in which each cell is specified by a coordinate of the form (x, y, z) (see figure 4). The three embeddings we will work with are the x-y plane ($z = 0$); the x-z plane ($y = 0$), and the y-z plane ($x = 0$). Suppose the user clicks on the point (3,5) on the x-y plane. This would

then specify a line on both the x-z plane ($x = 3$) and the y-z plane ($y = 5$). The user would then click on a point on one of these lines (say (3,4) on the x-z plane) to fix the z-coordinate as $z = 4$, resulting in the unique point (3,5,4).

Figure 3 - Three embeddings of a 3-dimensional space.

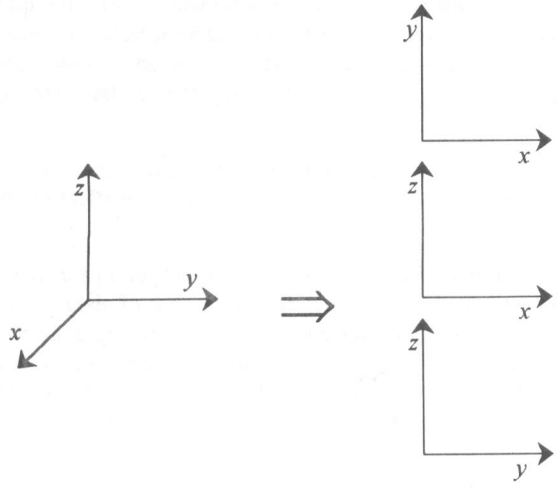


Figure 4(a) - User clicks on a point in the x-y plane, fixing lines in the other two embeddings.

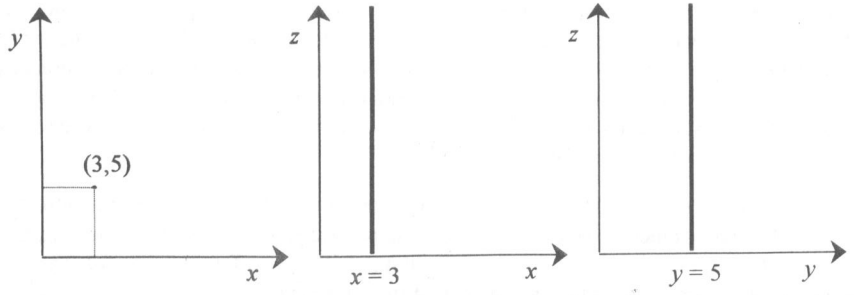
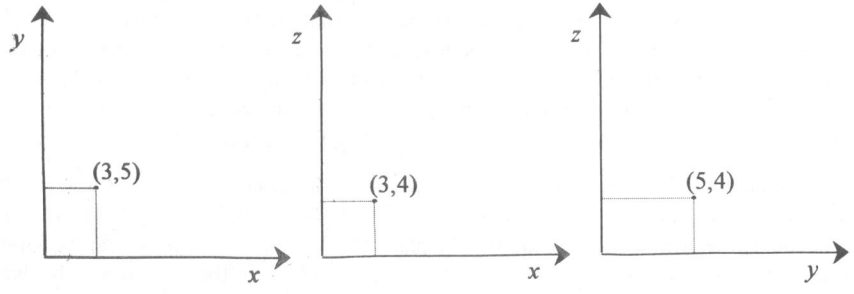


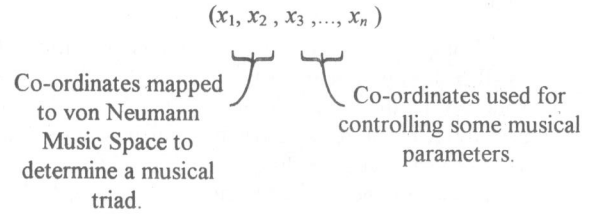
Figure 4(b) - User clicks on a point on the line in the x-z plane, fixing the point in the 3-dimensional space.



This system is very similar to that used in 3-D graphics packages, and will provide a useful starting point on which to build the interface. Practical experience with the working prototype will indicate desirable adaptations.

A further development of this idea is to use n-dimensional automata for control purposes, but to map the results to a two-dimensional von Neumann Music Space, with the 'extra' cell coordinates controlling some musical parameters (see figure 5). This would offer an alternative to using the articulations of the standard CAMUS system. Some study must be given to the efficiency of this system, specifically as to whether the calculations required for the extra dimensions for the control leads to an increase or decrease in operating speed.

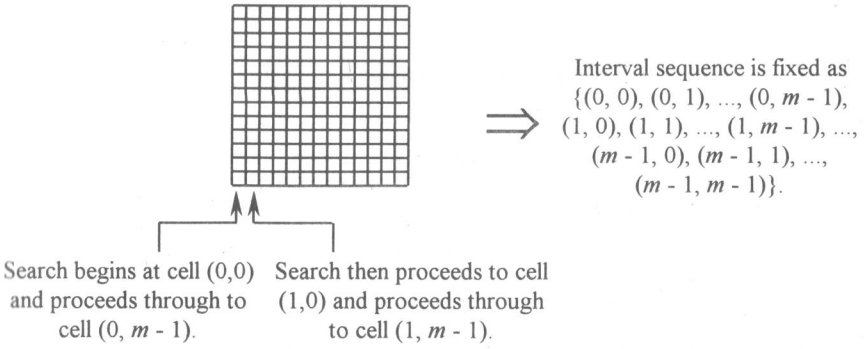
Figure 5 - Alternative mapping for an n-dimensional automaton.



The second technique which can be used to increase the complexity of the musical output works with standard 2-dimensional control automata, although it still requires considerably more processing power than the current system.

As it stands, CAMUS scans through the cells of the Game of Life sequentially, playing live cells as and when it comes across them. What this means in terms of complexity is that no more than one cell will be active at any one time, and that the triads generated are always played in the same order (see figure 6). In other words, even if the notes from two cells overlap in time, there is an imposed maximum of six concurrent notes.

Figure 6 - Interval sequence for an ($m \times m$) array.



To overcome this problem, we consider each timestep of the automata as contributing one sonic event. This has the distinct advantage over the present one-cell-per-sonic-event technique in that the number of simultaneous notes that the system can produce is limited only by the size of the control automata. For example, a 40 × 40 array would allow a

maximum of 40 × 40 × 3 = 4800 simultaneous notes. Also, since the cells are essentially calculated and played in parallel, the fixed interval sequence no longer proves troublesome - each cell, and thus each interval, has an equal likelihood of exposition at any given time. However, there are two further sources of difficulty which now present themselves.

The first is concerned with real time performance: We are now generating music at a higher level, and thus we need considerably more processing power than before. If we consider a timestep of x seconds, this means that for real-time output, we must calculate the next timestep of the automata, determine the live cells and their corresponding instrumentations, sort the cells in temporal order at triad level, sort the cells in temporal order at note level, update the score files and output the musical data all within the x-second window. Some study must be given to the feasibility of producing real-time output using this technique.

The second source of difficulty arises as a result of another limitation of the CAMUS system. At present, CAMUS does not, in any way, distinguish between triads. Thus each live cell contributes sonically to the composition, whether or not the user desires its presence. This is especially relevant if we have more than one cell active at once, particularly if we wish to avoid excessive dissonance. To address this, it is useful to define metric and span functions on the von Neumann Music Space, *NMS*.

By a *metric function* on the von Neumann Music Space, we mean a function, d: $NMS \times NMS \rightarrow \mathbb{Z}$ such that the following conditions hold:

(*i*) $d(\mathbf{x}, \mathbf{y}) \geq 0 \ \forall \ \mathbf{x}, \mathbf{y} \in NMS$, and $d(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$.

(*ii*) $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x}) \ \forall \ \mathbf{x}, \mathbf{y} \in NMS$.

(*iii*) $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y}) \ \forall \ \mathbf{x}, \mathbf{y}, \mathbf{z} \in NMS$.

A *span function* is a function s: $NMS \rightarrow \mathbb{Z}$ such that $s(\mathbf{x}) \geq 0 \ \forall \ \mathbf{x} \in NMS$.

We define the span function on *NMS* by:

$$s(\mathbf{x}) = x + y,$$

where $\mathbf{x} = (x, y) \in NMS$. This idea can be taken one step further, to give the *upper span*, us: $NMS \rightarrow \mathbb{Z}$, defined by

$$us(\mathbf{x}) = y,$$

and the *lower span*, ls: $NMS \rightarrow \mathbb{Z}$, defined by

$$ls(\mathbf{x}) = x,$$

again, where $\mathbf{x} = (x, y) \in NMS$.

It should be noted that the functions us and ls are also known as *projection maps*. We have introduced the alternative notation in the above discussion to remain consistent with the notion of the musical interval spanned by a triad.

We now define the metric function on *NMS* by

$$d(\mathbf{x}_1, \mathbf{x}_2) = |x_2 - x_1| + |y_2 - y_1|$$

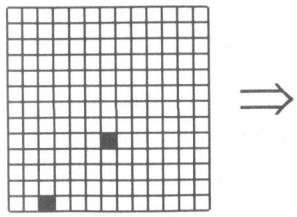
where $\mathbf{x}_1 = (x_1, y_1)$, $\mathbf{x}_2 = (x_2, y_2) \in NMS$.

In effect, we are defining the distance between two triads in the von Neumann Music Space to be the Manhattan distance ([Barnsley 1988]) between the intervals described by them.

It will frequently prove useful when using these functions to work modulo 12, so as to keep the system consistent with the idea of a simple musical interval (i.e. the number of semitones between the notes given as an integer between 0, the unison, and 11, the major seventh). This is particularly relevant to the upper and lower spans of a triad, where we wish to consider the consonance (or dissonance) of the individual intervals. For example, an interval of 16 semitones can be considered to be equivalent to an interval of 4 semitones, since both are essentially the 'major third'.

With these functions so defined, we have a notion of both triadic 'width' and 'distance'. Triad selection then becomes a simple matter of specifying precisely which combinations of spans and distances are important (see figure 7). Additionally, for a system in which many cells are active simultaneously, it would be sensible to include a parameter to control the maximum number of concurrent notes.

Figure 7 - Span and metric functions for the von Neumann Music Space.



The cell at position (0, 2) has a span of 0 + 2 = 2, and lower and upper spans of 0 and 2 respectively.

Similarly, the cell at position (6, 4) has a span of 10, and lower and upper spans of 6 and 4 respectively.

The distance between the cells is $|6-0|+|4-2| = 8$.

When CAMUS comes to play the cells, it searches a lookup table to evaluate the probability of such a configuration contributing a sonic event. The cells are then played or discarded according to this probability.

By implementing these methods, the complexity of the output can be increased dramatically, whilst still retaining a reasonably simple and manageable control system.

4. Evaluation Techniques

Evaluating compositional methods can present almost as many problems as defining and implementing the techniques themselves. In particular, we should be concerned with the following:

How efficiently does the technique in question utilise its resources? This point is very important if the user is to hear real-time feedback of compositions, particularly if the system is being used in a live performance, or with other composition systems. Inefficient use of resources could lead to glitches in the output, possibly throwing other players out of step.

How robust is the system? How well behaved is it? A good system will be both *well behaved*, that is it will produce output which is within the human hearing range and files

which store only the information needed to reproduce the music, and *robust*, that is impervious to erroneous input by the user and not prone to crashing or causing other programs to crash. No program can be guaranteed to be completely error-free. However, with extensive testing under many different operating environments and with many types of input, the system can at least be declared reasonably stable. Tidy programming techniques and a well-designed system should ensure good-behaviour.

How intuitive is the system? Is it possible for a new user to pick up the basics of the system and produce compositions straight away, or is there a steep learning curve which must first be surmounted? How effective is the user interface? These questions are all fundamental to the usefulness of any system. An interface which is too complex may alienate many users before they become familiar enough with it to obtain pleasing results. On the other hand, a system which is too simplistic may result in experienced users become frustrated at being 'pampered' through many stages of introduction and setup instead of getting on with the job in hand. There is a fine balance to be struck here, and care must be taken to get it right. Algorithmic composition systems are, after all, tools to aid creativity - few things inhibit the creative process more than struggling with a poorly designed interface.

How versatile is the system? Do all the compositions it helps create sound the same, or can the system produce music in a number of different styles? There is nothing wrong with designing a system to produce one particular style of music, nor is there anything wrong with a system which can produce many different styles of music. The key issue here is that the system can produce a variety of compositions. System versatility can often be improved by offering the user a variety of techniques which can be applied to the problem in hand.

The second approach to evaluation of a composition system is to consider the artistic merits of the musical output generated by the technique, and it is here that difficulties arise. The problem is that music is, by its very nature, subjective, whereas rigorous analysis demands a high degree of objectivity. Clearly, a balance must be struck, so that new techniques are not dismissed out of hand due to personal tastes, but nor are subtle artistic details omitted from consideration because of a very clinical approach to analysis.

5. Conclusion

Despite its limitations, the CAMUS system has already proven itself to be a viable mechanism for driving musical composition. It is hoped that by addressing its shortcomings, we can push the system further still, to produce compositions which are more complex, but still musically pleasing.

Acknowledgements

Many thanks to the Carnegie Trust for the Universities in Scotland, who generously provided the funding which has enabled this research. Thanks also to the University of Glasgow for providing both a research position and the facilities to undertake research in this field.

References

[Barnsley 1988] M. Barnsley, *Fractals Everywhere*. Academic Press, 1988.

[Miranda 1993] E. R. Miranda, *Cellular Automata Music: An Interdisciplinary Project*. Interface 22: 3 - 21, 1993.

[Miranda 1994] E. R. Miranda, *Music Composition Using Cellular Automata*. Languages of Design 2: 105 - 117, 1994.

[Wolfram, 1984] S. Wolfram, *Universality and Complexity in Cellular Automata*, Physics 10: 1 - 35, 1984.

Artificial Intelligence and Music: Computer and Brain

Author: Dr. Mladen Milicevic, The University of South Carolina, 6543 Haley Drive, Columbia, SC 29206, USA e-mail: MMladen@SC.edu

Abstract

This paper proposes that there are no objective aspects of music at all. To support this claim, the author uses the latest research from neuroscience, cognitive psychology, and neo-Darwinism. This paper also points out why the computer represents an inadequate means for Artificial Intelligence when used to emulate the workings of the brain. Furthermore, it addresses the cultural issues regarding the survival of computer music.

Introduction

Recently I came across people arguing about objective and subjective aspects of music. I thought that the argument was not worth pursuing because I tend to believe that the very concept of objectivity is subjective, and that there are no objective aspects of music at all. My assumption is that everything we know is a matter of interpretation, and there are as many interpretations as there are conscious organisms. To support this point of view I have to start with the study of the headquarters of subjectivity--the human mind.

For a long time the human mind and consciousness fell under the province of metaphysics and were generally considered as topics improper for scientific investigation. Even today, there is an enormous reluctance of scientists to deal with such subjective phenomena. Those who dare to wrestle with these slippery issues, often address the complexity of the mind via the Darwinian theory of evolution and cognitive psychology; or, as neuroscientists do, focus their interest on the structure and workings of brain and its functions. The best results will probably come from a marriage of these two strategies.

There are lots of theories, particularly in AI and cognitive psychology, that try to parallel computer and the brain, but when doing so we have to be extremely careful. The most important point to make is, that human brain developed over a period of more than 500 million years under the pressures of natural selection. This development certainly cannot be compared to a deliberate design of computer technology by the engineers. Old computer hardware and the software are often being completely

scrapped and redesigned from scratch, to reflect totally new fresh ideas and approaches. The only obstacle to this progress is market's requirement for backwards compatibility. I like to be able to run my old software on new hardware. On the other hand the human brain is entirely designed by evolution-forced backwards compatibility. We have a reptilian, paleomammalian, and neomammalian brains layered on top of each other, all running at the same time.

> If we remove the cerebral cortex--that part of our brain that has evolved over the past two million years or so--we essentially eliminate our humanity. Beneath the cortex is a brain that is not far different from that of a Bengal tiger, a French poodle, or an Arctic fox. We could, if we wish, remove even more to approximate the brain of a salamander or a rattlesnake. [Restak, 1984]

Our brain has an archaic multilevel design comprised of several brains each developed in different periods of evolution for different purposes and priorities. There is no way of going back, as in case of computers, to improve the design and rewire it for higher efficiency--we simply got stuck with our reptilian part of the brain.

Let me use an example from technological evolution to illustrate this point. The very computer keyboard that I'm using in writing this article, known as QWERTY keyboard for its letter layout, was designed at the time when fast typing on mechanical typewrites caused letters to jam. To remedy the problem and slow down the typing speed typewriter manufacturers came up with the most speed-inefficient layout of letters-- known as QWERTY. This is the keyboard we use today on the computer which does not have any letters to jam. But, can we go back and redesign the keyboard to achieve the most efficient and application appropriate letter layout. The answer is NO. Considering the number of people that today use QWERTY keyboards trying to improve their typing speed, it is near to impossible to make the change. Through technological evolution we got stuck with QWERTY, just as through biological evolution we got stuck with our limbic part of the brain, which guides our emotional responses and behavior. [Dennett 1995]

Another important point is that computer works very fast crunching numbers serially, one after another, operating at the speed of more than 10 million transactions per second. The human brain fires at much slower rate of a hundred spikes per second but makes up the speed loss with its massive parallelism. For example, more than

million axons go from each eye to the brain, all working simultaneously. Now we know for sure that the brain is built as a massive complexity of parallel-interacting neurons which lack a hierarchical order and central organizer, but do produce an emergent property--consciousness, which is mainly serial (subjective) in its nature. [Crick 1994] This is the reason why we have a sense of subjective unity regardless of the multilevel and multi-component make-up of the neurons involved. [Restak 1994]

Thinking and Logic

With the idea of developing machine intelligence additional problems arise when making the analogy between thinking and logic. Computer software is a written set of logical (this includes fuzzy logic too) instructions that conform to a specific syntactical structure which depends on the programming language in use. It is totally inadequate to think that syntax produces semantics. Semantic contents involve meanings, and syntax does not in itself deal with meanings.

> In the computer, the meaning is assigned by a human programmer. There is no ambiguity in the interpretation of physical states as symbols because the symbols are represented digitally according to rules in a syntax. The system is *designed* to jump quickly between defined states and to avoid transition regions between them; electronically, each component always goes to a "zero" or a "one." [Edelman, 1992]

Putting it simply: logic, as well as mathematics, operate in the abstract and, let me use that word here, "objective" realms; while semantics consider meanings which are subjective by default.

Here is an example that illustrates this point. Biological objects under evolutionary time have functional properties that are different from, for example, those of electrons. We cannot speak of the "abnormal" function of an electron as a physical object. On the other hand a proper function of a biological device depends on its evolutionary history. A human ear has a proper function to transduce a sound into a bio-electrical signal. There is also an evolutionary explanation for the construction of such a component (an ear) in a species, and this justifies the correspondence of this organ to "normal" ears in that species. Human ears work well or not; badly functioning ones, like Beethoven's, are abnormal and may need a hearing aid. In contrast, atomic particles do *whatever* they do, and whatever they do is part of their "working."

Within the evolutionary context, functions that propel the survival of the fittest are labeled as "normal" functions. Thus, each set of functions may be recognized as "normal" in relation how the system redundantly manages to perform that function. Since we know that the neural making of every human brain is different, as well as the environmental and historical context in which each brain develops, we may consider our brain/minds to be quite different systems. Different systems have different contexts that render all kinds of different "normals."

> Meaning derives from embodiment and function, understanding arises when concepts are meaningful in this sense, and truth is considered to arise when the understanding of a statement fits one's understanding of a situation closely enough for one's own purposes. (Notice the pragmatism!) Thus, there is no absolute truth or God's-eye view. [Edelman, 1992]

Meaning, Genetics, and Environment

To come up with a meaning it is necessary to have a conscious human being interacting with environment. This kind of being, unless raised in a deprivation chamber, is going to be loaded with myriads of subjective life experiences. Humans categorize "good" and "bad" experiences based on their interactions with the environment and the ability to detect, memorize, and compare exponentially growing repertoire of new good and bad things. Most of the human intentions are based on these subjective experiences and computers have neither intentions nor experiences.

The home of our conscious subjectivity is cerebral cortex--the last part of the brain to evolve. This is the part that produces consciousness and that we are the most proud of; because it can make complex decisions, create language, write music, play chess, and think of the black holes--which no other animal can do. Unlike other animals, humans are born with only 25% of its adult brain weight, and brain's final development has to be finished after birth. Though human genome specifies in great detail construction of a human body, vast number of the brain circuits are not preset by the genes and ought to be finished later in life. Since every human's being history and circumstances are different, every brain's wiring is going to be quite unique. Humans come to life having a brain endowed with automatic survival mechanisms which work adaptively in an array of socially permissible decision-making strategies. These nurturing conditions under which brain finishes its development are shaped by a certain

culture that conforms to "socially agreed" set of values. Educating the brain within such a system of social organization should guarantee the survival of that cultural group (species) within the given environment.

Hypothetically, if the mores of the cultural environment do not prescribe the practice of music, these survival mechanisms would certainly guide one to become non-musical. If Mozart was born in this kind of environment, in spite of his genome outline, he might prefer being a carpenter rather than a musician.

The Matter of the Mind

Humans categorize on culturally adopted set of values. Computers cannot do that, their values are specified in advance and imposed on the system by a human operator as a set of algorithms that define what to do under appropriate conditions and with appropriate error feedback. [Edelman 1992] It would be nice if we could talk about mind, as some psychologists do, only in terms of algorithms. Looking at different brains as if they were replicable machines or black boxes which can be understood solely in terms of inputs and outputs is an oversimplification. We ought to go inside the "black box" and try to figure out the mechanisms which operate our brains. The tissue organization and composition of the brain in form of groups of interacting neurons may be doing this job.

> These groups compete with each other in an effort to create effective representation, or maps, of the infinite variety of stimuli entering from the world. Groups that form successful maps grow still stronger, while other groups wither. [Horgan, 1996]

It is not by chance that brain of a person gifted with perfect pitch possesses a thickened area that processes sound--the auditory cortex, while a person with photographic memory has more neurons in its visual cortex. [Restak 1991] This is the feature that we call talent and it is genetically predisposed. The precise structure of a certain neural tissue of the brain exposed to a certain external environment will allow that tissue to develop, for example, a musical talent. For that reason it is wrong to believe that human brain of the youngster is an empty book that experience will write its story on. [Gazzaniga, 1992] If the neural circuits are not receptive to the environmental pressures the experiences "written" in the brain will be faint.

Brain's Age

So far I have dealt with two conditions required for design of another Mozart: a) genetic predisposition, and b) stimulating environment for that predisposition. There is yet another very important aspect in this picture to be considered and that is the age of the brain. When the brain is young it displays a neural plasticity which is capable of developing more and more neural connections required for a particular talent. This plasticity diminishes as the brain ages and may be one of the reasons why we get no musical talents suddenly erupted at the age of fifty. Even if the person had a thickened auditory cortex but the environment did not externally stimulate it at the most appropriate time--no Midori will be produced.

> Some circuits are remodeled over and over throughout the life span, according to the changes an organism undergoes. Other circuits remain mostly stable and form the backbone of the notions we have constructed about the world within, and about the world outside. The idea that all circuits are evanescent makes little sense. Wholesale modifiability would have created individuals incapable of recognizing one another and lacking a sense of their own biography. That would not be adaptive, and clearly does not happen. [Damasio, 1994]

Generally speaking, development of neural mappings for the "talents" that involve some kind of intensive body coordination, such a playing a violin or a game of basketball, tend to be locked-in very early in life. Other, more "disembodied" talents retain their neural circuit plasticity for much longer period of life.

A famous pianist said to me, about forgetting a familiar piece of music, "Muscle memory is the last to go," meaning by that term playing the piece automatically and without thinking about it. [Crick, 1994]

Emotion and Feeling

Let me now try to explain why the embodiment is so important for our understanding of the mind. The human body, as represented in the brain, provides a fundamental frame of reference for the neural processes that we experience as the mind. We use the physical state of our very organism as the ground reference for the mental constructions which we make about the environment we live in. [Damasio 1994] It is extremely important to understand that the human body and the brain constitute an inseparable interconnection that produces our consciousness. Everything we do, is

derived from the structural and functional ensemble of these two, rather than from brain alone.

> Can one fancy the state of rage and picture no ebullition of the chest, no flushing of the face, no dilatation of the nostrils, no clenching of the teeth, no impulse to vigorous action, but in their stead limp muscles, calm breathing, and placid face. [James, 1950]

In short, the background state of our body landscape provides a rather neutral "mood," against which we can judge any changes shaken by emotions. When the brain consciously appraises emotional changes in that equilibrium, we are having an emotional response--a feeling. Pretty much like being aware of the goose bumps while listening an effective piece of music. Conscious feeling of these goose bumps creates in your brain a memorized history of your body state under the given circumstances. Feeling depends on the juxtaposition of an image of the body, correlated to an image of something else; such as the auditory image of a piece of music. [Damasio 1994] Thus, later on in life, under the similar listening conditions, your brain may recreate this correlation of the images, and you may experience the feeling of goose bumps again.

Our individual identity of selfhood is firmly grounded on this illusion of living sameness, against which we can be aware of the infinite changes in our environment and consequently in our body. For this reason the sheer fact of a change in a room temperature, while listening to piece of music, may dramatically affect you emotional responses. Here it lies one of the central problems of the western medicine which subspecializes in treating diseased organs and system throughout the body but rarely addresses its most precious product--the mind. [Damasio 1994] Needless to say, that computers are not only disembodied; but cannot by themselves provide a meaningful relation between symbols and world entities.

Attention and Memory

We also have to be aware that not all operations of the brain correspond to consciousness, as I mentioned earlier in the case of the background body felling. Awareness starts when we consciously focus on a point of interest. Here, I would like to concentrate on consciousness which involves very short term of memory and it is closely associated with attention. Attention is, as William James said, "withdrawal from some things in order to deal effectively with others." In general attention and primary

consciousness involve some kind of a bottleneck. At first, the brain is processing the vast amounts of incoming information in parallel. Then the selective attention of hearing, for example, concentrates on one or a few objects at the time using the serial processing of the bottleneck--attending to one object after another. This is done by temporarily focusing on the objects of our interest while filtering out unattended information. [Crick 1994] Pretty much like listening to Bach's fugue while focusing our attention to the thematic workings of the dux and comes.

We may assume that primary consciousness of short-term memory deals with attention which is value-free perceptual categorization. This takes place *before* perceptual events contribute further to the alteration of neuronally structured and experience shaped value-dominated long-term memory. When short-term memory starts to contribute to the modification of a subjective long-term memory (this could be called a learning process) events are no longer in the remembered present, that is, they are no longer in primary consciousness. [Edelman 1992]

> Primary consciousness is required for the evolution of higher-order consciousness. But limited to a small memorial interval around a time chunk I call the present. It lacks an explicit notion or a concept of personal self, and it does not afford the ability to model the past or the future as part of correlated scene. [Edelman, 1992]

Musical Experiences

Now, just for the sake of experiment, lets assume for a moment that under the same external environmental circumstances, for me to perceive a middle C, certain groups of neurons and molecules in my head must behave in a very specific way. If the neural correlate of middle C in your head is exactly the same as in mine, which we know is probably not true, we may conclude that you hear middle C the same way I do. Stating this, we inevitably run into a problem with "exactly." The word "exactly" functions properly only in mathematics 1 + 1 = 2. On the abstract level "exactly" makes perfect sense but in practical world of everyday's living, nothing is exactly precise. Moving further on, to consider a major scale, things get indeed considerably obscure. When the brain receives sequences of musical tones, it does what it does with other patterns: it attempts to "interpret" them by using the information stored in its long-term memory about previous, similar experiences. This information may allow some aspects of a future signal to be anticipated--as it happens when we hear the first line of a

familiar song. This ability to extrapolate forwards on the basis of past experience is one form of that ability that we call "intelligence"; it can dramatically enhance an organism's chances of survival. Thus, if my neural correlate of middle C depends on my past experiences, which we assume is true, then my neural correlate for major scale is going to have totally different wiring than yours. My past musical experiences had grossly influenced my neural mapping which correlate in my brain with a concept of major scale.

> ... we constantly judge by comparison, and our judgment of any item depends upon what we are comparing it to at that moment. [Ornstein, 1986]

Different parts of brain handle specific mental processes but even this changes with our experiences. Comparing the brain activities while listening to music by trained musicians and people who have no specialized musical training shows very interesting results. Researchers discovered that musicians process music in their left hemisphere which is concerned with analyzing and comparing, for example the musical form and structure. Non-musicians' brains reacted to music with the right hemisphere in a more holistic way "I like this but not that"--which does not involve too much analysis. When non-musicians became more involved with appreciating music the PET (positron emission tomography) scan pattern of their brains showed a shift towards the left hemisphere because now they were equipped with new musical knowledge which enabled them to analyze. [Restak 1991]

Addressing the past experiences we have to also consider the way our brain handles the long-term memory. These memories are not stored in the brain photographically as intact individual events; and there are no stores of audio tapes, or albums of pictures. This is completely unlike computer-based memory which deals with exact reproductions. Human brain operates with a reconstructed version of the original--an interpretation. In order to compile a musical tune, the brain has to fire a certain set of neural mappings as a means to paraphrase "the music." These firing patterns trigger the momentary reconstruction of an *approximate* representation of the "Star Spangled Banner," for example. Your interpretation of music today, depends on: who you are, what are you doing at that moment, and your past experiences stored in the long-term memory. But, the next day, you are going to be different, what you will be

doing is going to be different, and your past experiences in long-term memory will change as well.

The "mass of soothing sound" your mother made while singing lullabies to you in childhood, is reduced to Twinkle, Twinkle Little Star, later on in life. Our memory of a certain musical piece is influenced not only by previous knowledge but also by events that happen between the time an event is perceived and the time it is recalled. [Ornstein 1991] Furthermore, we can only recall memories that are related to our present situation--where you are and what are you doing. If you are composing an orchestral piece your brain more likely focuses on recalling memories related to the instrumental ranges, rather than memories of how to change a flat tire on your car.

> So our memories, as exact, recorded, fixed images of the past, are an illusion. We believe we are stable, but this is one of the built-in illusions of the mental system. We believe we remember specific events, surely. Yet we don't. We make them up on fly. We change our minds all the time, from our estimate of the odds on a bet, to how we view our future. And we are unaware that the mind is doing this. [Ornstein, 1991]

Limitations of Artificial Intelligence

All this is pointing out that human mind deals exclusively with subjective phenomena and what we call objective is nothing but what most people agree to within a given socio-cultural context. Yes, we may listen to Beethoven's symphony in terms of air-pressure waves and on that level probably most people would have similar experiences. The question is, what is the use of doing that.

Herein lies the insurmountable problem of Artificial Intelligence--to emulate the human mind, science's final frontier. Holding that the brain is nothing more than a very complicated machine whose properties can be duplicated with computers seems very implausible. Even the system like the Internet is absolutely trivial compared to a brain. If there is a computer model of the brain/mind to be manufactured, it would have to mimic the human development of going to kindergarten, playing around the house, listening to music, and being traumatized by your neighbor's dog. Marvin Minsky, one of the founders of AI, made a statement which essentially undermines the whole project of AI in human terms.

> Minsky confessed that he would love to know what Yo-Yo Ma, the great Japanese cellist, felt like when playing a concerto, but Minsky doubted whether such an experience would be possible. To share Yo-Yo Ma's experience, Minsky explained, he would have to possess all Yo-Yo Ma's memories, he would have to become Yo-Yo Ma. But in becoming Yo-Yo Ma, Minsky suspected, he would have to cease to be Minsky. [Horgan, 1996]

This claim is the admittance that everything we know is nothing but a subjective interpretation shaped by person's biological make-up of the brain and neural correlates of the past experiences which define the self of that very person. Now, we are left with exclusively subjective properties of music to deal with, and in this dealing we may use the computers, because they are the most powerful heuristic tools we have in our attempt to understand the matter of the mind. [Edelman 1992]

Memetic Cultures

Intelligence and a culture are corequisites for each other. What makes the primary difference between our species and all others is; our reliance on cultural transmission of information, and hence on cultural evolution. Animals do interchange information within a biological rather than a cultural context. Bird mating calls certainly fall under the category of sonically transmitted information which is specific to a given species, but those species have limited intelligence and undoubtedly no bird-culture. This does not mean that animals have no minds, it simply means that *by human standards* those "primitive" minds produce no cultural history.

Dawkins' meme, has a peculiar but powerful role to play in our understanding of the human culture. This is the way he defines it:

> Examples of memes are tunes, ideas, catch-phrases, clothes fashions, ways of making pots or of building arches. Just as genes propagate themselves in the gene pool by leaping from body to body via sperm or eggs, so memes propagate themselves in the meme pool by leaping from brain to brain via process which, in the broadest sense, can be called imitation. If a scientist hears, or reads about, a good idea, he passes it on to his colleagues and students. He mentions it in his articles and his lectures. If the idea catches on, it can be said to propagate itself, spreading from brain to brain. [Dawkins 1976]

The important rule for memes, as for genes, is that they must constantly replicate. This replication is a mindless process not necessarily for the good of anything; replicators flourish that are good at replicating--for whatever reason. Meme X spread among the people because X is a good repilcator. [Dennett 1995] Let's take a moment and look at one particular meme case--the success of a four-note meme at the

beginning of the Beethoven's Fifth Symphony. It certainly has little to do with a pitch-set of its "internal" design, i.e. the way a musical piece is compositionally structured, but much more to do with the design this meme presents to the world, its phenotype, the way it affects the minds and other memes in a particular socio-cultural environment.

Conclusion: The Future of Computer Music

If we take all this into account, it is clear that as long as computer music composition is iconoclasticly focused on its internal structure such as: fractals, fibonacci numbers, solar systems, palindromes, permutations, interpolations, pitch-sets, algorithms, etc. etc.; and less focused on how the cultural environment reacts to it; computer music memes are not going to replicate and spread themselves further beyond the narrow technically oriented facilitators. The question: Do we compose only for our idiosyncratic selves or for the audiences of our culture?

References:

- [Crick, 1994] Francis Crick. *The Astonishing Hypothesis.* Charles Schribner's Sons, New York, 1994.
- [Damasio 1994] Antonio R. Damasio. *Decartes' Error.* A Grosset/Putnam Book, New York, 1994.
- [Dawkins 1976] Richard. *The Selfish Gene.* Oxford University Press, Oxford, 1976.
- [Dennett, 1995] Daniel C. Dennett. *Darwin's Dangerous Idea.* Simon & Schuster, New York, 1995.
- [Dennett, 1991] Daniel C. Dennett. *Consciousness Explained.* Back Bay Books: Little, Brown and Company, Boston, 1991.
- [Edelman, 1992] Gerald M. Edelman. *Bright Air, Brilliant Fire.* Basic Books, New York, 1992.
- [Gazzaniga, 1992] Michael S. Gazzaniga. *Nature's Mind.* Basic Books, New York, 1992.
- [Horgan, 1996] John Horgan. *The End of Science.* Helix Books, Reading, Massachusetts, 1996.
- [James, 1950] William James. *The Principle of Psychology.* vol 2, Dover, New York, 1950.
- [Ornstein, 1991] Robert Ornstein. *The Evolution of Consciousness.* Prentice Hall Press, New York, 1991.
- [Ornstein, 1986] Robert Ornstein. *Multimind.* Houghton Mifflin Company, Boston, 1986.
- [Restak, 1994] Richard M. Restak. *The Modular Brain.* Simon & Schuster, New York, 1994.
- [Restak, 1991] Richard M. Restak. *The Brain Has a Main of its Own.* Harmony Books, New York, 1991.
- [Restak, 1984] Richard M. Restak. *The Brain.* Bantam Books, New York, 1984.

Using Computer Programs as Generators of Compositions

Aditya P. Mathur*

Department of Computer Science

Purdue University

W. Lafayette, IN 47907, USA

apm@cs.purdue.edu http://www.cs.purdue.edu/people/apm

June 16, 1997

Abstract

An approach to music composition that utilizes a computer program as a generator of music is reported. This approach allows a composer to select a computer program P and develop a mapping of static and dynamic events within the program to musical elements of the desired composition; the musical elements being, for example, theme, pitch, and instrumentation. This mapping is then specified formally using a language called the Listen Specification Language (LSL). A compiler for LSL then compiles program P and the specification into object code. When interpreted the object code generates MIDI data sent to a synthesizer which in turn generates audio. The music so generated depends on the mapping and the input to the program. The run-time system associated with LSL allows the composer to alter various characteristics of the music interactively, i.e. during program interpretation. This approach has so far been used to generate short works based on simple sort programs and compilers. The effect is often pleasing and depends to a great degree on the mastery of the composer in determining a suitable mapping.

1 Introduction

Music composition is a complex task making use of knowledge gathered by the composer from a variety of experiences. Traditionally, it is the human composer who

*This work was supported in part by NSF award CCR-9102331. All reports and programs referenced in this paper may be obtained by writing to the author at the address given or by sending a request via email.

serves as the generator of the composition. The composition itself has a variety of elements. At an abstract level there may be an overall theme. At less abstract levels there are subthemes, instrumentation, melodies, rhythm, and several others. Through a complex derivation process a composer creates and combines these elements into a single composition. The basis of the derivation process and the "object" that drives the creation and combining of these elements is essentially a complex mental process.

Mostly out of curiosity, it was decided to investigate the use of computer programs as "objects" that could be used to derive compositions. In this sense of the word "object", a computer program is treated as a generator of composition. A computer program, hereafter referred to as simply "program", is a statement of logic for solving a well specified problem. This statement itself is always made in a language known as a programming language. The program is a static object when written; it becomes a dynamic object when the logic it uses is interpreted. The interpretor's behavior is controlled by the logic of the program. The interpretor is known as a "computer". The behavior of the computer, while interpreting a program P, is commonly referred to as the behavior of the program P itself. This research is about experimenting with ways to map static elements of P and its behavioral patterns to the elements of music. Observations from such experiments might lead to (a) improved understanding of the composition process, (b) new forms of music, (c) new ways of generating music, and (d) other forms of human knowledge.

The remainder of this paper reports the status of this research. Specifically, Section 2 develops an analogy between behavioral elements of programs and those of music. Section 3 provides an overview of a language, named LSL, developed to specify a mapping between program behavior and music. How one uses LSL and a system that incorporates LSL is described in Section 4. A summary of this work and our conclusions so far appear in Section 6.

2 Programs and music

According to a functional view of computer programs, a program is considered to be a collection of functions that operate on data. The functions "call" each other according to a pattern governed by the logic that interconnects them and the data input to the program during its interpretation. A function is considered "active" when it is called by another function and remains active until it terminates. A function that calls another function gets "suspended".

An object oriented view of computer programs treats programs as objects that

belong to classes. Objects can invoke methods belonging to themselves or to other objects. A class is a "static" entity and is used as a template to generate a "dynamic" entity. For our purposes, a function and a method are different names for the same thing which does computation on data. A class aids in encapsulating data and functions that operate on this data. This allows objects to communicate with each other only via methods. Any data transfer across objects is via method parameters.

Programs may be sequential or parallel. In sequential programs only one function may be active at any instant in time; multiple functions may however be suspended. Parallel programs allow multiple active functions at any time instant.

There are several, possibly infinite, ways of mapping static and dynamic elements of a program to elements of music. Table 1 shows one mapping for a subset of possible program elements; an explanation of some entries of this table follows. One way to imagine the mapping between programs and music is to consider each function in one to one correspondence with a symphonic "track" or a distinct "instrument". The execution of a function, or a method, may correspond to the elaboration of a theme. Thus, when a function is called, a theme is initiated and elaborated while the function is active. The theme gets suspended when the corresponding function does. Access to data items is mapped to accented tones or arpeggiations. Evaluations of expressions and conditions is mapped to melodic motives. Assignment of values to variable data items is mapped to the start or the end of a phrase.

Parts of the mapping in Table 1 are derived from what one might understand to be a "meaning" of a function; the other parts are mostly arbitrary. Though a formal basis for reasoning and specifying such a mapping is presented in the next section, it is this mapping which brings in the element of creativity in using programs as generators of music.

3 Specification of program-music mapping

A formal basis for the program-music mapping, borrowed from previous work [2], is illustrated in Figure 1. To design a language for specifying a variety of mappings, a quantification of two domains is established. Let E be the domain of occurrences during the execution of any program. Function call, function return, expression evaluation, are examples of such occurrences. Let S be the domain of all possible sound patterns that may be associated with each element of E. Note sequences "C4E4G4", "D3F#4A5" are sample sound patterns that may be articulated in a variety of ways. A mapping from E to S is an association of sound patterns in S to occurrences in

Table 1: A sample mapping of program elements to the elements of music.

Element type	Program element	May be mapped to
Dynamic		
	Function call	Theme initiation or resumption
	Function execution	Theme elaboration
	Suspended function	Theme in background or suspended
	Data access	Accented note(s)
	Array scan	Arpeggiation
	Assignment	Start or end of a phrase
	Expression evaluation	Melodic motive
	Condition evaluation	Melodic motive
	Function call pattern	Rhythm
Static		
	Function name	Pitch or instrument
	Data name	Pitch or instrument

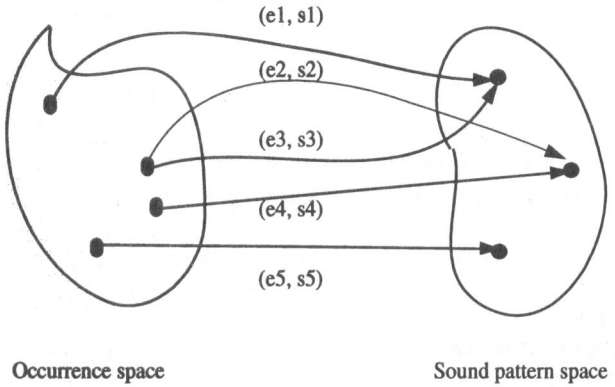


Figure 1: A domain based view of program-music mapping.

E. Such a mapping is specified as a set of pairs (e, s) where $e \in E$ and $s \in S$. The program-music mapping for any program P is a set $\{(e_1, s_1), (e_2, s_2), \ldots, (e_n, s_n)\}$, where each $(e_i, s_i), 1 \leq i \leq n$, is an association of an occurrence to a sound pattern.

Note that the above specification is static. It merely states "what" sounds are to be emitted when P behaves in a certain way; the sounds emitted when P is interpreted depends on the sequence of occurrences during one interpretation of P. This sequence may differ from one interpretation to another due to variations in input data.

An illustrative program-music mapping is shown in Figure 2. This mapping, reported earlier in [1], specifies how the behavior of a program to sort numbers is to be mapped to sound. The program itself is found in [1]. The music related commands begin from the fourth command in the figure. The `syncto` command specifies that the music is to be played with 120 quarter notes per minute. Each `notify` command specifies the sound to be generated when an event occurs. For example, the first `notify` command specifies that the start of program execution is to be mapped to the `Begin_snd` which is a predefined sound for the Roland SC-55 synthesizer. The `dtrack` command specifies that changes to the value of the data item named `temp` are to be mapped to the `Flute2_snd` which is another predefined sound for the same synthesizer.

Arbitrary patterns of notes are used to specify a sound such as `Flute2_snd` in the above example. A pattern contains a note sequence, instrument on which it is to be played, and how the notes within the pattern are to be played. One may specify arbitrary chordal patterns to be played once when an event occurs or continuously until a **pause** command is issued during interpretation. Multiple continuously played

```
begin auralspec
specmodule paper-example
begin paper-example

        syncto mmabs q = 120;

        notify rule = prog_begin using Begin_snd;
        notify rule = prog_end using End_snd;

        notify rule = for_body_begin
                using Sticks_snd
                in func = ''bubble_sort'' and func = ''selection_sort'';

        dtrack temp
when rule = function_entry:''exchange''
                until rule = function_return:''exchange''
                using Flute2\_snd;

end paper-example;
end auralspec.
```

Figure 2: Sample program-music mapping for a "selection sort" program.

patterns form a theme and may be interrupted or resumed during program interpretation.

4 Using Listen

To generate music from a program, a system named **Listen** has been developed. Using **Listen** one may follow the steps given below to map a program P to music.[3]

1. Develop a mapping from P to the elements of music. Create a formal specification S of this mapping in LSL.
2. Use **Listen** to compile[1] P and S into an interpretable program P'.
3. Interpret P' on a computer.[2] During program interpretation the events and activities in P are mapped to sound data played through a MIDI device such as the Roland SC-55 synthesizer.

During the interpretation of P' the listener is able to control the mapping specified as S. Thus, for example, a theme may be turned "off" or "on" at will. This allows a listener flexibility to experiment with the composition generated by the program. **Listen** allows the tempo to be set statically through an LSL command. The tempo can be altered during program interpretation. Examples of music generated by using the **Listen** system are found in [1, 3].

5 Programs as compositional sources

In this section we argue that the proposed approach to music composition is a novel contribution to the process of music composition. To do so, we first define a *source* of music composition as any object that is used by a composer or motivates the composer to generate all or any part of the composition. Emotions, events in life, elements of nature around us, factory manufactured elements around us, mathematical functions, and many others are all possible sources. Let S denote the set of these sources. It is not clear how to enumerate S or whether S is indeed enumerable. It is also not clear whether or not S is finite. Considering the variety of objects in the universe that can serve as sources, we conjecture that S is infinite. Let $SS = 2^S$ be the power set of

[1] Compilation of a program is its transformation from a high level language in which it is written, such as C, to a low level language which a computer understands.

[2] The current **Listen** system operates on any Sun Sparc computer under the Solaris operating system. It is available upon request from the author.1

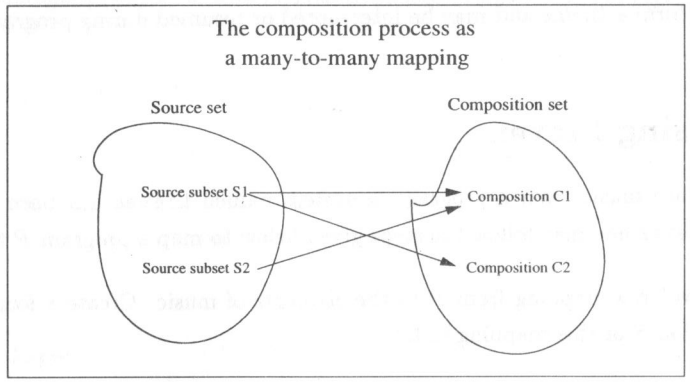


Figure 3: A functional view of the compositional process.

S; it is referred to as the *source set*. SS is the set of all subsets of elements of S. An element of SS is a *source subset*. Notice that the empty set is a part of SS which accounts for the possibility that a composition is generated out of "nothing".

During the composition process, one knowingly or unknowingly selects an element of SS and maps it to a composition. This is a view of the composition process and certainly one that allows us to place, in its proper perspective, the proposed composition process. Let C denote the *composition set* which is the set of all possible musical compositions. Then, according to our view, the compositional process P is considered the following mapping: $P : SS \rightarrow C$. This mapping, illustrated in Figure 3, is many-to-many as one element of SS can be mapped to two or more elements of C and that one element of C can be derived from two or more elements of SS.

The proposed composition process adds a new source of composition, namely computer programs, to S and hence to SS. As the set of such programs is infinite, the process provides a significant enhancement of the source subset. This addition allows a composer, be it the computer programmer or someone else or even another computer program, to select one or more programs and, possibly, other objects from SS and map them to compositions. It is this possibility, generated by the proposed process, that we consider the key novelty of the approach.

From a programmer and program user's point of view, mapping a program to music is akin to mapping it to visual elements such as animations. Just as visual animations aid one in the understanding of a program's behavior, so do aural mappings of a program. Of course, whether or not an aural mapping can be considered as music is a subject of debate and not considered here.

6 Summary and conclusions

An approach to the generation of music from computer programs is described. A language named LSL has been developed to specify the mapping between the elements of a program and music. A system, named `Listen`, that compiles arbitrary C programs and an LSL specification, has been developed. Experiments carried out so far have been mostly trivial in nature. Relatively small computer programs have been mapped to music. Music so generated is sometimes exciting enough for the listener to begin dancing; on other occassions it is meaningless and appears to be random noise. A key to the generation of "pleasing" music when using `Listen` is the LSL specification; of course what is pleasing to one may not be so for others. By varying the specifications and the data input to the program a wealth of compositions may be obtained.

`Listen` 3.0 system was originally developed for application in another area of Computer Science, namely "program debugging." The system has several limitations when considered for program-music mapping. An enhanced version, `Listen` 4.0, is currently under development. Once available the system will allow experimentation with large programs that are perhaps capable of generating larger musical works such as symphonies.

References

[1] D. Boardman, "LISTEN: An Environment for Program Auralization," Master's Thesis, Department of Computer Science, Purdue University, W. Lafayette, IN 47907, August 1994.

[2] D. Boardman and A. P. Mathur, "Preliminary Report on Design Rationale, Syntax, and Semantics of LSL: A Specification Language for Program Auralization," Technical Report, SERC-TR-143-P, August 1993, available from Software Engineering Research Center, Department of Computer Science, Purdue University, W' Lafayette, IN 47907.

[3] D. Boardman, G. Greene, V. Khandelwal, and A. P. Mathur, " LISTEN: A Tool to Investigate the Use of Sound for the Analysis of Program Behavior," *Proceedings of the Nineteenth Annual International Computer Software & Applications Conference (COMPSAC'95)*, IEEE Computer Society Press, August 9-11, 1995, Dallas, Texas, pp 184-193.

Acknowledgements

Thanks to my teachers Verna Abe and Helen Brown who introduced me to the formal elements of music and encouraged my enthusiasm for pursuing it in new directions. Thanks also to Dave Boardman, Vivek Khandelwal, Geoff Greene, Nate Nystrom, and Howard Chen for the endless hours they put into the development of `Listen` 3.0; they have created a system that allows unlimited and exciting experimentation in music composition using computer programs as generators.

An Intuitive Interface for using Spectral Modeling as a tool for Composition among Colombian Composers

Juan Reyes
jreyes@uniandes.edu.co
Mauricio Rincon
mrincon@indy.uniandes.edu.co
GIIM
Grupo de Investigaciones en Informática Musical
MOX: Centro de Computación Avanzada
Universidad de Los Andes
Santafe de Bogota - Colombia

Abstract

This is a description of one of the various computer music projects currently at La Universidad de Los Andes. The paper outlines the reasons for getting into this project and the benefits that will come out . It focuses on a description for a graphic user interface developed specially for Spectral Modeling Synthesis and aimed to be used intuitively by musicians with little or no knowledge on UNIX commands as well as minimal knowledge on Spectra and The Fourier Transform.

Definitions

Spectral Modeling Synthesis (SMS) was developed by Xavier Serra in 1989 at Stanford University. It follows the idea of sound analysis procedures to extract synthesis parameters out of real sounds. The analysis procedure detects partials by means of the Fast Fourier Transform and other complex mathematical procedures such as Linear Predictive Code. These partials are then subtracted from the original sound, and the remaining sound component or residual is represented as a time-varying filtered non linear white noise component. Synthesis can be achieved by a combination of additive synthesis for the sinusoidal part and subtractive synthesis for the noise part. This strategy can be used for either generating sounds (synthesis) or transforming (sound processing). The software package is written in C, and runs on most UNIX environments.

Motivation

As part of the various resources offered by the Centro de Computación Avanzada at Los Andes, and as a suggested by Xavier Serra the authors considered a good option to implement SMS as one of the tools for research and development at the center. The reasons being as a continuing research on analysis techniques, which were started by Carlota Mojica and Camilo Rueda in 1990, and also because of the attractive technology of deterministic and stochastic approaches to analysis and synthesis of sound .It was acknowledged that these were continuing steps toward various subjects of interest, including sound compression, partials extracting and speech recognition which previously found their way at various departments at the university.

It should not be denied that there was also great excitement on implementing computer music and signal processing techniques to parallel computing systems like a CRAY J-916 supercomputer at the center. This is to be able to carry on research on parallel processing of signals. Typically this type of processing has been used on graphics simulation, computer modeling, and computer aided design. Pseudo-spectral algorithms have been used for seismic modeling, to evaluate hydrocarbon and petroleum prospects. We believe that the same sort of research, can be applied to sound processing techniques, due to the time domain nature of sound, and the enormous processing power which is needed to model sound in real time

(Mathews, 1969). Fourier analysis is a complex mathematical procedure which can be accomplished to a greater detail for reverberation and modeling of acoustic spaces. Infinite response filters and convolution can be very intricate and power demanding. Phase vocoder and resynthesis techniques have been widely known as time consuming procedures which require a lot of machine power as well as speed. By implementing some of the SMS procedures we think we can apply these ideas further to parallel computing giving a big hint to our research. A graphical interface is the first step towards hands on experience, since most of these machines still rely on UNIX operating systems (Hernandez, 1995).

Benefits

SMS presents various benefits for composers and music related research. At first and with a reasonable training on manipulating the spectra of sounds, end users can get perceivable results, and sound processing can be treated as sound dissection. This is helpful for finding out spectra of known timbres as well as complex timbres seldom used in musical context. Thereafter this results are worth the effort, for resynthesis and synthesis for achieving new and rich sounds. This gives the composer the benefit of working with sound like an artist resembling the process of creating a sculpture. Nevertheless , this context is far from being ideal to reproduce sounds of traditional instruments, although is ideal for natural sound sources (Serra-Smith, 1990).

In spectral models, description of sound characteristics depends upon what the listener, or to what the composer perceives. In this, the timbre or spectral characteristics of the sound are described. Then sound generation is carried out from the perceptual data. Therefore, by analyzing a particular sound, its perceptual parameters can be extracted and then analyzed. As a result it is possible to synthesize the original sound, and the parameters can be modified in such a way, that the resulting different sound maintains aspects of the original sound analyzed (Serra, 1996).

Project Description

This project aims to provide research on an interdisciplinary basis. It involves both artistic and scientific disciplines. The goal is to create a tool that is intuitive and helps to define and manipulate in a musical context, what is known as a sound object. This is an interface which demands portability to the various environments on which SMS has been implemented and could also be maintained, upgraded, and used as part of other compositional environments like PatchWork and Common Lisp Music, therefore guaranteeing its longevity. It should also provide room for future enhancements and improvement. TCL/TK was found to be the appropriate environment because of its standard and performance among UNIX systems. The use of an X-windows application will also assure the CRAY J-916 implementation.

There was also an in-depth analysis to the performance and procedures of the analysis and synthesis techniques of SMS in order to obtain statistics and data for the project. Tests were performed on DECAlphas, SGI, NeXT and Pentiums to some extend. Finally due to the extensive tools and libraries offered on the SGI, this was chosen as the machine for development at beginning stage. Future development will include the other machines. There has been a close cooperation with some local composers as well as interface developers in order to get input for various suggestions on this approach. Previous graphical interfaces for SMS were also taken into account (Aguiar, Serra 1994) as well as the SMS web site Java application. There was also a version of SMS for PC and Windows95, that use a very neat graphics interface for SMS synthesis.

The Design of an Intuitive Interface

Music activities in the Colombia focus on two major aspects. First musicology and technical skills for performance and second by researching folklore. There has been a great effort to

develop high musical standards, and traditional instrument skills by importing or mimicking methods from European countries, and the United States. Folklore is studied from the compositional, and anthropological sides mainly. For the most people, skills do not achieve a competitive standard , therefore it is to our belief, that computer music techniques can help Colombians to be able to achieve a minimum standard at a worldwide level (Reyes, 1996). Our project has been based on this assumption and suggests that for a composer with an average level of technical expertise on music theory and traditional techniques, the sound approach is very appealing. In particular SMS brings this out, therefore helping to solve the problem for some composers as having to write music in traditional ways.

With these tools the soundscape approach for composition is very appropriate. For our own purposes it gives us the advantage that work can be done in a different way. It is not instrumental, but rather a piece where the artist, matter and context mix to make the sound alive. It is done so, by spreading it over a time line on tape or hard disk, and frozen on time, like an artist over canvas. This interface is aimed at people with little computer background but with some knowledge on spectra, music composition and aesthetics. It is of high priority to limit the use of programming and computer commands, therefore giving the artist freedom to concentrate on his / her subject matter.

Our definition of intuitive interface is based upon the logic SMS has been constructed. It follows a modular approach which resemble the different built-in algorithms on the original program. Namely there is a module for each procedure with windows for its parameters. In addition, there is a module for file management of input and output processes, and there are modules which control extended SMS functions. These are a sound player application, an spectral analysis graphing program, and the Internet applications. The advantage of this modular design is that SMS processes can be divided into building blocks. Thereafter each block can be treated separately. The module has all the parameters relevant to the process with suggested default values. Thus, each parameter can be learned and exploited, since the user is focused on one process at a time. Visual indicators of processing events and times inform the user the status of the program.

On line documentation on html format is included for most of the SMS parameters with suggestions on how better to perform a particular task. This documentation was part of the original package and also there are pointers to the SMS web site as well as references. Since this interface is aimed to Colombian composers and Spanish native speakers most of the documentation has been translated to Spanish. The windows and modules have also been translated and show the Spanish counterpart of the English parameter. This is part of our intuitive approach to design which will be evaluated accordingly in order to see if language is a barrier for music software end users.

Interface Description

Flexibility and simplicity have been definitive in the design of this graphic user interface, so that the user with little or no knowledge on spectral manipulation could manage information without risking all the possibilities of SMS algorithms. For flexibility all standard SMS options are included on surface windows with capabilities for invoking more complex options from behind. For producing a simple execution the interface offers coherent manipulation of the information. The distribution of its components are sprout in terms of its functionality. Full Description of SMS parameters can be found on various articles by Julius Smith and Xavier Serra listed on the references. On this paper we are limited to mention their effect on the design of the interface.

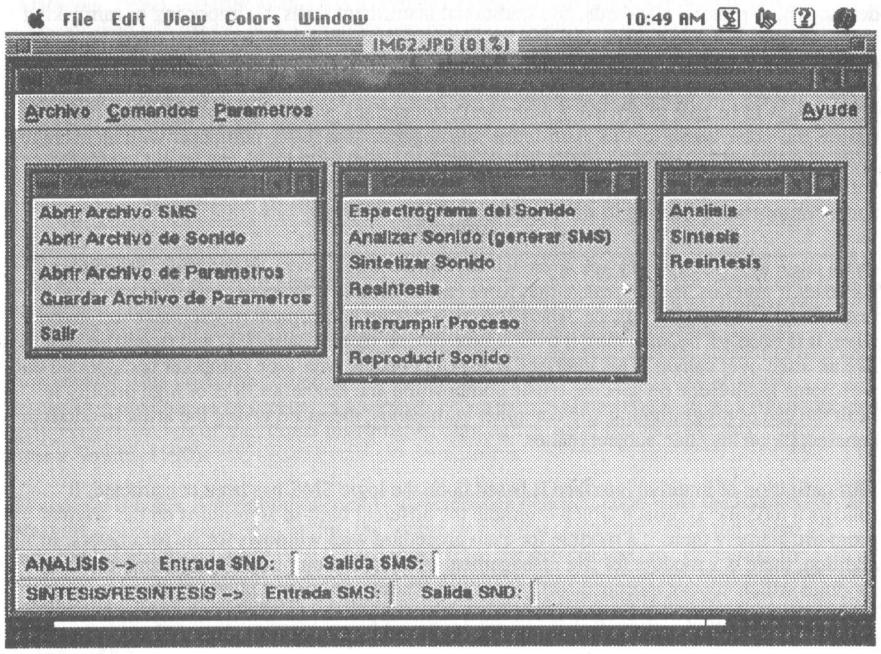


Figure - 1

The interface consists of four basic options (see figure 1): File, Commands, Parameters, Help. The first option involves operations that deal with files. There are three different kinds of file formats. The first is the sound format which defaults to NeXT sound format: .snd. This standard has been imported to other UNIX systems for sound manipulation. The second format is the result of the sound analysis or namely the .sms format. This file stores the spectral information of the sound. It contains deterministic information as well as the non-linear stochastic part. This file is always a subproduct of an .snd file. The third kind of file manages parameters for SMS performance. It contains parameters which are used for analysis and synthesis of the sound. In the synthesis case the parameters file can be viewed as a music score because its parameters describe sound manipulation on the time domain. The file manipulation option has the following functions: Open a .sms file to be processed. Open a .snd file to be played or analyzed. Open a parameters file, which loads a corresponding data structure into memory. Save a parameters file, and quit SMS. The second of the four main options involves operations that take place over SMS functions and are called command options. The following commands are available to the user. Sound spectrogram. This is useful for getting parameters for the analysis of the sound and it is performed before SMS takes place. There is also an option for a .sms file spectrogram.

The second command of the SMS functions is the perform analysis on a soundfile. It performs the spectral analysis of the sound. The input soundfile has been previously selected in the file options with values previously selected in the parameter options.. The third command of the commands option is the perform synthesis of a .sms file. The sms file has been previously selected at the file options, and its values must have been previously selected at the options

parameters. The fourth command performs an interrupt giving the user the option of continuing or stopping the process. The fifth command is a play command for the synthesized .snd soundfile.

The third of the basic options of the interface, is the parameters option which are going to be used as part of the analysis and synthesis processes (see references). For the analysis parameters there are windows for choosing general analysis parameters, peak detection parameters, peak continuation parameters, pitch detection parameters, partial cleaning parameters, stochastic analysis parameters, and tracing parameters. For example, in the general parameters windows there are options for an output .sms filename, type of sound (harmonic or enharmonic), harmonic sound to be analyzed, type of analysis window, window size, and frames per second. There is an option for reverse analysis. On the stochastic analysis window there are options for performing stochastic analysis as well as for the number of coefficients and the cutoff frequencies for filtering a spectrum for the analysis to take place. Some examples with default parameters for different kinds of sounds are planned as guidance for the user.

For the synthesis parameters of the parameters option there is a window which gets the soundfile name and creates a parameters file with various synthesis blocks as well as the envelope pairs. There is also a compression/expansion factor that can be applied to the existing .sms file. This window can bring up amplitude parameters, frequency parameters, modulation parameters and hybrid creation. Most of them can be controlled by means of a vector or a list which can be given in terms of a function. This interface provides a list editor on which an element can be added or deleted from the list. There is also a function or envelope editor that provides pairs within a time vs. amplitude constrains.

The fourth of the interface options is the help option. This was designed with the simple and intuitive interface on mind. It provides a definition by command to every single process that is performed with SMS. It gives a simple tutorial so that the user can get hands on experience to the basic functions of the software. There is also a brief definition with references for further study on the Fast Fourier Transform, and for manipulating sound by means of spectrum analysis.. There is a description with generic examples of most parameters for analysis and synthesizing. This was highly inspired on the SMS home page on the Internet at http://www.iua.upf.es/eng/recerca/sms.html.

Future work

There are plans to integrate the graphic user interface with algorithmic composition packages such as Patchwork and Common Music. This because, it is desirable to unify and get some standards so that this work will not end here. These packages already support syntaxes widely known on the computer music community and are modular and adjustable to any composer desires. As suggested by Xavier Serra there are also plans to extend to graphic user interfaces that use JAVA. In this case processing will still be done at UNIX operating system level, and JAVA will be used to call upon commands of the mother system. This is to some extend, an effort to integrate our work to the work at Phonos in Barcelona. As for applications of SMS we are looking forward to do research on sound compression, partial extracting and more on voice recognition. It is our intent to get musicians and artists working with sound and spectra. The results will be compositions which use this technique. From the electrical engineering standpoint , further research on digital signal processing, peak detection, and linear predictive code will be done.

Conclusions

We think that the approach of a graphic user interface gives the power of intuition and stimulates exploration of creativity among many users that did not even know this technique ever exist. Therefore, a friendly interface can reach more musicians and researchers to get more

and better jobs accomplished. From the beginning we did not try the reinvent approach, but rather to improve over something and therefore giving room for more improvement. This project has given us the insight of research on interfaces and different approaches to its design. This is where the interdisciplinary approach makes its fundamental contribution. From the technical standpoint we know have found the boundaries of what can be achieved with these techniques. From the artistic standpoint we can find the control to what can be done. As a result there has been interest from the advanced computing group in fields such as signal processing for computer music applications. We have been able to involve more people which are not only computer scientists. At this point we feel somehow confident to know what can be accomplished and what is feasible, on this kind of research. With this we hope that research in this field will continue at Los Andes and that this tools will become other kinds of instruments for musical creation.

References

Cook, P. 1995. *"Introduction to Video and Audio Compression Techniques"*. Proceedings of SIGGRAPH, 1995.

Flanagan D. 1996. *"Java in a Nutshell"*, Sebastopol CA USA. O'Really Associates

Harris, F.J. 1978. *"On the use of windows for for harmonic analysis with the discrete Fourier Transform"*, IEEE Proceedings, vol 66, pp. 51-83

Hernandez J.T. et al. 1995. *" Proyecto de Computacion Avanzada de la Facultad de Ingenieria de La Universidad de Los Andes."*.; CIFI. Bogota, Colombia Universidad de Los Andes.

Maher, R.C. and Beauchamp J.W. 1994. *"Fundamental Frequency Estimation of musical signals using a two way Mismatch Procedure."* Journal of the Acoustical Society of America 95(4):2254–2263

Mathews, M.V. et al. (1969) *"The technology of Computer Music"*. Cambridge, MA. USA.: MIT Press.

Makhoul, J. 1975. *"Linear Prediction: A Tutorial Review."* IEEE Proceedings vol 63:pp. 561-580

McAulay, R.J. and Quatieri T.F.. 1986. *"Speech Analysis/Synthesis based on a Sinusoidal Representation."* IEEE Transactions an Acoustics, Speech and Signal Processing. 34(4): 744-754.

Mojica C. 1991. "Una Interfaz para Analisis de Sonido Mediante el Vocoder de Fase"; Thesis dissertation, Santafe de Bogota- Colombia Universidad de Los Andes.

Moore, F. Richard, 1990, *Elements of Computer Music*, Englewood Cliffs, USA, Prentice-Hall

Redeker M. S. and Ross C.P. 1991. *"Pseudo-spectral algorithms and seismic modeling"*. Cray Channels Mendota Heights, MN USA.. Cray Research inc.

Reyes, J., March 1996, *Proyectos Sobre Informatica Musical,* Bogota, Colombia, MOX, Centro de Computaci—n Avanzada, Universidad de Los Andes.

Serra, X. 1989. *"A system for Sound Analysis/Transformations/Synthesis based on a deterministic plus Stochastic Decomposition"*. Ph. D. Dissertation, Stanford University.

Serra, X. and Smith J., 1990. "Spectral Modeling Synthesis: *A sound Analysis/Synthesis System based on a Deterministic plus Stochastic Decomposition.*" Computer Music Journal 14(4): 12-24

Serra, X. 1995. *"Current Perspectives in the Digital Synthesis of Musical Sounds"*.; Phonos Foundation, Pompeu Fabra University. Barcelona Spain

Serra, X. 1996. *"Musical Sound Modeling With Sinusoids Plus Noise"*.; Phonos Foundation, Pompeu Fabra University. Barcelona Spain.

Strum, Robert D., 1989, *"First Principles of Discrete Systems, and Digital Signal Processing"*, Menlo Park, USA, Addisson-Wesley

Welch, B. 1995." *Practical Programming in TCL and TK"*, Englewood Cliffs, NJ.USA. Prentice Hall.

Pattern Reuse in Tonal Music Improvisation and Accompaniment Systems

Geber Ramalho

Departamento de Informática, Universidade Federal de Pernambuco

Cx. Postal 7851, 50740-540, Recife-PE, Brasil

glr@di.ufpe.br — www.di.ufpe.br/~glr

Abstract

Previously stored melodic and rhythmic patterns can be sequentially combined and adapted to compose new melodies and rhythmic lines. This pattern reuse technique has been increasingly applied in the design of tonal music improvisation and accompaniment systems, particularly in jazz styles. The technique appears to be promising since it aids the system designer to deal with the well-known "knowledge acquisition bottleneck", which is a fundamental issue in building intelligent systems. However, the success of this technique depends on critical design choices concerning patterns' nature, representation, indexing, preference and adaptation. Despite its increasing importance, patterns reuse advantages and fundamental issues have not been widely discussed so far. Based on the case based-reasoning theoretical framework (Kolodner, 1993) and on my own experience in building jazz bass playing system (Ramalho, Rolland & Ganascia, 1997; Ramalho, 1997), I survey here the existing music improvisation and accompaniment systems, to shed a light on the design choices involving patterns reuse.

1. Introduction

The last 10 years have witnessed the development of tonal music improvisation and accompaniment systems in computer music scenario. These systems' basic task is to automatically generate melodic lines (e.g., saxophone and bass), rhythmic lines (e.g., drums and percussion) and/or chord voicing chaining (e.g., piano) in a particular music style according to a given chord grid. A chord grid is a sequence of ciphered chords (e.g., Fm7, Bb9, Ebmaj7), as typically found in "real/fake books" (Sher, 1991). Most of music improvisation and accompaniment systems are dedicated to jazz, especially bebop (Baggi, 1992; Brown & Sidley, 1993; Giomi & Ligabue, 1991; Hidaka, Goto & Muraoka, 1995; Hodgson, 1996; Horowitz, 1995; Johnson-Laird, 1991; Pachet, 1990; Pennycook, Stammen & Reynolds, 1993; Ramalho & Ganascia, 1994; Spector & Alpern, 1995; Ulrich, 1977; Walker, 1994). We also find other musical styles, such as rock, reggae, bossa nova, etc. (Ames & Domino, 1992; Band-in-a-box, 1995; Levitt, 1993). The tonal music improvisation and accompaniment systems have been basically used as arrangement assistants, to avoid the detailed specification of the parts of some instruments, and as rehearsal partners. For instance, the computer plays the parts of some instruments (e.g., the rhythm section), while the user plays another part (e.g., the solo), and vice-versa.

Designing music improvisation and accompaniment systems is a quite complicated task. First, if the system is to be used as a rehearsal partner, it must work in real time, in order to be able to interpret what the user (or other systems) has been playing and react accordingly. Second, it is hard to elicit and represent the knowledge used by musicians to bridge the strikingly large gap between the music they actually play and the chord grid's simplistic guidelines. In fact, the knowledge acquisition process, which aims at "transferring" knowledge concerning a given task from the expert to the computer, is the most difficult process in building intelligent systems. This is particularly complicated for music creation activity because of its empirical and intuitive character.

To tackle the knowledge acquisition bottleneck in musical tasks, some researchers have adopted the reuse of patterns as the basic strategy for building music improvisation and accompaniment systems. The whole idea is to retrieve and adapt previously stored melodic/rhythmic patterns and put them side by side to compose a new melodic/rhythmic lines. Patterns are, in fact, easy to acquire, and they represent knowledge *in extension* (Woods, 1985). Moreover, the musical results obtained by some of the above cited systems are really impressive. These points confirm pattern reuse as a promising technique for building music improvisation and accompaniment systems.

However, the importance and critical issues of patterns reuse has not been widely discussed so far. These issues include the determination of patterns' nature, representation, indexing, preference and adaptation. Based in the case based-reasoning theoretical framework (Kolodner, 1993) and on my own experience in building jazz bass playing system (Ramalho, Rolland & Ganascia, 1997; Ramalho, 1997), I survey the existing music improvisation and accompaniment systems, to shed a light on the design choices involving patterns reuse. I claim that case-base reasoning is a powerful framework for highlighting and proposing solutions for the main issues in pattern reuse. Patterns can be seen as cases, since they represent musical passages actually played that can be reused in a similar context in future. During the last ten years, case-based reasoning has been widely employed with success by the Artificial Intelligence community as an alternative methodology for developing knowledge-based systems, lightening the knowledge acquisition effort.

Usually, a pattern is designates a *recurrent* musical structure. I employ the term "pattern" in a loose sense, since most of the above cited system impose no restrictions on the occurrence frequency of their patterns. A pattern may be a melodic fragment that occurred once in the analyzed corpus.

In the next Section, I discuss in more detail why patterns are used in the existing music improvisation and accompaniment systems. I also introduce some fundamental concepts of case-based reasoning. In Section 3, the fundamental issues concerning pattern reuse are presented and the trade-offs of some design choices are analyzed. I draw some conclusions and point out some directions for further research on pattern reuse.

2. Why reuse Patterns?

One of the main problem in designing music improvisation and accompaniment systems is the acquisition of the knowledge used by musicians. Most of the time, musicians cannot explain their choices at the note level in terms of rules, constraints or any known formalism (Pachet, 1990; Ramalho, 1997). They usually justify what they have played in terms of more "abstract" *musical properties*, such as swing, tension, contour, density, balance, etc. However, the interaction between these concepts is not simple to determine, and there is no logical rule chaining that can directly instantiate them into notes. This is particularly true for knowledge related rhythm, dynamics and sound effects. For instance, a soloist may not be able to explain the choice of each note (pitch, amplitude and duration) of a passage, even though he has consciously used a given scale

The musician's difficulty in explaining choices analytically is partially a consequence of the empirical way they learn to create music. Especially in jazz, musicians learn to play by listening to and imitating performances of famous musicians (Ramalho, 1994; Sabatella, 1996). Through this learning process, the musicians acquire a set of examples of melodic/rhythmic phrases and chord chaining which can be *reused* in the future. The strongest evidence of this claim is the musicologists' work on the identification of the catalogue of patterns used by famous jazzmen, such as Charlie Parker (Owens 1974), Bill Evans (Smith 1993), Miles Davis (Baker 1980), John Coltrane (Kernfeld 1983), Lester Young (Rottlieb 1991). Figure 1 illustrates some melodic phrases frequently used by Miles Davis in a major II-V chord sequence. This does not mean that playing jazz is limited to putting side by side previously known patterns. Between their "favorite phrases", musicians play "original" phrases or variations of their favorite ones. However, there is enough evidence testifying that musical examples (patterns) reuse plays an important role in creating solos and accompaniments.

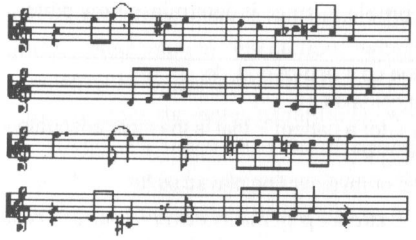


Figure 1 - Five typical melodic patterns from Miles Davis on |Dmin7 |G7 | (a major II-V) as identified by David Baker (Baker, 1980)

Departing from the musicological or cognitive standpoint, what is important for computer scientists is that pattern reuse is an extremely powerful technique to attack knowledge acquisition problems. First, patterns can be easily acquired either by consulting experts or the literature, such as (Aabersold, 1979; Coker, 1970). Second, patterns represent knowledge in extension, since they implicitly codify solutions (e.g.,

appropriate phrases) for a given musical problem (e.g., a chord sequence). For instance, it is hard to identify fine-grained "rules" that indicate how to *build* a bass melodic phrase for a given chord sequence, such as Fm7(b5) Bb7, in different contexts (e.g., with respect to the location of the chord sequence within the song, to what the musician himself has played just before, etc.) and according to some desired musical properties (e.g., density, melodic contour, tension, etc.). Figure 2 shows two bass line fragments that could be respectively retrieved, transposed and reused to Fm7(b5) Bb7, in the case one wants a simple and smooth phrase (upper fragment), or a slightly dissonant and syncopated phrase in the medium range, containing several notes and using mainly chord notes.

Figure 2 - Two bass line fragments as played originally by Ron Carter in Stella by Starlight (Aabersold, 1979)

Some computer scientist have then adopted the reuse of pattern as the systematic strategy (Ulrich, 1977; Baggi, 1992; Band-in-a-box, 1995; Ramalho & Ganascia, 1994; Hodgson, 1996) or as an additional strategy (Spector & Alpern, 1995; Pennycook, Stammen & Reynolds, 1993; Walker, 1994) for generating music. Previously stored melodic and rhythmic patterns are sequentially combined and adapted to compose new melodies and rhythmic lines. The basic loop of the systematic pattern reuse algorithm is shown in Table 1. The current pattern *reuse context* C may be characterized by the following elements (see Figure 3, for illustration): the melodic or rhythmic line played so far by the system; the parts played so far by the other musicians/systems; the chord grid-related information (previous, current and future chords; current position within the song; song style; song tempo, etc.). Choosing what constitutes the relevant information for describing the context in which the pattern will be reused is a central problem, since it determines how patterns must be indexed.

```
While the current position cp < end-of-grid, do:
    Describe the current context C;
    Search in the library L for a pattern P that is the most adequate with respect to C;
    Adapt P to C, obtaining P';
    Add P' to the melodic or rhythmic line played so far
```

Table 1 - Basic loop of the systematic pattern reuse algorithm

The existing systems employ different kinds (from melodic fragments to pitch interval sequences) and size (from one-chord to many-measure fragments) of patterns, as well as different representation schemes (from simple character lists to sophisticated object-oriented representations). They also adopt various policies for choosing the good pattern (from random to a measure of the similarity between the current reuse context and the original one) and for adapting it to the current context (from no adaptation to significant modifications). To discuss these critical design choices, I use case-based reasoning (Kolodner, 1993) as a theoretical framework. Patterns reuse can

be studied within the framework of case-based reasoning, since patterns can be seen as cases: they are episodes that can be reused in a similar context in future. This is the very definition of case-based inference mechanism. A case is composed by a description of a problem and its respective solution. When a new problem is presented to a case-based system, its relevant features (indexes) are described. Next, the system performs a index-guided search into the case-base in order to retrieve a case whose problem part is the most similar the new given problem. The solution associated to the previously stored problem is adapted to solve the new one.

3. FUNDAMENTAL ISSUES ON PATTERN REUSE

In this section, I enumerate the main issues on pattern reuse, taking as examples the existing music improvisation and accompaniment systems. I analyze the consequences of some possible design choices and give some suggestions for good design.

3.1. Pattern nature

What kind of patterns the pattern library (case base) must be composed of? This is one of the most preliminary and important questions the designer of a improvisation and accompaniment system must be aware of. The pattern's nature can be determined regarding to two aspects: content and length.

The *content* is related to the dimensions (e.g., pitch, beginning, duration, amplitude, timbre) that are taken into account in the pattern description. According case-based reasoning paradigm, the closer a pattern is to a transcription of an actual improvisation or accompaniment fragment performed by a professional musician, the better it is, since it carries more information (and knowledge). In this perspective, the patterns used in Ulrich's system are very poor, since they are a mere sequence of degrees of an abstract scale and make no reference to the original durations. By loosing rhythm information, Ulrich's system discards one of the musical knowledge's aspects that is really hard be formalized. Moreover, by codifying various dimensions altogether, one can "grasp" their mutual interaction, which is also difficult to formalize. The rhythmic patterns in Baggi's system (16 for piano and 22 for drums) are richer, but they do not consider the notes' amplitude, which could be a useful information for swinging. Hodgson's (more than 64 saxophone melodic fragments), Ramalho's (more that 250 bass melodic fragments) and Band-in-box (about 20 fragments per instrument in a given style) code their patterns in more detail, including pitch, duration and amplitude information.

The choice of patterns *length* should take into account the musical plausibility (fixed vs. variable length) and the reasoning granularity (local vs. global information). Fixed-length patterns are not musically plausible. The "natural" structure for building melodic lines is the (melodic) phrase, which has different lengths (Lerdahl & Jackendoff, 1983; Narmour, 1989). However, unlike natural language, there is no clear definition for the beginning and end of these phrases. They may overlap, or there may be silences between them. Because of this difficulties, some systems adopt a fixed-

length pattern generally equal to one measure. As regards the reasoning granularity, one should find a compromise to guarantee both continuity and quick reaction. On one hand, it is difficult to control the overall coherence of the melodic/rhythmic line using too short patterns (e.g., lasting less than 2 beats), since the line generated is to fragmented. Moreover, some musical properties (such as tension or density), that may serve as guidelines for choosing the best pattern, cannot be measured adequately for too short patterns. On the other hand, too long patterns (e.g., lasting more than 4 measures) are inadequate in real-time applications where the system acts as a rehearsal partner. In fact, while the system is playing a pattern, many important events may occur in the environment (e.g., the soloist is playing chromatic scales). The system will not be able to react quickly, changing what it intended to play accordingly. The best compromise regarding both plausibility and granularity employed by the existing systems so far is to choose patterns corresponding to a single chord (band-in-a-box's and Pachet's systems) or a particular chord chunk (Hodgson's and Ramalho's systems). Chord chunks, such as II-V, II-V-I, VI-II-V-I, are chord sub-sequences typically found in jazz, bossa nova and pop songs. Since they are recurrent, they improve reusability. Chord chunks play, in fact, an important role in musicians reasoning, since they are well-defined harmonic references (Ramalho, 1997). The majority of patterns identified by the musicologists cited in the preceding section were played on chord chunks.

It is worthwhile to state that the fact that a melodic or rhythmic fragment implicitly codify musical knowledge does not depend on the recurrence of that fragment. Thus, a musical fragment may be useful whatever is its recurrence in a corpus. In fact, the majority of the existing systems, the patterns are not actual "patterns" in the strict sense of the word.

3.2. Pattern representation

How patterns should be represented? The issues and guidelines associated to the choice of representation language for patterns are basically the same applied to systems dedicated to tonal music tasks. The adequacy of the language will facilitate the inferences the system will be capable to perform. For instance, the melodies of Spector and Alpern's system are represented by a 64-number list, each one corresponding to a pitch and having the same duration (eighths). Ulrich's system represents a chord such as Cmaj7 by "C (4 7 11)", where the numbers correspond to the semitone interval to the root. These are poor representations. As they don't make any reference to the notion of diatonic interval, many concepts of harmony, such as the determination of enharmonic spelling of a note, cannot be easily manipulated.

The computer music community has been discussing the representation problem for the last ten years. Many languages and platforms have been developed to support composition and analysis tasks: Pla (Schottstaedt, 1983) Flavors Band (Fry, 1991), Common Music (Taube, 1991), MODE (Pope, 1991), MusES (Pachet, 1994), etc. These works reinforce the conviction that object-oriented programming is the most appropriate paradigm to represent and manipulate (tonal) musical concepts.

3.3. Pattern indexing

Which are the good indexes for storing patterns? Indexing is the central issue in pattern reuse. One must find a pertinent set of descriptions (indexes) so as to guarantee that the good pattern will be appropriately and quickly retrieved in a given context. This a general problem in case-based reasoning known as "the indexing problem" (Kolodner, 1993). In the case of musical patterns, there are to groups of indexes: those describing the melodic and rhythmic pattern it self, and those characterizing the context within which the pattern were played or must be played (e.g., underlying chords, tempo, position within the song, next chords, pattern played just before).

The first group of indexes are composed of the vocabulary used by musicians to talk about melodic/rhythmic fragments (and chord voicing in the keyboard case) of a particular instrument in a musical style. They may correspond the global musical properties, such as scale preference, dissonance, tessitura, melodic contour, dynamics, etc. These properties, features or facets can be seen as abstract descriptions of a set of notes. Abstraction description contributes to reusability, since it facilitate the computation of the similarity between two melodic/rhythmic fragments. For instance, it is more computationally complex and less intuitive to look for a particular sequence of notes, such as D E F# G A, in the pattern library, than to look for a melodic fragment whose property is "slightly ascending contour" or "notes played in G major diatonic scale".

Excepting their duration (length), the patterns global properties are not still taken into account in most of the current systems. Baggi's system includes pattern density, and Hodgson includes melodic contour, dissonance and other few properties. Ramalho's system incorporates the most exhaustive list of properties (pitch contour, tessitura, scale, dissonance, construction style, inversion, repeated notes, rhythm style, density, syncopation, amplitude contour, loudness, leading note, pull down, drop, swing and classicness) for describing bass line fragments in jazz style. The use of rich indexing vocabulary enables a more accurate retrieval, which is a advantageous characteristic. However, it also demands a further knowledge acquisition effort, since the designer must identify all the relevant features.

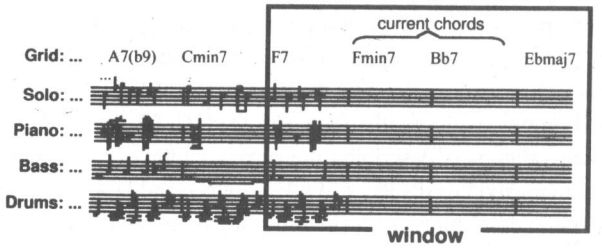


Figure 3 - Filtering the context description

The second group of indexes are composed of the features that characterize the context of pattern reuse. For computational complexity reasons, one cannot include the description of the entire melodic/rhythmic line played by the system so far, the

complete parts of the other instruments played so far or the whole grid. Thus, one might choose the relevant context features as well as determine a temporal window for filtering the context. Typically, this window is defined around the current chord(s), including some preceding and subsequent measures (see Figure 3).

The majority of the current systems represent only the pattern's underlying current chords (e.g., Fm7 Bb7, in Figure 3). Band-in-a-box, Hodgson's and Baggi's extend the context representation to include the next subsequent chord (e.g., Ebmaj7, in Figure 3) or the interval between the roots of the last current chord and the next subsequent one (e.g., a descending forth, in Figure 3). Hodgson's and Ramalho's also consider the previously played chords (e.g., Cm7 F7, in Figure 3) as well as the previously played melodic pattern. Ramalho's enrich the current grid segment description by adding information about the local tonality, the chord chunk type (e.g., II-V-I), position within the song and within the chorus, song style, song tempo, etc. Figure 4 shows the Ramalho's interface for the acquisition a pattern, represented as a case in a particular case-base called Musical Memory. Each pattern is indexed with respect to its musical global properties (right-top), the underlying grid context (right-bottom), the previously played pattern's properties (left-top) and grid context (left-bottom).

There are two global approaches for describing the context. First, one represent the context in which the pattern as originally played. This approach, that follows case-based reasoning paradigm, is fully adopted by in Ramalho's and partially by the other systems. Second, the pattern is described only with respect to the future contexts in which the pattern may possibly be reused. This is Band-in-a-box's approach. For instance, instead of stating that a pattern *p* lasting 4 beats was played on a C major chord, Band-in-box informs that *p* can be used on major chords or, simply, any 4-beat chord. The advantage of the former approach is its flexibility due to the fact that it avoids an early generalization. The complete information is stored (C major), and only during the adaptation phase the constraints are relaxed allowing the pattern to be reused on majors chords or any 4-beat chord.

3.4. Pattern choice

Which are the criteria for preferring a particular pattern in the library? The retrieval process aims at finding a pattern whose description satisfies the best the given *query*, which is generally composed by the description of both the current context and desired musical properties. Formally, the query Q = (C, D), where C = $\{c_1,...,c_k\}$ may be a set attribute-value pairs describing the current context (e.g., chords = 'Cm7 F7', tempo = 120, previouslyPlayedNotes = 'F# E G', etc.), and D = $\{d_1,...,d_j\}$ may be a set attribute-value pairs describing the desired musical properties (e.g., dissonance = low, rhythmStyle = quarter-based, etc.). The policies adopted by the existing systems differ in following aspects: the description richness of C; the description richness of P; and the retrieval function F(Q,L) which defines how C and P are considered for choosing the pattern into the pattern library L.

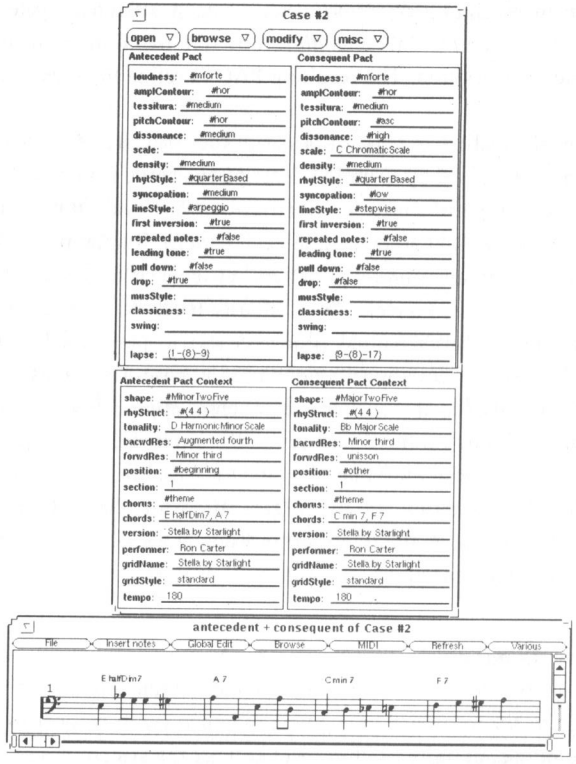


Figure 4 - Case in Ramalho's system obtained from the bass line played by Ron *Carter on STELLA BY STARLIGHT (Aabersold, 1979).*

The query contents P and Q are obviously related to the pattern indexing vocabulary. As discussed in the previous sub-section, most of the existing systems restrict indexing to few features of the current context, and none or very few pattern's musical properties are included. Ramalho's, Hodgson's and Baggi's systems employ a larger indexing vocabulary, and then have more control on pattern choice.

Context description can be quite easily enriched, but the inclusion of musical properties as additional criteria in pattern retrieval demands more domain knowledge a as well as the implementation of other reasoning mechanisms. In Baggi's systems, the values of three user-manipulated knobs (hot/cool, dissonance/consonance, free/as-in-ness) are the inputs of a neural network that activates in a coherent manner various musical properties (output). The disadvantage of Baggi's current approach is that the properties do not change along the performance, unless the user changes the three knobs. The musical properties should be set according to the dynamic environment changes (e.g., the drummer starts to play "hot riffs"), the position within the song (e.g., improvisation vs. theme exposition), etc. In Ramalho's system, this problem is solved by the notion of PACTs (Potential ACTions)which are sophisticated objects aimed at representing the actions a musician intend to execute, such as "play lot of notes", "play

the major scale in ascending direction", etc. At each inference cycle (see Table 1), some PACTs are *activated* and then *combined* by rule-based mechanisms according to the current context description. The resulting PACTs refine the query to be formulated to the case base.

The retrieval function is random-based in the majority of the existing systems. The query criteria expressed by C and D descriptions serve just to restrict (weight up) the random choice. A particularity of Band-in-box retrieval strategy is the use of a user-provided priority weight associated to each pattern. Ramalho's system is the only one that do not use any randomness in pattern retrieval. It retrieves the pattern that is the most similar to the given query. The similarity measure between the query (called target case) and the patterns (called source cases) is performed using a k-nearest neighbor classification, a common technique used in case-based reasoning (Aamodt & Plaza, 1994). Similarity-based retrieval is a coherent approach when the system has a rich description of the environment features and desired musical properties. Since this approach is more time consuming than the random-based one, the system designer has to pear a particular attention to the organization of the pattern library. Some hierarchical organizations, as the one used in Ramalho's, or parallel search strategies (Kolodner, 1993), may speed up the retrieval process and guarantee a real time application.

3.5. Pattern adaptation

How to adapt the retrieved pattern to the current situation? The adaptation process' goal is to try to assure harmonic coherence (with respect to the underlying chords) and melodic continuity (with respect to what the system has played so far).

Among the different stages of pattern reuse (see table 1), adaptation is the less studied and developed one. There are two reasons for this. First, some transformations are complicated. For instance, there is no general procedure for adding/deleting notes in a melodic fragment. Second, one cannot apply some transformations on the retrieved pattern without changing its other properties. For example, it is too difficult to change the melodic contour (e.g., from descending to ascending) of a fragment without changing is harmonic content. One need to order transformations to control their mutual interference, they need to be ordered. Unfortunately, ordering any set of non-orthogonal musical transformations is an opened issue.

Because of the above cited reasons, the majority of the existing systems restrain the adaptation to simple and quite orthogonal ones, such as time shift, tempo change, pitch transposition, and amplitude change. Hodgson's and Ramalho's, however, change the pitch and/or duration of some notes, typically the last ones, in order to "soften" patterns chaining. Baggi's makes more radical changing by introducing passing notes in the bass line so as to augment density. This system also change frequently a fifth by a flat fifth in the piano and bass parts in order to augment tension.

In order to attenuate the limitations of the patterns adaptation, Ramalho's has introduced an adaptability measure as an additional criterion for pattern choice during the retrieval process (Ramalho, 1997). This strategy has been recently employed in other case-based systems devoted to design tasks (Smyth & Keane, 1994; Smyth & Keane, 1995)

4. Conclusions

The reuse of melodic and rhythmic fragments is a promising technique in designing musical improvisation and accompaniment systems. Even in other composition tasks without chord grid guidance, such as those studied by Cope (Cope, 1991), this technique applies. Pattern reuse partially solves the knowledge acquisition problem, since patterns are easily available and codify important aspects of musical knowledge that are difficult to be formalized. However, a great amount of knowledge is still needed to guide the indexing, retrieval and adaptation of the patterns. The systems that incorporate this knowledge extensively are those that exhibit the best musical results. In fact, the intelligence of a system based of pattern reuse is in its capability of interpreting a given situation in order to choose the most adequate case. The integration of different reasoning paradigms (case-, rule-, constraint- and neural net-based) so as to incorporate appropriately the available musical knowledge is the main research direction in building tonal musical improvisation and accompaniment systems.

References

Aabersold, J. (1979). *Play-along: Paying Dues.* . New Albany: Janey Aabersold Pub.

Aamodt, A. & Plaza, E. (1994). Case-Based Reasoning: Fundamental Issues, Methodological Variations, and System Approaches. *Artificial Intelligence Communications*, 7(1), 39-79.

Ames, C., & Domino, M. (1992). Cybernetic Composer: an overview. In M. Balaban, K. Ebicioglu, & O. Laske (Eds.), Understanding Music with AI: Perspectives on Music Cognition (pp. 186-205). Menlo Park: The AAAI Press.

Baggi, D. (1992). NeurSwing: An Intelligent Workbench for the Investigation of Swing in Jazz. In D. Baggi (Eds.), *Computer-Generated Music* (pp. 79-93). IEEE Computer Society Press.

Band-in-a-box (1995). Band-in-a-box Pro 6.0. Canada: PG Music Inc.

Baker, D. (1980). *Miles Davis Trumpet.* Giants of Jazz Series. Lebanon: Studio 224 Ed.

Brown, D., & Sidley, S. (1993). The Expression of Aesthetic Principles as Syntactic Structures and Heuristic Preferences and Constraints in a Computer Program that Composes Jazz Improvisations. In AAAI Spring Symposium Workshop on Artificial Intelligence & Creativity, (pp. 133-6). Stanford: The AAAI Press.

Coker, J. (1970). *Patterns for Jazz.* . Lebanon: Studio Publications/Recordings.

Cope, D. (1991). *Computers and Musical Style.* . Oxford: Oxford University Press.

Fry, C. (1991). Flavors Band: A Language for Specifying Musical Style. In S. Pope (Eds.), *The Well-Tempered Object: Musical Applications of Object-Oriented Software Technology* (pp. 49-63). Massachusetts: MIT Press.

Giomi, F., & Ligabue, M. (1991). Computational Generation and Study of Jazz Music. *Interface*, 20(1), 47-63.

Hidaka, I., Goto, M., & Muraoka, Y. (1995). An Automatic Jazz Accompaniment System Reacting to Solo. In *International Computer Music Conference*, (pp. 167-70). Banff: International Computer Music Association.

Hodgson, P. (1996). Modelling Cognition in Creative Musical Improvisation (unpublished poster). In *International Conference on Music Perception and Cognition*. Montreal.

Horowitz, D. (1995). Representing Musical Knowledge in Jazz Improvisation System. In *Fourth Workshop on Artificial Intelligence and Music, IJCAI-95*, (pp. 16-23). Montreal:

Johnson-Laird, P. (1991). Jazz improvisation: a theory at the computational level. In P. Howell, R. West, & I. Cross (Eds.), *Representing Musical Structure* (pp. 291-325). London: Academic Press.

Kolodner, J. (1993). *Case-Based Reasoning*. . San Mateo: Morgan Kaufmann.

Levitt, D. (1993). A Representation for Musical Dialects. In S. Schwanauer & D. Levitt (Eds.), *Machine Models of Music* (pp. 455-69). Massachusetts: The MIT Press.

Lerdahl, F., & Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. . Massachusetts: The MIT Press.

Narmour, E. (1989). *The Analysis and Cognition of Basic Melodic Structures: the Implication-Realization Model*. . Chicago: University of Chicago Press.

Pachet, F. (1990). Representing Knowledge Used by Jazz Musicians. In *International Computer Music Conference*, (pp. 285-8). International Computer Music Association.

Pachet, F. (1994). The MusES system: an environment for experimenting with knowledge representation techniques in tonal harmony. In *First Brazilian Symposium on Computer Music - SBC&M '94*, (pp. 195-201). Caxambú Computer Science Brazilian Society (SBC).

Pennycook, B., Stammen, D., & Reynolds, D. (1993). Toward a Computer Model of Jazz Improviser. In *International Computer Music Conference*, (pp. 228-31). Tokyo: International Computer Music Association.

Pope, S. (1991). Introduction to MODE: The Musical Object Development Environment. In S. Pope (Eds.), *The Well-Tempered Object: Musical Applications of Object-Oriented Software Technology* (pp. 83-106). Massachusetts: MIT Press.

Ramalho, G. (1997) *Construction d'un agent rationnel jouant du jazz*. Thèse de doctorat, Université Paris VI.

Ramalho, G., Rolland, P.-Y., & Ganascia, J.-G. (1997). An Artificially Intelligent Jazz Performer. *to appear in Journal of New Music Research*.

Ramalho, G., & Ganascia, J.-G. (1994). Simulating Creativity in Jazz Performance. In *Twelfth National Conference on Artificial Intelligence*, (pp. 108-13). Seattle: The AAAI Press.

Sabatella, M. (1996). *A Jazz Improvisation Primer*. Available in http://www.fornet.org/~marc/primer.

Schottstaedt, B. (1983). Pla: A Composer's Idea of Language. *Computer Music Journal*, 7(1), 11-20.

Smyth, B., & Keane, M. (1994). Retrieving Adaptable Cases: The Role of Adaptation Knowledge in Case Retrieval. In *Topics in Case-Based Reasoning* (pp. 209-20). Springer Verlag.

Smyth, B., & Keane, M. (1995). Experiments on Adaptation-Guided Retrieval in Case-Based Design. In *First International Conference on Case-Based Reasoning*.

Spector, L., & Alpern, A. (1995). Induction and Recapitulation of Deep Musical Structure. In *Fourth Workshop on Artificial Intelligence and Music, IJCAI-95*, (pp. 41-8). Montreal.

Taube, R. (1991). Common Music: A Music Composition Language in Common Lisp and Clos. Computer Music Journal, 15(2), 21-32.

Ulrich, W. (1977). The Analysis and Synthesis of Jazz by Computer. In Fifth International Joint Conference on Artificial Intelligence, (pp. 865-72). Massachusetts.

Walker, W. (1994) A Conversation-Based Framework for Musical Improvisation. Ph.D. Thesis, University of Illinois at Urbana-Champaign.

Woods, W. (1985). What's in a link? In R. Brachman & H. Levesque (Eds.), Readings in Knowledge Representation Los Altos: Morgan Kaufmann.

Who composed *Entre l'Absurde et le Mystère*?

An Introduction to Music and Artificial Intelligence[1]

Eduardo Reck Miranda
L AMESM - Laboratorio de Musica Eletroacustica de Santa Maria
and
Dept. of Music - Glasgow University
eduardo@music.gla.ac.uk

Abstract

One of the key feature that distinguishes humans from other animals is the fact that we are intrinsically musical. Music is generally associated with the expression of emotions, but it is also common sense that the intellect plays an important role in musical activities. The interplay between these two elements figures in the research agenda of a variety of scientific fields, including Neuroscience, Cognitive Sciences and Artificial Intelligence (AI), to cite but a few. This chapter introduces some fundamental issues of AI and its interplay with music. We pay special attention to the use of grammars in music and knowledge representation techniques.

1. The Musical Brain

From a number of plausible definitions for music, the one that frequently stands out in musicological research is the notion that music is an intellectual activity; that is, the ability to recognize patterns and imagine them modified by actions. We understand that this ability is the essence of the human mind: it requires sophisticated memory mechanisms, involving both conscious manipulations of concepts and subconscious access to millions of networked neurological bonds. In this case, it is assumed that emotional reactions to music arises from some sort of intellectual activity.

Different parts of our brain do different things in response to the stimuli we hear. Moreover, music is not detected by our ears alone; for example, music is also "heard" through the skin of our entire body (Storr 1993). The brain's response to external stimuli, including sound, can be measured by the activity of the neurons. Two measuring methods are commonly used: PET (Positron Emission Tomography) and ERP (Event-Related Potential). PET measures the brain's activity by scanning the flow of radioactive material previously injected into the subject's bloodstream. Despite its efficiency, this method is rather controversial because the long term side effects of the radioactive substances to the health of the subject are not entirely kown. ERP uses tiny electrodes placed in contact with the skull of a person to measure the electrical activity of the brain. As far as the health of the subject is concerned, ERP is safer than PET, but the measurement is different. Whilst PET scans give a clear cross-sectional indication of the

[1] This article was originally written for the special issue on Music and AI of the Contemporary Music Review due to appear in 1998.

area of the brain where the bloodflow is more intense during the hearing process, ERP gives only a voltage level vs. time graph of the electrical activity of the areas of the brain where the electrodes have been placed.

Our understanding of the behaviour of the brain when we engage in any type of musical activity (e.g., playing an instrument or simply imagining a melody) is merely the tip of an iceberg. Both measuring methods have brought to light important issues that have helped researchers uncover the tip of the iceberg. PET scans have shown that listening to music and imagining listening to music activate different parts of the brain and ERP graphs have been particularly useful to demonstrate that the brain expects sequences of stimuli that conform to established circumstances. For instance, if you hear the sentence "A musician composes the music", the electrical activity of your brain will tend to run fairly steadily. But if you hear the sentence "A musician composes the dog", the activity of your brain will display significant negative electrical response immediately after the word "dog".

The human brain seems to respond similarly to musical incongruities. Such behaviour obviously depends upon one's understanding of the overall meaning of the language in hand. A number of enthusiasts believe that we are born "programmed" to be musical, in the sense that almost no-one have difficulties in finding coherence in simple tonal melodies (Robertson 1996).

2. Understanding Intelligence with AI

The understanding of the behaviour of the human brain is not, however, identical to the understanding of intelligence. Physical measurements of brain activity may certainly endorse specific theories of intelligence, but not all theories seek endorsement of this sort.

One of the goals of AI is to gain a better understanding of intelligence, but not necessarily by studying the inner functioning of the brain. The methodology of AI research is largely based upon logics, mathematical models and computer simulations of intelligent behaviour (Luger and Stubblefield 1989).

Of the many disciplines engaged in gaining a better understanding of intelligence, AI is one of the few that has special interest in testing its hypotheses in practical day-to-day situations. The obvious practical benefit of this aspect of AI is the development of technology to make machines more intelligent; for example, thanks to AI computers can play chess and diagnose certain types of diseases extremely well.

It is generally stated that AI as such was "born" in the late 1940's, when mathematicians began to investigate whether it would be possible to solve complex logical problems by automatically performing sequences of simple logical operations. In fact AI may be traced back far before computers were available, when mechanical devices began to perform tasks previously performed only by the human mind, including some musical tasks (Levenson 1994).

Is intelligence synonymous to the ability to perform logical operations automatically, to play chess or to diagnose diseases? Answers to such types of questions tend to be either biased to particular viewpoints or ambiguous. The problem is that once a machine is capable of performing such types of activities, we tend to cease to consider these activities as intelligent. Intelligence will always be that unknown aspect of the human mind that has not yet been understood or simulated.

3. Towards Intelligent Music Machines

Music is without doubt one of the most intriguing activities of human intelligence. By studying models of this activity, researchers attempt to decipher the inner mysteries of both music and intelligence. From a pragmatic point of view, however, the ultimate goal of Music and AI research is to make computers behave like skilled musicians. Skilled musicians should be able to perform highly specialized tasks such as composition, analysis, improvisation, playing instruments, etc., but also less specialized ones such as reading a concert review in the newspaper and talking to fellow musicians. In this case the music machine would need to have some basic understanding of human social issues, such as sorrow and joy. Will computers ever display such highly sophisticated and integrated behaviour? More optimistic enthusiasts believe so (Minsky 1985).

As happens in other areas of AI research, however, computers have so far been programmed to simulate most specialized tasks, but fairly independently from each other. Current research work is now looking for ways to integrate the ability to perform a variety of such tasks; e.g., mechanisms from systems for music analysis are aggregated to systems for composition in order to allow for the computer to autonomously compose music in the style of analysed pieces.

It is debatable whether musicians want to believe in the possibility of an almighty musical machine. Musicians will keep pushing the definition of musicality away from automatism for the same reasons that scientists keep redefining intelligence. Nevertheless, AI is helping musicians to better operate the technology available for music making and to formulate new theories of music (Balaban et al. 1992).

4. Formal Grammars

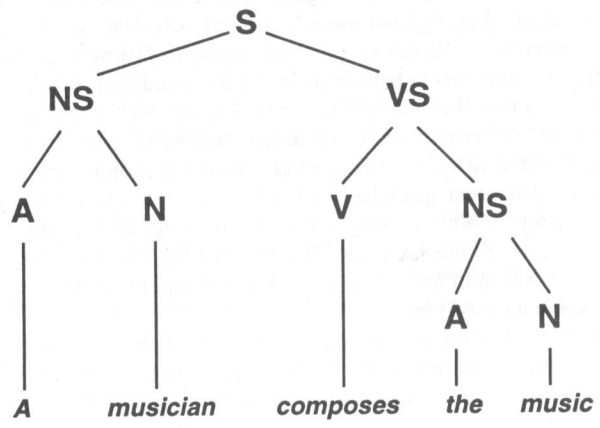
The notion of formal grammars is one of the most popular, but also controversial, notion that has sprung from AI research to fertilize the grounds of these flourishing new theories of music (Cope 1991). Formal grammars appeared in the late 1950s when linguist Noam Chomsky published his revolutionary book *Syntactic Structures* (Chomsky 1957). In general, Chomsky suggested that people are able to speak and understand a language mostly because they have mastered its grammar. According to Chomsky, the specification of a grammar must be based upon mathematical formalism in order to thoroughly describe its functioning; e.g., formal rules for description, generation and transformation of sentences. A grammar should then manage to characterise sentences objectively and without guesswork. Chomsky also believed that it should be possible to define a universal grammar, applicable to all languages.

The study of the relationship between spoken language and music is as old as the music of the Western culture. It is therefore not by accident that Music and AI research has been strongly influenced by linguistics and particularly by formal grammars. Many musicologists believed that Chomsky's assumptions could be similarly applied to music (Lerdhal and Jackendoff 1983; Cope 1987; Holtzman 1994). A substantial amount of work inspired by the general principles of structural description of sentences has been produced, including a variety of useful formal approaches to musical analysis; e.g., Schenkerian-like techniques (Cook 1987; Smoliar 1979;).

4.1. A Brief Introduction to Formal Grammars

Figure 1 below illustrates an example of a grammatical rule for a simple affirmative sentence: "A musician composes the music".

Figure 1: Example of a grammatical rule.



The above rule is saying that:

(a) **S = NS + VS** (a sentence **S** if formed by a noun-sentence **NS** and a verb-sentence **VS**)
(b) **NS = A + N** (a noun-sentence **NS** is formed by an article **A** and a noun **N**)
(c) **VS = V + NS** (a verb-sentence **VS** is formed by a verb **V** and a noun-sentence **NS**)

Such a rule can be programmed onto a computer in order to generate sentences automatically. In this case, the computer also must be furnished with some sort of lexicon of words to choose from. For example:

A = {the, a, an}
N = {dog, computer, music, musician, coffee}
V = {composes, makes, hears}

Note that the lexicon defines the three classes of words required by the rule: articles (**A**), nouns (**N**) and verbs (**V**). Given the above rule and lexicon, the computer can be activated to generate meaningful sentences, such as "A computer composes the music" or "A dog hears the musician", but also nonsense ones, such as "A musician composes the dog" and "A coffee hears the computer". The production of non-sense could be alleviated by defining tighter rules, but a notion of semantics would also be necessary for better results.

The rule above is called *generative* because it is used to generated sentences from scratch. *Transformational* rules work similarly to the generative ones. In this case, the computer is usually programmed to verify if the sentence to be transformed is syntactically correct. This verification is often done by simply matching the generative rule that would have generated the sentence. For example, a transformational rule to change the order of simple sentences could be defined as follows:

IF:

$S_{(o)} = NS_{(n)} + VS_{(m)}$ *and*
$NS_{(n)} = A_{(n)} + N_{(n)}$ *and*
$VS_{(m)} = V + NS_{(m)}$

THEN:

$S_{(t)} = NS_{(m)} + VS_{(n)}$ *and*
$NS_{(m)} = A_{(m)} + N_{(m)}$ *and*
$VS_{(n)} = NS_{(n)} + V$

In plain English, the rule above reads as follows: "*If the sentence to be transformed is composed of a first noun-sentence followed by a verb-sentence, and the first noun-sentence is formed by an article and a name, and the verb-sentence is formed by a verb and a second noun-sentence of the same format of the first noun sentence, then the transformed sentence will be formed by the second noun-sentence followed by a new verb-sentence composed of the first noun-sentence followed by the verb.*"

Applying this transformation rule to the sentence "A musician composes the music" will give the result "The music a composer composes". Punctuation could also be included in the rule in order to produce "The music, a composer composes."

Generative and transformational rules to generate and transform musical "sentences" can be similarly defined.

4.2. An Example of a Music Formal Grammar

In order to define a grammar for music one should carefully consider what the constituents of the rules will be; e.g., notes, phrases, melodies, chords, etc. For the purposes of the example below, we defined the constituents of our grammar in terms of 5 fundamental notions:

(a) the notion $\mathbf{R_n}$ of a reference note (e.g., $\mathbf{R_1 = C4}$)
(b) the notion of interval $\mathbf{I_n}$ between two notes (e.g., $\mathbf{I_1}$ = **perfect 5th**)
(c) the notion of direction $\mathbf{D_n}$ of the interval (e.g., $\mathbf{D_1}$ = **upwards**)
(d) the notion of sequence $\mathbf{SEQ_n}$
(e) the notion of simultaneity $\mathbf{SIM_n}$

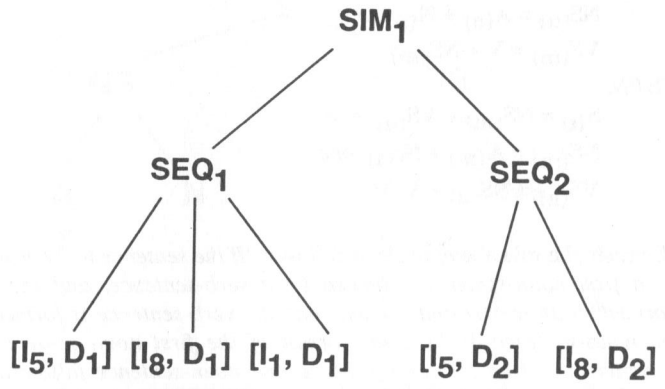
An example of a generation rule from our grammar is defined as follows (Figure 2):

(a) $SIM_1 = SEQ_1 + SEQ_2$

(b) $SEQ_1 = [I_5,D_1] + [I_8,D_1] + [I_{11},D_1]$

(c) $SEQ_2 = [I_5,D_2] + [I_8, D_2]$

Figure 2: An example of a music grammar.



The lexicon of our grammar inclues the following:

I = {minor 2nd, major 2nd, minor 3rd, major 3rd, perfect 4th, augmented 4th, perfect 5th, minor 6th, major 6th, minor 7th, major 7th, octave, none}

D= {upwards, downwards, none}

The pair $[I_5,D_1]$, for example, indicates that the interval is a **perfect 4th** (the fifth element of the set **I**) and the direction of the interval is **upwards** (the first element of the set **D**). Thus, the rule above reads as follows: "*A certain musical passage is composed of two sequences played simultaneously. One sequence is formed by 3 notes and the other is formed by 2 notes. The notes of the former sequence are calculated from a given reference note in this order: a perfect firth upwards, a minor 6th upwards and a major 7th upwards. The notes of the latter sequence are calculated, using the same reference note, in this order: a perfect 4th downwards and a minor 6th downwards*". By establishing that the reference point **R=C4**, the musical passage shown in Figure 3 can be generated:

Figure 3: Musical passage generated by the grammar.

Note that rhythm has not been included in our grammar in order to keep our examples as simple as possible .

A transformational rule for the above type of passage could, for example, create a new sequence SEQ_3 by joining the two sequences into a single simultaneous event ($SIM_2=SEQ_1+SEQ_2$), followed by the original first sequence SEQ_1(Figure 4):

IF:

$SIM_1 = SEQ_1 + SEQ_2$ *and*

$SEQ_1 = [I_5,D_1] + [I_8,D_1] + [I_{11},D_1]$ *and*

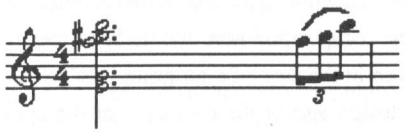
$SEQ_2 = [I_5,D_2] + [I_8, D_2]$ *and*

THEN:

$SEQ_3 = SIM_2 + SEQ_1$ *and*

$SIM_2 = SEQ_1 + SEQ_2$

Figure 4: An example of a musical transformation.



The computer could be programmed to produce an entire musical composition by successive activation of a variety generative and transformational rules (Figure 5):

Figure 5: Musical material produced by successive activation of rules.

The music produced by computer programs of this kind is usually labelled as *algorithmic composition*. Algorithmic composition, however, does not necessarily mean composition produced by grammars, but all music produced automatically by any kind of computer program. Most composers working with computers nowadays tend to be cautious with such labelling.

5. Who composed *Entre l'Absurde et le Mystère*?

Entre l'Absurde et le Mystère is a piece for chamber orchestra produced by CAMUS, a computer system designed by the author (Miranda 1993; 1994). CAMUS uses cellular automata-based simulations of biological behaviour to produce sequences of music structures (e.g., melodies, chords, clusters, etc.).

The public warmly applauded its performance by The Chamber Group of Scotland in 1995 in Edinburgh[2]. Martyn Brabbins, the conductor, was reluctant to believe that a computer had generated the piece and generally members of the audience found that the piece was pleasant. The general wonder of that evening was: "Was the piece really composed by a computer?"

This question is debatable and has serious ideological implications. In our point of view, a distinction between author and meta-author should be made in such cases. The ultimate authorship of the composition here should be to the person who designed and/or operated the system. Even in the case of a program that has the ability to program itself, someone is behind the design and/or the operation of the system. Similarly, one would hardly consider that Pierre Boulez's *Polyphonie X* (Nattiez 1993), for example, was composed by the serialism system.

[2] The first Brazilian performance of this piece was in Porto Alegre, in October 1995, by Orquestra Sinfonica de Porto Alegre (OSPA).

6. Semantics

Formal grammars are suitable for the description of the syntactical rules of a language but they do not guarantee meaningful formations. If one wishes to program a computer to produce meaningful sentences automatically, some notion of semantics must be included in the system. Although semantics is a concept primarily related to spoken languages, it also applies to music to a certain extent; e.g., musical semiotics.

Semantics, often referred in AI jargon as *declarative knowledge*, is a crucial aspect of AI research. A substantial amount of research work is dedicated to the design of methods to represent knowledge effectively (Brachman and Levesque 1985).

7. Knowledge Representation

Designers of AI systems require knowledge representation techniques that provide representational power and modularity. They must capture the knowledge needed for the system and provide a framework to assist the systems designer to easily organize this knowledge (Bench-Capon 1990; Luger and Stubblefield 1989).

7.1 The internal representation hypothesis

The primary assumption in AI is that mental activity is mediated by internal representations. Although there is no consensus as to what these representations actually are (some regard them as neurophysiological states, whilst others may define them as symbols or even images). The traditional approach to AI assumes that intelligent activity is achieved through:

(a) the use of symbols to represent a problem domain
(b) the use of these symbols to generate potential solutions to problems
(c) the selection of a suitable solution to a problem.

The use of an adequate knowledge representation technique is therefore one of the most important keys for the design of successful AI systems.

7.2 A brief survey of knowledge representation paradigms

7.2.1 Logic representation: first-order predicate calculus

A number of logics have been developed in philosophy and mathematics to represent knowledge; for example propositional calculus and first-order predicate calculus. The first-order predicate calculus is largely used in AI systems.

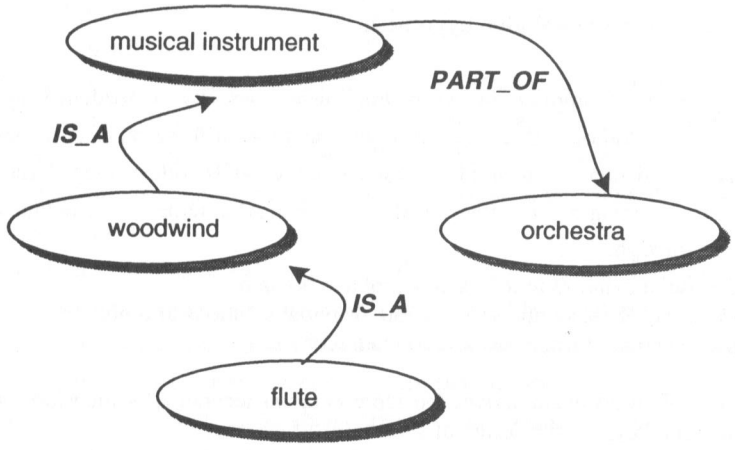
The first-order predicate calculus provides a well-defined language for describing and reasoning about qualitative aspects of a system. It can denote objects of a domain by using simple symbols and can express relationships between objects, assertions and denials of these relations, and logical relations between these statements (Luger and Stubblefield, 1989).

The first-order predicate calculus is sufficiently general to provide a foundation for other models of knowledge representation. AI problem domains however often require large amounts of highly structured interrelated knowledge. Some high-level notion of structure is needed to help the systems designer represent complex concepts in a coherent way. The first-order predicate calculus alone does not provide this help.

7.2.2 Network representation: graphs

A network representation also provides the means to denote objects of a domain and relations between them by using simple symbols. The advantage of network representations over logic representations is that the former can provide some high-level notion of structure that helps the systems designer to represent taxonomically structured information. The philosophy behind a network representation is that one reasons about a concept or object by relating it to other concepts or objects of the domain.

Figure 6: An example of a simple semantic network.



Graphs technique are an example of network representation; it provides a means to explicitly represent objects and relations by using nodes and arcs. A number of graphs techniques have been developed and used in AI systems; for example, conceptual graphs and semantic networks.

A semantic network, for instance, consists of a network of nodes linked by arcs, so that nodes represent concepts, or objects, and arcs represent relations between them; both nodes and arcs are usually labelled (Figure 6).

In Figure 6, there are 4 objects ("musical instrument", "orchestra", "woodwind" and "flute") and 2 types of relations ("*PART OF*" and "*IS A*"). This semantic network represents the following facts:

A musical instrument is part of the orchestra.
A woodwind is a musical instrument.
A flute is a woodwind.

Other facts can be inferred from the network, in addition to the facts which are explicitly represented. For example:

A flute is part of the orchestra.

One of the advantages of a graph-based representation is that facts come from the definition of links and associated inference rules that define specific mechanisms, such as the inheritance mechanism. In the case of the above example, "flute" inherited the fact that it is part of "orchestra".

In itself a graph-based notation of relationships is not so different from the first-order predicate calculus. The power of a network representation is that it provides an explicit method to represent objects and relations, and promotes the organization of knowledge into class hierarchies and the inheritance mechanism.

7.2.3. Structured representation: frames

Network representations allow for the representation of knowledge using explicit links between single objects in a knowledge base. Structured representations however extend network representations by providing a means to organize large networks of knowledge into a collection of separate networks, each of which represents some stereotyped situation or class of objects.

Frames technique is a type of structured representation. Frames technique allows for the representation of complex structures by encapsulating multiple attributes of situations, or objects, into single units, or classes of objects, in the domain.

A frame is a data structure whose components are called slots. Slots have names and accommodate various types of information: a value, a link to other frames or procedures to calculate its value. A slot may also be left incomplete.

As in semantic networks, the most useful feature of frames is the inheritance mechanism; when a frame represents a class of objects and another frame represents a superclass of this class, then the class frame inherits from the superclass the values for its incomplete slots. Examples of frames:

FRAME: *musical instrument*
part_of: *orchestra*

FRAME: *woodwind*
is_a: *musical instrument*
excitation_method: *air stream*
resonance_method: *pipe*

Note that slots are similar to the arcs of a network representation. Slots however have the advantage that they can hold procedures to perform some function, in addition to links to other concepts.

Structured representations thus extend network representations by representing complex objects as interconnected structured single entities, rather than as one single large network.

8. Conclusion

Mathematics and logics undoubtedly play a dominant role in the formalisation of intelligence for AI research. But is formalization the right approach to express intelligent behaviour? Is it right to distinguish between mind and body, semantics and syntactic, knowledge and abstract representation schemes?

The great majority of AI work to date assume that intelligence can be simulated by encapsulating chunks of data into static "packets" of information. Intelligent activity is then performed by an "engine" that picks and combines appropriate packets of information stored in memory in order to achieve specific goals. In this case, knowledge is often classified into two main groups: *declarative* (e.g., the semantics of the grammar or the "meaning" of the packets of information) and *procedural* (e.g., the grammar itself or the how the engine should function).

In fact humans store knowledge in a more complex way. Our brains are stubborn systems which cannot be deconstructed so neatly. Human intelligence is formed by both conscious and unconscious elements distributed at different levels of layers in our mind. Only the conscious ones can be objectively accessed and manipulated. We seem to not have access to the levels which do most of our thinking. When we think, we certainly change our own rules and rules that changes the rules, and so on, but we cannot change the lower layers; i.e., neurons always function in the same way.

Modern approaches to AI seek inspiration from this neurophisiological model. Traditional AI research methods are not necessarily inspired by neurophysiology but have, nevertheless, produced fruitful results. Perhaps the most fruitful of them all is the conclusion that there are many types of intelligence and each have their own characteristics. Music certainly involves a very distinct type of intelligence and it is up to Music and AI researchers to find the right approaches to it.

9. References

Balaban, M., Ebicioglu, K. and Laske, O. (Editors), 1992, Understanding Music with AI: Perspectives on Music Cognition, Menlo Park: AAAI Press/MIT Press.

Bench-Capon, T.J.M., 1990, Knowledge Representation: An Approach to Artificial Intelligence, London: Academic Press.

Brachman, R.J., and Levesque, H.J. (Editors), 1985, Reading in Knowledge Representation, San Mateo: Morgan Kaufmann Publishers.

Chomsky , N., 1957, Syntactic Structures, The Hague: Mouton. Cook, N., 1987, A Guide to Musical Analysis, London: J.M. Dent & Sons Ltd.

Cope, D., 1991, Computers and Musical Style, Oxford: Oxford University Press.

Cope, D., 1987, An Expert System for Computer-assisted Composition, Computer Music Journal, (11)4: 30-46. MIT Press.

Holtzman, S.R., 1994, Digital Mantras, Cambridge/MA: MIT Press.

Lerdhal, F. and Jackendoff, R., 1983, A Generative Theory of Tonal Music, Cambridge/MA: MIT Press.

Leveson, T., 1994, Measure for Measure: A Musical History of Science, New York: Simon & Schulster.

Luger, G.F. and Stubblefield, W.A., 1989, Artificial Intelligence and the Design of Expert Systems, Redwood City: Benjamin/Cummings.

Miranda, E.R., 1993, Cellular Automata Music: An Interdisciplinary Project, Interface 22:3-21.

Miranda, E.R., 1994, Music Composition Using Cellular Automata, Languages of Design 2:105-117.

Minsky, M. 1985, The Society of Mind, New York:Simon & Schuster.

Nattiez, J-J., 1993, The Boulez-Cage Correspondence, Cambridge: Cambridge University Press.

Robertson, P., 1996, Music and The Mind, television production for Channel 4 first shown May 1996, UK.

Smoliar, S., 1979, Computer Aid for Schenkerian Analysis, Communications of the ACM, 1979:110-115.

Storr, A., 1993, Music & The Mind, London: Harper Collins Publishers.

Audio Workshop, a program for audio synthesis and processing

Victor E P Lazzarini
Nucleo de Musica Contemporanea
Departamento de Arte
Universidade Estadual de Londrina
e-mail: Victor.Lazzarini@lda.palm.com.br

ABSTRACT

This paper describes AUDIOWORKS, a synthesis and processing program, for the Win32 platform. This software was developed with a twofold purpose: to create a simple environment where the beginner can explore basic audio synthesis and processing procedures and to provide the basis for the development of a larger project: the creation of an object library for sound manipulation. Sound synthesis and processing are normally taken up by larger packages like Csound and Cmusic, which offer a comprehensive set of tools for the user. However, packages such as these are, in a general sense, difficult to grasp by the beginner. Although AUDIOWORKS initially offers a limited number of synthesis and processing options, it approaches the subject in a more simple and intuitive way. The creation of a basic set of objects to deal with sound processing is a long-term project that aims to provide a cross-platform C++ class library for users to create their own custom-built applications. The development of the AUDIOWORKS code provides some guidance for the realisation of that project.

I - Introduction:

Sound synthesis and processing programs, such as Music V (Mathews, 1969), Cmusic (Moore, 1990) and Csound (Vercoe 1992), are basic elements of contemporary electroacoustic music practice. They provide the tools for the creation of sounds that is part of the activity of the composer involved with computer-generated music. Most of these systems were developed from a family of audio processing programs that are situated at the basis of the evolution of computer music (Dodge and Jerse, 1985), dating back to 1957 with the MUSIC I program, written by Max Mathews (Manning, 1993). They inherited many of the design features of the first packages, the most notable being the interface with the user. Although it provides a very flexible platform for the development of the synthesis procedures, it is, nevertheless, quite complex and remote for an inexperienced user.

These packages are usually referred to as sound compilers, because they involve processes that are similar to the coding and compilation of a program. Some of them, like Cmix, even involve the compilation of custom-built user programs. The usual steps of the user when dealing with such packages are: defining a code for the signal flow of the "instrument" used in the synthesis or processing, writing a second code to control these pre-defined parameters (a "score") and, using the suitable commands, interpreting these codes ("compiling" the sound). This process creates an output sound, which is normally stored in a file. The fact that the user can code almost any signal path possible by the program and control many different parameters of the process makes these packages very flexible and powerful. On the other hand, for the beginners, they represent a challenge that sometimes can discourage their study of computer music.

The development of a small program that aimed to provide a smoother introduction to sound synthesis and processing formed the foundation of AUDIOWORKS. It originated as simple synthesis program as a means of teaching basic concepts of computer music. The necessity for the development of tools directed to introduce computer music practice to non-initiated composers and students is clear. However, there is little point on a research that leads to a Csound or Cmusic clone. AUDIOWORKS is being developed to provide a

platform where the beginner can work in a more intuitive way. It is a work-in-progress project, being constantly expanded and revised. At the present state, it can provide alone the user with the means to produce a basic piece of computer-generated music. It works, in a non-realtime manner, with sound stored in files, as this is the typical way in which the more advanced sound manipulation packages function. A simple intuitive graphic interface, with menus and dialog boxes **(figure 1)**, provides the communication with the user. It also has a simple sound playing function that can perform the soundfiles created by the program.

AUDIOWORKS has been succesfully used by the author as an aid to teaching computer music practice and composition in the Music Technology course offered at the Universidade Estadual de Londrina. It has proved its potential as a comprehensive beginner's tool for computer music composition. Also, the development of the program has been shaped by its application and by the feedback received from the students.

II - Basic characteristics of the program

AUDIOWORKS is a Win32 platform application. It can run under Windows95 and WindowsNT operating systems. Its graphic interface consists of common Windows features, such as menus and dialog boxes. Other aspects of the graphic interface are still under-developed as the main work was done on the synthesis and processing capabilities of the software. The use of some of the features of the graphic interface provided by the Windows GUI rends the program simple to operate. It was an important point of the program to use a common environment to help the user understand the basic aspects of sound creation and manipulation.

The program works with sound stored in files of the type RIFF-WAVE, supported by the Microsoft Windows API (Pope and Van Rossum, 1995), as it is the most common soundfile format used in that platform. There are plans to support other formats, such as AIFF. The program can read and write mono or stereo files, of any sample rate, with 16-bit precision only. No support for 8-bit files is provided, since this program is aimed at high-quality audio production.

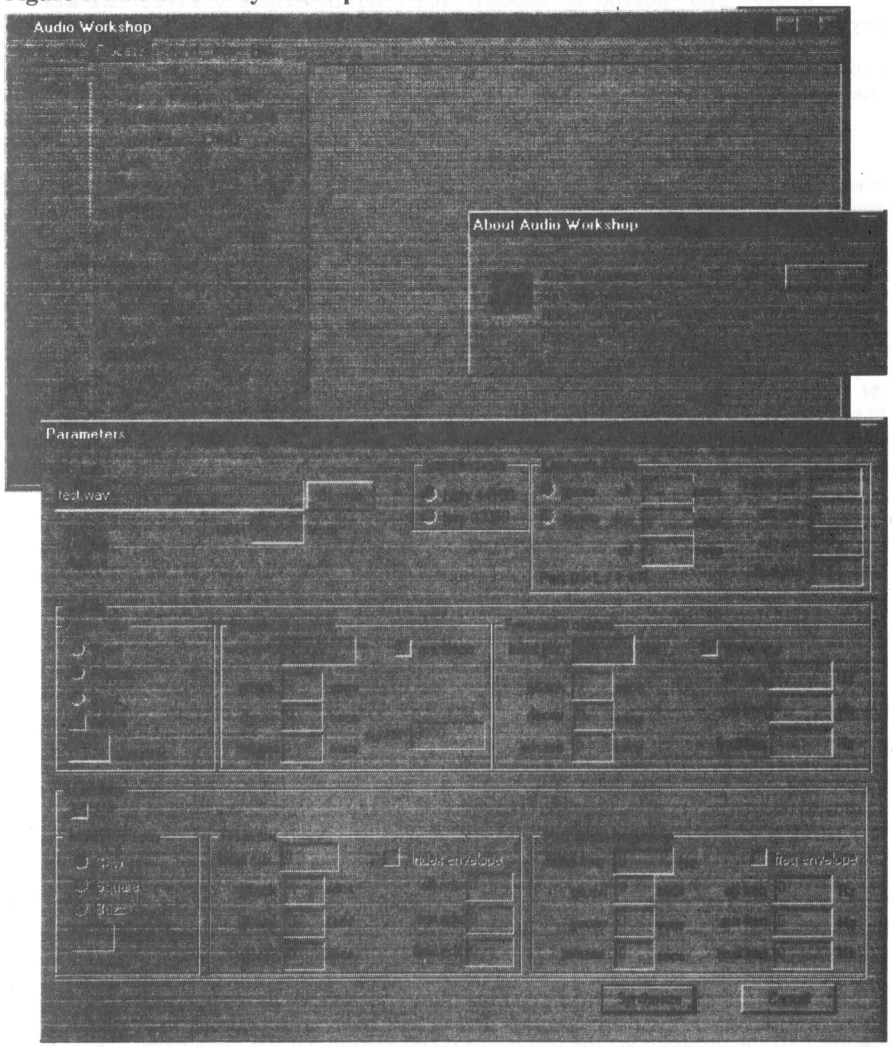
There are three basic sections in the program: synthesis, processing and soundfile utilities. The first section creates a soundfile by simple table and FM synthesis, whereas the second can process and modify input soundfiles. The soundfile utilities available are: performance and properties display. A help file describing the program functionality is available.

III - The synthesis section

This section can be accessed by the user via the synthesize menu. This comprises four items: parameters (Ctrl+P), save parameters (Alt+S), load parameters (Alt+L) and exit (Ctrl+X). The parameters option opens the synthesis parameters dialog box. The user can then fill the box with his choices and press the synthesize button. A summary of the synthesis parameters follows:

Filename: name of the soundfile to be created, of type RIFF-wave(*.wav), 16-bit precision. Overwrites existing files. Browse button opens the filestore for searching.
Duration: duration of the soundfile to be created, in seconds.
Sampling rate: option of high and low SR.

Figure 1. Audioworks synthesis parameters window and Process menu.



Channels: option of creating a mono or stereo file. The stereo option opens the pan envelope.
Waveform: option of saw, square or buzz waves, with up to 99 harmonics for both carrier and modulator (independently).
Alternatively, a noise generator can be employed, by checking the appropriate box.
Amplitude control: the user has two options, to create a steady-amplitude sound or to use the ADSR envelope

Max amp: in the first case, simply the soundfile's amplitude, in the range of 0 - 32000. In the second case, it is the amplitude after the attack period.

Envelope: check this box to use the ADSR envelope generator.

Attack: time period in seconds at the beginning of the soundfile, when the amplitude rises from zero to **max amp**.

Decay: time period in seconds starting immediately after the **attack** period when the amplitude varies from **max amp** to **sustain**.

Sustain: amplitude of the sustain period, in the range of 0 - 32000, which starts at the end of the **decay** period and finishes before the **release** period.

Release: time period in seconds which the amplitude goes from **sustain** to zero. It is calculated backwards from the end of the file.

Frequency control: the user has two options, to create a steady-frequency sound or to use the envelope to vary the frequency in time. The envelope parameters of both carrier and modulator are similar to the amplitude envelope, with the addition of:

Initial freq: the oscillator initial frequency.

Att freq: the frequency reached after the attack period.

Final freq: the oscillator final frequency.

FM: check this box to modulate the main oscillator frequency with a second oscillator.

FM Index: as before, the user has two options, to create a steady-index frequency-modulated sound or to use the envelope to vary the modulation index in time. The envelope parameters are similar to the other envelope generators.

The user, therefore, has two main options for strategies of sound synthesis: simple wavetable or white noise sound generation and frequency-modulation. The generation of simple waveforms is made by a continuous sampling of a table filled with one cycle of a specified wave, with a specified number of harmonics (Moore, 1990; Dodge and Jerse, 1985), whereas white noise is obtained by a pseudo-random function. Frequency modulation is extensively described elsewhere (Chowning, 1973). It defines the technique where one oscillator (the modulator) modulates the frequency of a second (the carrier). This is called a distortion technique, and is a mathematically efficient way of creating complex sounds. Its basic parameters are the ratio of frequencies of the two oscillators and an index of modulation, which is intuitively related to the amount of distortion or modulation injected in the carrier signal.

All the basic parameters are present for the user to experiment with simple synthesis algorithms. Frequency, amplitude and timbre can be easily accessed and varied, according to the user's will. A useful experience can be drawn from the manipulation of the parameters, which will be important for the beginner when facing advanced synthesis systems. The configuration of the synthesis parameters window can be saved and retrieved via an option that writes a proprietary file (*.par) containing that information. These files can be saved or loaded into the program via the respective commands in the synthesis menu. This option gives the user the possibility of building a personnal library of sound synthesis.

IV - The processing section

The process menu lets the user access the processing functions of the program. It has a number of options in the form of dialog boxes. They can also be called by particular key strokes:

Process Menu

1)Allpass filter (Ctrl+A)
2)Apply envelope (Ctrl+E)
3)Butterworth (Ctrl+B)
4)Comb filter (Ctrl+C)

5)Delay (Ctrl+D)
6)Extract (Ctrl+E)
7)Filterbank (Ctrl+F)
8)Flanger (Ctrl+G)
9)Interleave (Ctrl+I)
10)Loop (Ctrl+L)
11)Pan (Ctrl+N)
12)Pitch Shifter (Ctrl+H)
13)Reson (Ctrl+R)
14)Reverse (Ctrl+V)
15)Splice (Ctrl+S)

The processing section covers the main areas of soundfile manipulation: amplitude modification, filtering, mixing, splicing, reversing, panning and delay/reverberation. The amplitude modifier is a straightforward four-stage envelope, which can be applied to an input soundfile. The simple mixer lets the user mix two soundfiles of the same type, with individual gain control.

The filters employed for the time-varying resonator (reson) and filterbank are of the feedback type. They are standard two-pole infinite impulse response (IIR) filters described by the general equation:

$$y(n) = a_0x(n) - b_1y(n-1) - b_2y(n-2)$$

Both filters have controls for the half-power bandwidth, centre frequency and gain. The resonator incorporates a four-stage centre frequency envelope. The filterbank can employ up to ten parallel fixed filters with individual controls. The butterworth filters are second-order filters of that particular design, which has a maximally flat pass-band (Dodge and Jersey, 1985). The program offers an option of high-, low- and band-pass, as well as band-reject responses, with a time-varying centre frequency.

The allpass and comb filters pass the soundfile through a delay line in the usual configurations, with delay time and gain controls. The flanger employs a variable delay line modulated by a LFO, with modulation, LFO rate, feedforward and feedback gain controls. The delay function passes the soundfile through a number of delays of any specified length. There is a reversing option which simply writes the soundfile backwards.

The extract function lets the user extract channels from multichannel files and the interleave mixes two mono soundfiles as a single stereo. The splice option joins together two soundfiles with a user specified crossfade time. The loop function creates looped repetitions of a user specified length. Other processing operations will constantly be added to this section, including spectral analysis and resynthesis. With the present functionality, the user can produce small works with soundfiles created and processed by the program alone.

The separation of synthesis and processing sections mirrors one common feature of the first (analog) synthesizers, where oscillators and modifiers were separate sections of a machine (Naumann and Wagoner, 1985). This is a very good didactic characteristic of the voltage-controlled synthesizer, which lets the user manipulate the sound in different compartments. Subsequently, one can have a better understanding of the processes involved. The processing section is very comprehensive and flexible. Thus, it offers a set of tools that are easy to manipulate, although powerful enough to let the user experiment with different

techniques of sound modification.

V - Soundfile playing and properties display

Using the menus Sound and Info, the user can perform the soundfiles created and, in addition, display some information on any soundfile:

Sound Menu

1) Play

Plays the last processed or synthesized soundfile.

2) Play soundfile

Opens the play soundfile dialog box, which prompts the user the name of the soundfile to be played.

Info Menu

1) Properties:

Opens the properties dialog box that shows general information on the soundfile.

These two utilities add some functionality to the program, making it possible for the user to immediately hear the results of his manipulations and obtain some information, such as number of channels, sampling rate, etc. on any soundfile of the RIFF-WAVE format.

VI - The Sound Manipulation Class Library project

A secondary goal of the AUDIOWORKS development is to provide the basis for a larger project to be developed at the Nucleo de Musica Contemporanea of the Universidade Estadual de Londrina. It is an interdisciplinary project to be developed jointly by the Arts, Computer Science and Mathematics department. It aims to build a library of objects to deal with sound manipulation, directed towards electroacoustic music. The final products of this project will be, on a lower level, C++ classes that could be employed by programmers and composers to build their own applications, and on a higher level, applications, such as AUDIOWORKS, ready to be employed in a compositional environment.

The study carried out for the development of AUDIOWORKS will be directed to the creation of objects that deal specifically with certain aspects of sound synthesis and/or processing. The modular nature of object-oriented languages, such as C++ (Stroustrup, 1991), makes them a tool flexible enough to create such objects. The inheritance and encapsulation characteristics of these languages will facilitate the user when dealing with a new problem, or when developing new objects.

VII - Conclusion

In this paper it was shown that sound synthesis and processing packages such as Csound and Cmusic represent a challenge for the beginner in computer music. They are advanced tools that require a certain basic knowledge of the subject prior to their usage. As compositional tools they are very efficient and research aiming to rival their functionality is therefore useless. On the other hand, a program that introduces basic computer music tools to the beginner is a real necessity. One of the aims of the AUDIOWORKS program was to fill that

gap, providing high-level tools whose employment could be easily understood by the inexperienced user. The program provides the means of synthesizing simple and complex sounds by wavetable and FM synthesis. It lets the user apply a comprehensive number of processing operations on the basic parameters of sound. Complementing its functionality, it can play soundfiles and display some information on them. AUDIOWORKS has been employed successfully by the author in his music technology course, as an aid to teaching computer music composition.

References:

Chowning, J (1973). "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation". *Journal of the Audio Engineering Society,* 21(7), 1973, pp.526-534.

Dodge, C & Jerse, T (1985). *Computer Music: Synthesis, Composition and Performance.* Schirmer Books, New York.

Manning, P (1993). *Electronic and Computer Music.* Oxford University Press, Oxford.

Mathews, M (1969). *The Technology of Computer Music.* MIT press, Cambridge, Mass..

Moore, F (1990). *Elements of Computer Music.* Prentice Hall, Englewood Cliffs.

Naumann, J & Wagoner, J D (1985). *Analog Electronic Music Techniques.* Prentice Hall, Englewood Cliffs.

Pope, S T & Van Rossum, G (1995). "Machine Tongues XVIII: A Child's Garden of Sound File Formats". Computer Music Journal, 19(1), 1995, pp. 25-63.

Stroustrup, B (1991). *The C++ Programming Language.* Addison-Wesley Publishing Co., Reading, Mass..

Vercoe, B (1992). *Csound, A Manual for the Audio Processing System.* MIT, Cambridge, Mass..

Acknowledgements

The author would like to thank CAPES for the support of the research that lead to the development of this program and CNPq who are currently funding the author's stay at the Universidade Estadual de Londrina.

WAV2MID - Conversor de melodias no formato WAVE para o formato MIDI

MÁRCIO DA COSTA PEREIRA BRANDÃO, RICARDO STACIARINI PUTTINI,
LUIS ANTÔNIO BRASIL KOWADA e CARLOS ANTÔNIO JORGE LOUREIRO
brandao@cic.unb.br, puttini@guarany.unb.br,
kowada@ime.usp.br e guto@linf.unb.br
Laboratório de Processamento Espectral
Departamento de Ciência da Computação
Universidade de Brasília, Brasília, DF
CEP 70910-900 BRASIL

Resumo

Este trabalho apresenta uma proposta para a transcrição de uma melodia armazenada em um arquivo de áudio no formato WAVE, cantada ou executada em um instrumento, para um conjunto de comandos indicando a execução das notas musicais da melodia em um arquivo no formato MIDI.

O processamento realizado consiste na detecção da frequência fundamental (*pitch*[1]) e na delimitação temporal das notas musicais. Para a detecção do *pitch*, foi desenvolvido um algoritmo baseado na formulação de Máxima Verossimilhança (*Maximum Likelihood*). A eliminação de erros grosseiros no conjunto de *pitches* estimados é feita utilizando um filtro da mediana de ordem cinco. A delimitação do início e duração de cada nota é feita a partir de um estimador de curto termo para a energia do sinal. Esta análise também é utilizada na determinação das pausas.

Abstract

This paper presents a transcription algorithm from a melody stored on an WAVE audio file, sung or played by a musical instrument, to a set of commands showing the execution of musical notes to be stored on a MIDI file.

The signal processing consists of fundamental frequency (*pitch*) detection and temporal delimitation of the musical notes. An algorithm for pitch detection based on a Maximum Likelihood formulation is developed. Gross errors on the set of estimated pitches are eliminated by means of a median filter of 5th order. The start and duration of each note and the pauses are identified through an energy short-term estimator.

[1] A origem do termo *pitch* está associada a uma característica perceptual. Todavia, atualmente está consagrado seu uso como sinônimo de frequência fundamental.

1. Introdução

As duas formas mais usuais de se armazenar uma música digitalmente consistem na amostragem temporal do sinal sonoro relativo à musica ou no armazenamento das instruções relativas a uma linguagem musical, tal como uma partitura. Este segundo modo é largamente utilizado pelos músicos e compositores, devido às diversas vantagens que ele oferece [MMA 96]. A principal delas é a flexibilidade em alterar os elementos da música, tais como compasso, ritmo, instrumento e valores, além de permitir a visualização da peça na forma de uma partitura convencional. A geração deste tipo de arquivo a partir da execução da música é feita através de um dispositivo de interface, que liga o instrumento musical ao computador. Por este dispositivo são enviados os comandos relativos às ações realizadas sobre o instrumento. A comunicação entre instrumentos musicais de diferentes fabricantes e modelos foi possível graças ao estabelecimento do protocolo MIDI (*Musical Instrument Digital Interface)* em 1982, que definiu o modo de comunicação entre instrumentos musicais, e da posterior padronização no formato de arquivos utilizados no armazenamento destas informações [MMA 96].

Estas facilidades enumeradas acima, atualmente, estão disponíveis apenas para o usuário que possui um instrumento com um dispositivo apropriado para comunicação com outros instrumentos musicais, utilizando o protocolo MIDI. Mas esta interface ainda não está disponível para determinados tipos de instrumentos. O caso em que isto se torna mais evidente ocorre quando a música é cantada por vozes humanas.

Esse trabalho tem por objetivo a geração de um arquivo no formato MIDI a partir de instrumentos que não possuam interface de comunicação MIDI. Mais especificamente, deseja-se converter uma melodia, cujo sinal é gravado em um arquivo por amostragem temporal, para o formato de arquivos MIDI, que contém uma seqüência de comandos indicando, entre outros, as execuções de notas musicais. O tipo de arquivo de entrada escolhido é o padrão WAVE por ser um formato popularmente utilizado para arquivos de áudio.

Essa conversão é realizada com a utilização de técnicas de processamento digital de sinais. Basicamente, o sinal de áudio digital (melodia digitalizada) é dividido em blocos que são processados separadamente. Para cada bloco são estimadas as frequências presentes e respectivas energias. Essas informações são então agrupadas e utilizadas na identificação da altura musical das notas (*pitch*) e pausas (silêncio) bem como a duração das mesmas. O procedimento para detecção de *pitch* é discutido na seção seguinte. Em seguida são discutidas a identificação de pausas, notas e suas respectivas durações bem como a geração do arquivo MIDI correspondente. Os resultados obtidos para um grupo representativo de instrumentos musicais são então apresentados.

2. Estimativa da Frequência Fundamental

Para que seja possível efetuar a conversão proposta, necessita-se obter do sinal amostrado a altura musical percebida pelo homem. Neste trabalho, considera-se a altura musical correspondendo a frequência fundamental (*pitch*) do sinal sonoro em questão. Apesar da detecção automática do *pitch* ser um campo de pesquisa relativamente antigo, não há ainda algoritmos que sejam completamente confiáveis e simples

computacionalmente [Deller 1993]. Em [Rabiner 1976], pode-se encontrar um estudo comparativo de diversos tipos de algoritmos para este fim. Os principais algoritmos, desenvolvidos desde meados da década de 60, são projetados especialmente para sinais de voz e utilizam *a priori* características especificas desse tipo de sinal, como por exemplo a independência e separabilidade entre a fonte de excitação e a resposta do trato vocal [Deller 1993]. Essas características geralmente não são válidas para muitos instrumentos musicais [Laroche 1994]. Além disso, os algoritmos para estimação do *pitch* de sinais de voz (falada) são projetados para operar em uma faixa de frequências inferior a três oitavas, enquanto um estimador para aplicações musicais deve apresentar bons resultados em uma faixa de, pelo menos, seis oitavas. Uma proposta de estimador de *pitch* para sinais gerados por instrumentos musicais, operando em uma faixa de frequências de 40 Hz a 2500 Hz pode ser encontrada em [Tucker 1978].

O algoritmo utilizado no presente trabalho consiste numa adaptação da abordagem de [Wise 1976], fundamentada na formulação de Máxima Verossimilhança, onde o período fundamental é estimado para um bloco de amostras r_j de comprimento K_0 ($0 \le j < K_0$), procurando-se o máximo da função g(P):

$$\hat{P} = \underset{P_{min} \le P \le P_{max}}{\arg max} \left[g(P) = \frac{1}{N} \sum_{l=1}^{N-1} \phi_R(lP) \right] \quad (1)$$

onde:

P_{min} e P_{max} correspondem aos valores de procura mínimo e máximo.

$N = \lfloor K_0 / P \rfloor$[2], corresponde ao número de termos do somatório.

$\phi_R(k)$ é uma estimativa de curto termo para a função de autocorrelação (fac):

$$\phi_R(k) = \sum_{j=0}^{K_0-1-k} r_j r_{j+k} \quad (2)$$

Um dos principais problemas apresentado pelo algoritmo de Wise consiste na decisão por valores múltiplos do período fundamental. Esse fato se torna ainda mais crítico quando se amplia a faixa de frequências para operação do algoritmo. Na formulação aqui proposta, aplica-se um fator de ponderação que decai exponencialmente a cada termo do somatório (1), resultando em um peso maior para os primeiros termos do somatório. Neste caso, o fator pelo qual a soma deve ser dividida corresponde à soma dos pesos:

$$\hat{P} = \underset{P_{MIN} \le P \le P_{max}}{\arg max} \left[g(P) = \frac{1-\beta}{1-\beta^{N-1}} \sum_{l=1}^{N-1} \beta^{l-1} \phi_R(lP) \right] \quad (3)$$

Para valores adequados de β ($0{,}6 < \beta < 0{,}9$), verificou-se uma melhora considerável no estimador, reduzindo-se significativamente a ocorrência de decisão por múltiplos do período fundamental.

---

[2] $\lfloor X \rfloor$ indica o maior inteiro menor do que X.

Para sinais periódicos, a função de autocorrelação apresenta um máximo local no valor de defasamento correspondente ao período fundamental [Deller 1993]. Desse modo, a estimativa só será considerada válida se o valor da fac para o *pitch* estimado constituir um máximo local da mesma. Caso contrário, realiza-se uma nova procura descrita por (3) entre $2P_{min} \leq P \leq P_{max}$.

Devido à resolução temporal exigida pelo algoritmo, é recomendável que os sinais sejam amostrados com uma frequência (f_s) de, pelo menos, 22,05 KHz. Além disso, é possível obter-se um aumento da resolução temporal através de uma interpolação na fac estimada [Wise 1976].

Com essas modificações, o algoritmo mostrou-se suficientemente preciso para operação na faixa de frequências de 50Hz a 3600Hz. A figura 1 apresenta a forma de onda de um trecho de sinal sonoro e sua respectiva função de autocorrelação, usada na estimativa da frequência fundamental.

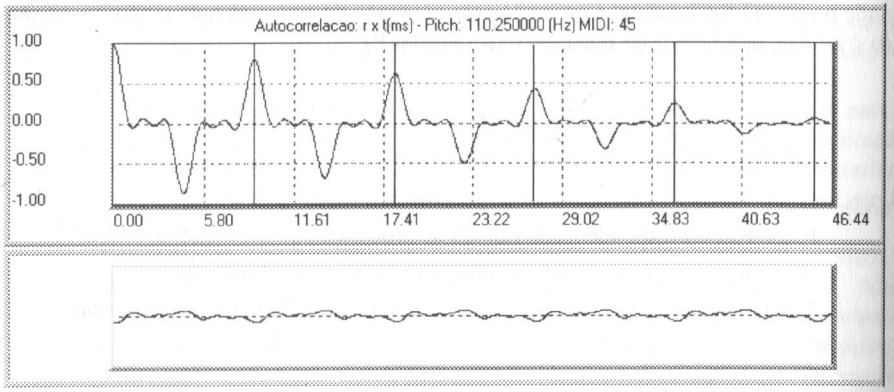


Figura 1: Função de autocorrelação e estimativa da frequência fundamental (em cima) e forma de onda para 46ms de um A3, f = 110Hz, produzido por uma clarineta.

A escala musical temperada é composta de 12 notas musicais por oitava, e o intervalo (razão de frequências) entre notas consecutivas é denominado semitom, correspondendo a uma razão de $2^{1/12}$. No formato MIDI, a primeira nota (f_0) possui a frequência fundamental de 8,1758 Hz. A frequência da n-ésima nota é dada por:

$$f_n = f_0 * 2^{n/12} \quad (4)$$

A partir da frequência estimada pelo algoritmo de máxima verossimilhança (f_{ML}), calcula-se o número da nota MIDI pelo inteiro mais próximo de n:

$$n = 12 * \log_2(f_{ML} / f_0) \quad (5)$$

onde:

$$f_{ML} = \frac{f_s}{\hat{P}} \quad (6)$$

3. Identificação de Pausas

Para a identificação de pausas, utiliza-se uma estimativa para a energia de cada bloco, dada por:

$$\overline{E} = \phi(0) = \sum_{j=0}^{K_0-1} r_j^2 \quad (7)$$

A energia de cada bloco é comparada com um limiar correspondente a uma fração da energia média do arquivo [Rabiner 1975 (II)]. O valor para este limiar adotado foi de 1% da energia média, correspondendo a uma relação sinal/ruído maior que 20dB. Desse modo, o bloco é classificado como pausa se sua energia estiver abaixo desse limiar e como nota, caso contrário.

4. Duração das Notas e Pausas

Para a identificação das notas, pausas e suas respectivas durações, a frequência fundamental e a energia estimadas para cada bloco são ordenadas. Erros isolados na estimativa do *pitch* podem ser eliminados comparando-se o valor estimado para o bloco com aqueles estimados para os blocos vizinhos. Neste trabalho é utilizado um filtro de mediana (valor central) de cinco pontos [Rabiner 1975 (I)]. A figura 2 ilustra o procedimento de eliminação de erro (suavização da lista de *pitches*).

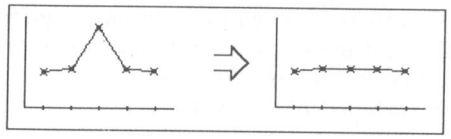


Figura 2: Eliminação de erro na estimativa da frequência fundamental

Primeiramente, as pausas são identificadas, a partir da lista de energias, sendo sua duração associada ao número de blocos de pausa consecutivos. Em seguida, a partir da lista de *pitches* suavizada, associa-se a duração de cada nota ao número de blocos consecutivos com o mesmo *pitch* estimado. Assim, cada mudança no valor de *pitch* estimado para blocos consecutivos identifica o término de uma nota e o início de uma outra.

Finalmente, resta analisar se dois elementos seguidos da lista de *pitches* com o mesmo valor correspondem à mesma nota tocada duas vezes, ou a uma nota tocada com um tempo mais prolongado. A transição entre duas notas é precedida do decaimento (*release*) da última nota e sucedida do ataque da nota seguinte. Para a maioria dos instrumentos musicais, a energia do sinal apresenta um decréscimo gradual durante a fase de *release* e um aumento na fase de ataque. Nestes casos, uma transição entre duas notas pode ser identificada por um vale pronunciado no contorno de energia do sinal (figura 3), estimada recursivamente a cada amostra por (8) [Deller 1993].

$$E_n = E_{n-1}(1-\tau) + r_n^2\tau, \quad 0.01 < \tau < 0.001 \quad (8)$$

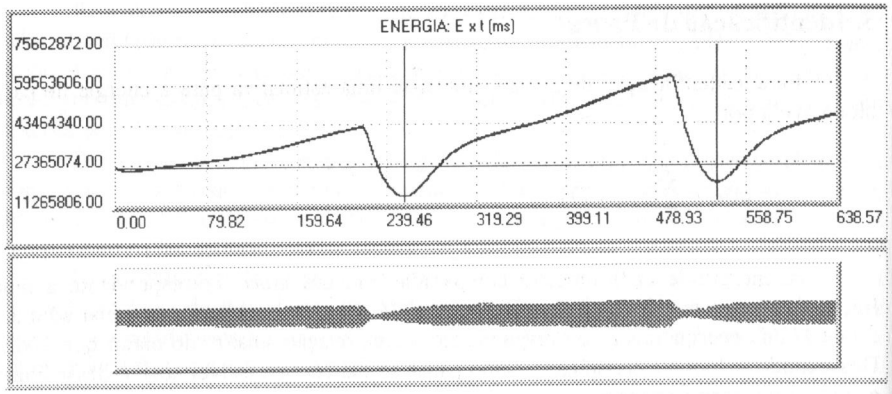


Figura 3: Estimador de curto termo para energia e detecção da transição entre notas

5. Montagem do Arquivo MIDI

A saída é dada num arquivo binário do tipo MIDI. Além do cabeçalho, há duas trilhas: uma com os valores referentes ao compasso e ao tempo, e outra com os dados relativos as notas.

Na trilha de dados, especifica-se o instrumento que reproduzirá a melodia, o qual pode ser definido pelo usuário ou utilizado um instrumento padrão. A escolha do instrumento é feita de acordo com a tabela de timbres definida pelo padrão General MIDI [MMA 1996]. O resto da trilha de dados contém a sequência de comandos MIDI (Note On e Note Off) que define o início e a duração das notas musicais. Não foram implementadas características auxiliares do som tais como: *pitch bend*, *key velocity* (sensibilidade ao toque) e *aftertouch* (pós-toque).

6. Resultados Obtidos

Os procedimentos descritos nas seções anteriores foram implementados em software para ambiente Windows, utilizando o compilador Borland C++ 4.5. A plataforma mínima consiste em um microcomputador PC-486 com placa de som SoundBlaster ou compatível.

No programa implementado, é disponibilizada uma saída sonora para o arquivo gerado (MIDI), que pode ser comparada com a saída sonora do arquivo contendo o sinal original (WAVE). Como apenas foi proposta a detecção da altura musical das notas e suas durações, esta comparação deve ser observada apenas do ponto de vista melódico e rítmico.

Além da avaliação subjetiva pela reprodução dos arquivos de entrada (WAVE) e saída (MIDI), foram realizados diversos testes, cujos resultados são apresentados a seguir:

- Escala Natural de Dó Maior em 5 oitavas: Dó 3 (nota MIDI nº 36, f = 65,4 Hz) ao Dó 8 (nota MIDI nº 84, f = 2.093Hz). A melodia da escala foi gerada por um dispositivo de síntese FM (OPL3 - Yamaha, timbre *Acoustic Grand Piano*) e amostrada a uma taxa de 44,1KHz. Todas as notas da escala foram identificadas com sucesso.
- Melodia de *Pour Elise* (L.V. Beethoven) executada por diversos instrumentos (sons amostrados de um teclado Roland W30) e cantada por uma voz masculina e outra feminina, todas amostradas a 44,1KHz. Os resultados são apresentados na tabela 1. Uma comparação entre o arquivo MIDI original usado na geração da melodia e o arquivo MIDI gerado pelo programa desenvolvido é apresentada no apêndice.

Timbre	Numero de Notas da Melodia	Numero de Notas Erradas
Piano	21	0
Flauta	21	0
Sax	21	0
Clarineta	21	0
Violino	21	1
OPL-3 (síntese FM)	21	0
Voz Feminina	12	0
Voz Masculina	12	1

Tabela 1: Resultado de testes com a melodia de *Pour Elise*

Os resultados aqui apresentados mostram a validade dos procedimentos discutidos. A maioria dos erros na estimativa da frequência fundamental ocorreram isoladamente, em geral nos blocos contendo transição entre duas notas diferentes. Esses erros foram completamente eliminados pela suavização aplicada à lista de *pitches*. Os erros que persistiram correspondem à estimativa de um múltiplo do período fundamental, evidenciando as limitações do algoritmo. Outras estratégias de eliminação para esse tipo de erro podem ainda ser testadas, com objetivo de corrigir o problema.

7. Conclusão

O presente trabalho apresentou uma proposta de um conversor de melodias no formato WAVE para arquivos de comandos no formato MIDI. Foi desenvolvido um algoritmo para detecção de frequência fundamental para uma grande faixa de frequências (50Hz a 3600Hz) e que permite sua utilização com sinais gerados por diversos instrumentos musicais e pela voz humana. Os resultados obtidos mostram a aplicabilidade das técnicas propostas.

Apêndice

A seguir é apresentada uma comparação entre o arquivo MIDI utilizado para geração da melodia amostrada (arquivo WAVE) e o arquivo MIDI gerado pelo programa desenvolvido. Ao se comparar as notas, pode-se observar que elas são idênticas em ambos os arquivos, exceto por pequenas diferenças na duração das mesmas, devido à quantização introduzida pela divisão do arquivo em blocos de tamanho fixo. Esse tipo de erro pode ser diminuído, utilizando-se um maior número de blocos (blocos superpostos, com amostras em comum).

Arquivo MIDI original		Arquivo MIDI gerado pelo Wav2Mid	
// teste.mid		// clarinet.mid	
MThd 00000006		MThd 00000006	
0001 0002 0078 //	format, trackcount, resolution	0001 0002 0060 //	format, trackcount, resolution
MTrk 00000019		MTrk 00000013	
0016: 00	FF 58 04 04 02 18 08 // tact	0016: 00	FF 58 04 04 02 18 08 // tact
001E: 00	FF 59 02 00 00 // key	001E: 00	FF 51 03 0B 10 08 // tempo
0024: 00	FF 51 03 09 27 C0 // tempo	0025: 00	FF 2F 00// end of track
002B: 00	FF 2F 00// end of track		
MTrk 00000088		MTrk 000000AF	
0037: 00	FF 21 01 00 // meta	0031: 00	C0 00 // program
003C: 00	90 4C 4C // note on	0034: 00	90 4C 7F // note on
0040: 3C	4C 00 // note on	0038: 24	90 4C 00 // note on
0043: 00	4B 3F // note on	003C: 00	90 4B 7F // note on
0046: 3C	4B 00 // note on	0040: 1E	90 4B 00 // note on
0049: 00	4C 45 // note on	0044: 00	90 4C 7F // note on
004C: 3C	4C 00 // note on	0048: 1E	90 4C 00 // note on
004F: 00	4B 47 // note on	004C: 00	90 4B 7F // note on
0052: 3C	4B 00 // note on	0050: 1E	90 4B 00 // note on
0055: 00	4C 42 // note on	0054: 00	90 4C 7F // note on
0058: 3C	4C 00 // note on	0058: 1E	90 4C 00 // note on
005B: 00	47 40 // note on	005C: 00	90 47 7F // note on
005E: 3C	47 00 // note on	0060: 1E	90 47 00 // note on
0061: 00	4A 3C // note on	0064: 00	90 4A 7F // note on
0064: 3C	4A 00 // note on	0068: 18	90 4A 00 // note on
0067: 00	48 38 // note on	006C: 00	90 48 7F // note on
006A: 3C	48 00 // note on	0070: 1E	90 48 00 // note on
006D: 00	45 38 // note on	0074: 00	90 45 7F // note on
0070: 3C	45 00 // note on	0078 :1E	90 45 00 // note on
0073: 00	2D 3E // note on	007C : 00	90 2D 7F // note on
0076: 3C	2D 00 // note on	0080: 1E	90 2D 00 // note on
0079: 00	34 3D // note on	0084:1E	90 34 7F // note on
007C: 3C	34 00 // note on	0088: 1E	90 34 00 // note on
007F: 00	39 38 // note on	008C: 00	90 39 7F // note on
0082: 3C	39 00 // note on	0090: 1E	90 39 00 // note on
0085: 00	3C 38 // note on	0094: 00	90 3C 7F // note on
0088: 3C	3C 00 // note on	0098: 1E	90 3C 00 // note on

Arquivo MIDI original (cont.)		Arquivo MIDI gerado pelo Wav2Mid	
008B: 00	40 45 // note on	009C: 00	90 40 7F // note on
008E: 3C	40 00 // note on	00A0: 1E	90 40 00 // note on
0091: 00	45 40 // note on	00A4: 00	90 45 7F // note on
0094: 3C	45 00 // note on	00A8: 18	90 45 00 // note on
0097: 00	48 32 // note on	00AC: 00	90 48 7F // note on
009A: 3C	48 00 // note on	00B0: 1E	90 48 00 // note on
009D: 00	4C 3A // note on	00B4: 00	90 4C 7F // note on
00A0: 3C	4C 00 // note on	00B8: 1E	90 4C 00 // note on
00A3: 00	51 37 // note on	00BC: 00	90 51 7F // note on
00A6: 3C	51 00 // note on	00C0: 1E	90 51 00 // note on
00A9: 00	54 3C // note on	00C4: 00	90 54 7F // note on
00AC: 3C	54 00 // note on	00C8: 1E	90 54 00 // note on
00AF: 00	58 43 // note on	00CC: 00	90 58 7F // note on
00B2: 3C	58 00 // note on	00D0: 1E	90 58 00 // note on
00B5: 00	4D 43 // note on	00D4: 00	90 5D 7F // note on
00B8: 78	5D 00 // note on	00D8: 3C	90 5D 00 // note on
00BB: 00FF	2F 00 // end of track	00DC: 00FF	2F 00 // end of track

Referências Bibliográficas

Deller JR, J. R., Proakis, J. G. & Hansen, J. H. L. (1993). *Discrete-Time Processing of Speech signals*. MacMillan Publishing Company.

Laroche, J. & Meiller, J. L. (1994). Multichannel Excitation / Filter Modeling of Percussive Sounds with Application to the Piano. *IEEE Transactions on Speech and Audio Processing*, vol. 2, No. 2, (Apr 1994), pp. 329-344.

MMA - MIDI Manufacturers Association (1996). *The Complete MIDI 1.0 Detailed Specification*. Los Angeles, CA.

Rabiner, L. R., Sambur, M. R. & Schmidt, C. E. (1975). Applications of a Nonlinear Smoothing Algorithm to Speech Processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-23, No. 6, (Dec 1975), pp. 552-557.

Rabiner, L. R. & Sambur, M. R. (1975). An Algorithm form Determining the Endpoints of Isolated Utterances. *The Bell System Technical Journal*, vol. 54, No. 2, (Feb 1975).

Rabiner, L. R., Cheng, M. J., Rosenberg A. E., & McGonegal, C. A. (1976). A comparative performance study of several pitch detection algorithms. *IEEE Transaction on Acoustics, Speech, and Signal Processing*, vol. ASSP-24, No. 5, (Oct 1976), pp. 399-418.

Roads, C. & Strawn, J., editors (1987). *Foundations of Computer Music - 3rd edition*, MIT Press, pp 114-144.

Tucker, W. H. (1978). A Pitch Estimation Algorithm for Speech and Music. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-26, No. 6, (Dec 1978), pp. 597-604.

Wise, J. D., Caprio, J. R. & Parks, T. W. (1976). Maximum Likelihood Pitch Estimation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-24, No. 5, (Oct 1976), pp. 418-423.

Restauração de Gravações Analógicas via Controle MIDI

MARCELO MOREIRA

Núcleo Interdisciplinar de Comunicação Sonora (NICS) /
Departamento de Comunicações (DECOM)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (UNICAMP)
mmoreira@decom.fee.unicamp.br

ABSTRACT

This paper reports the partial results obtained with the implementation of a new processing system employed for restoration of old degraded phonographic materials. Its main feature is to combine both classical technics of time-frequency analysis/synthesis with a MIDI-coded version of a musical score acting as a source of information for specification and control of local filtering processes applied over the musical program under processing.

Introdução

Originalmente desenvolvidos para a análise de padrões de voz, os métodos de análise/síntese em tempo-freqüência (TF) encontram um campo de aplicação natural no processamento de sinais provenientes de fontes sonoras musicais [Pielermeier *et al.*, 1996]: análise de parciais, modificação e síntese de padrões espectrais, transcrição automatizada de partituras, visualização, processos de compressão/expansão, técnicas de controle e remoção de ruídos [McGee *et al.*, 1991], [Desain *et al.*, 1992]. O sistema descrito a seguir procura explorar novas possibilidades de aplicação da Transformada de Fourier de Curto Prazo como técnica a ser empregada na remoção e tratamento do ruído presente em registros de material fonográfico deteriorado. Em particular, é explorada a possibilidade de utilização da partitura do programa musical processado como fonte de

informação e controle no processo de especificação e implementação de filtros locais adequados a cada segmento do sinal.

Análise STFT de Sinais Degradados

O formalismo da Transformada de Fourier convencional [Oppenheim *et al.*, 1989] é um instrumento apropriado para a abordagem de sinais periódicos, transientes ou mesmo aleatórios, desde que estatisticamente estacionários. No entanto, são amplamente conhecidas as limitações das técnicas clássicas quando aplicadas à análise e processamento de sinais não-estacionários. Nestes casos, os métodos de processamento em tempo-freqüência tornam-se a opção mais adequada no que diz respeito à manipulação de programas musicais (cuja estrutura está apoiada na possibilidade mesma de criação deliberada de variação sobre o material sonoro). Entre estes métodos, optou-se inicialmente pela denominada Transformada de Fourier de Curto Prazo Discretizada no Domínio Temporal (*Discrete Short-Time Fourier Transform*, STFT). A idéia subjacente [Rabiner *et al.*, 1978] é a do cálculo da transformada discreta de Fourier nas vizinhanças de cada amostra do sinal digitalizado. A partir daí, o conjunto das transformadas pode ser processado, dando origem, após síntese e conversão em sinal analógico, a um novo sinal processado. Este fato cria então a possibilidade de implementação de um processo adaptativo, adequado às características espectrais específicas de cada região de curto prazo [Moorer, 1986], [Arfib *et al.*, 1993], [Berndtson, 1996].

Restringindo-se inicialmente à análise STFT de gravações fonográficas em discos de vinil de 33 e 48 rpm, é possível observar a discriminação de padrões espectrográficos correspondentes ao programa musical frente a componentes de ruído provenientes da degradação do registro analógico. Observam-se padrões espectrais de faixa larga e de longa duração, em evidente contraste com padrões instrumentais de maior concentração de energia. Deste modo, e encarando a representação espectrográfica como uma imagem degradada, é intuitivamente concebível que a uma operação de restauração da imagem espectrográfica, deverá corresponder uma intervenção efetiva no plano sonoro. De fato, as extensas regiões da imagem espectral

dominadas essencialmente por ruído de aparência granular, indicam a necessidade de um primeiro processamento global cujo efeito deveria ser o de remover ou apagar estas grandes regiões. Isto sugere, em uma primeira etapa, a utilização de filtros passa-baixas direcionais, cuja implementação pode ser facilmente obtida a partir do formalismo da STFT (cf. fig. 1).

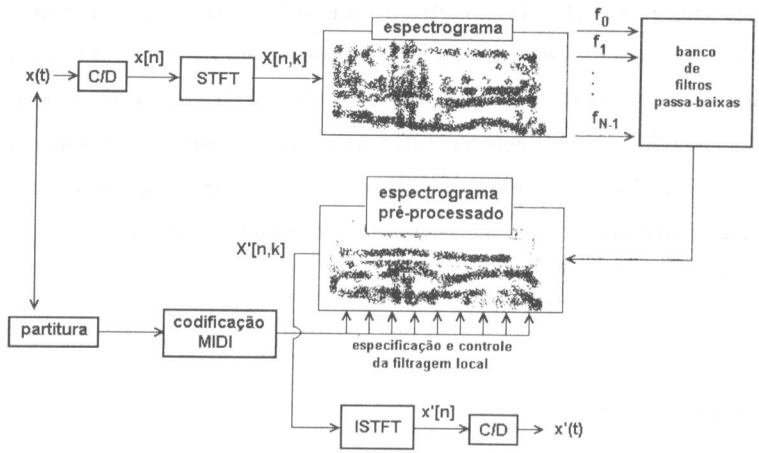


fig. 1 - Diagrama de Blocos do Sistema de Restauração via Controle MIDI

O Processamento Específico: Controle via Partitura

Realizado um tratamento global das componentes de ruído, impõe-se dividir a imagem em regiões de características uniformes e aplicar a cada uma destas seções um tratamento específico, adequado às particularidades de cada uma destes segmentos. As regiões correspondem a seções temporais da ordem de 10 a 30 ms de duração, nas quais o sinal pode ser assumido como localmente estacionário, encontrando-se próximas ao limite de resolução temporal do sistema auditivo humano [Moore, 1982].

Sobre cada uma destas seções deve ser aplicado um processamento adequado, baseado em filtros projetados segundo a análise do conteúdo harmônico de cada seção. Além de rotinas para reconhecimento de formantes, é possível utilizar-se como fonte de informação as indicações contidas na partitura correspondente ao programa musical sob

análise. De fato, este tipo de abordagem nada mais faz do que restabelecer a relação entre a representação quase-física fornecida pela análise espectral e os eventos abstratos fornecida por uma notação a ser decodificada pelo intérprete [Lesbros, 1996]. Então, para que o processo de interpretação se adeque ao processamento computacional, é necessário que este seja apresentado de forma convenientemente codificada. Em particular, a codificação MIDI satisfaz os vários do requisitos propostos para o projeto: fornecer indicações sobre localização, duração e intensidade dos diferentes eventos musicais. Aliada a métodos de detecção e rastreamento de componentes harmônicas comumente empregados em análise espectral de voz [Lim, 1983], [Maher, 1990], permite uma intervenção automatizada. Além da possibilidade de identificação e remoção automáticas de ruído, o projeto prevê também facilidades para eventual intervenção manual sobre o processamento, favorecendo processos de depuração e ajuste fino.

Resultados Parciais e Conclusão

Até a presente data, foram implementadas as rotinas STFT e um banco de filtros FIR projetados com o auxílio de janelas do tipo Kaiser (o que permite uma rápida manipulação da resposta dos filtros através da especificação de apenas três parâmetros). Encontram-se em andamento os estudos para adaptação do código MIDI para o controle dos filtros e os experimentos sobre a intervenção manual sobre o processo e a discriminação dos componentes de ruído residuais após o primeiro processamento global. Os resultados disponíveis no momento, obtidos com a utilização da abordagem tempo-freqüência, baseada na divisão do sinal em quatro bandas de áudio superpostas e na intervenção automatizada através da especificação de expansores em cada uma das bandas, tem revelado a presença de padrões de modulação que dão origem ao denominado "ruído musical". Este efeito pode ser parcialmente atenuado pelo ajuste mais cuidadoso do limiar e das durações do ataque dos expansores (*noise gate*); no que se refere ao último parâmetro, tem sido necessário respeitar o compromisso entre alguns efeitos de distorção harmônica introduzidos pela atribuição de uma duração de ataque mais curta e o aparecimento de *overshoots* devido a uma ação mais suave dos

expansores. Diferentes atribuições de valores de ganho mínimo para cada banda têm se mostrado eficazes para algumas gravações; porém, os melhores valores encontrados divergem de modo considerável em relação às especificações encontradas na literatura [Moorer *et al.*, 1986].

Na seqüência do projeto, pretende-se estudar ainda os efeitos e vantagens da aplicação de curvas de equalização fisiológicas sobre os processos de filtragem e da adoção do formalismo *wavelet* [Kronland-Martinet, 1988]. Em resumo, o sistema em desenvolvimento deve dar origem não somente a implementações mais sofisticadas, baseadas em ferramentas analíticas mais poderosas [Pielermeier *et al.*, 1996], como favorecer novas formas de interação entre partitura e espaço sonoro, fato desejável para o estudo de futuros métodos de síntese e composição musicais.

Bibliografia

[Arfib *et al.*, 1993] - Arfib, Daniel & Delprat, Nathalie, "Musical Transformations Using the Modification of Time-Frequency Images", *Computer Music Journal*, 17:2, pp. 66-72, Summer 1993.

[Berndtson, 1996] - Berndtson, Gunilla, "The KTH Rule System for Singing Synthesis", *Computer Music Journal*, 20:1, pp. 76-91, Spring 1996.

[Desain *et al.*, 1992] - Desain, Peter & Honing, Henkjan, "Time Functions Function Best as Functions of Multiple Times", *Computer Music Journal*, 16:2, pp. 17 - 34, Summer 1992.

[Kronland-Martinet, 1988] - Kronland-Martinet, Richard, "The Wavelet Transform for Analysis, Synthesis, and Processing of Speech and Music Sounds", *Computer Music Journal*, 12:4, pp. 11-20, Winter 1988.

[Lesbros, 1996] - Lesbros, Vincent, "From Images to Sounds, A Dual Representation", *Computer Music Journal*, 20:3, pp. 59-69, Fall 1996.

[Lim, 1983] - Lim, Jae S., ed, *Speech Enhancement*, Prentice Hall, Englewood Cliffs, New Jersey, 1983.

[Maher, 1990] - Maher, Robert C., "Evaluation of a Method for Separating Digitized Duet Signals", *Journal of the Audio Engeneering Society*, 38(12), pp. 956-979, 1990.

[McGee *et al.*, 1991] - McGee, W. E. & Merkley, Paul, "A Real-Time Logarithmic-Frequency Phase Vocoder", *Computer Music Journal*, 15:1, pp. 20-27, Spring 1991.

[Moore, 1982] - Moore, Brian C. J., *An Introduction to the Psychology of Hearing*, Academic Press, London, 1982.

[Moorer *et al.*, 1986] - Moorer, J.A. & Berger, M, "Linear-Phase Band Splitting: Theroy and Applications", *Journal of the Audio Engeneering Society*, 34(3), pp. 143-152, 1986.

[Oppenheim *et al.*, 1989] - Oppenheim, Alan V. & Schafer, Ronald W., *Discrete-Time Signal Processing*, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[Pielermeier *et al.*, 1996] - Pielermeier, William J.; Wakefield, Gregory H. & Simoni, Mary H., "Time-Frequency Analysis of Musical Signals", *Proceedings of the IEEE*, vol. 84, no. 9, september 1996.

[Rabiner *et al.*, 1978] - Rabiner, Lawrence. R. & Schafer, Ronald W., *Digital Processing of Speech Signals*, Prentice Hall, Englewood Cliffs, New Jersey, 1978.

Time-Frequency Distributions for Timbre Morphing: The Wigner Distribution versus the STFT

C. J. Hope and D. J. Furlong,
Dept. of Electronic and Electrical Eng.,
Trinity College Dublin,
Ireland
email: Dermot.Furlong@tcd.ie, cjhope@alf2.tcd.ie

Abstract

The objective of this paper is to investigate and compare the timbre representation potential of the STFT and the Wigner Distribution in the context of timbre morphing. Both Time-Frequency distributions can be interpreted as the energy in time and frequency of a signal. There are however, intrinsic differences between the distributions. The Wigner distribution of multi-component signals exhibits cross terms, while the STFT representation inherently involves smearing in both time and frequency directions. However, the Wigner distribution cross terms can be reduced through the use of an appropriate smoothing window, thereby providing the basis for improved timbre feature analysis and processing.

1 Introduction

Of the elements of musical composition, timbre has, at least until modern times, been the most neglected. The musical potential inherent in timbre manipulation has, of course, been realized for centuries but for the greater part this has been regarded as a subtlety best left to the performer as he/she sought to enhance their musical expression. Only with the development of electronic processing devices has timbre come to be regarded as a true compositional element which can be brought under specifiable control for musical effect. It is almost certainly the case that those compositions to date which rely heavily on timbre variations (*e.g.* works by Stockhausen, Boulez and others) for effect are but the warp and weft of a tapestry which has yet to be revealed, and that, as timbre processing becomes easily available to mainstream composers through the development of effective tools for 'timbre manipulation', it is likely that the utility and effectiveness of timbre as a medium for composition will become fully apparent. It might be suggested that our current situation regarding the development of tools for the exploitation of timbre as a compositional element may be compared to that of harmony and the development of effective keyboard instruments in the sixteenth century. With historical hindsight, it is clear that the increasing availability of keyboards at that time allowed relatively easy compositional development of 'vertical blocks' of sound, an aspect which was not so easily conceived or experimented with for the case of string or voice which, for the greater part, demanded ensemble performance of harmonies. As a consequence, the use of harmony as a compositional element 'flowered' following keyboard development.

With the arrival of electronic processing, timbre is now becoming accessible as a compositional element. Computer synthesis techniques allow both faithful mimicking

of so-called 'natural' tones and artificial generation of new sonorities. To date, the basis for timbre representation and manipulation has been what is referred to as the 'classical' concept of tone quality which relates timbre primarily to its spectral composition, *i.e.* to the pattern of harmonics inherent in different instrument tones. However, synthetic processing techniques clearly demonstrate the inadequacy of 'harmonic recipes' for timbre representation. For example, a piano note played backwards is perceived to have a very different tonal quality from the same note played in its normal sequence, although the spectral composition of both normal and reversed notes is identical. In short, there is now a very significant body of research evidence which clearly indicates the importance of *both* spectral and transient or temporal note characteristics for perceived tone. The significance of these findings for artificial synthesis and ultimately for composition is very noteworthy.

Firstly, it is becoming clear that traditional instruments allow exploration of a very small part of what might be called 'timbre space'. Secondly, it is evident that computer processing techniques, as are inherent in many digital synthesizers, are not being exploited to the full in that a comprehensive palette of timbres is not typically being made available to the composer/artist. A more mature approach to timbre processing would be based on what is referred to as 'joint time-frequency distributions' which make both transient and spectral information accessible. One example of such distributions is the Spectrogram, familiar to those working with speech processing. However, it can be shown that the Spectrogram is not an optimal joint time-frequency distribution as it introduces significant 'smearing' of temporal and spectral information which can distort, or even hide, both transient and spectral aspects which may be of musical significance. As a consequence of the realisation of the inadequacies of both the spectrum (*i.e.* harmonics) and the Spectrogram for timbre representation, attention has been brought to bear on the mathematical foundations of time-frequency distributions in an effort to identify an optimal distribution which would allow accurate representation of both temporal and spectral information.

As a result of such studies the Wigner Distribution, originally developed for joint distribution in Quantum Mechanics, has been applied to signal analysis and its superiority has been established for accurate time-frequency representation. In the context of the representation of musical timbres, the Wigner Distribution allows accurate analysis of both temporal and spectral details, and thereby facilitates synthetic processing of both transient and steady-state aspects of musical tones. This in turn opens up previously unexplored vistas in timbre space in that timbre manipulation is not confined to spectral processing. With an ability to 'see' the full detail of temporal and spectral composition of musical timbres comes the desire to manipulate these same details for the purposes of creative, musical expression.

One approach to the exploration of 'timbre space' is based on the possibilities inherent in 'morphing' between two chosen timbres. The concept of visual morphing is familiar to many from 'special effects' processing in cinema. A good example is where we are shown a young face gradually and smoothly age as it moves from youth to old age. This is achieved by using two frames - one young and the other old - and manipulating various 'perceptually significant' parameters so that the presented face appears to move from the youthful 'starting' frame to the aged 'finishing' frame. An exactly analogous process may be applied to musical timbres. For example, we may

choose a violin 'starting' timbre and a flute 'finishing' timbre and manipulate various important (spectral and temporal) features so that we can move smoothly between them. By so doing we open up new parts of timbre space which nevertheless are not entirely unknown to us in that we would at least be operating between the known boundaries of 'violin' and 'flute', or whatever other timbres we had chosen at the outset. Thus, morphing would provide the composer with access to a much richer 'timbre palette' than heretofore in that previously unexplored regions of timbre space could be made available for creative manipulation. By basing such morphing tools on Wigner Distributions of musical tones, rather than Spectrograms, representational distortions would be minimized and detailed spectral and temporal features of real instrument tones would be made much clearer for the purposes of computational analysis and synthesis.

When attempting to develop a morphing algorithm, the intrinsic problems involved in the analysis of timbre must first be considered. That is, the sounds to be morphed must first be analysed to determine those spectral and temporal features which characterise timbre. Then, in the implementation of timbre morphing, a number of problems must be addressed relating to the attempt to smoothly blend two, perhaps very different, 'feature sets'. The term *'feature'* is introduced by Holloway, Tellman and Haken [1] in their description of a timbre morphing algorithm. A feature is any part of a sound that contributes to its timbre. While, as stated previously, the classical position assigned timbre representation solely to the pattern of harmonic steady state amplitudes, the more modern position gives full acknowledgement to the importance of the details of the temporal evolution of individual harmonics. Consequently, the feature sets involved in timbre synthesis and morphing require extraction of both temporal and spectral details. Thus, the accurate extraction of these features from a given timbre representation is possibly the most important step in attaining an effective synthesis of a 'morphed' timbre. It is the purpose of this paper to present some preliminary results which indicate the superiority of the Wigner Distribution as a joint time-frequency representation which may be used as a basis for timbre morphing.

2 History of Timbre Representation

Fourier Analysis is a powerful technique for estimating the frequency content of many real signals, including speech and musical tones. The *Fourier Transform* (FT) of a non-stationary signal, f(t), is given in equation 1, which allows the generation of a signal spectrum

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-j2\pi ft}\,dt \qquad (1)$$

To allow a greater generality in the study of timbre space it is necessary to combine time-amplitude representations, with frequency-amplitude relationships. This is so as to allow observation of how the spectral components of a musical sound evolve over time (*i.e.* how the spectra develop).

The *Short-Time Fourier Transform* (STFT) or the *Spectrogram* [2], was originally developed as a time-frequency-intensity representation for speech analysis. It is a simple extension of the FT, where the FT is repeatedly evaluated for a running windowed version of the time domain signal. Each FT gives a frequency domain 'slice' associated with the time value at the window center. The STFT is represented analytically by

$$F_w(t,f)=\int_{-\infty}^{+\infty} f(\tau)w(\tau - t)e^{-j2\pi ft}d\tau \qquad (2)$$

where t indicates the position of the window on the time axis. The Spectrogram can be used for time-frequency analysis of musical tones. It enables us to observe the temporal and spectral characteristics at any point in the 'timbre space'. Slaney, Covell and Lassiter [3] make a point of using Magnitude Spectrograms, which allow changes to be made to the Spectrogram, in the knowledge that the phase will be recovered in the inverse Spectrogram. The simplest case they use for the purpose of morphing involves two quasi-stationery tones. These each have well defined spectral shape, pitch, rhythm, and other perceptually relevant auditory dimensions. For example two different pitches cause sounds to be perceived as two different auditory objects. This is but one of the difficulties faced in performing a successful morph, and in this case, it can be overcome by scaling the frequencies in the STFT which allows interpolation between the different pitches. This, unfortunately, does not work for drastic pitch changes as formants are also moved. Apart from such intrinsic morphing difficulties, it is also the case that other problems arise due to the nature of the joint time-frequency representation used. For example, the Spectrogram has a significant shortcoming in that it involves an intrinsic trade-off between time resolution and frequency resolution. The reason for the time-frequency resolution trade-off is the Fourier domain property that a window and its spectrum cannot both be arbitrarily narrow. The implication is that improvements in the identification of spectral detail from a Spectrogram can only be achieved at the expense of deterioration in temporal resolution. Given that this is not entirely satisfactory in the context of timbre morphing, we are put upon to identify joint time-frequency representations that facilitate better joint time-frequency resolution. One such representation is the Wigner Distribution.

The Wigner Distribution (WD) was proposed by Wigner in 1932 for application in quantum mechanics. It has more recently been recognised as a powerful tool for time-frequency analysis of signals, where with some care, it can be interpreted as a distribution of the signal energy in time and frequency. Claasen and Mecklenbrauker [4] show that the WD can be evaluated from both the time signal f(t)

$$W_f(t,\omega)=\int_{-\infty}^{+\infty} e^{-j\omega\tau} f(t + \tau/2)f^*(t - \tau/2)d\tau \qquad (3)$$

and from the Fourier Transform, F(ω), of the time signal f(*t*)

$$WF(\omega,t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} e^{+j\Omega\tau} f(\omega + \Omega/2)f^*(\omega - \Omega/2)d\Omega \qquad (4)$$

The two distributions have the relation:

$$W_F(\omega,t) = W_f(t,\omega) \qquad (5)$$

It can also be shown that the WD is the most basic or primary time-frequency distribution, as defined by Cohen [5], [6]. In the context of timbre representation, an important attribute is the spread of the square magnitude of any joint time-frequency distribution. We can make our choice among the many distributions of the Cohen class with the simple demand of minimum spread. In the case of the WD, it can be shown that this smearing/spreading will be a minimum and will be greatly reduced in comparison to that of the STFT, hence allowing better feature identification for morphing. While the STFT suffers from the important drawback of improved frequency resolution being obtained only at the expense of time resolution and vice versa, the WD overcomes this drawback. So, the WD minimises the inherent averaging over time and frequency of the STFT, but, as has often been pointed out, at the expense of the introduction of cross products.

Recently, Pielemeier and Wakefield [7] have looked at other time-frequency analysis methods for representation of musical signals, including the constant-Q modal distribution, which can be used as the basis for estimation of temporal and spectral parameters. They dismiss the possibility of using the WD for the representation of musical timbres due to the distortion introduced by the cross terms previously mentioned. It is the purpose of this paper to show that effective smoothing of the WD can remove the bothersome interference products while still providing minimal spreading of temporal detail. It is suggested that such smoothed WDs can provide a useful basis for the extraction of those features relevant to timbre morphing.

3 The Wigner Distribution - theme and variations!

So far, the WD has just been considered for continuous-time signals. To perform a practical computer-based measurement the WD needs to be evaluated from a discrete-time signal. Windowing and sampling the continuous-time signal, Claasen and Mecklenbrauker [8] show that

$$PWD(n,\omega)=2\sum_{k=-L+1}^{L-1} e^{-j2k\omega}p(k)g(n,k) \qquad (6)$$

The new function in equation 6 is the discrete Pseudo Wigner Distribution (PWD). In this case,

- p(k) = *w*(k)*w**(-k) is the window time correlation function, and
- g(n,k) = f(n+k) f*(n-k) is the signal time correlation function.

Comparing the PWD with the Spectrogram, it can be seen that both are spread versions of the WD. However, significantly, for the case of the PWD, the spreading is only in the frequency direction. It is also noticeable that there are interference terms present in the PWD, which are not present in the Spectrogram. These add clutter to the Wigner distribution representation. While there are no methods by which to reduce the spreading or smearing evident in the spectrogram, the cross-terms in the PWD can be removed by introducing a smoothing window. In equation 7, the smoothing window is included in the PWD and the smoothed discrete Pseudo Wigner Distribution (SPWD) is generated.

$$SPWD(n,\omega)=2\sum_{k=-L+1}^{L-1} e^{-j2k\omega} p(k) \sum_{l=-m+1}^{m-1} z(l)g(n,k) \qquad (7)$$

Here, z(l) is the smoothing window. In general the window length for w(k) is 2L and is therefore reasonably large. The smoothing window length for z(l) is 2m and is generally of short duration, as if it is too large it smoothes too much of the required PWD signal. It can be shown that the optimum smoothing window type is Gaussian [9].

4 Graphical Comparison of Spectrograms and Wigner Distributions

In order to investigate the relative performance of the STFT and WDs with regard to accurate representation of temporal and spectral detail, an examination of a number of synthesized signals has been undertaken. For the purposes of demonstration, particular cases of the STFT and WDs of single and dual chirps and musical signals are presented. As will be seen, the spreading or smearing of the STFT (fig. 5.1) relative to the WD (fig. 5.2) is very evident for single chirps. However, when the multi-component dual chirp is used the WD cross terms become all too obvious (fig. 6.2). It is shown that the introduction of a Gaussian smoothing window effectively eliminates this interference (fig. 6.3).

When the SPWD is applied to real musical signals - in this case, the example used in figures 7.1, 7.2 and 7.3 is a portion (0.12s) of the quasi-steady state section of a synthesized tone of fundamental pitch 440Hz, sampled at 5000Hz - the clutter seen in figure 7.2, which previously was the principal objection to the use of the WD for timbre representation, is seen to be absent (fig. 7.3). From this it can be suggested that the spectral and temporal features required for timbre morphing can be more accurately extracted using the SPWD (fig. 7.3) rather than the STFT (fig. 7.1) representation. The analysis of a longer sample should improve the resolution of the STFT, however, the relative superiority of the SPWD would remain. The temporal and spectral smearing of the STFT makes it more difficult to identify the correct location of any of the harmonics in figure 7.3. Each, is smeared over an average range of approximately 185Hz, while each is smeared temporally beyond the extents of the analysis window. The SPWD of the same tone (fig. 7.3), displays much better spectral resolution than that of the STFT(fig. 7.1), and temporally it remains within the analysis

window. Based on these comparisons, over similar analysis window lengths, the PWDF appears to provide for more accurate feature identification - than that provided by the STFT - within the signal.

For example, in relation to a morph between two quasi-stationary sounds, Slaney, Covell and Lassiter [3] point out that the feature in the first sound needs to be slowly moved to its corresponding position in the second sound during the morphing process, as features have often moved relative location between the sounds. Using the SPWD, it would be much easier to identify any changes in features. This point is all the more important when seen in the light of Iverson and Krumhansl's [10] work, where they showed that although the onset of a tone is not required for similarity judgement between tones, it is required to enable listeners to identify an instrument. In the following, graphical comparisons of the relative performance of the STFT and the WD for the various test signals will be presented.

5 Single Chirp

STFT

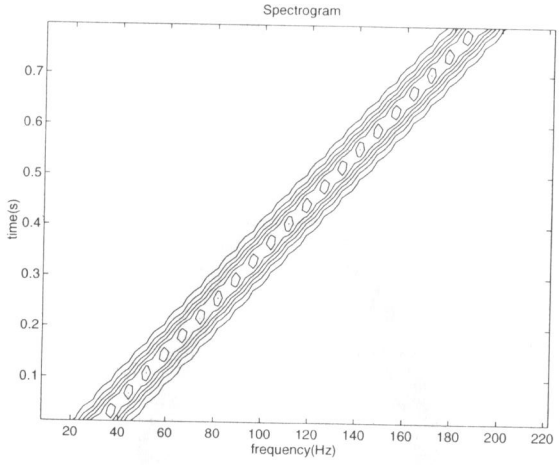


Figure 5.1. The Spectrogram of a single chirp - length 354, sampled at 444Hz using a Hanning window of length 60, overlap 55 - rising in frequency from 20Hz to 200Hz.

PWD

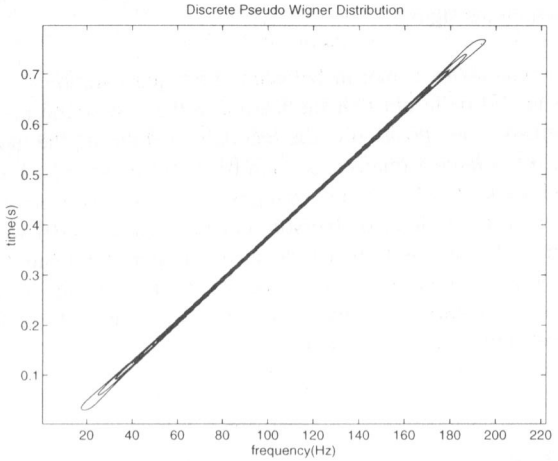


Figure 5.2. The Discrete PWD of a single chirp - length 354, sampled at 444Hz, windowed using a Hanning window of length 354 - rising in frequency from 20Hz to 200Hz.

6 Dual Chirp

STFT

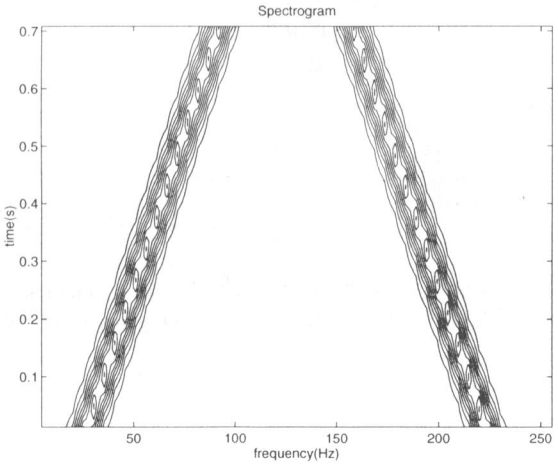


Figure 6.1. The Spectrogram of a dual chirp - length 362, sampled at 511Hz, windowed using a Hanning window of length 60, overlap 55 - rising in frequency from 10Hz to 100Hz, and falling in frequency from 230Hz to 140Hz.

PWD

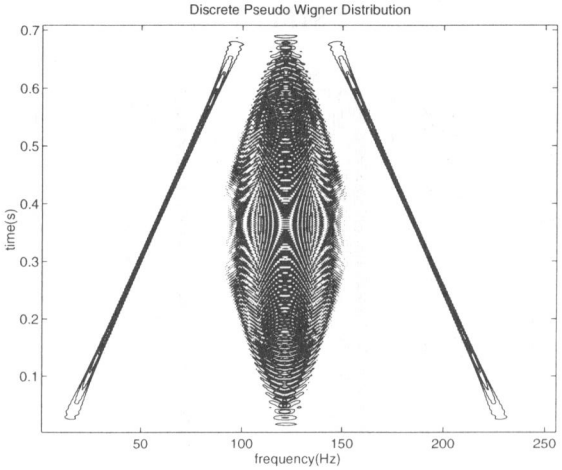


Figure 6.2. The Discrete PWD of a dual chirp - length 362, sampled at 511Hz, windowed using a Hanning window of length 362 - rising in frequency from 10Hz to 100Hz, and falling in frequency from 230Hz to 140Hz.

SPWD

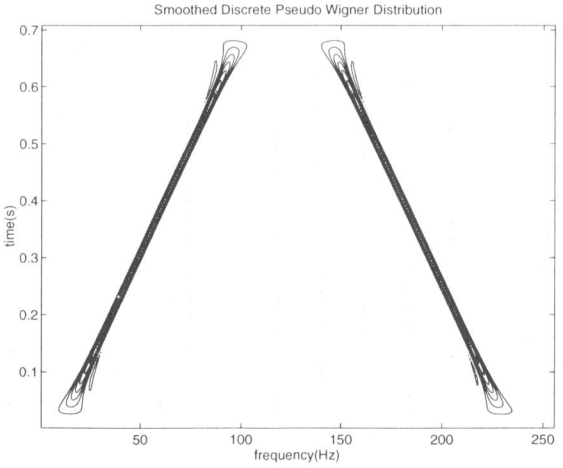


Figure 6.3. The Smoothed Discrete PWD of a dual chirp - length 362, sampled at 511Hz, windowed using a Hanning window of length 362 and a Gaussian smoothing window of length 16 - rising in frequency from 10Hz to 100Hz, and falling in frequency from 230Hz to 140Hz.

7 Synthetic Tones

STFT

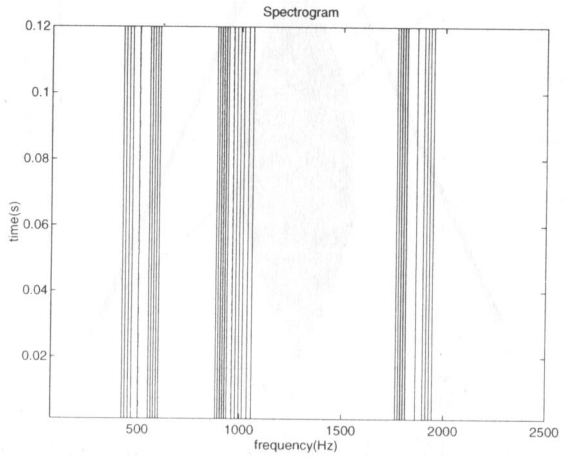


Figure 7.1. The Spectrogram of a synthetic tone - length 600, sampled at 5kHz, windowed using a Hanning window of length 60, overlap 55 - with fundamental frequency 440Hz.

PWD

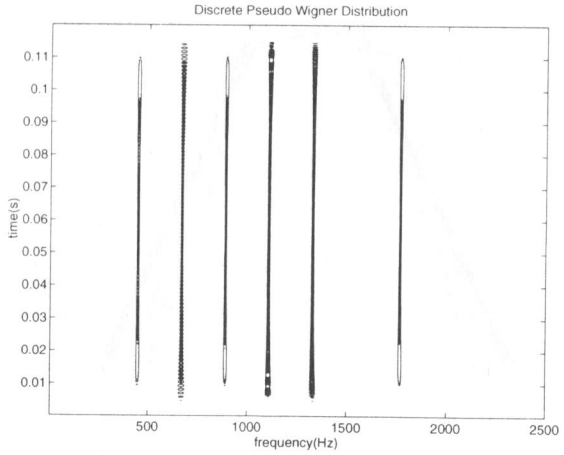


Figure 7.2. The Discrete PWD of a synthetic tone - length 600, sampled at 5kHz, windowed using a Hanning window of length 600 - with fundamental frequency 440Hz.

SPWD

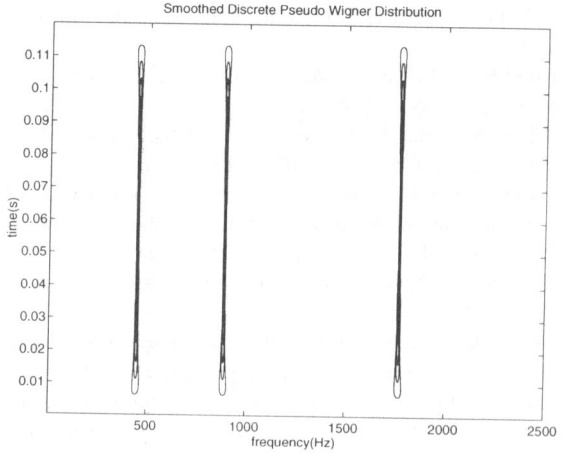


figure 7.3. The Smoothed Discrete PWD of a synthetic tone - length 600, sampled at 5kHz, windowed using a Hanning window of length 600 and a Gaussian smoothing window of length 16 - with fundamental frequency 440Hz.

8 Conclusions

From the analysis presented, and from the graphical presentations given, it is clear that the WD can be used for effective representation of musical timbres. When appropriate smoothing is applied the cross terms, which introduce unwanted clutter for multi-component signals, can be eliminated at the expense of the introduction of a degree of spectral smearing. It is notable, however, that, firstly, this smearing is not as significant as that of the Spectrogram, and secondly, that temporal smearing is not introduced. This is a significant advantage over the STFT which is constrained to a trade-off between temporal and spectral resolution. For the purposes of musical synthesis deriving from timbre morphing, the increased accuracy of the WD representation will allow more accurate extraction of those features which characterise musical timbre.

9 References

[1] B. Holloway, E. Tellman and L. Haken, "Timbre Morphing of Sounds with Unequal Numbers of Features," *J. Audio Eng. Soc.* vol. 43, no. 9, pp 678-689, September 1995.

[2] L.R. Rabiner and R.W. Schafer, "Digital Processing of Speech Signals," *Prentice-Hall, Englewood Cliffs, NJ* 1978.

[3] M. Slaney, M. Covell and B. Lassiter, "Automatic Audio Morphing," in *Proc. ICASSP, IEEE, Atlanta, GA,* May 7-10, 1996.

[4] T.A.C.M. Claasen and W.F.G Mecklenbrauker, "The Wigner Distribution - A Tool for Time-Frequency Signal Analysis, part 1: Continuous-Time Signals," *Phillips Journal of Research,* vol. 35, no.3, pp. 217-250, 1980.

[5] L. Cohen, "Generalized Phase-Space Distribution Functions," *J. Math. Phys.,* vol. 7, pp. 781-786, 1966.

[6] L. Cohen and H. Margeneau, "Probabilities in Quantum Mechanics," in *Quantum Theory and Reality,* M. Bunge, Ed. (Springer, Berlin), chap. 4, pp. 71-89, 1967.

[7] W.J. Pielemeier and G.H. Wakefield, "A high-resolution time-frequency representation for musical instrument signals," *J. Acoust. Soc. Am.,* vol. 99 (4), Pt. 1, pp 2382-2396, April 1996.

[8] T.A.C.M. Claasen and W.F.G Mecklenbrauker, "The Wigner Distribution - A Tool for Time-Frequency Signal Analysis, part 2: Discrete-Time Signals," *Phillips Journal of Research,* vol. 35, no.4/5, pp. 276-300, 1980.

[9] J.C. Andrieux, R. Feix, G Mourgues, P. Bertrand, B. Izrar and V.T. Nguyen, "Optimum Smoothing of the Wigner-Ville Distribution," *IEEE Transactions on Acoustics, Speech and Signal Processing,* vol. ASSP-35, no. 6, pp. 764-769, June 1987.

[10] P. Iverson and C.L. Krumhansl, "Isolating the dynamic attributes of musical timbre," *J. Acoust. Soc. Am.,* Vol. 94 (5), pp 2595-2603, November 1993.

SÍNTESE ADITIVA E WAVESHAPING PELO MÉTODO POLINOMIAL

Edwin M. Loboschi, Cesar J. B. Pagan e Yaro Burian Jr.

Faculdade deEng. Elétrica e de Computação da UNICAMP

Campinas 13081-970 SP

e-mail: pagan@fee.unicamp.br

Paulo S. dos Santos

PS Digital

R. Milão 271, Amparo 13900-000 SP

A escolha de uma técnica de síntese freqüentemente envolve a comparação entre as possibilidades criativas e a capacidade de prever o efeito de cada parâmetro no resultado final. Até o início dos anos 70, havia principalmente o uso das técnicas de síntese aditiva e subtrativa [1,3]. Na síntese aditiva, várias componentes senoidais são combinadas para formar o som desejado, enquanto que na subtrativa, predomina a utilização de filtros para tratar o sinal de entrada (geralmente uma forma de onda complexa). Posteriormente a síntese FM [2], que em sua forma mais simples trata de uma função senoidal, cuja fase é modulada por outra função senoidal, foi amplamente explorada com diversas variações à proposta inicial, inclusive sendo a primeira técnica digital numericamente tratável para geração de espectros dinâmicos complexos. Na técnica de waveshaping, uma função de transferência é aplicada ao sinal de entrada (geralmente senoidal), gerando um espectro complexo. Outros métodos conhecidos para geração de audio tem sido usados, cada qual com suas peculiaridades, sendo que, nos equipamentos comercias é comum a implementação de um conjunto de técnicas.

Podemos citar o equipamento Prophet 5 da empresa Sequential Circuits, que utilizando basicamente os módulos controlados por tensão disseminados por Robert Moog, de certa forma, utiliza sínteses aditiva (dois conjuntos de osciladores onde cada conjunto pode gerar alguns tipos de formas de onda) e subtrativa (utilização de um filtro, no caso controlado por tensão).

Nossa proposta de síntese parte do método matemático de reprodução de uma função qualquer a partir da expansão em série de polinômios ortogonais. As bases de funções trigonométricas, nas quais é realizada a expansão de uma série de Fourier, não são a única forma de se representar uma função periódica arbitrária. Quando se trata de música, as funções trigonométricas são freqüentemente usadas, sendo a composição harmônica dada pelos coeficientes a_k das funções de base:

$$F(t) = \sum_{k=0}^{\infty} a_k \phi_k(\omega_k t) \qquad (1)$$

O espectro de freqüências de uma função de base $\phi_k(\omega_k t)$ é dado diretamente por sua Transformada de Fourier. Entretanto, como o domínio das funções trigonométricas estende-se de $-\infty$ a $+\infty$, o som deveria estar soando infinitamente para que fosse possível encontrar uma frequência única que o representasse. De fato, não há razão para preferir funções trigonométricas quando se trata de escolher uma base de funções para se fazer a expansão de uma função periódica arbitrária.

No método que estamos desenvolvendo, bases ortogonais de polinômios de Legendre, Chebyschev e Hermite são usadas para gerar funções de transformação do sinal de entrada $F_{in}(t)$ à semelhança do que é feito pela técnica de waveshaping. O sinal de saída $F_{out}(t)$ é dado pela operação:

$$F_{out}(t) = \sum_k a_k(t) P_k(F_{in}(t)) \qquad (2)$$

para a qual P_k é o polinômio de ordem k de determinada base. Note que $a_k(t)$ é função do tempo.

Temos dois casos especiais a considerar: No primeiro, fazendo as $F_{in}(t)$ funções senoidais, o resultado $F_{out}(t)$ equivale a uma síntese aditiva, para a qual os envelopes das harmônicas são dados por combinações lineares dos coeficientes $a_k(t)$. No segundo, considerando constantes os coeficientes dos polinômios, nosso método corresponde à síntese por waveshaping convencional, com a função de transferência fixada pelos coeficientes $a_k(t)$.

Os parâmetros a serem determinados neste método de síntese são:

- A base de polinômios ortogonais
- A forma e a frequência da função de entrada $F_{in}(t)$
- A forma e a frequência (quando aplicável) da função de cada coeficiente $a_k(t)$.

A implementação computacional do método mostrou que esta técnica é rica em possibilidades de timbres e de espectros dinâmicos, possibilitando resultados musicalmente muito agradáveis e muitas vezes inusitados. Nosso programa parte das idéias expostas acima para gerar amostras de timbres (arquivos do tipo WAVE). Um desenvolvimento em série de polinômios ortogonais é feito à maneira da equação (2). O usuário realiza as escolhas apontadas acima, e escolhe a forma funcional dos coeficientes e da função de entrada entre senoidal, dente de serra, quadrada, *sample-and-hold*, ruído e triangular, tendo portanto uma possibilidade muito maior de síntese, comparando com a síntese waveshaping convencional. Atualmente, estamos trabalhando no sentido de gerar estes sinais em tempo real para que possam ser acessados via MIDI.

Nosso grupo desenvolveu ainda outro programa para explorar a possibilidade de reconstrução de um som a partir desta técnica, o qual realiza a decomposição de uma onda séries de polinômios ortogonais.

Para decompor em séries de polinômios ortogonais fazemos a função de entrada $F_{in}(t)=\alpha t$, onde $\alpha=1/T(S)$ e T(S) é o tempo de duração do *setor* a ser examinado. Antes disso, o programa faz a transformada rápida de Fourier dos primeiros 2048 pontos, obtém a frequência da componente de frequência mais baixa e calcula o valor do período correspondente. Feito isso, o vetor que representa a onda é dividido em setores, e estes são escolhidos de maneira que as suas fronteiras correspondam aos zeros da função amostrada, sendo o número de pontos de cada setor próximo ao número de pontos correspondente ao período calculado por FFT. O número de amostras varia conforme o setor. Os coeficientes da série são calculados então para cada setor da onda, obtendo-se os valores de $a_k(S)$ e $\omega_k(S)$, ou seja, a amplitude da k-ésima componente e a frequência angular do desenvolvimento no setor S.

A partir deste programa é possível obter a evolução temporal dos coeficientes $a_k(S)$. Para o mesmo som, diferentes escolhas de $\{\phi_k\}$ levam a diferentes conjuntos $\{a_k(S)\}$.

Os programas foram escritos em QBasic e no momento estão recebendo interface em Visual Basic. Estes programas estarão disponíveis brevemente nos endereços eletrônicos dos autores.

REFERÊNCIAS

[1] Giovanni de Poli, Sound Synthesis by Fractional Waveshaping; J. Audio Eng. Soc vol 32,11 pp 849-861

[2] Miller Puckette, Formant-Based Audio Synthesis Using Nonlinear Distortion; J. Audio Eng. Soc. Vol 43, ½ pp 40-47

[3] Daniel Arfib, Digital Synthesis of Complex Spectra by Means of Multiplication of Nonlinear Distorted Sine Waves; J. Audio Eng. Soc. Vol 27,10 pp 757-768

Utilização de Estrutura de Dados espaciais para Representação de Recursos Composicionais em Música.

Ana Miccolis
Engenharia de Sistemas e Computação - COPPE
Universidade Federal do Rio de Janeiro
miccolis@cos.ufrj.br

Abstract

Traditional musical composition uses a conventional notation based on pitches and discrete values. Traditional music representations have being used with good performance in many musical software. There are many products, like Finale and Encore, that represent this kind of notation. They offer the possibility of writing for many instruments, inputing data through a midi keyboard or by passing arguments to the score. But the progress of musical invention increases the need of others structures, able to represent non conventional music, enabling new approaches and new possibilities of composition. In this paper it will be proposed an alternative way to represent non conventional music aided by computer.

Introdução

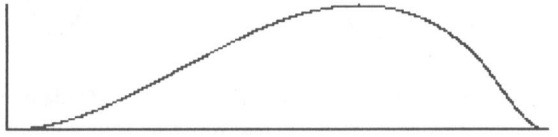
A composição de música tradicional utiliza uma notação convencional baseada em pentagramas e claves para determinar alturas e valores de notas musicais. Representações tradicionais de música em computador tratam com eficiência os recursos de composição baseados neste tipo de construção. Para esta forma de composição, representada na notação musical convencional, já existe atualmente boas ferramentas de auxílio computacional. Produtos como Finale e Encore oferecem a possibilidade de escrita para vários instrumentos, tanto a partir de um teclado midi, como a partir da entrada de parâmetros na edição da própria partitura. Contudo, a evolução da pesquisa em composição musical fez com que o modelo de representação tradicional não correspondesse mais às necessidades dos músicos. Sentiu-se a necessidade de outros tipos de representações que oferecessem maior flexibilidade na composição musical assistida por computador. Estas representações deveriam desenvolver-se de forma a considerar estéticas mais amplas do que aquelas até então exploradas pela música convencional. Este paper visa propor uma alternativa para a representação não convencional de música por computador. A partir de alguns estudos em ferramentas de software para composição musical será discutido como estender o modelo tradicional a fim de oferecer recursos composicionais que atendem aos requisitos de estéticas musicais não convencionais.

Representação de Recursos Composicionais não Convencionais

Segundo [Bel 96], a notação musical ocidental favoreceu uma representação gráfica de música a qual deve sua simplicidade a quantização convencional de parâmetros principais como altura e tempo. Linhas em uma partitura musical clássica pressupõe uma divisão em oitavas de doze tons. Da mesma forma, barras de medida indicam um tempo de intervalo igual a um número inteiro de ocorrências numa unidade convencional. Contudo a medida que a composição musical vai avançando em refinamentos de intonação microtonal ou padrões de duração não usuais, símbolos adicionais vão se fazendo necessários para representar estas tendências. No domínio da música assistida por computador existe vários produtos que trabalham a representação musical tradicional. Programas como Finale e Encore são especializados em notação musical comum e execução de parâmetros pertinentes a volume, velocidade de ataque e outras características referentes a uma contagem de tempo metronômica. Um ambiente de software para composição musical baseado em representação de eventos de sons é uma alternativa proposta por Bel em contraposição as descrições gráficas ou numéricas.

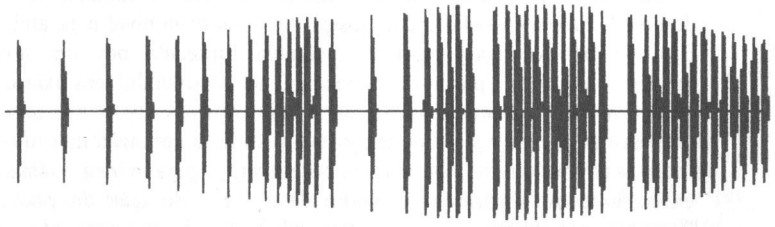
O objetivo principal deste paper é propor o desenvolvimento de ferramentas para geração de recursos auxiliares à composição musical, através de uma representação gráfica de relacionamentos entre diferentes parâmetros os quais são de difícil manipulação através da notação convencional. A partir de uma representação não tradicional o compositor poderia praticar a manipulação de parâmetros composicionais a fim de criar estruturas musicais bastante complexas. Um tipo de notação não convencional que pode ser empregada na criação de música por computador é aquela que ao invés de representar notas em pentagramas oferece uma série de curvas. Estas curvas descrevem o comportamento de características sonoras e funcionam como uma partitura gráfica a qual pode ser modificada e ouvida pelo compositor durante o processo de criação. Este paper seleciona alguns tipos de recursos composicionais que podem ser representados através de técnicas para tratamento de dados espaciais. A vantagem de utilização deste tipo de representação está na possibilidade de criação de estruturas complexas pelo compositor. Algumas pesquisas nesta área apontam para a necessidade de alternativas que possibilitem novas formas de representação musical. Um ambiente para criação gráfica e representação de música gerada por algoritmos genéticos denominado GenComp foi proposto por [Iazzeta 96]. O GenComp é uma aplicação desenvolvida em Macintosh utilizando os recursos ferramentais do MAX. Essa aplicação emprega algoritmos para gerar composições musicais as quais são baseadas em representação gráficas de diferentes parâmetros musicais como densidade, tensão, velocidade, duração, distribuição e tempo. Estes parâmetros podem ser representados graficamente em curvas que controlam o comportamento da música a ser gerada. [Gordon 96] em seu algoritmos de composição musical não emprega geração de notas (usualmente associada com equipamento MIDI), mas sim formas de ondas. Ele considera este tipo de representação o elemento mais importante em suas

peças. A partir de interporlação fractal ele constroi ondas baseadas em formas triangulares e permite sucessivas evoluções do formato original, manipulando a amplitude no tempo. Equipados com computadores e programas de síntese por software, como o Csound ou equivalentes, pode-se abordar de forma bem sistemática a complexificação de funções controladoras, assim como o seu armazenamento, catalogação e reprodução. Recursos como a modulação de freqüência podem ser empregados para a criação de ritmos e outros eventos do tipo 'pseudo-naturais' [Caesar 96]. Na representação desses ritmos surgem pelo menos duas dificuldades principais. A grafia convencional não se adequa a este propósito pela característica da continuidade das evoluções nas vozes 'naturais' e pela discontinuidade da notação da música ocidental. Esta grafia não oferece meios para representar inflexão contínua de alturas microtonais. Utilizando técnicas de modulação de freqüência pode-se gerar funções complexas para a aceleração de pequenos grãos integrantes das vozes de sapos e grilos, por exemplo. A forma da figura a seguir representa a controladora que acelerou, deslocou a altura para cima, aumentou a amplitude ('volume') e diminuiu o tamanho dos grãos e a largura de banda do sonograma de um dos 'cri' da voz de um grilo.



Forma da Função Controladora do 'cri' da voz de um grilo

Modulando esta forma por si mesma, isto é, fazendo tanto a portadora quanto a moduladora gerarem esta mesma forma, pode-se obter uma seqüência de 'cris' cada vez mais rápidos e agudos, que, tão logo chegados ao topo, depressa retornam ao estado inicial. A figura a seguir ilustra esta forma.

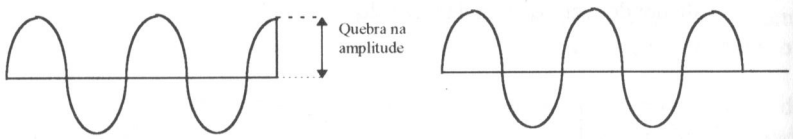


Resultado de FM com portadora e moduladora na forma da figura anterior

Entre os recursos composicionais que podem ser representados através de técnicas para tratamento de dados espaciais pode-se citar os diversos tipos de envelopes de som.

Envelopes

Há vários tipos de envelopes de som, com finalidades e características distintas. Os sons produzidos por sintetizadores muitas vezes possuem um problema característico: a produção de um ruído entre o fim de uma formação e o início de outra devido a mudanças na amplitude de uma forma de onda. Geralmente isso ocorre após a interrupção de um som e o início de outro, cujo valor de amplitude seja muito mais largo que zero. Um mecanismo para produzir um efeito de suavização entre estas transições de silencio e som pode ser produzido a partir da utilização de um envelope de som.



Onda Original finalizada com Ruído Onda envelopada

Envelopes também dão formas a sons e em alguns casos ajudam a determinar seu timbre. Um som de piano, por exemplo, sem seu ataque característico torna-se irreconhecível. Invertendo-se a dinâmica de uma nota do piano pode-se produzir um efeito semelhante ao de um instrumento de arco, porém sem a fricção característica do instrumento. A aplicação desse recurso pode ser encontrada na obra de Schaeffer entitulada "Solfejos do Objeto Sonoro".

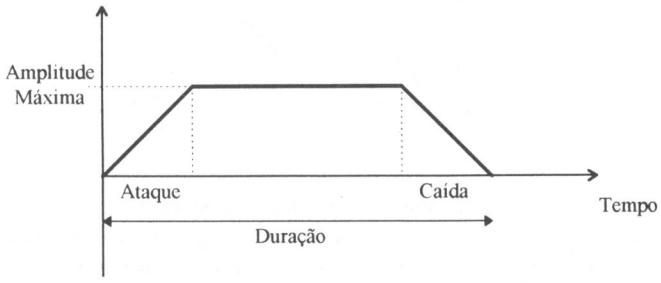
Criação de Envelopes no Csound

A formação de envelopes no Csound será utilizada como forma de ilustrar a possibilidade de manipulação destes recursos para produção de efeitos em composição musical. Csound é uma ferramenta de software desenvolvida para auxílio à composição musical disponível para ambiente PC. Baseado na descrição acústica detalhada fornecida por um arquivo de orquestra e outro de partitura, oferece inúmeras possibilidades para criação de música não convencional. Pelo menos dois arquivos textos são utilizados no processo de composição. O primeiro para conter a definição da orquestra com todas as rotinas que o Csound reconhece e outro para conter as instruções de execução como dinâmica, momento de ataque., duração do processo, etc. Para maiores detalhes sobre a disponibilidade de recursos oferecida pelo Csound, sugere-se a leitura de [Fischman 91]. O conceito básico da implementação do Csound está na forma como os sons são produzidos.

Osciladores são combinados para criarem juntos unidades mais complexas a partir de tipos básicos de formas de ondas como senoidais, quadradas, triangulares, dente de serra, ruído branco, ruído rosa, etc. No csound a utilização de controladores é uma alternativa para configurar não osciladores que produzem o som, mas módulos que controlam outros osciladores. Uma aplicação comum de controlador utilizada pelo Csound é a geração de envelopes de amplitude. Estes controladores podem ser programados para produzirem diversos tipos de sinais. No Csound um módulo de geração de funções foi implementado para prover geradores de sinais de diversos tipos, a partir de rotinas, cuja numeração própria permite identificá-las dentro da definição de uma orquestra. Dessa forma, GEN10 é utilizada para produzir ondas senoidais e combinações da fundamental e seus harmônicos, GEN07 é empregada com o objetivo de produzir formas de ondas que possam ser construídas a partir de linhas retas, como ondas quadradas e triangulares. Sugere-se a leitura do quarto capítulo em [Vercoe 92] para melhor detalhamento de rotinas geradoras no Csound. A construção de envelopes a partir destas funções geradores é um recurso muito rico na atividade de composição musical. Discutiremos a seguir alguns dos mais comuns meios de criação de envelope no Csound, como LINEN, LINSEG, EXPSEG.

LINEN

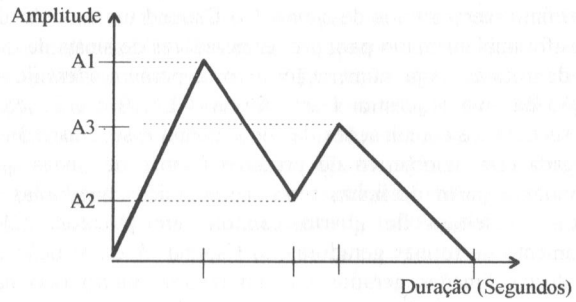
A primeira maneira e também a mais direta de produção de um envelope no Csound é através do uso do comando Linen. Ele pode produzir um envelope linear como o mostrado na figura abaixo, para alteração da amplitude de um som, controlando o seu ataque característico.



Para conseguir-se envelopes mais flexíveis, pode-se utilizar o comando LINSEG, onde formas mais complicadas podem ser especificadas. A partir da construção de uma lista de amplitudes e tempo que cada valor leva para ser atingido, esta função calcula o envelope correspondente.

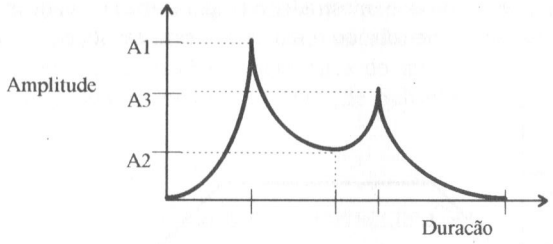
No exemplo abaixo, utiliza-se interpolação linear para calcular os valores entre as três amplitudes dadas A1, A2 e A3.

LINSEG



EXPSEG

Outra forma de geração de envelopes pode ser empregada com o comando EXPSEG, que funciona de maneira muito próxima ao LINESEG, diferenciando-se dele pelo fato de utilizar segmentos exponenciais ao invés de interpolação linear entre os pontos dados.

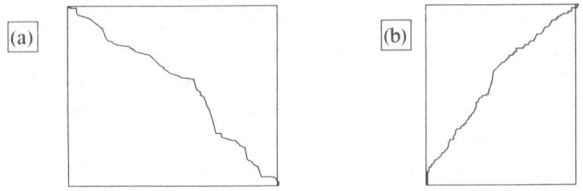


Outras duas maneiras muito freqüentes de produção de envelopes de som, podem ser conseguidas através do gerador de envelopes ENVLPX ou usando um oscilador para gerar o envelope desejado. Este último meio apresenta vantagens sobre as rotinas geradoras pois permite formas mais diversificadas de envelopes. Utilizando-se tabelas de funções pode-se determinar a forma do envelope. Um recurso que ampliaria as possibilidades de composição com utilização de envelopes de som seria o de criação de um envelope a partir de pontos de interseção de dois envelopes. Para isso, seria necessário desenvolver-se uma função que pudesse analisar o comportamentos de dois envelopes e a partir dos pontos comuns entre eles apresentar um terceiro, cuja interpolação linear entre os pontos de interseção desse origem a um novo envelope.

Aplicação de Envelopes de Som Utilizando Técnicas de Representação de Linhas Poligonais

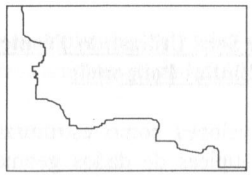
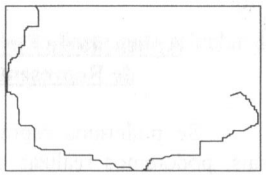
Se pudermos representar envelopes como estruturas de dados espaciais, poderemos realizar operações típicas de dados geométricos com operações booleanas como união, interseção ou diferença. Todas estas operações são de fundamental importância quando de trata de composição baseada em tipos de filtros, envelopes e outros recursos, utilizados na composição não tradicional. Para este tipo de composição, outras formas de representação além das clássicas são necessárias para lidar com parâmetros que não se restringem apenas a valores discretos em uma escala de sons. O caso da criação de envelopes de som será ilustrado, com o objetivo de exemplificar a geração de recursos composicionais que podem utilizar uma representação espacial. A partir de estruturas que possibilitem o tratamento espacial do dado, sugere-se a criação de envelopes com características híbridas para manipulação do som.

O problema de determinação dos pontos de interseção de duas linhas poligonais pode ser resolvido utilizando-se uma representação hierárquica de aproximação para curvas, do tipo BSPR[Burton 77] ou Strip Trees [Samet 90]. Quando uma linha poligonal é representada como uma BSPR (Binary Searchable Polygonal Representation), certos conjuntos de lados consecutivos podem ser examinados simultaneamente. Se um lado neste conjunto puder conter um ponto que esteja sendo procurado, então um subconjunto pode ser examinado, constituindo basicamente uma busca binária. Denomina-se seção a um conjunto de lados consecutivos e seção primitiva aquela que contiver apenas um lado. Uma seção será dita simples quando for monotônica nas coordenadas x e y, como ilustra a figura abaixo.



Uma vantagem de optar-se por representação do tipo BSPR em lugar de Strip trees, está no fato de não limitar-se a regularidade da curva representada, permitindo-se trabalhar com curvas que sejam não monotônicas tanto em X como em Y.

Um ponto significativo de uma linha poligonal é um vértice onde exista um local extremo tanto em x ou em y ou em ambos. Pontos extremos são sempre pontos significativos. Pontos significativos partem um alinha poligonal em regiões maximais simples a qual pode ser denominada uma seção básica. BSPR permite trabalhar com regiões compostas, isto é, seções contendo várias seções básicas, como ilustra a figura a seguir.

(c)  (d) 

Cada seção que for examinada deverá respeitar as seguintes condições:

(1) A localização do primeiro vértice do primeiro lado, e do último vértice do último lado da seção deverá ser conhecida. Estes dois vértices serão denominados pontos finais da seção.

(2) Os valores Mínimo e Máximo das coordenadas X e Y dos vértices da seção serão conhecidos. Estes quatro valores determinam o menor retângulo com lados paralelos aos eixos, contendo a seção. A este retângulo dá-se o nome de SR(section retangle). As figuras anteriores (a), (b), (c) e (d) mostram SR para quatro seções.

(3) Exceto no caso da seção primitiva, uma seção irá ser a união de duas outras seções.

Considere o problema de encontrar os pontos de interseção de duas seções. Se os SRs das seções não se interceptarem, a interseção entre elas será impossível. Se as seções forem seções primitivas, o ponto de interseção pode ser facilmente calculado, se existir. Finalmente se os SRs se interceptarem e ao menos uma seção for não primitiva, então uma das seções poderá ser dividida em duas subseções. O processo inteiro pode então ser repetido e desta forma a terceira condição implica em uma hierarquia de árvore binária de seções. Os pontos para divisão da curva devem ser escolhidos como aqueles no qual a monotonicidade em X ou em Y seja quebrada.

O problema de determinar os pontos de interseção de duas linhas poligonais requer que se pesquise dois BSPR simultaneamente. Duas seções correntes, uma lista de resultado e uma pilha serão necessárias. O algoritmo a seguir ilustra o procedimento para tratamento deste problema.

Passo 1: Inicializar as seções correntes CS(1) e CS(2) com as duas linhas poligonais pesquisadas. A lista e a pilha estarão ambas vazias.

Passo 2: Se os SRs de CS(1) e CS(2) não se interceptarem, vá para o passo 6.

Passo 3: (Se a área de SR de CS(2) for maior que a área de SR de CS(1) e CS(2) não for uma seção primitiva) ou

(Se CS(1) for uma seção primitiva)
Então Faça i = 2;
Senão Faça i = 1.

Se CS(i) for uma seção primitiva vá para o passo 5.

Passo 4: Divida CS(i) em duas seções NS(1) e NS(2), e faça CS(i) = NS(1).
Se i = 1, então coloque o par (NS(2), CS(2)) na pilha.
Se i = 2, então coloque o par (CS(1), NS(2)) na pilha.
Vá para o passo 2.

Passo 5: Se CS(1) e CS(2) (ambos lados simples) interceptarem-se, adicione o ponto de interseção à lista de resultado.

Passo 6: Se a pilha estiver vazia, retorne.
Caso contrário, faça o par de seções correntes (CS(1), CS(2)) serem iguais ao par do topo da pilha, remova este par da pilha e vá para o passo 2.

Conclusão

A representação de recursos composicionais em forma de dados espaciais é uma alternativa que amplia o potencial de manipulação de parâmetros na construção de obras musicais não convencionais. Esse tipo de representação permite que se opere com os dados trabalhando-os com operações booleanas como interseção, união ou diferença. Na geração de envelopes e filtros, dois recursos de fundamental importância para a música não tradicional assistida por computador, o emprego das operações citadas anteriormente amplia a forma de obtenção dos efeitos sonoros desejados. Neste paper, um envelope de som, resultante da interseção de outros dois envelopes, foi proposto como meio de ilustrar uma aplicação que seria viável caso o envelope fosse representado como uma estrutura de dados espaciais. Utilizou-se para isso a estrutura de BSPR (Binary Searchable Polygonal Representation) a fim de representar uma curva de envelope de som. Outra aplicação que seria de grande utilidade na criação de música assistida por computação seria a possibilidade de criação de filtros a partir da diferença de determinadas regiões. Representando-se os filtros como estruturas espaciais, poderia-se implementar algoritmos que tratassem dessa operação, utilizando-se para isso a linguagem C, que é a linguagem na qual foram desenvolvidas as rotinas geradodas (GenRoutines) do Csound. O Csound contém atualmente 21 rotinas geradoras para manipulação de diversos parâmetros, contudo nenhuma delas explora os recursos composicionais como dados geométricos. O acréscimo de rotinas geradoras que pudessem dar esse tratamento aos parâmetros utilizados comumente na criação de orquestra do Csound contribuiría para que o usuário desse produto pudesse ampliar o domínio de amostras sonoras.

Referências

[Bel 96] - Bernard Bel. A symbolic-numeric approach to quantization in music. In III Simpósio Brasileiro de Computação e Música, Recife, Pe.

[Burton 77] - W. Burton. Representation of many-sided polygons and polygonal lines for rapid processing. Communications of the ACM 20, 3 (March 1977).

[Caesar 96] - Rodolfo Caesar. Artefatos FM para a produção de ritmos pseudo-naturais. In III Simpósio Brasileiro de Computação e Música, Recife, Pe.

[Gordon 96] - Gordon Monro. Algorithmic composition at the waveform level. In III Simpósio Brasileiro de Computação e Música, Recife, Pe.

[Fischman 91] - Rajmil Fischman. Musical Applications of Digital Synthesis and Processing Technics , Keele, Abril 1991.

[Iazzeta 96] - Iazzeta, F. GenComp: An Environment for Graphic Creation and Representation of Music Generated with Genetic Algorithms. In III Simpósio Brasileiro de Computação e Música, Recife, Pe.

[Samet 90] - Samet, H. The design and analysis of Spatial Data Structures. Addison-Wesley Reading, MA, 1990.

[Vercoe 92] - Vercoe, Barry L. Csound - Manual for the Audio Processing System and Supporting Programs. M.I.T., Cambridge, Massachusetts.

Agradecimentos

Esta pesquisa é apoiada pelo CNPQ e pela Escola de Música da U.F.R.J. Contou com a orientação do Prof. Rodolfo Caesar e a colaboração de Murilo Moss Barquette.

Representação e Recuperação Baseada em Conteúdo de Partituras Musicais em Bases de Dados Orientadas a Objetos

Marisa Beck Figueiredo
marisa@icmsc.sc.usp.br

Caetano Traina Júnior
caetano@icmsc.sc.usp.br

Agma J. Machado Traina
Agma@icmsc.sc.usp.br

Departamento de Ciências de Computação e Estatística
Instituto de Ciências Matemáticas de São Carlos- USP
Caixa Postal 668 - 13560-970 - São Carlos - SP
Tel. :(016) 274-9128 Fax: (016) 274-9150

Abstract

This paper presents a manner to represent the Musical Score characteristic in an Oriented-Object Database, supporting storage, manipulation, search and retrieval methods. The information search is treated with particular interest, because they can provide resources for that the database manager can run content-based searches. So, it is possible to analyse musical information stored to answer the queries performed in a database. The system described here utilizes an object-manager called SIRIUS/GO that supports the data model SIRIUS [Biaj96]. This management is made through of creating a new class of data type, characterized as Musical Score. This class is included among others SIRIUS semantic builders, and it is homogenized by the orthogonal criterium of the SIRIUS Data Model.

1. Introdução.

Quando é necessário armazenar um grande volume de informação, é tradicional utilizar-se *Gerenciadores de Bases de Dados* para esta tarefa, os quais estruturam as informações em uma *Base de Dados*. Essas estruturas devem seguir um conjunto de regras, que definem o arcabouço conceitual de representação dos dados adotado. A esse conjunto de regras denomina-se *Modelo de Dados*, o qual permite a um projetista efetuar a *Modelagem de Dados* de uma determinada aplicação. Nos últimos anos, vem tomando grande impulso o desenvolvimento dos chamados Modelos de Dados Orientados a Objetos. Nesses modelos, a idéia básica é a representação dos conceitos e coisas do mundo real através de *"objetos"*, dos quais descreve-se tanto sua *estrutura* (atributos, propriedades, etc.) quanto seu *comportamento* (ações e procedimentos que são tomados pelos objetos em respostas aos estímulos externos que recebe) [Elma94].

No geral, os modelos de dados orientados a objetos [Banc92] [Bert93] [Zand95] têm uma maior capacidade de representação do que os modelos ditos "tradicionais", pois além de serem capazes de representar o comportamento dos elementos do mundo real - o que os modelos tradicionais não fazem - ainda possuem um conjunto de elementos de modelagem mais rico. Um dos elementos de modelagem que acrescentam poder de representação aos modelos orientados a objetos é sua capacidade de permitir a definição de novos tipos de dados, através da definição de "Classes" de objetos. *Tipos de dados* usuais, "nativos" dos gerenciadores em geral são por exemplo: números inteiros, números reais, bytes, e cadeias de caracteres [Banc92]. A esses tipos nativos, os projetistas de dados acrescentam os tipos que são específicos de seus domínios

de aplicação, através da definição de "tipos de objetos", que nada mais são do que a definição da estrutura e comportamento dos objetos que sejam do tipo definido.

O comportamento de um objeto é definido por um conjunto de *métodos*, que define como o objeto reage frente a determinadas solicitações externas. Essas solicitações incluem o fato deles serem operados com objetos de mesmo tipo ou de tipos distintos. Por exemplo, objetos nativos de tipo número inteiro podem ser somados, multiplicados, etc. entre si, ou objetos de um domínio específico de tipo custo podem ser somados com um número real. A estrutura de um objeto é definida como o conjunto de propriedades, ou *atributos*, associados ao objeto. Cada atributo pode receber um ou mais *valores*, dentro do conjunto de valores permitido pelo tipo de dado desse atributo.

Este trabalho é desenvolvido utilizando um modelo de dados orientado a objetos específico, denominado SIRIUS [Biaj96], o qual incorpora todos os conceitos de orientação a objetos descrito, além de estender o conceito de tipos de dados com conceito de *"característica de atributos"*. Esse novo conceito agrupa os tipos de dados dos atributos que podem ser associados aos objetos, de acordo com o conjunto básico de operações que são permitidas sobre os atributos de cada característica. Por exemplo: os tipos de dados inteiro, real, real-de-precisão-dupla, etc. podem ser operados pelas quatro operações aritméticas, podem ser comparados por igualdade e ordem, etc., constituindo a característica de *números*. Os tipos de dados cadeia-de-caracteres, texto_ASCII, texto_RTF (padrão Rich Text Format), etc. podem ser concatenados, ter partes substituídas, podem ser comparados por sub-cadeias, etc., constituindo a característica de *texto*. Essas operações podem ser efetuadas mesmo entre dados representados de forma diferente (por exemplo, números com tipo de dado inteiro podem ser multiplicados por números com tipo de dado real, textos com tipo de dado RTF podem ser pesquisados por sub-cadeias com tipo de dados ASCII, etc., mas não se pode somar um número com um texto).

Este trabalho explora esses conceitos quando aplicados à representação de partituras musicais. Nesse contexto, Partitura passa a ser uma característica de dado, que pode ser representada de diferentes maneiras, cada uma sendo um tipo de dado. Este trabalho explora a representação de partituras através do padrão SMF ("Standard MIDI File") de arquivos MIDI ("Musical Instrument Digital Interface") [Gome93], [Hube95], [Stol93], [Ratt95] embora, dentro dos recursos propiciados pelo modelo de dados adotado, outras formas de representação de partituras possam ser igualmente suportadas. Para que uma nova característica de atributos possa ser suportada por um gerenciador de dados, além de definir pelo menos um tipo de dado, é necessário definir também o conjunto de operações básicas associadas à essa característica.

Este trabalho explora as operações de armazenagem, manipulação, busca e recuperação de dados necessárias para que a Característica Partitura possa ser suportada por um Gerenciador de Dados orientado a objetos. Para isso, na seção 2 descreve-se o que considera-se ser uma partitura, e como ela pode ser representada através de um arquivo MIDI. Na seção 3 descrevem-se os passos necessários para a armazenagem e recuperação de partituras em bases de dados, dando-se atenção especial às etapas que devem ser cumpridas para facilitar sua recuperação através de buscas baseadas no conteúdo musical de uma partitura. A seção 4 apresenta algumas características particulares do modelo SIRIUS, e como este oferece suporte para esta nova característica. Por fim, a seção 5 apresenta as conclusões do trabalho.

2. Constituição de Partituras e o Padrão SMF.

Uma partitura é basicamente uma representação bi-dimensional da seqüência de sons que constituem uma música. As dimensões que estão representadas são: as notas musicais que cada instrumento deve soar, e o tempo, o instante quando soa cada nota. Na realidade, uma partitura representa mais dimensões do que as duas citadas, uma vez que uma quantidade representação

dessas duas dimensões é essencial e suficiente para a apresentação deste trabalho[1].

O padrão MIDI [Gome93], [Hube95], [Stol93], [Ratt95] foi inicialmente concebido como uma maneira de instrumentos MIDI musicais trocarem informações sobre a execução de uma peça musical, incluindo-se o início e finalização de quando os instrumentos MIDI devem soar quais notas, e os parâmetros musicais que definem como cada nota deve ser produzida. Na terminologia empregada em sua definição, cada elemento de informação transmitido é denominado um *evento MIDI*. Por exemplo, são eventos MIDI a definição de qual instrumento deve tocar um determinado trecho, o início de uma nota, a finalização de uma nota, um pedal a ser acionado, etc.

Sendo um padrão para troca de dados em tempo real, o padrão MIDI não representa a dimensão temporal, pois cada evento deve ser executado tão logo a mensagem seja enviada (ou o evento deve ser enviado tão logo o executor acione uma tecla, botão, pedal, etc.). Assim, para que representem uma partitura, a seqüência de eventos MIDI deve ser temporizada, acrescentando a cada evento a dimensão temporal. Assim, todos os eventos passam a estar indexados no tempo, e é possível a representação estática da seqüência de eventos que constituem a execução de uma música, e portanto uma Partitura. A incorporação da temporização aos eventos MIDI permite a armazenagem dos dados em arquivos de computador, e isso foi padronizado através da criação do "Standard MIDI File" - padrão SMF [Stol93].

Os arquivos padrão MIDI apresentam uma estrutura que divide os dados em blocos, existindo duas estruturas: o *header* e a *trilha*. O *header* permite o armazenamento de informações gerais da partitura tais como o tipo do arquivo, número de trilhas e o andamento da música. O tipo de arquivo indica variações sobre a estrutura de armazenagem do arquivo, e estão padronizados os tipos 0,1 e 2. Um arquivo de tipo 0 armazena toda a sequência de informações musicais em apenas uma trilha; os arquivos tipo 1 separam as informações musicais em várias trilhas, que são armazenadas sequencialmente no arquivo, e executadas em paralelo; os arquivos tipo 2 também armazenam as trilhas sequencialmente, mas estas são executadas algumas em paralelo e outras seqüencialmente [Stol93].

Trilha é uma estrutura da qual pode haver uma ou mais, dependendo do tipo do arquivo. As trilhas armazenam os eventos da partitura propriamente dita, tais como: as notas que soam e param de soar, o tipo dos instrumentos MIDI, parâmetros de expressão, etc. Todos os eventos são vinculados no tempo, através da indicação de quanto tempo leva para que cada evento ocorra depois do evento anterior ter ocorrido. Pelo fato dessa indicação ser a diferença de tempo entre dois eventos consecutivos, ela é referenciada como *delta de tempo* [Stol93].

Numa trilha de um arquivo MIDI padrão, os eventos são classificados em três tipos: os eventos MIDI, eventos exclusivos do sistema e os meta-eventos.

Os eventos MIDI correspondem aos relacionados diretamente com a execução musical, como o ato de pressionar e soltar teclas, indicação de instrumentos MIDI, as variações de volume, uso de controladores, e demais parametrizações na geração de som. Os eventos exclusivos do sistema são eventos não padronizados, específicos de cada fabricante de instrumentos MIDI para permitir acomodar as particularidades de cada instrumento. Os meta-eventos são os que não afetam a execução musical, mas permitem aos sistemas de software acrescentar informações úteis tanto para o próprio software, como para documentar a partitura. Esses eventos podem por exemplo conter um texto com o nome do instrumento, o nome do autor e a letra da música. A inclusão de meta-eventos em um arquivo MIDI não é obrigatória, e fica a critério do software ou do copista que o utiliza. Além disso, apenas alguns dos meta- eventos possíveis têm seu significado padronizado [Stol93].

---

[1] Outros parâmetros dependem da notação/representação utilizada, e são portanto dependentes do Tipo de Dados adotado, não afetando a definição de Partitura como uma Característica.

A configuração de um sistema MIDI suporta a transferência de dados entre dois instrumentos MIDI através da definição de no máximo 16 canais de comunicação virtuais. Usualmente emprega-se um canal para cada instrumento conectado, embora instrumento aqui possa significar tanto o instrumento físico quanto a forma de geração de som, ou timbre. Por exemplo, um Piano pode ser um instrumento, porém a geração do timbre de Piano pode ser feito por um sintetizador capaz de emular um piano e também o timbre de outros instrumentos MIDI simultaneamente. Cada canal pode se referir a um máximo de 128 timbres. A alocação de timbres de instrumentos MIDI a cada uma das 128 possibilidades não é padronizada através do SMF, porém a convenção *General Midi* fez uma padronização dos timbres, permitindo que qualquer seqüência musical criada de acordo com a especificidade do padrão GM seja executada corretamente em qualquer equipamento que atenda a esse padrão [Stol93].

3. Suporte a Partituras em uma Base de Dados.

Um arquivo MIDI usualmente armazena informações a respeito de uma única música. Os eventos MIDI e os meta-eventos contêm todas as informações necessárias para compor a partitura dessa música, sendo que os eventos exclusivos do sistema adequam a execução da música para um instrumento (ou um conjunto de instrumentos MIDI) específico, e portanto não serão consideradas neste trabalho como integrantes de uma partitura.

Dispondo-se das partituras em arquivos, sua armazenagem em uma base de dados pode ser feita de duas maneiras: copiando-se as partituras para "dentro" da base de dados, ou mantendo-se as partituras nos arquivos originais e gravando-se o nome do arquivo na base de dados. O último enfoque não garante consistência das informações armazenadas na base de dados, pois o usuário pode modificar o arquivo sem que o gerenciador da base de dados tenha conhecimento disso. O primeiro enfoque é então o adotado neste trabalho.

Uma outra decisão de projeto corresponde ao tratamento que deve ser dado na armazenagem da partitura, a qual pode ser vista como um objeto da base, ou como o valor de um atributo. Este trabalho adota a segunda opção, pois permite associar partituras a qualquer objeto. Por exemplo, caso o usuário queira armazenar músicas como objeto, basta criar explicitamente o tipo de objeto Música, o qual pode ter um atributo chamado Partitura, monovalorado com característica de partitura, e cujo valor é a partitura propriamente dita. Dessa maneira, a característica partitura é suportada restringindo-se às informações puramente musicais, ou seja, o valor de um atributo de característica partitura inclui apenas as informações correspondentes aos eventos MIDI de um arquivo SMF. Os meta-eventos passam a ser outros atributos, com características diferentes de partituras, associadas aos objetos aos quais a partitura está atribuída. No exemplo acima, os objetos de tipo música poderiam ter, além do atributo Partitura, um atributo Data_de_Publicação de característica data, e atributos Autores e Letra de característica texto.

3.1. Arquitetura do software.

O Gerenciador de Objetos utilizado foi o sistema SIRIUS [Biaj96], conforme descrito na seção 4. Porém o gerenciador oferece suporte apenas para as operações de armazenagem, manipulação, busca e recuperação de partituras. Para que o usuário possa interagir com o sistema, é necessária a existência de um aplicativo que disponibilize uma interface para essa interação. Para isso foi construída uma interface de acesso, através de um Editor de Partituras, cuja arquitetura básica é mostrada na figura 1. Esse editor atende a dois propósitos: por um lado permite que o usuário efetivamente solicite as operações de armazenagem, manipulação, busca e recuperação de partituras; por outro lado foi o mecanismo através do qual as idéias e a

implementação realizadas foram validadas. É importante notar que os conceitos descritos neste trabalho não estão restritos ao escopo deste editor, porém o mesmo pode ser utilizado como um guia para o desenvolvimento de outros aplicativos que utilizem a armazenagem de partituras em bancos de dados orientados a objetos. O Gerenciador foi construído em ambiente Windows-95.

O gerenciador de objetos [Biaj96] permite que arquivos padrão MIDI sejam armazenados diretamente na base de dados como atributos de característica partitura, bem como permite recuperar tais atributos recriando arquivos padrão. Ao invés de arquivos padrão MIDI, pode-se armazenar e recuperar as partituras para estruturas em memória, permitindo assim que os aplicativos (no caso o editor de partituras) possam operar sobre esses dados. O Editor foi construído de maneira que arquivos padrão MIDI bem como partituras em memórias possam ser ouvidas.

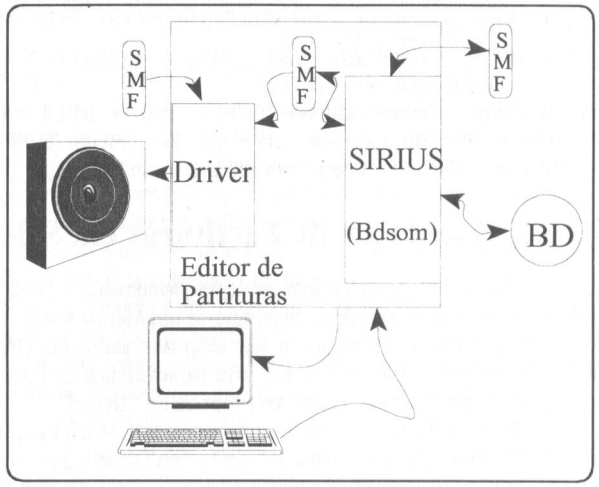


Figura 1- Arquitetura do Software

3.2. Estruturas de Armazenagem de Partituras.

Os arquivos padrão MIDI [Stol93] possuem muitas informações que não correspondem às informações musicais. Apenas os Eventos MIDI são necessários para a efetiva representação de uma Partitura. Assim, quando um arquivo é recebido pelo Gerenciador de Objetos, o mesmo é separado em suas trilhas componentes, e cada uma destas é filtrada, para que somente permaneçam os Eventos MIDI. Os Eventos Exclusivos de sistema são descartados, pois não fazem parte da música em si, mas de uma configuração de instrumentos MIDI.

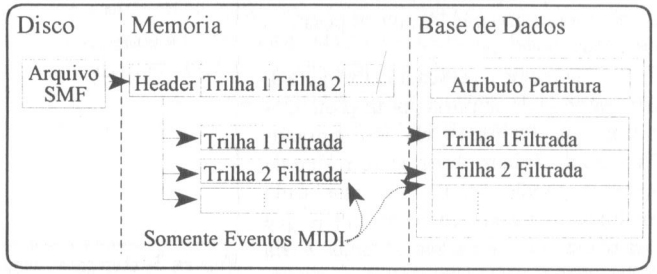


Figura 2- Estrutura de Armazenagem de Partitura

Os meta-eventos são retirados, e submetidos ao próprio Gerenciador de Objetos como atributos Texto com tipos pré-definidos para serem associados ao mesmo objeto onde a partitura está sendo associada. Assim, os atributos partitura são armazenados com um conjunto de informações padronizadas, que correspondem estritamente ao conteúdo musical de um arquivo padrão MIDI. Os outros atributos são armazenados como atributos normais, e após a finalização de uma operação de armazenagem, não são distintos dos que um usuário poderia armazenar

manualmente. A figura 2 mostra as transformações sofridas por um arquivo padrão MIDI até ser armazenado na base de dados. Igualmente, quando uma operação de recuperação é solicitada, um arquivo padrão MIDI é pré-montado buscando as informações MIDI e atributos de outras características. Por exemplo, o seguinte comando (na linguagem SQL estendida suportada pelo Editor):

```
SELECT Autor, Trilha.Instrumento, Partitura
FROM Hinos_Nacionais
WHERE Hinos_Nacionais.Pais = "Brasil
INTO HinoBrasil.MID
```

recria o arquivo HinoBrasil com todos os eventos MIDI gravados, com o atributo textual Instrumento gravado como um meta-evento Instrumento (0xFF04) em cada trilha, e um meta-evento Autor (0xFF08) na primeira trilha gravada [Stol93].

4. Representação de Partituras em SIRIUS.

Um modelo de dados abrangente freqüentemente é capaz de modelar a si próprio. A um modelo com essa propriedade denomina-se um *Meta-Modelo de Dados*. SIRIUS é um meta-modelo, e portanto pode representar suas próprias construções [Biaj96]. O trabalho aqui descrito estendeu SIRIUS para o suporte à nova característica de Partitura, e essa extensão pode ser representada utilizando os conceitos próprios de SIRIUS.

A figura 3 mostra a modelagem da característica Partitura utilizando a notação gráfica de SIRIUS. Nessa figura mostra-se o meta-tipo Objeto, que representa um objeto que define o tipo de todos os objetos, e mostra-se o meta-tipo Atributo, que representa um objeto que define o tipo de todos os atributos. O arco rotulado com Atribui/Atribuído_a representa o fato que Atributos podem ser atribuídos a objetos. A notação  representa o fato de que Partitura é uma especialização de Atributos. Atributos associados a objetos são notados através de uma reta ligando o nome do tipo do atributo ao objeto, tal como a indicação que Atributos possuem uma Característica.

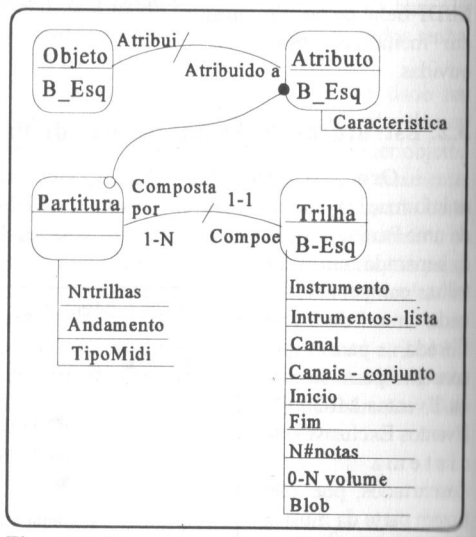


Figura 3- Representação de Partituras no Sirius

Tal como representado na figura 3, uma partitura é composta por uma ou mais trilhas. As Partituras têm os atributos Número de Trilhas, Andamento e Tipo do Arquivo padrão MIDI, pois esse dado modifica a maneira como as trilhas que compõem a partitura são tratadas. Cada Trilha possui os Atributos: Instrumento, usado quando toda a trilha corresponde à dados de apenas um Instrumento; Lista de Instrumentos MIDI, usado quando a trilha concatena dados de diversos instrumentos MIDI; Canal correspondente à trilha; Conjunto de Canais, quando a trilha utiliza mais de um canal; Início e Fim, para indicar quando ocorre respectivamente o primeiro e o último evento da tripla; N# de Notas, para contar a polifonia máxima necessária para a trilha; Volume, para indicar o volume relativo das notas que soam nessa trilha em relação à partitura em geral; e Blob, que

corresponde à sequência de eventos MIDI que constitui a trilha propriamente dita. Os atributos Instrumento e Canal são utilizados para arquivos tipo 1 e 2, para indicar a associação da trilha a um canal MIDI e a um Instrumento. Já os atributos Lista de Instrumento e Conjunto de Canais são utilizados para arquivos tipo 0.

Além das informações descritas, a meta-modelagem da característica Partitura inclui a definição de todo o padrão MIDI e todo o padrão SMF. Além disso, foi pré-definido também o padrão General MIDI. Assim, sempre que um usuário declara a intenção da utilização de atributos com a característica Partitura em sua aplicação, o sistema incorpora automaticamente ao seu esquema de dados: a definição do Tipo de Objeto Instrumento; a definição de 196 objetos de tipo Instrumento, um para cada um dos instrumentos MIDI do padrão GM; a definição de 11 tipos de atributos correspondentes aos meta-eventos padronizados pelo padrão GM; e a definição de 12 tipos de atributos correspondentes aos controladores de parâmetros musicais definidos pelo Padrão MIDI [Stol93]. Todos os tipos de Atributos definidos têm sua sintaxe automaticamente estendida para que possam ser associados a objetos de todos os tipos aos quais o usuário definir que podem ser associados atributos com a característica Partitura.

A definição dos objetos tipo Instrumento e dos Tipos de Atributos permite que o sistema possa extrair informações dos arquivos padrão MIDI armazenados, e armazená-las como atributos dos objetos. Isso facilita as operações de busca envolvendo o conteúdo das partituras, tais como consultas do tipo: "Quais as músicas que utilizam determinado instrumento?".

As consultas envolvendo a localização de seqüências de notas são implementadas de maneira semelhante à busca de sub-cadeias em textos. Assim as mesmas operações de comparação de cadeias (seqüências de caracteres) são permitidas envolvendo seqüências de notas.

As operações de recuperação de partituras são feitas de duas maneiras: geração de arquivos MIDI, e geração de um "response set" em memória. A geração de arquivos é feita individualmente, arquivo por arquivo. A geração de um "response set" permite que um conjunto de atributos sejam gerados, um a um, e submetidos individualmente a tratamento pelo aplicativo. Em ambos os casos, a recuperação dos dados segue o conjunto de atributos que o usuário indica que devem ser incluídos. É interessante notar que com esse enfoque, é possível atribuir dados a partituras manualmente, ou seja, o usuário manipula os atributos textuais de uma partitura independentemente da partitura. Quando uma Partitura deve ser recuperada, os atributos indicados somente são incluídos na estrutura recuperada caso eles tenham um valor definido.

A linguagem de acesso ao Editor de Partituras é uma extensão da Linguagem SQL, adaptada para a manipulação de Objetos [Bert93], adaptada para a manipulação de Partituras, e adaptada para o suporte dos conceitos do SIRIUS. As operações de manipulação de partituras implementadas restringem-se por ora àquelas necessárias para a homogeneização do suporte de partituras para as operações de comparação, sumarizações e busca em partituras.

5. Conclusões.

Este trabalho mostra como integrar o suporte à armazenagem de partituras em um Gerenciador de Objetos, de maneira homogênea com os demais conceitos suportados pelo Gerenciador. Utilizando o enfoque adotado, Partituras são uma característica de atributos a mais suportada pelo gerenciador, da mesma maneira que este suporta números, textos e datas. Usualmente Partituras são atributos de objetos cujo tipo é alguma especialização de Música, porém o enfoque adotado permite ao projetista da aplicação configurar livremente a estrutura que tais objetos possam ter.

Esse enfoque permite manter uma base de partituras de uma maneira bastante uniforme do ponto de vista estrutural, mesmo que as informações armazenadas provenham de fontes

diversas. Assim, muitas formas variadas de consultas podem ser emitidas e obter do sistema uma resposta consistente. Tais consultas são baseadas tanto em informações textuais associadas às partituras, quanto em informações musicais obtidas pelo conteúdo da partitura. As informações textuais podem provir dos próprios arquivos padrão MIDI utilizados para a coleta de informações, quanto podem ser inseridas individualmente pelo usuário. A linguagem de consulta é baseada na linguagem SQL, que é o padrão para acesso a bases de dados relacionais, e vem se tornando muito comum em Gerenciadores de Objetos.

O sistema é composto basicamente de duas partes: um editor de partituras, que provê o acesso à interface com o usuário e com o módulo de geração de som, e que interpreta a linguagem de comando da interface; e pela extensão feita no Gerenciador de Objetos SIRIUS, para suporte à Característica de Atributos Partitura. O Editor está implementado em plataforma Windows 95, escrito em Borland C++. O Gerenciador SIRIUS possui versões para plataforma Windows e SunSolaris, e a extensão implementada mantém a compatibilidade com ambas as plataformas, uma vez que o suporte oferecido pelo Gerenciador de Objetos restringe-se à armazenagem e recuperação de partituras, não dependendo de "drivers" específicos para a reprodução das partituras. Assim, este sistema aproveita todos os recursos de SIRIUS como uma arquitetura Cliente/Servidor multiplataforma, com estações cliente operando em Windows.

6. Bibliografia.

[Banc92] Bancilhon, F.- **"The O_2 Object-Oriented Database System"**, Proceedings of the 1992 ACM Sigmod - International Conference of Management of Data, San Diego, Califórnia, vol.2, No.1, p.7, June 2-5, 1992.

[Bert93] Bertino, E.; Lorenzo, M.- **"Object-Oriented Database Systems"**,International Computer Science Series, Addison-Wesley, 1993.

[Biaj96] Biajiz, Mauro - **"Representação de Modelos de Dados Orientados a Objetos através de Parametrização de Abstrações"** - Tese de Doutorado apresentado ao IFSQ - Instituto de Física de São Carlos, setembro/96.

[Elma94] Elmasri, R.; Navathe, S.B.; **"Fundamentals of Database Systems"**, Addison Wesley Publishing Company, 1994, 2a.edição.

[Gome93] Gomes, T.A.; Neves, A.- **"Tecnologia Aplicada à Música"**, Editora Érica, 1993.

[Hube95] Huber, M.D.- **"The MIDI Manual"**, SAMS, Prentice Hall Computer Publishing, 1995.

[Ratt95] Ratton, M.- **"Criação de Música e Sons no Computador"**, Editora Campus, 1995.

[Stol93] Stolz, A.- **"The Sound Blaster Book"**, Abacus, 1993.

[Zand95] Zand, M.; Collins, V.; Caviness, D.- **"A Survey of Current Object-Oriented Databases"**, Database Advances, vol.26, no.34, pp.14-29, February 1995.

ProbSys - Um Sistema Probabilístico Voltado à Criação Musical

GILBERTO CARVALHO*
VLADIMIR AGOSTINI CERQUEIRA**

Abstract

This paper aims to describe the ProbSys, a probabilistic system for musical composing. Established in propositions by José Vicente Asuar, the system looks at to generate the parameters in a musical discourse by means of probability curves peculiar to every one of them. For this, it is used a set of curves previously stored in disk and, at every new section in the work to be generated, it is constructed a tree with pointers to every element of that set.

0. INTRODUÇÃO

O projeto descrito neste artigo foi idealizado a partir das idéias de José Vicente Asuar[1] e realizado entre agosto de 1995 e julho de 1996. Nele trabalharam os autores deste artigo e o bolsista Sérgio Silva Gomes, aluno da Escola de Música da UFMG.

Apesar da época já recuada em que o conhecido autor chileno escreveu suas proposições, estas nos pareceram adequadas por três motivos principais. Em primeiro lugar, o autor expõe seus procedimentos através de uma linguagem simples e da qual estão ausentes formalismos que seriam de difícil compreensão por alunos de uma escola de música, tais como os expostos por Xenakis[4] e Lorrain[2]. Além disto, levando em conta o fato de que o projeto original de Asuar exigiu soluções extremamente caras quando de sua realização, a implementação num microcomputador foi vista como um desafio. Finalmente, o projeto de Asuar se nos apresentava como um documento histórico sobre a utilização musical de computadores na América Latina. Levando em conta que o ele foi idealizado e realizado há vinte e cinco anos atrás, este último aspecto não é de forma alguma irrelevante.

Neste artigo, procuraremos apresentar de forma concisa o funcionamento do sistema resultante da nossa implementação.

1. O SISTEMA

O ponto chave do sistema, ao qual demos o nome de **ProbSys**, é a utilização de curvas de probabilidade para gerar todos os parâmetros componentes de um discurso musical. Este é composto de quatro ferramentas:

a) **Write**
Escreve em disco um conjunto de curvas padrão que serão posteriormente utilizadas nos cálculos dos parâmetros.

* Professor da Escola de Música da UFMG . Av. Antônio Carlos, 6553 - Campus da Pampulha. 30.220-330 - Belo Horizonte - Minas Gerais. Brasil. agmc@oraculo.lcc.ufmg.br
** Aluno da Escola de Música da UFMG e bolsista de Iniciação Científica pelo CNPq.

b) **Custom**
Possibilita a customização de curvas preexistentes.

c) **Designer**
Possibilita o desenho de novas curvas.

d) **Probsys**
Gera um texto musical definido por instruções constantes de um arquivo de entrada fornecido pelo usuário e por um conjunto de curvas de probabilidade previamente gravadas em disco. É a ferramenta mais importante do sistema, já que as outras somente servem de suporte à sua atuação.

Neste texto nos deteremos somente no modo de operação da ferramenta principal (que, a partir de agora, chamaremos pelo nome) e, mesmo assim, não será possível aborda-la em todos os seus detalhes.

Na hierarquia do **ProbSys**, a estrutura mais baixa é a **obra musical**. Uma obra musical pode ser constituída por várias **seções**. Cada seção contém uma **árvore** e vários **estratos**. Cada estrato contém um conjunto de **eventos**, os quais são gerados com uma certa **tipologia** numa determinada **sintaxe**. Isto pode ser resumido assim:

1.1. Seção

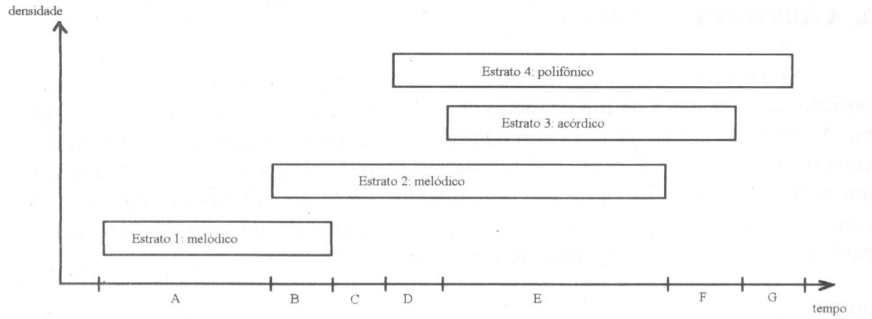
Denominamos **seção** às maiores divisões do texto musical a ser gerado. Sempre, ao início de uma seção, uma nova árvore de curvas de probabilidades é calculada. Uma seção compreende tanto os estratos quanto a árvore de curvas que será por eles utilizada.

1.2. Árvore

Para a descrição da árvore de curvas, ver abaixo no item 2 (A Árvore de Curvas).

1.3. Estrato

O estrato é uma estrutura que compreende, dentro do texto musical a ser gerado, uma textura básica, um ponto de partida e uma duração. Podemos, através de combinação de estratos, obter uma ampla gama de texturas mistas, como no exemplo abaixo:



Sobre os tipos de textura possíveis num extrato, ver abaixo no item 3 (Cálculo de parâmetros).

1.4. Evento

O evento é a menor unidade musical a ser gerada. É composto dos parâmetros musicais clássicos: altura (ou freqüência), intensidade, instante de ataque e duração.

1.5. Sintaxe

A sintaxe diz respeito tanto ao modo como os eventos de um determinado estrato serão calculados quanto ao modo através do qual eles serão armazenados no arquivo de saída. Existem dois tipos de sintaxe no **ProbSys**:

a) **SICX**: gera uma saída no formato *.scx (que pode ser convertida para MIDI).[1]

b) **Csound**: gera uma saída no formato de um *score* de Csound.[2]

1.6. Tipologia

Tipologia é o modo como é calculada a freqüência de um evento nos estratos melódicos e acórdicos, e como são calculados a freqüência, duração e instante de ataque nos estratos polifônicos. Existem dois tipos de tipologia:

a) **Espaço estriado**: os parâmetros de um evento são tratados de modo discreto.

b) **Espaço liso**: os parâmetros de um evento são tratados de modo contínuo.

Para a utilização de espaço liso em estratos melódicos e acórdicos é necessário faze-lo dentro da sintaxe **Csound**.

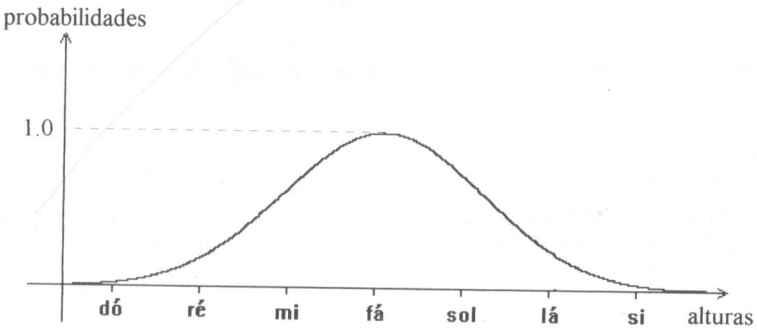
---

[1] SICX (Sistema Integrado de Composição e eXecução) é um projeto que vem sendo desenvolvido por Gilberto Carvalho no Departamento de Teoria Geral da Música, Escola de Música da UFMG. O ProbSys é parte integrante do SICX.

[2] Para um descrição das possibilidades do Csound, consultar Vercoe[3].

2. A ÁRVORE DE CURVAS

O discurso musical tradicional pode ser interpretado como uma determinada disposição dos elementos de um conjunto de parâmetros durante um certo intervalo temporal. No discurso tradicional, o valor dado a cada parâmetro é função de uma escolha do compositor. Se, ao invés disto, condicionamos as diferentes possibilidades de aparição de um parâmetro a uma curva de probabilidades, teremos um reflexo musical das características desta curva. Tomemos, por exemplo, as alturas de uma escala diatônica como parâmetros e uma curva de distribuição normal:



O resultado privilegiará as notas mi , fá e sol, uma vez que estas possuem as maiores probabilidades de ocorrência. Se nosso conjunto de parâmetros for formado por apontadores que levem a outras curvas de probabilidade, chegaremos à construção orgânica de uma árvore cujos nodos serão escolhidos a partir do nodo pai e de um conjunto de curvas possíveis.

Cada escolha realizada por uma curva leva a outra curva num nível superior da árvore, e assim por diante, até atingirmos, no último nível, os nodos que serão utilizados para o cálculo dos parâmetros musicais. Quando, por qualquer razão, houver um impasse na determinação de um parâmetro, existem indicadores que forçam a descida ao nível necessário para que seja escolhido um novo caminho a um novo nodo que resolva o problema.

Levando em consideração o que foi dito, foi implementada uma árvore a partir da estrutura de dados mostrada abaixo:

```
struct treenode {
      int curve;
      int nsons;
      float lifetime;
      struct treenode *father;
      struct treenode *sons[MAX_SONS];
};
```

Cada nodo é do tipo *treenode* e seus membros são descritos a seguir:

- *curve* é o número da curva de probabilidades representada pelo nodo.
- *nsons* é o número de filhos do nodo.
- *lifetime* é o tempo de vida da curva de probabilidades representada pelo nodo.
- *father* é um apontador para o pai do nodo.
- *sons* é um *array* de apontadores para os filhos do nodo.

Partindo da raiz, cada novo nodo é calculado levando em conta as características probabilísticas de seu pai. Para o cálculo do número de filhos de cada um dos nodos também é utilizada a curva pai, mas desta vez o conjunto sobre o qual esta curva atuará tem como extremos os números máximo e mínimo de filhos, sendo que os valores intermediários são encontrados através de interpolação linear.

O campo *lifetime* controla o tempo de atuação de uma curva. O *lifetime* se define pelas características da própria curva e pelos dados de entrada fornecidos pelo usuário. Extinto o *lifetime* de uma curva, o programa retorna a raiz da árvore e sobe novamente até seu nível mais alto, através de um possível novo caminho, encontrando uma nova curva de escolha paramétrica.

No **ProbSys**, o usuário pode indicar a profundidade (de quantos níveis se compõe a árvore), o máximo e o mínimo tempos de vida desejados para as curvas, o máximo e o mínimo números de filhos de um nodo, a curva-raiz e quais curvas deseja utilizar como filhos (ele pode usar curvas já existentes, modifica-las ou construir outras tantas). Naturalmente, existem valores *default* para todas estas variáveis. Finalmente, qualquer parâmetro do texto musical a ser gerado pode ser considerado como elemento de um conjunto a ser colocado no último nível da árvore. Para chegar à curva associada a um determinado parâmetro o programa percorre, nível a nível, um percurso ditado pelas características probabilísticas de cada nodo alcançado.

3. CALCULO DE PARÂMETROS

No **ProbSys**, podemos considerar três tipos de texturas básicas:

a) **Textura melódica**: não existe simultaneidade temporal de eventos.

b) **Textura acórdica**: existe simultaneidade de eventos dependentes.

c) **Textura polifônica**: existe simultaneidade de eventos independentes.

Todo evento no **ProbSys** estará contido em uma destas texturas básicas. Para a criação de outros tipos de textura é necessário utilizar uma combinação das já existentes.

Como cada estrato é capaz de conter somente um tipo de textura, é necessário, durante o cálculo de cada um deles, realizar a escolha:

```
Se TEXTURA = "Homofônica"
    Se TEXTURA = "Melódica"
        chame função Melódica;
    Senão
        chame função Acórdica;
Senão
    chame função Polifônica;
```

Cada evento de um trecho musical saído do **ProbSys** pode ser entendido como a soma dos seguintes parâmetros:

- Altura (ou freqüência) do evento
- Intensidade do evento
- Instante de ataque do evento
- Duração do evento

Como, no instante inicial, o programa sobe a árvore independentemente para cada parâmetro, isto leva a diferentes curvas para a escolha de alturas, intensidades, instantes de ataque e durações, sem esquecer dos tempos de atuação das curvas (*lifetime*) de cada um dos respectivos parâmetros.

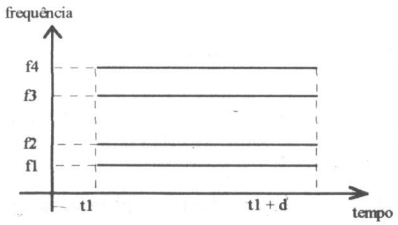
Resta ainda um tópico a respeito do calculo de parâmetros. É o que denominamos de "direcionalidade" ou "incidência". Esta variável define o número de consultas que se fará a uma curva no momento do cálculo de um determinado parâmetro. Após realizar as consultas, armazena-se os resultados e verifica-se qual deles foi mais freqüentemente encontrado. Este valor, então, é considerado o resultado final. Com esta técnica simples, é possível a eliminação temporária de elementos de baixa probabilidade.

3.1. Textura melódica

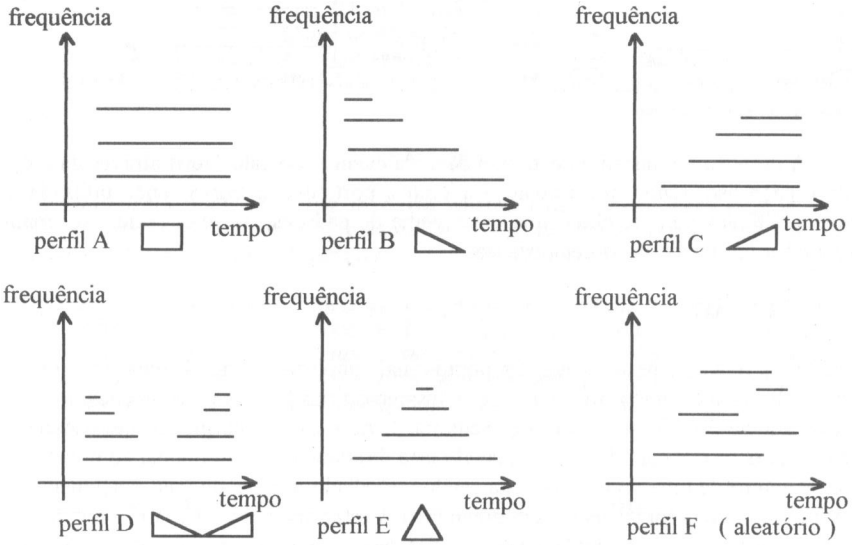
Na textura melódica, antes que se escolha um novo evento, determina-se seu instante de ataque como sendo igual ao ataque do evento anterior mais sua duração. Desta forma, não haverá simultaneidade, mas apenas sucessão de eventos.

3.2. Textura acórdica

A forma mais simples possível de um acorde é aquela em que seus componentes iniciam e terminam de soar simultaneamente. Assim sendo:



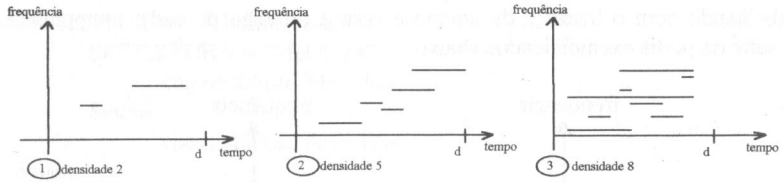
Trabalhando com o instante de ataque e com a duração de cada componente, obtemos a série de perfis exemplificados abaixo:



Tomando estes perfis como primitivas, implementamos uma função que, a partir de um evento melódico, constrói sobre ele eventos dependentes, gerando uma textura acórdica. Os parâmetros para a construção do acorde são a densidade, o perfil e a interválica. Para cada um destes parâmetros serão utilizadas curvas selecionadas pelo processo descrito anteriormente. Define-se como densidade o número de eventos dependentes simultâneos que deverão ser calculados a partir do evento melódico básico. O perfil é escolhido dentre os possíveis no conjunto mostrado acima, podendo variar entre os tipos 1 a 6. Para o calculo do perfil a duração do evento melódico básico é tomada como referência. A interválica é o conjunto das relações intervalares possíveis entre o evento melódico básico e os eventos dependentes. Estas podem ser indicadas em semitons ou através de proporções. As considerações anteriores sobre o tempo de vida (*lifetime*) de uma curva também são validas, naturalmente, para as curvas obtidas para densidade, perfil e interválica.

3.3. Textura polifônica

Um estrato polifônico pode ser, obviamente, o resultado da superposição de dois ou mais estratos melódicos. Porém, um tipo especial de polifonia, baseado no sexto perfil acórdico, pode ser obtido preenchendo, de modo assíncrono, uma determinada duração D por meio de uma densidade N de eventos. Tal processo pode ser considerado como uma variante daquele empregado na síntese granular no que diz respeito à geração dos grãos, residindo a principal diferença na dimensão dos eventos gerados, os quais são muito maiores no nosso caso.



Estrato Polifônico: 3 exemplos do processo

A duração e o instante de ataque de cada evento são calculados através da técnica do espaço liso, sendo diretamente resultantes, portanto, da curva sendo utilizada. A textura polifônica permite obter uma vasta gama de padrões, os quais podem ser muito úteis em formas musicais contemporâneas.

4. CONCLUSÃO

Procuramos mostrar, neste artigo, os pontos mais importantes do sistema por nós implementado. Como tópico principal a ser considerado nas futuras manutenções do sistema está aquele do gerenciamento de memória. É necessária uma grande quantidade de memória para a operação do sistema, sendo esta diretamente proporcional ao número de estratos dentro de uma seção. Isto está sendo remediado através de uma implementação em Windows que já foi iniciada e em breve deverá estar disponível. Como resultado prático do projeto, além das implementações das ferramentas do sistema, desenvolvemos uma peça para quinteto de sopros, uma peça para trompa solo e uma peça para fita magnética.

5. REFERÊNCIAS

[1] ASUAR, José Vicente. ¿Musica con computadores, como hacerlo?. *Revista Musical Chilena*, 1972, 118, 36-74.

[2] LORRAIN, Denis. A panoply of sthochastic 'cannons'. In: ROADS, Curtis (Ed..). *The music machine*. Cambridge: MIT, 1989. p. 351-379.

[3] VERCOE, Barry. *Csound: a manual for the audio processing system and supporting programs with tutorials*. Media Lab (MIT), 1994.

[4] XENAKIS, Iannis. *Musiques formelles*. Paris: Stock, 1981.

A research in progress:

Music, new technologies and Latin America

Ricardo Dal Farra
Estudio de Música Electroacústica
Azcuénaga 2764 - (1640) Martinez - Buenos Aires - Argentina
Tel: (+54-1) 553-3015 // Fax: (+54-1) 827-0640 // Email: dalfarra@clacso.edu.ar

Abstract

Many composers born or living in Latin America have been very active on the electroacoustic/computer music field for a long time. On some countries of the region experiences started around 40 years ago.

This paper is a report (No. 2) from a work in progress searching to compile information about composers born or living in Latin America who have been producing electroacoustic and computer music. References are sorted by countries, including composers born or living in Argentina, Bolivia, Brazil, Chile, Colombia, Cuba, Dominican Republic, Ecuador, Guatemala, Mexico, Panama, Peru, Puerto Rico, Uruguay and Venezuela.

1. Latin America and the electroacoustic/computer music field

The electroacoustic and computer music field have in Latin America a long, interesting, strong, prolific and not very well known (even inside the region) history. My proposal with this research is to collaborate to know more about electroacoustic and computer music activities related, in a broad sense, with Latin America.

The following list name composers who had been working with electroacoustic medias and computers on their music. Most of them born in Latin America and are living here at the moment, some are living outside the region, and a few did not born in Latin America but are living here now.

It should be considered also that on the same list are included composers devoted almost exclusively to electroacoustic and computer music, together with composers that perhaps have only one electronic or mixed piece among their list of works. There was not discrimination, nor a selection process to decide who should be included or not; the names listed were taken from bibliographical information, program notes from concerts and recordings, and direct or indirect contact with composers and institutions.

This is a research in progress, and for sure that I am not mentioning everyone, as there are many composers that probably should be included and that I do not know or I forgot unintentionally.

1.1 Argentina: Mauricio Kagel, César Franchisena and Francisco Kröpfl were experimenting with electroacoustic medias since the '50s. The Universities of Córdoba and Buenos Aires were main centers of activities at the beginning. During the '60s, the Centro Latinoamericano de Altos Estudios Musicales (CLAEM) at the Instituto Torcuato Di Tella was a meeting point for students and composers from Latin America to learn and exchange ideas with the most interesting composers from Europe and the United States at that moment. At present there are several studios in Buenos Aires, and also in Córdoba, Rosario, and Santa Fé.

The list of composers born or living in Argentina that have been around electroacoustic and computer music is very long. Some of them are: Daniel Almada ("Linde", vibraphone and tape; he is living in Europe), Claudio Alsuyet ("Solo Saxo Barítono", sax and live electronics), Jorge Arandia Navarro, Isabel Aretz ("Hombre al Cosmos", piano and electroacoustics, 1993; now living in Venezuela), Oscar Bazán ("Austera", 1973, "El valle de las profecías", 1989), Enrique Belloc ("Rugosidades del inconsciente colectivo", 1995), Eduardo Bértola ("Dynamus", 1970), Gonzalo Bifarella, Osvaldo Budón ("Para el trato con el desierto", 1992), Miguel Calzón, José Luis Campana (he is living in France), Cecilia Candia, Edgardo Cantón ("I palpiti", 1966), Ofelia Carranza, Pedro Caryevschi ("Analogias paraboloides", 1970; now living in Israel), Graciela Castillo ("Concreción", 1966), Carlos Cerana ("Huellas digitales" electroacoustic music, 1990), Pablo Cetta ("...que me hiciste mal...", for tape, 1992), Gustavo Chab, Eduardo Checchi, Marcelo Cosentino, Teodoro Cromberg ("Gritos y gritarras", for tape, 1993), Ricardo Dal Farra ("Karma", tape, 1986; "Mel18", live interactive computer music, 1994), Mario Davidovsky ("Synchronisms No.2", for flute, violin, cello, clarinet and electronic sounds, 1964; "Synchronisms No.6", for piano and tape, 1970; living in the U.S.A.), Marcelo Delgado, Hilda Dianda ("Dos estudios en oposición", for tape, 1959), Pablo Di Liscia, Gerardo Dirié (living in the U.S.A.), Hugo Druetta, Pedro Echarte ("Estudiantina", 1965), Oscar Edelstein, Mariano Etkin, Carlos Ferpozzi, Beatriz Ferreyra ("Siesta Blanca", 1972), Sergio Fidemraizer (he is living in Spain), Juan Carlos Figueiras, Lionel Filippi, Héctor Fiore ("Las vaquitas son ajenas", for charango, computer and tape, 1993), César Franchisena ("Numancia", 1960; "Tres momentos mágicos", 1973; "Canticum", 1988), Martín Fumarola ("El peregrinar de la araña", 1995), Pablo Furman ("Sureña", for amplified violin and electronics, 1993; now living in the U.S.A.), Gerardo Gandini, Javier Garavaglia ("Arte Poética (I)", 1995; now living in Germany), Eleazar Garzón ("La morada del cóndor", 1994), Enrique Gerardi ("Lejanía", 1990), Arturo Gervasoni (now living in France), Silvia Goldberg, Luis Jorge Gonzalez (he is living in the U.S.A.), Carlos Grätzer ("Nio Aeln", 1989; he is living in France), Dante Grela ("Glaciación", 1979), José Halac ("India vieja que pretende volverse joven chupando sangre de condor", 1990; he is living in the U.S.A.), Rufo Herrera (he was born in Argentina, but have been living in Brazil for many years), David Horta, Alejandro Iglesias Rossi ("Angelus", for tape, 1996), Elsa Justel ("Ischihualasto", 1989; she is living in France), Mauricio Kagel ("Música para la torre", sound installation using recorded sounds, 1954; "Transición II" for piano, percussion and 2 tape recorders, 1958-1959; "Szenario", for strings and tape, 1983; he have been living in Europe for many years), Damián Keller ("Persecución", 1987), Armando Krieger ("Contrastes", for 2 pianos and tape, 1963), Francisco Kröpfl ("Ejercicio de texturas", 1960; "Orillas", 1988), Bernardo Kuczer, Eduardo Kusnir ("La Panaderia", 1970; "Buenos días, fué un buen día", 1991; now living in Venezuela), Alcides Lanza ("Interferences I", 1966; "Ekphonesis III", 1969; he is living in Canada), Javier Leichman, Claudio Lluán, Fernando López Lezcano ("Three Dreams", 1993; he lives in the U.S.A.), María Eugenia Luc (she is living in Italy), María Teresa Luengo ("Absolum", for tape, 1973), Fabián Luna, Tomás Luzián, Ricardo Mandolini ("Círculos fosforescentes en fondo negro", 1982; "De mí huían los pájaros", 1983; he lives in France), Rolando Mañanes, Virtú Maragno, Lucía Maranca, José Maranzano ("Mnemón I", 1970), Edgardo Martinez, Patricia Martinez, Mario Marcelo Mary (he is living in France), Martín Matalón (he is living in France), José Mataloni, Luis Mihovilcevic, Raúl Minsburg, Daniel Miraglia, Jorge Molina, Antonio Moliterni, Nelly Moretto ("Composición 9b", 1966), Luis Mucillo, Luis Naón (he is living in France), Jorge Naparstek, Ricardo Nillni ("Entropogel", 1988; now living in France), Pablo Ortiz (he is living in the U.S.A.), Ricardo Palazzo, Juan Carlos Pampín, Graciela Paraskevaídis (she lives in Uruguay), Alberto Paulín, Stella Perales, Alberto Perduca, Ricardo Pérez Miró, Eduardo Piantino, Sergio Poblete, Fernando Polonuer, Guillermo Pozzati, Jorge Rapp ("Evocaciones", 1988), Augusto Rattenbach, Miguel Angel Rondano, Carlos Roqué Alsina (he lives in Europe), Michael Rosas Cobian (he lives in England), Jorge Rotter, Jorge Sad, Carmelo Saitta (born in Italy), Patricia San Martín, Luis Maria Serra ("Invocation", 1969), Daniel Schachter, Sergio Schmilovich, Carlos Simkin, Miguel Angel Sugo, Antonio Tauriello,

Eduardo Tejeda ("Estudio Electrónico No. 1", 1968), Daniel Teruggi ("E cosi via", 1985; "Syrcus", 1992; "Saxtenuto", 1994; he lives in France), Alicia Terzián ("Canto a mi misma", 1987), Virgilio Tosco ("Complejo No. 2", 1965), Claudio Tripputi, Horacio Vaggione ("Música Electrónica I", 1960; "Fausto", 1966; "Ash", 1990; he lives in France), Gabriel Valverde, Osvaldo Vázquez, Mario Verandi, Julio Viera, Jorge Villar, Alejandro Viñao ("Go", for tape, 1981; "Chant d'Ailleurs", for soprano and computer, 1992; he lives in England), Ezequiel Viñao (now in U.S.A.), and Daniel Zimbaldo ("Naturaleza muerta", 1984; he lives in Spain).

1.2 Bolivia: Alberto Villalpando (born in La Paz, 1940) is considered one of the main driving forces on contemporary music in his country. He have been working with electroacoustic medias for many years. Some of his pieces are: "Bolivianos..." (1973); "Desde el Jardín de Morador" (1990-91); and "De los Elementos" (1991).

Composers like Agustín Fernández (now living in Europe), Javier Parrado, Florencio Pozadas ("CM-Op.1", for percussion and tape, 1968), Cergio Prudencio ("Awasqa", 1986), Juan Siles Hoyos, Nicolás Suárez ("Chica Aruma", 1994) and Gerardo Yañez have been working with electroacoustic medias on their music too. There had been also some collaborative works by Oscar García and Sergio Claros ("ES ZAS").

1.3 Brazil: Reginaldo Carvalho and Jorge Antunes were pioneers on the electroacoustic music field in Brazil (= Brasil). "Si bemol" (1956), "Temática" (for tape, 1956), "Troço I" (for tape, 1956), "Troço II" (for tape, 1957), "Alegria de Natal" (for mixed chorus and tape, 1963-64), "Piano Surpresa No.1" (for tape, 1965), "Cleta" (for tape, 1966), "Cemiterio sem Flores" (for tape, 1966) and "Caleidoscópio III" (for tape, 1967) are among the early works by Carvalho (born in 1932). Jorge Antunes composed "Pequena peça para MI bequadro e harmônicos" during 1961, and have been very active working with electronic medias since then; some of his compositions are: "Valsa Sideral" (for tape, 1962), "Cinta Cita" (for tape, 1969), "Proudhonia", (for mixed chorus and tape, 1972), "Catastrophe Ultra-Violette" (for male chorus, orchestra and 3 tapes, 1974), "Intervertige" (for string quartet, wind quintet, 2 percussionists and electronic equipment, 1974), "Mixolydia" (for Theremin and tape, 1995), "Agenda pour un petit futur" (for tape, 1995).

Brazil have an interesting activity around electroacoustic and computer music. At present there are several studios and research groups working; some of them are based at University of Brasilia, State University of Campinas, Federal University of Minas Gerais (at Belo Horizonte), University of São Paulo, Federal University of Bahia (at Salvador), Federal University of Paraíba, Federal University of Rio Grande do Sul (at Porto Alegre), Federal University of Santa Catarina, Federal University of Espírito Santo (at Vitória), and Studio PANaroma at UNESP.

Among the composers who have been working on the field are: Celso Aguiar (born in Palo Alto, U.S.A., he grew up in Salvador, Bahia), Paulo Álvares (now living in Germany), Aluízio Arcela ("Cinco Hierarquias Espectrais", 1993; "/cartas/rs95.car", 1995), Rodolfo Caesar ("Curare I", "Carne da pedra", "Volta Redonda", "Nemietoia"), Guto Caminhoto, Eduardo Campolina, Lindembergue Cardoso, Gilberto Carvalho, Clodomiro Caspary, Fernando Cerqueira, Paulo Chagas ("Ellipse", for tape, 1986; he is now living in Germany), Sérgio Igor Chnee, Rodrigo Cicchelli Velloso ("Latitudes Emaranhadas", 1992-93), Rodolfo Coelho de Souza ("Diálogos", for tape, 1990), Willy Corrêa de Oliveira ("Materiales", 1981), Ilza Maria Costa Nogueira, Luis Carlos Csëko ("Sound", for voice and tape, 1992), Vânia Dantas Leite ("Sforzato/Piano", for tape, 1994), Jocy de Oliveira ("Estória II", for voice, percussion and tape, 1967; "Wave Song", for piano and tape, 1977; "For Cello", for cello and tape, 1996), Raul do Valle ("Encadeamento", for tape, 1979), Rogério Duprat ("Experimental Music", 1963), Aylton Escobar ("Quebradas do Mundaréu", for tape, 1975), Marlene Fernandes, Sílvio Ferraz ("Casarío, terreiro com igreja ao fundo", for tape, 1995, Sérgio Freire

("Baurembi"), Denise Garcia ("Um dia feito d'agua", 1993), Anselmo Guerra de Almeida ("Proporcões"), Didier Guigue (he born in France), Eduardo Guimarães Alvares ("Estórias", 1987), Rufo Herrera ("Ambitus Mobile I"; he was born in Argentina, but have been living in Brazil for many years), Hubert Hans Hoffmann, Fernando Iazzetta ("PerCurso", for tape, 1996), Arthur Kampela ("Textórias", 1994; he is now living in the U.S.A.), Damián Keller (born in Argentina; see that section), Hans-Joachim Koellreutter, Fábio Kon, Harry Lamott Crowl Jr. ("Convivium", for tape, 1986), Marc Lannelongue (now living in France), Victor Lazzarini ("Vozes Dentro", for tape), Igor Lintz Maués ("15015", for tape, 1981; "Antes o mundo nao existia", for tape, 1989; "Durch unsere stadt zum tor hinaus" for tape, 1990; he is now living in Austria), Mauricio Alves Loureiro ("On behalf", for piccolo clarinet and tape, 1991), Vanderlei Lucentini, Mikhail Malt ("λ 3.99", for tape, 1994; now living in France), José Augusto Mannis ("Cyclone", for tape, 1983; "Synapses", for french horn and live electronics, 1987), Jônatas Manzolli ("Luvas de Pelica"; "Turbulências"), Roberto Martins, Silvia Matheus ("Influx", for tape; she is living in the U.S.A.), Chico Mello ("Todo Santo", 1989; now living in Germany), Gilberto Mendes ("Nascemorre", 1963; "Son et Lumière", 1968; "Santos Football Music", 1969), Flo Menezes ("La (de)marche sur le grains", for tape; "Parcours de l'entité", for amplified flutes, metal percussion instruments and digital tape), Ronaldo Miranda, Jaime Mirtembaum, Amaro Borges Moreira Filho, Hélcio Müller, Lelo Nazário, Jose Maria Neves ("UN-x-2", for tape, 1971), Gil Nuno Vaz, Jamary Oliveira, Marcos Olívio, Eduardo Paiva, Aquiles Pantaleão ("Materialma", 1995), Luis Roberto Pinheiro ("Fuzuê"), George Randolph, Eduardo Reck Miranda ("Italo Calvino takes Jorges Borges on a taxi journey in Berlin", for tape, 1993; "The Flying Scotsman", for tape, 1996; he is now living in England), Luiz Augusto (Tim) Rescala ("Ponto, Linha e Plano", for clarinet and computer, 1990), Antonio Celso Ribeiro, Frederico Richter ("Metamorfose", "Estudo Eletrônico"), Claudio Santoro ("Mutationen I", 1968; "Estudo para fita magnética", 1976), Lourival Silvestre, Wilson Sukorski ("A Batalha de Adrianopolis", 1994), Tato Taborda, Ricardo Tacuchian, Antonio Carlos Tavares, Vera Terra, Geraldo Henrique Torres Lima, Livio Tragtenberg ("Festival de Estranhezas", 1993), Correia Vasconcelos, Ernst Widmer ("ENTROncamentos SONoros", for piano, 5 trombones, strings and tape, 1972; born in Europe), Bernadete Zagonel, Edson Zampronha ("Modelagem V", for tape, 1995-96).

Conrado Silva (born in Uruguay, see that section for more references) is another composer who helped the development of electroaocustic music in Brazil.

1.4 Chile: Some of the earliest electroacoustic music works in Latin America were "Nacimiento" (1956) by León Schidlowsky (born in Santiago, 1931; later he adopted the Israel's nationality), "Los Peces", composed in 1957 by Juan Amenabar (Santiago, 1922), and "Variaciones espectrales" by José Vicente Asuar (Santiago, 1933) produced during the next year. Asuar built the first Electronic Music Studio of Chile in 1958.

Since then, Chilean composers like Miguel Aguilar-Ahumada, Francesca Ancarola, Jorge Arriagada, Gustavo Becerra-Schmidt ("Historia de una Provocación", 1972; he is now living in Germany), Gabriel Brncic ("Dialexis", 1966; "Batucada amenazante para los que huyen", 1969; he is now living in Spain), Eduardo Cáceres ("Metalmambo", 1994), Fernando Carrasco Pantoja, Rolando Cori Traverso ("Fiesta", 1989), Ernesto Holman Grossi, Jorge Martinez Ulloa, Mario Mora, Cristián Morales Ossio, Iván Pequeño ("Motete con Huesillos", 1972), José Perez de Arce, Guillermo Rifo, Iris Sangüesa, Juan Carlos Vergara Solar and Santiago Vera-Rivera have been working with electroacoustic medias on their compositions.

1.5 Colombia: Blas Emilio Atehortúa ("Cantico delle creature", mixed piece from 1965), Jacqueline Nova ("Resonancias I", for piano and tape, 1969; "Cantos de la creación de la tierra", 1972; she born in Belgium in 1937, and died in 1975), and Fabio Gonzalez Zuleta ("Ensayo electrónico", for tape, 1965) were some of the first composers to work with electroacoustic music in Colombia.

Andrés Posada ("Catenaria", for tape, 1990) and Juan Reyes ("Diálogos por Paz", "El Espectador", "Las Meninas"; born in Bogotá, 1962) have been working during the last years with electroacoustic and computer medias, and also directing electronic music studios, Posada at the University of Manizales (now this studio is closed) and Reyes at the University of los Andes. Ricardo Arias ("Sinfonía Global"), Carlos Mauricio Bejarano ("Aparato I", for tape, 1990), Guillermo Carbo ("Frecuencia modulada", 1990), Fabio Fuentes, Roberto García, Guillermo Gaviria, Víctor Hernández ("El designio de la paz"), Francisco Iovino, Gustavo Lara, Catalina Peralta ("Episodios sin conexion de la vida de un artista", for tape, 1989), Andrés Pinzón Urrea, Andrés Rojas, Ana María Romano ("Carreras de aves y pájaros"), Roberto García ("PinPon Romero") and Germán Toro, are some of the Colombian composers that have been working with electroacoustic and computer medias on their music.

1.6 Cuba: Juan Blanco (born in 1919) have been the main force around the development of electroacoustic music in Cuba. Some of his pieces are: "Música para danza" (for tape, 1961), "Texturas" (for orchestra and tape, 1963-1964), "Espacios II" (for tape, 1984), "Circus-Toccata" (for tape, 1983), "Cinco Epitafios" (for tape, 1994). Many of his works ask for a large spatial distribution of sound sources, and complex multimedia installations ("Poema espacial No. 3", "Contrapunto espacial III").

Edesio Alejandro, Calixto Alvarez, Sergio Barroso ("Yantra VI", for piano and tape, 1976-1979; he is now living in Canada), Juan Marcos Blanco, Aurelio de la Vega ("Tangents", for violin and tape, 1973; he is now living in the U.S.A.), Orlando Jacinto García ("Metallic Images" for percussion and tape, 1991; he is now living in the U.S.A.), Argeliers León ("Saturnalia", for tape), Jesús Ortega ("Picassianas I", 1981), Tania León (she is now living in the U.S.A.), Pedro Pablo Pedroso, Ileana Pérez ("Conversations", 1995; she is now living in the U.S.A.), Alain Perón, Juan Piñera ("Pámpano y cascabel", guitar and tape, 1995), Armando Rodriguez (now living in the U.S.A.), Julio Roloff (he is now living in the U.S.A.), Julio Ruda, Roberto Valera, and Marietta Veulens ("Paz, meditación y metáfora") are some of the composers born in Cuba that have been working with electroacoustic medias. Composer Carlos Fariñas ("Aguas territoriales", for tape; "Impronta", for piano, 4 percussion players and tape; he is now living in Germany) have been also very active around the electroacoustic and computer music field during the last years.

1.7 Dominican Republic: Alejandro José (born in 1955) have been crossing the sea to live on his natal land or Puerto Rico alternatively, promoting electroacoustic and computer music in both places. He have been working on psychoacoustic research for many years. Some of his compositions are: "Tangentes" (1988), "Ecofonía I" (1992), and "Todo es Uno" (1995).

1.8 Ecuador: Mesías Maiguaschca (born in 1938; he lives in Europe) have been working with electroacoustic medias and computers for many years. He produced tape pieces ("Hor Zu" in 1969; "Ayayayayay" in 1971), mixed compositions ("Exercises", for violin and synthesizer, 1972-73; "FMelodies II", for cello, percussion and tape, 1981-1984), and works using live electronics too ("Segundo cuarteto de cuerdas", 1967).

Milton Estevez ("Apuntes con refrán", for orchestra and electronic sounds; "Patch 13" for tape, keyboard and percussion), Diego Luzuriaga ("Viento en el viento", for 2 flutists, percussion, electronic keyboard and computer; he is now living in the U.S.A.), Jorge Campos ("Glissandi", for Theremin and tape, 1996; he is living in Russia) and Pablo Freire ("Zeluob 3", for tape) have been also working with electroacoustic and computer medias on their compositions.

1.9 Guatemala: Joaquin Orellana (Guatemala, 1937) composed tape and mixed compositions like "Meteora" (1968), "Humanofonía" (1971), "Primitiva I" (1973), and

"Híbrido a presión" (1982). Also composer Igor de Gandarias have been working with electroacoustic medias on his music.

1.10 Mexico: A pioneer of electroacoustic music composition in his country, Héctor Quintanar ("Sideral I", for tape, 1968; "Símbolos", for chamber group, tape, slides and lights, 1969) was director of the first electronic music studio there, organized during 1969-1970. Raúl Pavón ("Fantasía Cósmica", 1982), an engineer interested both in electronics and music, started to promote the use of electronic musical instruments in Mexico many years before the first studio was built.

Among the Mexican composers that have been producing electroacoustic works are: Javier Alvarez ("Papalotl", for piano and tape; "Temazcal", for maracas and tape; he is now living in England), Guillermo Dávalos, Manuel de Elías ("Pro Pax", for tape), Manuel Enríquez ("Viols", for tape; "Conjuro", for double bass and tape; Enriquez was one of the main Mexican composers of instrumental contemporary music), Julio Estrada ("eua'on", for tape, 1980), Bernardo Feldman, Antonio Fernandez Ross, Carlos Jiménez Mabarak, Mario Lavista ("Contrapunto", 1972), Arturo Marquez ("Mutismo", for 2 pianos and tape), Eduardo Mata ("Los Huesos Secos", 1963), Víctor Manuel Medeles, Samir Menaceri, Roberto Morales-Manzanares ("Agua derramada", 1983; "Servicio a Domicilio", 1991), Francisco Nuñez ("Follajes", for violin, cello, double bass and electronics), Gabriela Ortiz, Manuel Rocha Iturbide ("Transiciones de fase", 1993-1994), Vicente Rojo, Antonio Russek ("Para espacios abiertos", 1981; "Ohtzalan", 1991), Arturo Salinas, Eduardo Soto Millan.

In spite the music by Conlon Nancarrow (born in the U.S.A., he have been living in Mexico for more than 50 years) is not directly related to electroacoustic or computer music composition, some of his pieces were presented (or performed) in concerts through computer controlled electromechanical systems, many of his compositions were translated from the original piano rolls to MIDI files, and even were produced computer generated versions from some of his works ("Study No.21"; "Study No.37") by composers like Rick Bidlack and Robert Willey.

1.11 Panama: David Soley (born in Ancon, 1962; he have been living in the U.S.A. for many years) composed several works using computers, as "Torso-trozos", for tape, in 1993, and "Linea", for Zeta violin, Radio Baton, sampler, sample playback and tape, during 1994-1995.

1.12 Peru: César Bolaños (born in 1931) and Edgar Valcarcel (born in 1932) are two of the main names related to the beginning of electronic music in Peru.

The first tape piece composed by Bolaños was "Intensidad y Altura" in 1964, produced at the Instituto Torcuato Di Tella of Buenos Aires. Among his compositions there are pieces for tape alone, compositions using live electronics, and multimedia works; some of them are: "Lutero, Yavi" (1965), "Dos en el Mundo" (1966), "Alfa-Omega" (1967), and "Flexum" (1969). Among the pieces composed by Valcarcel are: "Cantata" (for chorus and tape, 1967), "Invención" (for electronic sounds on tape, 1967), "Flor de Sancayo" (for piano and electronic sounds on tape, 1976), "Zampoña Sónica" (for flute and tape, 1968-1976).

Enrique Pinilla composed "Prisma" at the Columbia-Princeton Electronic Music Center in 1967.

Rajmil Fischman ("Los dados eternos", 1991; "Sin los cuatro", 1994; he is now living in England), Celso Garrido Lecca, Alejandro Nuñez Allauca ("Gravitacion humana", 1970), Olga Pozzi Escot (she have been living in the U.S.A. for many years), Arturo Ruiz del Pozo ("Lago de Totoras"), Pedro Seiji Asato, and José Roberto Sosaya Wekselman, are also composers who have been working with electroacoustic medias.

1.13 Puerto Rico: composers like Luis Manuel Alvarez, Rafael Aponte Ledée ("Cuídense de los ángeles que caen", for tape, 1974), Héctor Campos Parsi ("Arawak", for tape, 1970), José Montalvo ("Cuatro Estudios", for oboe and synthesizer, 1983), William Ortiz ("Composicion electrónica", for tape, 1978), Roberto Sierra ("Entre terceras", 1988), and Raymond Torres Santos ("Otoao", tape, 1984) have been producing electroacoustic and computer music works.

Alejandro José (born in Republica Dominicana; see his references on that section) is another composer mainly dedicated to composition with electronic medias. Eduardo Kusnir (born in Argentina) is now living in Caracas, Venezuela, but was teaching electronic music on Puerto Rico for sometime during the '70s, and again during the '90s.

Francis Schwartz (born in the U.S.A., 1940) have an important catalog of compositions, including tape pieces ("Calígula", 1975), mixed music ("Triangular Study", for trumpet, harp and tape, 1971) and multimedia works too ("Auschwitz", for tape, aromas, lights, dancer, temperature manipulation and slides, 1968).

Carlos Vázquez is a very active and prolific composer that produced many works using electroacoustic medias and computers. Some of his pieces are: "Sobre la inmiscusión" (for tape, 1975), "Alborada isleña" (for tape and computer, 1982), "Juracán" (for live electronics, 1992), "Los Ciclos de Luisa" (for tape, dancer and slides, 1994).

1.14 Uruguay: The first electroacoustic works by Sergio Cervetti (born in 1941), Coriun Aharonián (born in Montevideo, 1940), and Conrado Silva (born in Montevideo, 1940; now living in Brazil), were produced during the '60s. Some of their pieces are: "Studies in Silence" (1968), "Bits & pieces and Moving Parts" (1977) and "Something Borrowed, Something Blue" (1979-1980) by Cervetti; "Que" (1969) and "Secas las pilas de todos los timbres" (1995) by Aharonián; "Crónica" (1972), "Ulises" (1973), and "Pericón" (1989-1990) by Conrado Silva.

Ariel Martinez (now living in Argentina) composed several electroacoustic works during the '70s. Among them: "El glotón de Pepperland" for tape (1970), produced at the Instituto Torcuato Di Tella of Buenos Aires (CLAEM). Antonio Mastrogiovanni (born in 1936), composed "Secuencial II" (1970) also at the same center in Buenos Aires. León Biriotti (born in 1929), composer, conductor and oboist, composed "En la morada de la muerte" on the early '70s; he also produced several mixed pieces and works using live electronics.

Composers like Alejandro Barbot, Jorge Camiruaga, Fernando Condon, Carlos da Silveira, Ernesto Donas, Ulises Ferretti, Luis Jure, Jaime Kuckierwar, Diego Legrand, Beatriz Lockhart ("Ejercicio I", 1970), Alberto Macadar, Daniel Maggiolo, Leo Maslíah ("Llanto", 1980), Graciela Paraskevaídis (she was born in Argentina), Carlos Pellegrino ("Sin", 1974), Renée Pietrafesa, ("Juegos extraños", 1977), Pablo Sotuyo ("Retratos", 1989), and Héctor Tosar ("La gran flauta", 1988; he is one of the big names among the Uruguayan composers of contemporary instrumental music; born in 1923), have been working with electroacoustic medias or computers too.

1.15 Venezuela: During 1966-1967 was organized in Caracas the Estudio de Fonologia Musical of Instituto Nacional de Cultura y Bellas Artes (INCIBA) by José Vicente Asuar (from Chile; see his references on that section) and Alfredo del Mónaco (Born in Caracas, 1938), who composed on that studio "Cromofonías I" in 1967, being the first electroacoustic piece produced on that country. Later del Mónaco moved to New York where he produced several tape and mixed works ("Syntagma (A)", for trombone and electronic sounds on tape, 1972) at the Columbia-Princeton Electronic Music Center. Del Mónaco is now living in Venezuela.

That first studio was closed but after some time another laboratory was organized, being composer Antonio Estevez the director ("Cosmovibrafonía I", "Cromovibrafonía múltiple"). Later, Eduardo Kusnir (born in Buenos Aires, Argentina, in 1939; see also the Argentina and Puerto Rico´s references) started more sustained activities.

Some of the composers who also have been working with electroacoustic and computer medias are: Diana Arismendi, Josefina Benedetti, Roberto Cedeño, Roberto Chacón, Alvaro Cordero, Raúl Delgado Estevez, Juan de Dios Lopez, Julio D'Escrivan ("Salto Mortal", 1989), Adina Izarra ("Vojm", 1988), Ricardo Lorenz, Alfredo Marcano, Servio Marín ("Retour au silence", 1985; now living in the U.S.A.), Gustavo Matamoros ("Truly Yours", 1987; he is now living in the U.S.A.), Musikautomatika (experimental electroacoustic group with Alvise Sacchi, Luis Levin, Mirella Lopez, and Stefano Gramitto), Fidel Rodríguez Legendre, Alfredo Rugeles ("Thingsphonia", for tape, 1978), Federico Ruíz, Juan Francisco Sans, Edgar Saume, Rodrigo Segnini-Sequera ("Pekuek", 1994; now living in Japan), Jacky Schreiber ("Un Mundo dentro de un Mundo", 1994), Adrián Suárez Pérez, Ricardo Teruel ("Nuestra cultura vegeta", 1976), Alonso Toro ("No me perdonan", 1995), Numa Tortolero, Victor Varela.

2. Final comments

This was not done pretending to be the final list of electroacoustic composers in Latin America. On the contrary, my intention is to present a beginning, a start for further communication among latinamerican composers themselves, and between the world community of composers interested on electroacoustic and computer music.

I would appreciate very much to receive comments, suggestions, critics, and information that could help to improve this research and future papers on the same topic. I would like to remark that this idea to compile information about Latin America's electroacoustic and computer music composers have no intention at all to discriminate nobody, or to split nothing, or to divide groups. Nothing far away from my point of view and my wishes, the only idea here is to contribute to be integrated, to know more, to be communicated, and in contact with each other.

The Last revision of this paper was done on January of 1997.

3. Acknowledgments

I wish to thank all the people who helped on this research through the years. Special thanks to Jorge Antunes, José Augusto Mannis and Igor Lintz Maués (Brazil), Pablo Sotuyo (Uruguay), Rodrigo Segnini-Sequera (Venezuela), Aurelio Tello (Peru-Mexico) and Juan Reyes (Colombia).

4. References

- Aretz, Isabel (eds.). (1977). *América Latina en su música*. Mexico: Siglo XXI Editores.

- Auza León, Atiliano (1989). *Simbiosis Cultural de la Música Boliviana*. Bolivia.

- Béhague, Gerard (1983). *La música en América Latina*. Venezuela: Monte Avila Editores.

- Dal Farra, Ricardo (1994). *Some comments about electroacoustic music and life in Latin America* and CD companion's program notes. In *Leonardo Music Journal* No. 4 (pp. 91-98). U.S.A.: MIT Press Journals.

- Dal Farra, Ricardo (1996). *Electroacoustic and Computer Music in Latin America*. In *Proceedings of the 1996 International Computer Music Conference at Hong-Kong* (pp. 165-168). U.S.A.: ICMA.

- Degláns, Kerlinda and Pabón Roca, Luis E. (1989). *Catálogo de Música Clásica Contemporánea de Puerto Rico*. Puerto Rico: Pro-Arte Contemporáneo.

- Grela, D., Gianotti, A. y Lens, M.L. (1992). *Catálogo - Obras Musicales Argentinas producidas entre 1950 y 1992*. Argentina: Departamento de Ciencia y Técnica, Instituto Superior de Música, Universidad Nacional del Litoral.

- *Guia da Música Contemporânea Brasileira 1995-1996*. Brasil: Musicon.

- Lintz Maués, Igor (1989). *Música Eletroacústica no Brasil*. Brasil: Universidade de São Paulo, Escola de Comunicações e Artes.

- Paz, Juan Carlos (1968). *Introducción a la música de nuestro tiempo*. Argentina: Editorial Sudamericana.

- Segnini-Sequera, Rodrigo (1994). *Trabajo Especial de Grado: Comprender la música electroacústica y su expresión en Venezuela*. Venezuela: Universidad Central de Venezuela, Facultad de Humanidades y Educación, Escuela de Artes.

THE ELECTROACOUSTIC PRODUCTION OF THE L.M.E. OF THE NATIONAL UNIVERSITY OF CÓRDOBA

Martín Alejandro Fumarola
Laboratory of Electroacoustic Music
National University of Córdoba
Estafeta 56
RA-5001 Córdoba
Argentina
Phone: +54 51 234906
Fax: +54 51 217676
maralefu@goedel.filosofia.uncor.edu

Abstract

This paper reports the whole significant compositional and research work done in the L.M.E. (Laboratorio de Música Electroacústica) belonging to the MusicDepartment of the National University of Córdoba (Argentina). There is also reference to the high-end pedagogical activities developed as well as the foreseen projects for the near future. It is to be pointed out that the activities and work detailed are all those that reached a minimun level of professionalism and recognition. This report is an extension of the Studio Report presented in the Second Brazilian Symposium on Computer Music (Fumarola, 1995).

Electroacoustic and computer music pieces

this is the complete list of the works produced in the LME in the period ranging 1986-1996:

***"Exara 7"* by Eleazar Garzón (1988)**

***"El valle de las profecías"* by Oscar Bazán (1988)**

***"Los números"* by Oscar Bazán (1989)**

***"Argos"* by Martín Alejandro Fumarola (1988)**

***"Estatismo"* by Martín Alejandro Fumarola (1989)**

***"In Movile"* by Martín Alejandrto Fumarola (1992-93)**

***"La morada del cóndor"* by Eleazar Garzón (1994)**

***"Set In"* by Martín Alejandro Fumarola (1994)**

***"Agua sobre el cielo"* for saxophone and tape by Jorge Naparstek (1994)**

"La entidad en transformación constante" by José Mataloni (1995)

"El Peregrinar de la Araña" by Martín Alejandro Fumarola (1995)

In the national level, the two main forums for concert diffussion of the electroacoustic pieces produced in the L.M.E. are the *"Semana Nacional de los Medios y la Música Electroacústica"*, held every year in Buenos Aires, and the *"Jornadas Santafecinas de Música Electroacústica"*, organized every year in the city of Santa Fe by the National University of the Littoral. The piece *"El Peregrinar de la Araña"* is the only one that has been recorded in a commercial CD, the self-funded CD of the CEC (Canadian Electroacoustic Community).
Pieces by Oscar Bazán, Jorge Naparstek and Martín Fumarola have been performed in several festivals and concerts abroad. Composer Carlos Ferpozzi realized complementary parts of pieces finished in his private studio.

Research

Eleazar Garzón worked several years in the design and implementation of algorithms for his electroacoustic compositions. Roberto Rué is carrying out research activities related to microtonalism, which are granted by the Argentinian Secretary of Science and Technology. Martín Fumarola is participating in a software research and development project, developed in the framework of the *"Proyecto Universidades"*, as part of an agreement between the National University of Cordoba and IBM Argentina. The project concerns the design of algorithms for computer music composition written in Visual Age for C++ for Windows.

Pedagogical activities

Prof. Ariel Martínez was in charge of teaching electroacoustics from 1986 through 1992. In that time he carried out a unique metodology for teaching electroacoustic and computer music as well as he was the main responsible of both the foundation of the L.M.E. and its subsequential development. At the same time, Prof. Martinez was the Studio Director between 1986 and 1992.

Seminars and workshops

The two following seminars have been organized in cooperation with the Center for Acoustical and Luminotechnical Researchs (C.I.A.L.) of the National University of Córdoba:

- *"Música y Computadoras - Generación, procesamiento y control de señales de audio en sistemas digitales aplicados a la creación musical"* by Ricardo Dal Farra (1990)

- *"Semanticidad de los mensajes acústicos no-verbales"* by Carmelo Saitta (1991)

In 1994, a workshop about equalization and mixing in charge of Diego Losa was organized.

For 1997, a workshop on CSound by Pablo Di Liscia is scheduled to start on June, and later on, Prof. Maria Teresa Luengo will lecture as part of an agreement with the National University of Quilmes.

Staff

Unfortunately, and because of the lack of minimal academic seriousness, which normally is typical of the national universities in Argentina, there has not had any Studio Director from 1992 up to now. The L.M.E. currently depends on the Director of the Music Department of the National University of Córdoba and has a staff of only 2 Associate Composers-Researchers (Roberto Rué and Martín Fumarola).

References

Fumarola, Martín (1995). Laboratorio de Música Electroacústica del Departamento de Música de la Universidad Nacional de Córdoba. Proceedings of the Second Brazilian Symposium on Computer Music

necSO - Interface para um Sistema de Síntese e Processamento de Som (CSound)

MAURÍCIO ALVES LOUREIRO
CPMC - Centro de Pesquisa em Música Contemporânea
Escola de Música da Universidade Federal de Minas Gerais - UFMG
Av. Antonio Carlos 6627, 31270-010, Belo Horizonte, MG
mauricio@dcc.ufmg.br

HÉLDER SOARES DE SOUZA
Mestrado em Ciência da Computação - UFMG
helderss@dcc.ufmg.br

GUILHERME AUGUSTO SOARES DE CASTRO
Bacharelado em Música - UFMG
tiao@dcc.ufmg.br

HUGO BASTOS DE PAULA, LEANDRO DE FARIA FREITAS,
MARLON PERIGOLO DE REZENDE, WILLY GARABINI CORNELISSEN
Bacharelado em Ciência da Computação da UFMG
hugo@dcc.ufmg.br, freitas@dcc.ufmg.br,
marlonpr@dcc.ufmg.br, willy@dcc.ufmg.br

Resumo

Este trabalho descreve uma interface gráfica para o **CSound** que gera os arquivos de entrada *score* e *orchestra* a partir de dados armazenados em arquivos próprios. O programa inclui um ambiente gráfico intuitivo para especificação de tabelas de função, representação gráfica das unidades geradoras do **CSound** para construção de instrumentos e um ambiente intuitivo para composição com recursos para manipulação de *score*.

Abstract

This paper describes a graphical interface for the **CSound**, which generates the input text files *score* and *orchestra*, from data stored on its own file system. The program includes a graphical environment for function table specification, graphical representation to the **CSound** unity generators for instrument construction and an environment for composition with tools for *score* manipulation.

Introdução

A origem da tecnologia e da metodologia da computação musical moderna está relacionada com o desenvolvimento dos primeiros sistemas de síntese e processamento digital de som para aplicações musicais: MUSIC I a MUSIC V [Pope, 1993]. Estes programas pioneiros, escritos por Max Mathews nos laboratórios da AT&T e da Bell Co., na segunda metade da década de 50 e início da década de 60, deram origem a inúmeros

necSO - Interface para um Sistema de Síntese e Processamento de Som (CSound)

MAURÍCIO ALVES LOUREIRO
CPMC - Centro de Pesquisa em Música Contemporânea
Escola de Música da Universidade Federal de Minas Gerais - UFMG
Av. Antonio Carlos 6627, 31270-010, Belo Horizonte, MG
mauricio@dcc.ufmg.br

HÉLDER SOARES DE SOUZA
Mestrado em Ciência da Computação - UFMG
helderss@dcc.ufmg.br

GUILHERME AUGUSTO SOARES DE CASTRO
Bacharelado em Música - UFMG
tiao@dcc.ufmg.br

HUGO BASTOS DE PAULA, LEANDRO DE FARIA FREITAS,
MARLON PERIGOLO DE REZENDE, WILLY GARABINI CORNELISSEN
Bacharelado em Ciência da Computação da UFMG
hugo@dcc.ufmg.br, freitas@dcc.ufmg.br,
marlonpr@dcc.ufmg.br, willy@dcc.ufmg.br

Resumo

Este trabalho descreve uma interface gráfica para o **CSound** que gera os arquivos de entrada *score* e *orchestra* a partir de dados armazenados em arquivos próprios. O programa inclui um ambiente gráfico intuitivo para especificação de tabelas de função, representação gráfica das unidades geradoras do **CSound** para construção de instrumentos e um ambiente intuitivo para composição com recursos para manipulação de *score*.

Abstract

This paper describes a graphical interface for the **CSound**, which generates the input text files *score* and *orchestra*, from data stored on its own file system. The program includes a graphical environment for function table specification, graphical representation to the **CSound** unity generators for instrument construction and an environment for composition with tools for *score* manipulation.

Introdução

A origem da tecnologia e da metodologia da computação musical moderna está relacionada com o desenvolvimento dos primeiros sistemas de síntese e processamento digital de som para aplicações musicais: MUSIC I a MUSIC V [Pope, 1993]. Estes programas pioneiros, escritos por Max Mathews nos laboratórios da AT&T e da Bell Co., na segunda metade da década de 50 e início da década de 60, deram origem a inúmeros

outros, que ainda preservam as características dos 2 últimos da série, MUSIC IV e MUSIC V [Mathews, 1969].

Deste último, o mais eficiente da série, derivaram um grande número de variantes, dentre os quais citamos o MUSIC 360 escrito para o IBM 360 por Barry Vercoe do MIT - Massachussets Institute of Technology, o MUSIC 11 escrito para o PDP 11 por Roger Hale e Carl Howe do MIT. Dentre as mais recentes implementações do modelo, as mais conhecidas são o **cmusic**, escrito por Richard Moore da Universidade da California San Diego [Moore, 1985], e o **CSound**, escrito por Barry Vercoe [Vercoe, 1992], ambos implementados na linguagem **C**. Embora cada um apresente diferentes especificidades, incluindo inúmeros recursos adicionais de processamento e síntese, todos eles ainda possuem as mesmas características do modelo do MUSIC V.

Existem hoje sistemas mais eficientes tais como, **Accelerando** [Lent, Pinkston & Silsbee, 1989], **Kyma** [Scaletti, 1991] e **IRCAM Musical Workstation** [Lindermann et al. 1991], mas que no entanto são dedicados a hardware específico. Esstes sistemas incluem interface gráficas que possibilitam uma interação mais intuitiva na especificação e edição de tabelas de onda, envelopes e espectro harmônico, além de prover também recursos de controle em tempo real, mesmo que limitado. Pacotes mais recentes como o **MODE** [Pope, 1991] incluem um ambiente gráfico para especificação de instrumentos, mixagem e gravação de arquivos, além de inúmeras ferramentas para geração de partituras (lista de notas).

Descrição do Projeto

Objetivos

Tanto o **CSound** quanto o cmusic, não apresentam qualquer tipo de interface gráfica, muito menos uma linguagem para trabalhar em um nível mais alto da estrutura composicional, na qual aspectos puramente musicais possam ser enfatizados. Gareth Loy, um dos autores de CARL[1], 10 anos depois de sua criação afirma: *"One way to think of the CARL software is a signal processing package begging for a real user interface!"* (Uma maneira de se ver o CARL, é como um pacote de programas para processamento de sinal implorando por uma real interface com o usuário!) [Loy, 1993].

O objetivo deste projeto é desenvolver um ambiente para criação musical, intitulado **necSO**, implementado no **CSound**. Além de facilitar a utilização deste poderoso sistema de síntese e processamento de som, **necSO** oferecerá também um ambiente intuitivo para a edição e controle dos componentes da composição musical.

Seguindo o modelo de outras linguagens de alto nível existentes, **necSO** procura abstrair da noção de *instrumentos* acionados por uma *lista de notas*, imposta pelo compilador do **CSound**, mantendo transparente para o usuário esta estrutura baseada nos arquivos de

[1] O sistema CARL (Computer Audio Research Laboratory) se constitui em um pacote de programas para processamento e síntese digital, que inclui o compilador cmusic.

entrada do compilador (`.orc` e `.sco`). Para isso **necSO** possui um sistema de arquivos próprio e um interpretador que gera aqueles arquivos, mas que será ativado apenas na fase final do processo. Toda a especificação de *instrumentos* e *listas de notas* está contida nos arquivos gerados pelo programa.

Implementação

O projeto de **necSO** utiliza tecnologia de programação orientada a objetos. A lista abaixo descreve suscintamente os objetos que foram definidos na implementação de **necSO**, mostrando também seus prinicpais atributos:

Função Tabela
Gera tabelas de função através das rotinas GEN's do **CSound**.

ATRIBUTOS:
- número da tabela (1 a 200)
- tempo de início
- tamanho da tabela
- número da GEN (1 a 21)
- lista de parâmetros específicos

Unidade Geradora
Unidade geradora ou modificadora de valores correspondentes às UG's do **CSound**.

ATRIBUTOS:
- nome da UG
- parâmetros de entrada
- número de parâmetros de entrada (1 a 16)
- nome da saída
- tipo da saída
- destino da saida

Instrumento
Conjunto de UG's conectadas e ligadas a uma unidade de saída para gerar ou modificar um sinal.

ATRIBUTOS:
- lista de UG's
- número de saídas (1, 2 ou 4)
- identificadores de saídas
- lista de parâmetros dos *p-fields* (do score)

Saída
Unidade que promove a saída de amostras para um arquivo de áudio.

ATRIBUTOS:
- varáveis de entrada
- número de entradas (1, 2 ou 4)

Score
Conjunto de parâmetros de execução ordenados, que irão alimentar um instrumento.

ATRIBUTOS:
- número de parâmetros
- lista de parâmetros

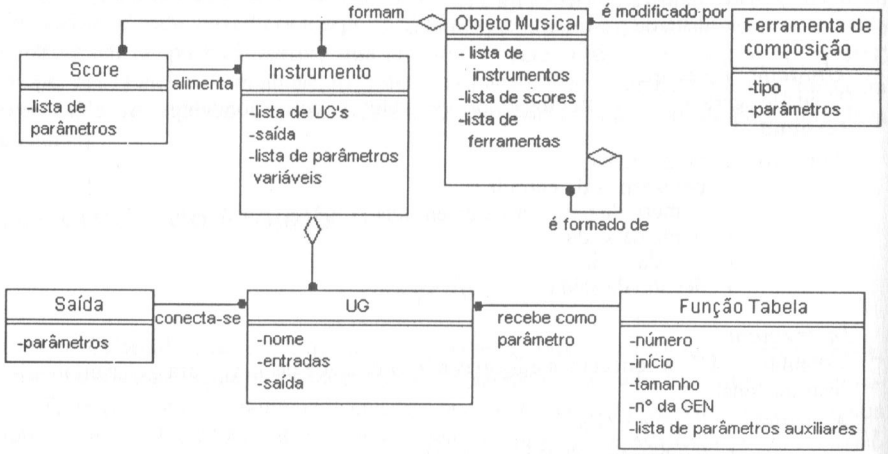
Objeto Musical
Um ou mais conjuntos agrupados de pares *instrumento/score* modificados ou não por uma ferramenta de composição.

ATRIBUTOS:
- lista de instrumentos
- lista de score associados
- lista de ferramentas de composição

Ferramentas de Composição
Funções que modificam e relacionam um ou mais pares de *instrumentos* e *scores*.

ATRIBUTOS:
- tipo de ferramenta
- nome da ferramenta
- lista de parâmetros
- lista de objetos musicais agrupados

A figura abaixo mostra um modelo dos objetos envolvidos no projeto de **necSO**, concebido segundo a modelagem de sistemas orientados a objetos descrita em por Rumbaugh [Rumbaugh, 1994].



Fiura 3: Modelo de Objetos do necSO

A implementação do **necSO** foi dividida em 3 módulos, cada um correspondendo a um dos 3 editores que integram o ambiente do programa:

Editor de Tabelas de Função

A fim de facilitar a especificação das *tabelas de função* a serem utilizadas pelo compilador, **necSO** inclui um ***Editor de Tabelas de Função* - FTed**, que fornece acesso gráfico às rotinas geradoras de tabela de função (GEN's) do **CSound**. A janela principal do **FTed**, consiste de quatro campos para entrada dos parâmetros comuns a todas as GEN's e uma área para preenchimento de parâmetros específicos para cada GEN selecionada na barra de ferramentas. O **FTed** gera e edita arquivos `.ft`, que contêm informações necessárias para gerar a linha de comando referente à tabela de função especificada (*f-statement*), a ser incluida no arquivo `.sco`. O menu principal dá acesso às funções padrão de manipulação de arquivos, edição, help e às funções específicas do editor

(Figura 1). A fim de tornar o processo de construção da tabela mais interativo, o editor fornece recursos de monitoração gráfica e auditiva. O usuário pode ver um diálogo que contêm a linha de texto do *f-statement*, a representação gráfica da tabela plotada, ou ouvir um som gerado por um instrumento padrão.

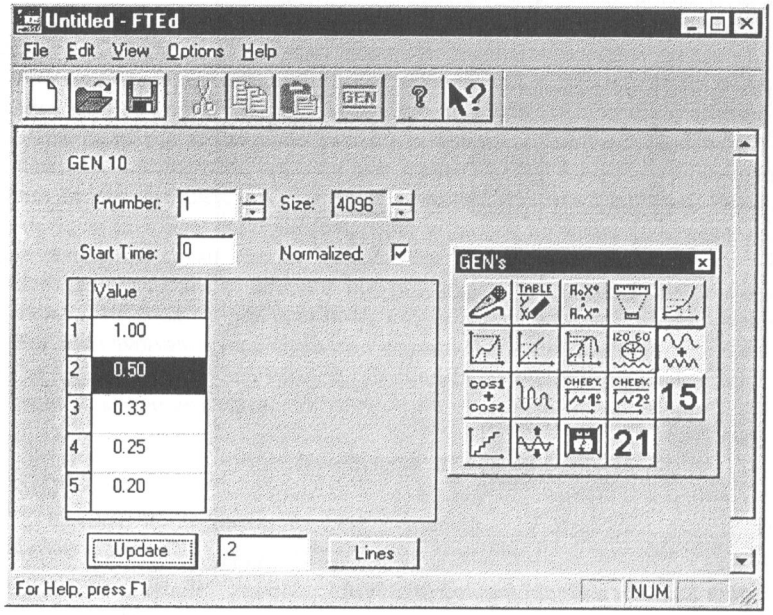


Figura 1: Editor de Função Tabela (FTed).

Editor de Instrumentos

No ***Editor de Instrumentos* - INSTed**, o usuário tem acesso gráfico a todas as *unidades geradoras* do **CSound** e a recursos para conectá-las graficamente. O editor contem uma área de trabalho onde o usuário constrói e edita os *instrumentos* graficamente, interligando as *unidades geradoras* (representadas por ícones), por linhas que conectam a saída de uma à entrada de outra. Através de duplo-clique, o editor mostra os campos para preenchimento dos parâmetros de entrada e nome da variável de saída da *unidade geradora* selecionada. O campo correspondente à *tabela de função* dá acesso ao ***Editor de Tabelas de Função***, ou mostra as tabelas disponíveis. O **INSTed** gera um arquivo `*.ins`, que será utilizado no ***Editor de Objetos Musicais***. Recursos de monitoração são também fornecidos neste editor.

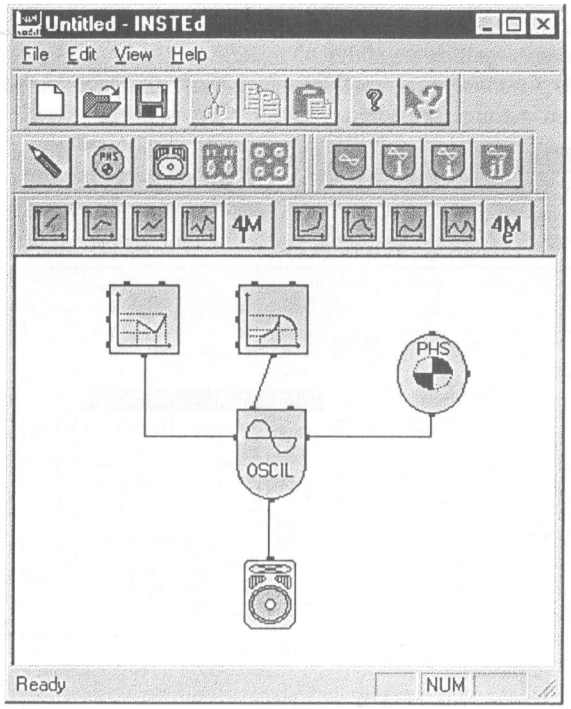


Figura 2: Editor de Instrumentos (INSTed).

Editor de Objetos Musicais

Esta é a janela principal de **necSO** onde o usuário cria e edita arquivos `.om` (*objetos musicais*), que contêm as informações necessárias para o interpretador gerar os arquivos de entrada do **CSound** (`*.orc` e `*.sco`). Um *objeto musical* pode ser criado do início, aplicando uma *lista de notas* a um *instrumento* criado no **INSTed**. O ***Editor de Objetos Musicais* - SOed**, mostra os campos para preenchimento da *lista de notas* que acionará o *instrumento* selecionado. Um *objeto musical* pode também ser criado a partir de um ou mais *objetos musicais* que podem ser processados ou relacionados através das *ferramentas de composição,* fornecidas pelo programa e que podem ser acessadas nesse editor.

Estas ferramentas são funções que podem gerar estruturas musicais a partir de um ou mais *objetos musicais*. Algumas delas representam implementações de funções do **CSound**, outras serão criadas para o ambiente de composição. *Ferramentas de transformação*, tais como transposição, ampliação, filtragem, deslocamento, quantização, loop e eco, modificam um *objeto musical* efetuando operações na lista de notas do arquivo *score*. *Ferramentas de relacionamento,* tais como soma, permutação, substituição, concatenação e desmembramento, relacionam 2 ou mais *objetos musicais*, agrupando-os e/ou transformando-os.

Conclusão

Uma composição é representada num sistema do tipo **CSound** por uma *lista de notas* relacionada a um conjunto de funções que definem um determinado processo de síntese, especificadas nos *instrumentos*. Esta estrutura impõe a restrição de que o evento acústico tem que ser uma *nota*. Mesmo que aparentemente ilimitada, a especificação de eventos através de *lista de notas* funciona bem apenas para ocasiões em que as informações necessárias para sua concepção é conhecida no instante em que ela é iniciada [Loy, 1989]. Além disso, por serem as notas nestes sistemas entidades independentes, a tarefa de se manipular um grupo de notas de uma maneira global é às vezes dificultada. Tais problemas conceituais, inerentes a sistemas deste tipo, tendem a continuar latentes por trás de qualquer interface. O projeto de **necSO** se propõe no entanto a oferecer recursos que que facilitam a construção de instrumentos e sua reutilização, além de oferecer alternativas para manipulação de *score,* que procuram tornar menos evidente a estrutura de *instrumentos* acionados por uma *nota*, e enfatizar paradigmas composicionais com operações que envolvam vários níveis de agrupamento de eventos musicais. Além disso **necSO** se apresenta como uma excelente ferramenta instrucional para aplicação em inúmeras áreas do ensino de música.

Bibliografia

Lent, K.; Pinkston, R. & Silsbee, P. 1989. "Accelerando: A Real-Time, General Purpose Computer Music System". Computer Music Journal. 13(4): 54-64.

Lindemann, Eric; Dechelle, François; Smith Bennett and Starkier, Michel. 1991. "The Architecture of the IRCAM Musical Workstation". Computer Music Journal, 15(3):41-49.

Loy, G. 1989. "Composing with Computers - A Survey of Some Compositional Formalisms and Music Programming Languages". In Mathews, Max V. and Pierce, John R. eds. Current Directions in Computer Music Research. Cambridge: MIT Press, pp. 291-396.

Loy, G. 1993. "Tools for Music Processing. The CARL System at Ten Years". In Goffredo Haus eds. Music Processing. Madison, WI: A-R Editions, Inc., pp. 267-300.

Mathews, Max V. 1969. The Technology of Computer Music. Cambridge: MIT Press.

Moore, F. Richard; Loy, D. Gareth and Dolson, Mark. 1985. CARL Startup Kit. San Diego: Computer Audio Research Laboratory.

Pope, Stephen Travis. 1991. "Introduction to MODE: The Musical Object Development Environment". In Pope, Stephen Travis, ed. The Well-Tempered Object. Cambridge: MIT Press, pp. 83-105.

Pope, Stephen Travis. 1993. "Machine Tongues XV: Three Packages for software Sound Synthesis". Computer Music Journal, 17(2):23-54.

Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F. e Lorensen, W. 1994. Modelagem e projetos baseados em objetos. Rio de Janeiro: Editora Campos.

Scaletti, Carla. 1991. "The Kyma/Platypus Computer Music Workstation". In Pope, Stephen Travis, editor, The Well-Tempered Object. Cambridge: MIT Press, pp. 119-140.

Vercoe, Barry. 1992. Csound - A Manual for the Audio Processing System and Supporting Programs with Tutorials. Cambridge, Massachussets: M. I. T., Experimental Music Studio. Media Laboratory.

Novas interfaces e a produção eletroacústica

Rodolfo Caesar
rcaesar@omega.lncc.br / caesar@acd.ufrj.br

Laboratório de Música e Tecnologia (LaMuT)
Escola de Música / Universidade Federal do Rio de Janeiro
Rua do Passeio 98, Centro 20021-290, Rio de Janeiro, RJ. Brazil

Abstract: Causes for the historic dispute between Musique Concrète and Elektronische Musik still surface for composers working in the domain of electroacustic music. Unsolved conceptual problems are present in old and new interfaces. Our difficulties have been reduced, but the main problems were not tackled: even with our computers we either choose a 'concrete' procedure, or a Köln-oriented writing.

Interfaces

No texto a seguir a palavra interface será usada em diversas acepções para diferentes modos de acesso a controles de processos de produção sonora/musical. Pode-se dizer que os instrumentos musicais apresentam tipos diversos de interface para o músico. Cada intérprete desempenha um determinado investimento físico para controlar a produção em certo âmbito particular a cada instrumento. O grupo 'instrumentos' tem, por sua vez, uma interface diferente da do computadores, que, com seus mouse e teclado apresenta acesso bem menos ergonômico. Mas a palavra interface tem também uma acepção mais localizada, que distingue os diversos modos de acesso aos computadores: os games tem um tipo de interface gráfica comandada por 'joysticks', em contraposição à interface simuladora de uma super-máquina de escrever para os programas de processamento de texto. No entanto se pode ainda falar de outras interfaces, superpostas a essa. A música instrumental 'digitalizada' gaba-se de uma interface própria chamada MIDI que consiste de um protocolo para envio de comandos entre máquinas. Os arquivos MIDI podem ser manipulados em programas que, por sua vez, apresentarão

interfaces relativamente diferentes entre si. Como se vê, são vários níveis sobrepostos em uma só palavra, que será usada aqui de modo não menos sobrecarregado.

Limites

Em comum, os instrumentos musicais tradicionais oferecem aos compositores um grande privilégio: a despeito de tantas transformações por que passa a tecnologia, os instrumentos musicais pouco se modificaram ao longo de cerca de dois séculos. Isso faz com que o repertório de sons de cada instrumento se manifeste dentro de um certo limite, e seja adotado por compositores que, como Stravinsky, prefiram exercitar a imaginação no campo delimitado entre 'sete sons e quatro acidentes'. Este conforto não é compartilhado pelos compositores que, ao entrarem no estúdio eletroacústico, rompem com os limites. O preço de sua decisão pode ser caro até para os mais experientes compositores pois, em seus quase cinqüenta anos, mudanças radicais ocorreram na tecnologia do estúdio de música eletroacústica - sempre alterando a relação entre homem, máquina e instrumento - flexionando os novos limites. A mais abrupta de todas essas mudanças foi a passagem do estúdio 'analógico' ao digital, com efeitos profundos não só no repertório de composições mas no destino de estúdios e compositores, que é o pano de fundo para o nosso tema.

Pré-história

Em seu início em fins dos anos 50, as músicas concreta e eletrônica eram feitas com equipamentos diferentes entre si. Enquanto o estúdio de Colônia se assemelhava a um laboratório de pesquisas acústicas, o de Paris identificava-se com o estúdio de gravação de rádio. Em comum, os dois tinham gravadores, mixers, amplificadores, monitores e filtros. Distinguiram-se por uma maior profusão de osciladores e moduladores de amplitude no estúdio de Colônia, enquanto o Club d'Essai abrigava mais microfones e suas próprias engenhocas, como variadores de velocidade e o 'Phonogène à coulisse', inventado por Schaeffer[1].

[1] Tratava-se de um leitor-gravador com uma dúzia de cabeçotes de leitura girando em torno de um eixo. Com velocidade controlada manualmente, os cabeçotes giravam ao mesmo tempo em que a fita magnética passava por eles, produzindo efeitos de estiramento temporal sem afetação de alturas, e transposição sem modificação da duração - como os vocoders e morphs mais tarde popularizados.

Depois da diluição da Música Concreta com a Música Eletrônica, o estúdio 'analógico' da nova 'música eletroacústica' ficou relativamente homogeneizado, compondo-se de: microfones, gravadores, amplificadores, mixers, sintetizadores, filtros, reverberadores e outros módulos de transformação de sons como delays, flangers, harmonizers. Todos esses aparelhos vieram a ser manualmente controlados em tempo real, acionados por potenciômetros, switches, pedais e teclados. Contava-se, ainda, com a operação sobre o som em suporte gravado: para a Música Concreta, a 'concretude' de um som residia não tanto em sua origem microfônica quanto no fato de ter existência no espaço físico e manipulável da fita magnética, acrescentando assim mais uma via de acesso ao trabalho, mais uma 'interface'. O compositor manipulava sons 'concretamente' existentes no registro da fita magnética, em atitude oposta à da Elektronische Musik, ou das músicas vocal/instrumentais, que se utilizam da 'abstração' seja em esquemas a priori ou no apoio da notação musical. Em Colônia predominava a busca - de cunho reducionista - do complexo a partir do ingrediente mais simples (da senóide em diante), enquanto que em Paris o sentido era inverso: partia-se da complexidade (do som gravado) para buscar o 'simples', isto é, a música, usando o ouvido como piloto. Em suma: o estúdio analógico, favorável à atitude 'concreta', oferecia aquela riqueza de acessos - batizados de 'manipulações' pelos compositores - pelas quais se podia interferir na natureza de um som e ouvir o efeito desta ação, simultâneamente. Exemplo de realização feliz em estúdio analógico é a obra 'Étude Élastique', de 1975, de Bernard Parmegiani, onde o autor, fisicamente ligado à sua ferramenta de trabalho, desenvolve técnicas surpreendentes, como a de jogar com a força retratora do motor do carretel do gravador dando puxões nas fitas (em contato com o cabeçote de leitura).

VC - patches

Beneficiados também pelas técnicas de síntese subtrativa, compositores eletroacústicos puderam realizar o repertório que confirmou, nos anos setenta, a viabilidade da música eletroacústica, a julgar pela proliferação dos estúdios institucionais daquela época. A síntese subtrativa oferecia uma interface manipulável em tempo real

pelo chamado Voltage Control. Os sintetizadores controlados por voltagem permitiam a contrução de 'patches', dispositivos resultantes da combinação dos módulos de geração de áudio e de controle. Esses patches (em si 'interfaces') favoreceram a produção de eventos de maior ou menor complexidade (desde a alteração de um parâmetro acústico em um som até o controle de grupos de eventos - às vezes em um só patch), cujo resultado gravado ainda poderia ser manipulado posteriormente, como qualquer som gravado em fita.

Se, para compositores como Parmegiani, os anos setenta foram o auge da produção, os anos oitenta viram a chegada dos sintetizadores digitais, os quais, embora mais poderosos para a fabricação de timbres, apresentavam interface menos apropriada para a experimentação da música eletroacústica. O primeiro sintetizador digital (com interface MIDI) disseminou a técnica de Modulação de Freqüência, permitindo a produção de timbres complexos com baixo custo. O preço real era de ordem mais estética: a interface MIDI. Para obter sons pouco reminiscentes dos instrumentos tradicionais, era preciso alguma ginástica, e maior esforço para criar patches. Aqui começa a mudança do estúdio analógico para o digital.

História recente

A entrada do computador no cenário musical trouxe consigo uma enorme dificuldade para os músicos acostumados a 'tocar' patches e manipular fitas. Sem as vias de acesso com as quais se acostumaram, compositores afeitos ao trabalho de composição 'de ouvido' do estúdio analógico recusaram as novas tecnologias e suas pobres interfaces restritas a teclado alfa-numérico e 'mouse', mantendo-se para sempre no domínio analógico (Michel Chion, 1991).

A 'música computacional', por seu lado, abria uma importante frente na qual cálculos 'a priori' e análises do conteúdo espectral dos sons serviriam de amparo para a síntese. Composta com o primeiro programa de computador para síntese de som, o Music V, de Max Matthews, a peça 'Mutations I' de J. C. Risset é paradigma de uma retomada prolífica da atitude 'determinista', porém indo muito além, porque agora apoiada nos conhecimentos da psicoacústica. Esta obra expõe um exemplo do poder da matemática dos sons, e do cálculo 'a priori' ausente no estúdio 'concretista': o glissando

'interminável', e o mergulho na análise espectral apontando para uma estética que mais tarde veio a firmar-se com o nome de 'Música Espectral'. Se a música espectral possui raízes anteriores datando da Música Concreta (!) ('Campanology', de Toshiro Mayuzumi, em 1958), ela precisou esperar a chegada dos computadores para florescer de fato.

No entanto a Música Espectral só tem disponibilizado a operacionalidade sobre o que é quantificável, tais como o parâmetro de altura e o conteúdo espectral dos sons. Outros parâmetros - ou critérios de percepção (na acepção schaefferiana) - menos quantificáveis, ficarão subtraídos do processo a menos que sejam colocados em vida acústica por novos dispositivos.

Tentativas nesse sentido vieram a partir de extrapolações a partir do padrão MIDI. Embora a serviço de uma uniformização que buscava parâmetros na música tradicional (pois trata-se de instrumentos para o mercado da música pop), o MIDI entrou em muitos estúdios como forma de recuperação do gesto instrumental, o controle manual necessário para o ato de compor. A partir de 1983/84 os estúdios digitais passaram a oferecer mais 'instrumentalidade' do que antes era disponibilizado pelo sistema analógico. Para muitos, no entanto, o protocolo MIDI foi também um canal para a reação, no sentido de que o repertório gestual das interfaces é, e soa, às vezes, por demais reminiscente do da música instrumental. Além disso, a falta de velocidade e a baixa resolução desse protocolo não permitem o controle de manipulações muito complexas e refinadas. A indústria de instrumentos chegou a anunciar um outro protocolo mais adequado, que deveria ter sido lançado em 1995.

Polarização

Atualmente quase todos os estúdios de música eletroacústica dispõem dos mesmos programas e interfaces, pela escolha dos quais ainda se pode interpretar uma certa polarização com respeito aos ideais da Musique Concrète e Elektronische Musik: de um lado programas de síntese e processamento reminiscentes do Music V, como Csound, beneficiando a escrita 'determinista' (pois cada evento deve ser previamente 'notado'). De outro lado interfaces descendentes ou inspiradas na música concreta como o Kyma, de Carla Scaletti, e o SYTER, do GRM, onde a noção de 'concretude' alia-se

à do 'tempo real' para manipulação e transformação de sons. O primeiro pólo privilegia o trabalho de concepção, e conseqüentemente de estruturação da macro-forma segundo dados quantificáveis. A segunda favorece a descoberta do 'inaudito', a pesquisa da musicalidade inerente ao 'objeto sonoro', pelo contrôle do ouvido. São estas as duas vertentes ainda não conciliadas pela via tecnológica. A música auxiliada por computador - talvez apenas por enquanto - está longe de resolver o impasse, porque, de um lado, transferiu o estúdio 'analógico' da música eletroacústica para o mundo digital. De outro propõe abordagens que se lançam de volta a Colônia, quando nós, compositores, não queremos um ou outro, mas um e outro.

O pesquisador Simon Emmerson (Emmerson, 1986) reputa aos programas de manipulação espectral (phase vocoders, morphs, etc.) a comunhão da música concreta com a eletrônica, graças à convergência do som gravado e do cálculo ou comando a priori em um mesmo e único campo de trabalho. Como já vimos, o caso da Musique Concrète depende muito mais de seu modo 'concreto' de manipulação do que da natureza dos sons aproveitados. Seu princípio sobreviveu na Música Eletroacústica graças à 'interface concreta' da fita gravada e ao 'patch', por permitirem a escuta analítica e a manipulação tanto de sons captados microfônicamente quanto de sons 'sintetizados'. A 'digitalização' do conceito de 'patch' em programas como Max (para MIDI) ou até o velho Turbosynth (para sintetizar e tratar sons) nunca devolveu a ergonomia e a funcionalidade dos 'patches' modulares do VC. A escuta do 'objet sonore' em tempo real em dispositivos como SYTER e Kyma não está acompanhada de vias de acesso (para o gesto fisico) correspondentes. A simulação 'fita gravada + processadores' é um recurso ainda interessante para a composição, mas apenas se baseia na reprodução de um nexo característico da 'fase analógica' da música eletroacústica, que necessita, agora, de interfaces em tempo real mais adequadas ao poder de geração e controle típicos do computador.

A título de curiosidade vale lembrar uma máquina recentemente aposentada, a (Machine Infernale) Publison, uma espécie de sampler com interessantes controles 'unicamente-em-tempo-real' (verso/reverso, timestretch) criada no início dos anos oitenta. Talvez pela falta de explorabilidade de seus recursos (em decisões anotadas e comandáveis a priori), os efeitos da Publison tornaram-se tão reconhecíveis que no final dos oitenta já se avaliava uma obra pela (não) recorrência aos seus efeitos.

Conclusão

Para superarmos uma condição - da composição - que a História apenas dissolveu sem ter resolvido, precisaremos buscar uma nova interface que permita uma mistura entre abordagem 'determinista' e experiência 'concretizante'. Lembremos que nossa música dos sons confia muito na apresentação daquilo que ainda não foi ouvido. Deste modo, uma pergunta do pesquisador Bénédict Maillard resumia, em 1982, a dificuldade da situação do compositor em estúdio digital: « Como pensar o inaudito? ». A pergunta fica ainda mais precisa ao elaborarmos: Como pensar o inaudito se não providenciarmos ferramentas que encontrem seu rastro? Para construirmos estas ferramentas deveremos ir mais longe, mais fundo em nós mesmos: temos que buscar mais recursos em nosso próprio aparelho conceitual, ainda insuficiente para a compreensão da experiência musical.

Bibliografia/Discografia

Caesar, Rodolfo, *O Zig-zag conceitual no estúdio de composição*, Anais do II SBC&M, Canela, 1995.

Chion, Michel, *L'Art des Sons Fixés*, Éditions Metamkine/Nota Bene/Sono-Concept, Fontaine (France), 1991.

Emmerson, Simon, ed., *The Language of Electroacoustic Music*, The Macmillan Press, London, 1986.

Schaeffer, Pierre, *La Musique Concrète*, col. Que sais-je, P.U.F., Paris, 1967.

Parmegiani, Bernard, *De Natura Sonorum*, INA C3001, INA-GRM, FRANÇA, CD.

Three-Threaded Invention

Aluizio Arcela

Computer Science Dept., University of Brasilia
70910-900 Brasilia-DF, Brazil
arcela@cic.unb.br

Abstract. It is described a client-server application for music where graphic servers are arranged in a way to provide visual counterparts to the real-time events generated by a sonic client. The client must read concurrently as many scores as is the number of servers in order to produce a coherent basis for a sound-image composition system. A Java implementation for three-servers-and-a-client is discussed while an extension of the program is proposed. The scores--called *spectral charts*--are those generated by the time-trees[1,2] and will work as raw material for a human real-time composition from a client interface. Properly speaking, the composition system works on a half-human-half-machine basis, for besides having a set of previously computed melodies, the process of melodic sequencing will continue by itself in an endless way when the human composer stops interacting with the system.

Music by Many Computers

If the communication among computers is possible by the so called client-server architecture, we may think in a group of computers as having the organization of an orchestra, such that each server will be running a different part of a piece which is being read and played by a client computer. As *spectral charts* are musical scores having visual information associated to each note, every server in the group must be running a real-time graphic program capable of interpreting visual data coming from the client.

The proper working of this program set depends on many factors all of them having the same degree of importance. An eventual inefficacy in any one of such factors will invalidate the expected result as it may be caused by the low speed of the client computer in the task of running three spectral charts in real-time, or by the insufficient graphic power of some server in painting the required image during the lifetime of the corresponding note. Evidently, the bandwidth of the physical network used to conect the computers is also one of these critical factors.

Besides allowing the conection among computers, modern programming languages, such as Java[4,5,7], have the means for the creation of *threads*. To music this represents a powerful tool for the writing of polyphony because threads are operating system related facilities which allow the execution of concurrent program segments.

Distributed Music

Three-Treaded Invention[*] is a program having a set of distributed objects created by instances of the classes to be described in the next sections. These objects communicate among themselves so as to orchestrate a distributed audiovisual output with the automatic input of melodies associated to the intervention of the composer from the

[*] A partial result from the project: *Objetos Intervalares em Java*, CNPq 352753/96-0 CC

users's interface. The composition model is that of a man-machine cooperation, where intuition is the ability assigned to the human composer--mainly in the timing he/she feels as the most suitable for melody and timbre renewal by pressing the interface buttons--while the heavy musical performance and crucial decisions are left to the computer. Among these crucial decisions is the mentioned melodic sequencing process, so that the human composer do not need to know what melody is the best to be concatenated to the current melody, but only signalize the moment the program must find it. Figure 1 ilustrates how the program is organized.

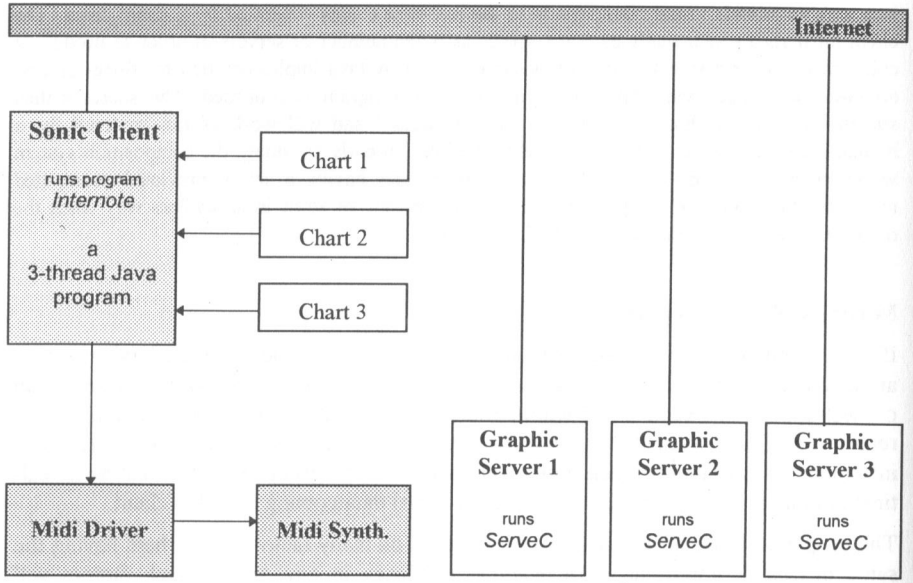


Figure 1

A Database of Melodies

The program works by running three melodies chosen from a database of previously computed melodies. Whenever the composer asks for a melody by pressing the melody button, or when the current melody reaches its end, the program picks a new melody from such group of melodies.

In *Three-Threaded Invention* the melody database is implemented as a set of files having enumerated names, as for example "*1.mel, 2.mel, ..., n.mel*", in order to facilitate the picking of a melody. Surely, the larger the number of files the greater the possibility for the composer to reach significant aesthetic results. However, in the construction of the melody database, the first rule to be followed is that a small but good set of melodies will work better than a large set having some bad melodies, even if the number of bad melodies is two or three. The melodies belonging to a really good group must hold formal relationship among themselves, in order to allow a place inside the program for a

deterministic, as opposed to a random, method for picking a new melody. Therefore, a bad melody--as employed here--is any melody having no relationship with any other in the database.

A Data Structure for Timbres

The assignment of a timbre to a melody is to be done with the help of data structures wherein all available instruments can be placed and their more important parameters being properly attached to each of them. Among such parameters are the average spectral contents, the attack time, the frequency range, and so on. The method for calculating the right timbre to play the current melody will operate by comparing the structural values of the melody--i.e. the group of intervals belonging to the melody, the rythmic patterns, the pitch range, and the overall duration--with the instrument's parameters, looking for that instrument which has the best set of properties in relation to an ideal spectral behaviour specified according to the entire melody or to a part of it, as it occurs when the human composer forces the program to pick a new melody.

The timbral calculation in case of *Three-Threaded Invention* is, however, not so complex as it is required above, for *Invention* uses a simple data structure, namely *Timbres[]*, which is an array of integers pointing to a subset of the general midi instruments, but without keeping their parameters. The composer may ask the program to change the timbre at any moment, while the program can decide for a new timbre when it starts a new melody by itself, that is, when a melody ends naturally without the intervention of the composer.

Part I Sonic Client Classes

A brief description in a Java-like way of the three classes belonging to the client program, namely *Internote*, *Performer*, and *Midi*, is found below. Many methods and variables have been omited for obvious reasons while some others have been abreviated to make the overall description easier.

The *Internote* Class

The role of *Internote* is to draw and administrate the user's interface after creating the COMPOSITION object. The first action of COMPOSITION is to create the threads *p1, p2*, and *p3* by instanciating three times the *Performer* class.

To start the threads, the composer may press the *playALL* button, or otherwise he/she may start individually each of them by pressing the *play* button inside the *Performer* box. When at least one conection is done the process of interactive algorithmic composition starts. The *Internote* class is defined as a subclass of the Java AWT *Frame* class, and looks like:

```
public class Internote extends Frame
{
  Performer p1,
                p2,
                p3;
 int timbres[] ={tb1, tb2, ... tbN}
```

```
    void desenhaInterface()
    { ... }

    Internote()
    {
        p1 =new Performer(serve1.IP_address, port);
        p2 =new Performer(serve2.IP_address, port);
        p3 =new Performer(serve3.IP_address, port);
        desenhaInterface();
    }

    public static void main(String[] args)
    {   Internote COMPOSITION =new Internote();
    }
}
```

The User's Interface

The basic rule for composing is that when a button is pressed at the user's interface, a change in both sound and image occurs in some way. However, there is no possibility of a direct control of the images but only of the sounds, for the images are the result of a mapping from the current music into graphics, and work as a second-level feedback path in the man-machine composition system. From the interface, the composer has three levels of control: (1) *Structural*, (2) *Positional*, and (3) *Fine-Adjust*.

The Structural Level

This is the top most level of the interface. It provides to the human composer the means for picking a new **melody** at the moment he wants as well as the means for changing the **timbre**. At this level, the composer handles large building blocks as long as melodies and timbres are previously computed units.

The Positional Level

At the positional level the composer may request coordinate changes of a virtual listener. The program will understand that as a reorganization of the acoustic projection that must take place as a result of a sound weight redistribution in the space. In terms of sound-image counterparts, the position of the virtual listener will contribute to determine the point-of-view for the images produced by the graphic servers.

As in the architecture adopted for *Three-Threaded Invention* there is a sound device only in the client computer, the overall implementation for the positional control is almost impossible. To fill all the 3D-space where the listener is imersed, there is a need for a more distributed way of organizing the program. Each graphic server must also be equipped with sound devices so that multiple audio channels become available in order to place the virtual sound sources anywhere in the listener space.

The Fine-Adjust Level

Here the composer can handle a set of controls for eventually adjusting the melodies. The **tempo** for the current melody may be varied either up and down in steps of a third, a fourth, a fifth, or an octave. The **transposition** can be changed in steps of 100 cents, 700 cents, or 1200 cents. Finally, the **intensity** can be varied in steps of 1, 10, or 50 dB.

The *Performer* Class

The threads *p1*, *p2*, and *p3* produced as a result of the instanciation of the *Performer* class by the COMPOSITION object have methods for reading the scores, sending midi messages to the midi device, calculating and sending to the graphic server color and geometric parameters of the same event, all of this on a real time basis. As an object defined as a *Performer* thread will take care of the execution of a note in the midi device, it must watch all the note's life in order to send a *note-on* message at its beginning--according to its starting time read from the score--and a *note-off* precisely at its end time. The *Chronometer* class defined below assists the process of picking the milisecond real time from the computer and comparing the elapsed time to check for the end of the note.

A key object to be created when this class is instanciated is *oM*-- an object of the *Midi* class. This object has the methods to cover most of the midi messages, as described below:

```
class Chronometer
{ int key,
      velocity,
      channel,
      end_time;
}


class Performer extends Thread
{
  Midi        oM   =new Midi();
  Chronometer cM[] =new Chronometer[16];
  double fb_midi   =(440/Math.pow(2,5))/Math.pow(2,(9/12));

  Socket          soquete;
  PrintStream     saidaLuz;

  public Performer(String server, int port)
  {
    for (int i=0; i<15; i++)
    { cM[i] =new Chronometer();
      cM[i].end_time =0;
      cM[i].key      =0;
      cM[i].velocity =0;
    }
    try
    { soquete  =new Socket(server, port);
      saidaLuz =new PrintStream(soquete.getOutputStream());
    }
    catch(IOException e) {}
  }
```

```
  public void run()
  {
    while(true)
    { read a command line from the chart;
      if (eof) { get a new chart;
                 calculate a new timbre;
               }
      if (comand.equals("note")
      {   read the note parameters;
          set the chronometers;
          oM.play(ins, dur, key, vel, ort, machine);
          ok=true;
          while(ok)
          {    read the geometric shapes associated to the note;
               calculate the color; (*)
               send values to the graphic server;
               if (end_of_instrument) ok=false;
          }
      }
      read chronometers to check for concluded notes;
    }
  }
```

(*) Although in the past some famous authors[6] had not believed in a real note-to-color transformation, a spectral component having amplitude a and frequency f will correspond to the following color in the *hsb* system[2,3]:

```
  void fromNOTEtoCOLOR(double a, double f)
  { double fr;
    if(f <=fb_midi) { h =0; s =0;
                    }
    else { fr =f/(Math.pow(2,octave(f)));
           h  =Math.IEEEremainder(360*((fr/fb_midi)-1), 360.0);
           s  =0.1*log(f/fb_midi, 2);
         }
    if (a*f*f <=0) b =0;
       else        b =invU*log((a*f*f)/af2_min, 10);
  }


  int octave(double f)
  {      if (f==0)          return =-999999;
    else if (f>= fb_midi) return = (int)(log(f/fb_midi, 2));
    else if (f<  fb_midi) return =((int)(log(f/fb_midi, 2))-1);
  }


}   //end of Performer class
```


The *Midi* Class

The implementation of this class presupposes the existence of a midi driver in the client computer. The *Midi* class used in *Three Threaded Invention* example below considers the driver **midixxx0** implemented for DOS by the device *bmidi.sys**.

```
class Midi
{
    File             drvmidi;
    FileOutputStream nomedrv;
    DataOutputStream mid;
    int    instr,
           dur_ms,
           key,
           veloc,
           orto,
           kanal;

 Midi()
 {
     try
     { drvmidi  = new File("midixxx0");
       nomedrv  = new FileOutputStream(drvmidi);
       mid      = new DataOutputStream(nomedrv);
     }
     catch (IOException e) {}
 }

 void play(int i, int d, int k, int v, int o, int c)
 {
     instr  = i;
     dur_ms = d;
     key    = k;
     veloc  = v;
     orto   = o;
     kanal  = c;
     note_on();
 }

 void note_on()
 { int pan;
   try
   { mid.writeByte(kanal+192);          //program change
     mid.writeByte(instr);              //general midi
     if (orto==0) pan=0; else pan=127;
     mid.writeByte(kanal+176);          //pan orto
     mid.writeByte(10);
     mid.writeByte(pan);
     mid.writeByte(kanal+144);          //note-on
     mid.writeByte(key);                //key midi
     mid.writeByte(veloc);              //veloc midi
     mid.flush();
   }
   catch (IOException e) {}
```

* Available for FTP from the URL: *http://www.lpe.cic.unb.br/servico/ftp/ftp-i.html#mps.*

```
    }

    void note_off(int kanal, int key, int veloc)
    { try
      { mid.writeByte(kanal+128);          //note-off
        mid.writeByte(key);                //key
        mid.writeByte(veloc);              //velocity
        mid.flush();
      }
      catch (IOException e) {}
    }

    void notes_off_canal(int canal_m)
    { try
      { mid.writeByte(canal_m+176);
        mid.writeByte(123);
        mid.writeByte(0);
        mid.flush();
      }
      catch (IOException e) {}
    }

    void notes_off_geral()
    { for (int i=0; i<15; I++) this.notes_off_canal(i);
    }

    void volume_canal(int vol)
    { try
      { mid.writeByte(kanal+176);
        mid.writeByte(7);
        mid.writeByte(vol);
        mid.flush();
      }
      catch (IOException e) {}
    }

} // end of Midi class
```

Part II Graphic Server Classes

There is in each server a program (*ServeC*) that loops forever, listening for a conection from a client machine. When a connection request occurs, *ServeC* creates a thread (*Conection*) that will open a Java AWT *frame*[3] for graphics output, and will take care of the reading of the visual data coming from the client through the Internet socket which is defined in the program. As the data are read in, a *Painting* object is asked to produce the images.

- ◊ ServeC
- ◊ Conection
- ◊ Painting

```
public class ServeC extends Thread
{
  ServerSocket serv_soc;
```



```
  int port;

    public ServeC(int p)
    {
      this.port=p;
      try { serv_soc =new ServerSocket(port);}
      catch (IOException e) {};
      this.start();
    }

    public void run()
    { Conection c = new Conection(tipoFigura);
      try
      { while(true)
          {
         Socket clie_soc = serv_soc.accept();
          String nomeH = serv_soc.getInetAddress().
          getLocalHost().getHostName();
          c.protocol(clie_soc, nomeH);
          }
      }
      catch(IOException e) {}
    }

    public static void main (String[] args)
    { new ServeC(2001);
    }
}
```

Because the multithreaded nature of the *ServeC* class above, any number of sonic clients may be connected to a particular graphic server, for to each of them a *Conection* thread is created to deal exclusively with the data coming from it. Here we may think in an extended architecture where more than one human composer may interact in the overall composition. This, however, requires really powerfull graphic stations.

In order to be a frame, the *Conection* class must be a subclass of the Java *Frame* class, while in order to be a thread, it must implement the Java *Runnable* interface. Therefore:

```
class Conection extends Frame implements Runnable
{
     Socket             client=null;
     DataInputStream    input;

     Thread  le_figura;
     String  nome_host;

     Painting painter;
     String shape;
     Figure figure[] = new Figure[8];
     String figType;


     public Conection(String f)
     { this.figType =f;
       le_figura= new Thread(this);
     }
```

```
public void protocol(Socket clie_soc, String nomeh)
{ client    =clie_soc;
  nome_host =nomeh;
  try
  { input =new DataInputStream(cliente.getInputStream());
  }
  catch (IOException e)
  {  try
     { cliente.close();
     }
     catch(IOException e2) {};
     System.err.println("Failed: " + e);
     return;
  }
  read_figura.start();
}

public static void main (String[] args)
{}
```

The *Conection*'s *run()* method--which is the thread's heart--will basically read clusters of graphic commands. When a cluster is completed (ending by a "Z" opcode), the *run()* method asks the *painter* objetc to apply its *drawing* and *painting* methods in order to accomplish the picture required by the commands.

```
public void run()
{ while(true)
  { get geometric values from the client socket;
    while("Z".equals(shape =input.readLine())) {};
    read the geometric parameters for shape in the socket;
    ask painter to draw and paint the shape;
  }
}

  }
} // end Conection


class Painting
{ drawing methods;
  painting methods;
}
```

Conclusions

Concurrent programming techniques applied to distributed computer architectures allows the composition of musical pieces at a large stitch, that is, rather than single-note building blocks, the smallest constructing parts should be whole melodies. Here we can remember some words about this matter by the german composer Gottfried Michael Koenig in an interview[8] carried out more than ten years ago: "... I could imagine a stage

in which a computer would respond not only with sound but also with structures with such speed that the composer could have as many compositions to consider in real time as he's now able to consider sound characteristics".

Best results are reached when the melodies belonging to the set of previously computed melodies have a formal relationship among themselves, as for example those melodies computed from the *time-trees*.

As far as the sound-image algebraic transformations[3] are coherent, the composition process helped by a visual aesthetic monitoring of the automatically created images provides a safe way of putting a true meaning into the piece under construction.

There are many interesting aspects which could be analysed in a distributed processing based composition system like *Three Threaded Invention*, but the possibility of having multiple audio channels--extending the compositional language by codifying the sound spacialization--is probably the more complex, both when we want to understand its possibilities and when we decide to implement it by placing real sound sources in the space.

References

1. ARCELA, A. "TimeTrees: The Inner Organization of Intervals". In *Proceedings of the XII International Computer Music Conference*, The Hague, 1986.
2. ARCELA, A. "Síntese de Imagens com Pedaços de Tempos", In *Humanidades*, 8(4):486-493, Brasilia, 1992.
3. ARCELA, A. *As Árvores de Tempos*, Hypertext available at http://www.lpe.cic.unb.br/teoria/teoria.html, University of Brasilia, 1995.
4. CAMPIONE, M. and WALRATH, K., *The Java Tutorial: Object-Oriented Programming for the Internet*, Addison-Wesley, 1997.
5. FLANAGAN, D. *Java in a Nutshell*, O'Reilly & Associates, 1996.
6. GOETHE, J.W. *Traité de Couleurs*, Paris, Triades, 1973.
7. LEA, D. *Concurrent Programming in Java*, Addison-Wesley, 1997.
8. ROADS, C. "Interview with Gottfried Michael Koenig". In C. Roads, ed., *Foundations of Computer Music*, Cambridge, MIT Press, 1985.

TAPE SOLO CONCERTS

All blue, I write with a blue pencil, on a blue sky (1996)

Celso Aguiar

This title was drawn from the writings of Walter Smetak (composer, instrument-builder, cellist and writer) to whose memory the piece is dedicated. The piece is about sound transformation, as a metaphor to the transformation of consciousness. Metallic percussion sounds are ever-present, while original cello sounds are broken into their rawest components. The basic cuisine for the piece was set up from these spices, and the dish is to be served hot.

The cello has its identity transformed: its defining harmonic series is turned inharmonic, sounding closer to the metallic percussion. The pitches from this now bent, inharmonic series, are used as framework for a melodic-timbral game (the 'blue pencil on a blue sky') played by cello and percussion.

The cello transformations were obtained with SMSplus, a CLM system built on top of Xavier Serra's Spectral Modeling Synthesis and developed by the composer.

All blue was originally composed in 1996 for four-channel tape - from which this stereo version was made. A procedure for modeling the physical properties of a room via feedback-delay-networks was employed (*Ball within a Box*, developed by Italian researcher Davide Rocchesso at CCRMA, with additional enhancements by the composer). A closer impression of the original virtual space is obtained when listening to the piece with headphones.

Unácrès (1996)

Ralf Ollertz

Together with *Pyrócua* and *Cral'une*, *Unácrès* makes part of a Trilogy about a travel through a desert in Chile. The piece was commissioned by the Academy of Arts of Berlin, in which studio it was produced. The material is based on my recording of different glass sounds. Glasses, crashed mirrors, bottles, broken windows, etc... Using Macintosh computers, the compositorial work was made with programs like Sound Designer, Pro tools, several GRM Tools, granular synthesis and, of course, traditional transposition and modulation technics. Like in all my works, the technic and programs I use, are never the beginning of the composition process, more a *compagnon* by realizing my issues and ideas

Chimeplay (1994)

Mara Helmuth

Chimeplay was written on the NeXT computer for my wedding in August 1994 to express the joy of that time. The sound sources were obsidian wind chimes, bells and the voices of my husband and myself, processed with Cmix and rt on the NeXT computer. Patterns heard in the percussion and voices come from musical algorithms based on pitches and rhythms of the wind chimes.

I used linear predictive coding to transform the voices, phase vocoding to stretch and deepen the wind chimes, and room simulation to create different environments. I wanted to create a world of chiming sounds which evolve through timbres from brilliant to heavy and dark, and even through the playfulness of the human voice.

Choi Hung (1996)

Juan Reyes

This is a piece for flute sounds, and modeling of timbres from the far east. The focus is on the sound of the Shakuhachi. The timbre, for the most, was achieved by using a technique known as Spectral Modeling Synthesis (SMS) developed by Xavier Serra . The inspirational sources for the piece are a Shakuhachi performance, a subway station in Hong Kong, and its environment at the time. Humidity, hot temperature, and wood highly inspire this soundscape. This piece was composed at MOX - Center for Advanced Computation at Universidad de Los Andes in Bogota - Colombia.

Am Steg (1992-96)

Javier Alejandro Garavaglia

The main sound material of this piece, is that of three notes played *sul ponticiello* (in german *am Steg*) on a viola. The richness in harmonics of this complex lead me to the composition of a piece, which should be a kind of study on spectra, consisting mainly on transpositions and reverberating of the main sound. The first version was finished in 1992, as I went on another project involving also viola sounds (which would become my piece called *pizz*, based only on one *pizzicato* on the C string of the viola). It's name was *Räume*, what in german means *Spaces*. In the beginning of 1996, I returned to this first *viola-project*- which hasn't been performed up to that time- as I decided to make some radical changes, leaving only the frame of the work, but adding more to it, for example, making filter chains to improve the spectral quality of the sound, letting some harmonics to resonate more than others, adding very low and very high FM and FM filtered frequencies by following the amplitude envelope of the original sounds and so on. The sense of space of the first version was not lost and that's why I called the piece quite the same but in another way, but I wanted also to mention the source of the sound in the title.

The piece can also be performed together with her sister pizz as a Viola cycle.

Um caminho no meio da noite (1996)

Paulo Vivacqua

The work is divided in four parts, which show internal articulations. The first two parts alternate two different materials, one internal, waves produced by additive synthesis build into the computer, and another one, external, with fragments of sound of short wave radio. Hardware employed: a PC 486 with audio and a short wave radio. Software employed: *Wave*, *Cool Edit*, for audio processing and *SAW* as a sequencer.

PerCurso (1997)

Fernando Iazzetta

The composition of PerCurso started about one year ago. At that time I was interested in using percussive sounds to create electronic music. For me, it was really challenging to transform those short and punctual sounds in a rich material to be used in music composition. All sonic material in PerCurso comes from a short fragment of clapping sounds presented at the very beginning of the piece. This fragment has been processed and transformed by using in a number of different software and techniques. The result was a large library of sounds which I have used to compose the piece.

Del cuadro a la postergación (1994))

Patricia Martínez

This piece is the attempt to effect the musical impression of a plastic work, to describe not the painting, but the first impact of the subject -my own impression in this case- of that almost automatic and hardly apprehensible moment of meeting to the visual, temporally embodied in music. The painting on which this work is based is called "The nail in the picture", and belongs to the contemporary german plastic Gunter Eucker.

The idea or concept that I have developed is the challenge of trying to transfer certain mechanisms of aesthetic and temporal perception of visual art to an art apparently opposite in essence, form and content, as music is. I wanted to turn, for the listener, during the hearing of a work, chronological time (implacably continuous) into psychological time, where the flow of music is transformed into sound paths seeking to exalt only "the instant".

In general, I worked on the elaboration of compact and regular sound structures, layers or walls of complex sound that are being juxtaposed, superimposed or distorted, gaining mobility and independence through the musical discourse. The organization of these materials is thought as the experimentation of a way of articulating the sound language by means of the fusion of timbre interweaving under permanent transformation.

Concreta (1997)

Aquiles Pantaleão

Concreta intend to discover the identity of the used materials, through the exploration of their spectral contents. The materials are deprived of their immediate recognized features - considered here *externals* - like contour, gesture, etc. in the attempt to emphasize tone color and internal space. Even though subsequent morphologic development of those materials an abstract course follow, strong qualities of body, mass, density and substance will be hold on.

The title is so much a reference to the reinforced concrete blocks that furnished most of the used sounds, as an homage -even if obvious - to the *musique concrète.*

Ramparts (1996)

Dennis Miller

Ramparts, for solo tape, was written in 1996 and received its premiere early in 1997. The piece is in two large sections that are similar in their use of granulated textures, but distinct in other ways, primarily register and dynamics.

The sonic material for the piece was generates by the Kyma Sound Design Workstation with processed via a Kurzweil K2000 sampler. Like many of the composer's recent works, the formal outline is defined by a balance of repeating, referential materials and unexpected, contrasting elements.

Theta Ticker (1996)

James Brody

Theta Ticker, composed in December of 1996, is a result of an ongoing (naïve) sound experiment to see if there might be some direct correlation between musical rhythm and brain waves. The obvious repeated sound is at a frequency of 5.2 Hz, right in the theta brain wave band. A penetrating quality to this sound seems to have some kind of hypnotic effect. If some listeners find the sound environment unendurable, they should feel free to leave for a minute.

Structurally the other sounds in the piece are organized in the proportions of the Fibonacci series, 1-1-2-3-5-8-13. The transformations of a small number of 'real-world' sound sources were accomplished using the Sound Forge and Cool Editor programs on a Pentium 150 computer in the composer's home studio. A final transfer was then made to Digital Audio Tape.

Recitativo elettronico II (1995-97)

Catalina Peralta

The main concept stays:an interchange between the different instrumental characters of a double stringbody, constructs a kind of Recitativo in space. On the tape:Indian Sitar and Violin; there were no other synthetical or electronic sounds. This work is based on motivs that at the same time are instrumental characters, configurations of instrumental soun objects. The "stringbody" expands and compresses itself contrasting with its own time structure, breaking it offen but going forward, enjoying the distance from recognizable auditive processes of pure acoustical instruments. The piece was produced at the Laboratorio de Composición Electroacústica, Depto.de Música, UNIANDES.

Into Darkness (1996)

Thomas Wells

My basic idea was to create an entire work from spectral manipulation of vocal sounds, using techniques of spectral interpolation, compression, and expansion. The primary sound source in this work is an excerpt from Don Carlo Gesualdo's Tenebrae, recorded

by the Hilliard Ensemble, and used by permission of ECM Records. The music is dark in mood, befitting the text and the style of Gesualdo's vocal setting.

The 60" excerpt on which the entire work is based was transferred directly from CD to hard disk using an Audio Digital Systems AES/EBU/SCSI interface developed at The Ohio State University under a National Endowment for the Arts Centers for New Music Resources Grant.

The recorded material was processed using phase-vocoder-based spectral modification programs written by Christopher Penrose, Eric Lyon, and myself -- programs to perform timbral interpolation, spectral expansion, compression, and inversion. These programs run on a NeXT 040 Cube in the Ohio State University Sound Synthesis Studios. The IRCAM Super Phase Vocoder, running on a DECStation 5000/200 was employed in this work for time expansion, and the Mark Dolson phase vocoder, running on a SUN 3/280 was used to provide analysis files for manipulation using cmusic scripts. Linear prediction was used, albeit very sparingly, using the mxv application, written by Doug Scott, and running on the NeXT 040 Cube.

Timbral interpolations were made between different excerpts of the Gesualdo samples, as well as between Gesualdo samples and software-synthesis produced sounds (made using FM, waveshaping, granular synthesis, and resonant-filter synthesis). The spectral modification techniques were applied, for the most part, successively, in order to achieve a desired timbral richness, and to distance the material somewhat from the original patterns and inflections of the Gesualdo. Mixing was done with the Lansky/Dickiert application. Recording was made on a Sony PCM-2500 DAT, recording directly digitally from the Audio Digital Systems interface.

TV Scherzo (1996)

Hwang Sung-ho

TV Scherzo, based on a dance, Fire, Mist (1991), materialized a theme, *TV satire* based on the events that occurred in 1994-1995. This is about the loss of our true senses in the reality reflected through TV, which causes confusion in our perception about time and space, reality and virtuality. In the manner of TV programs, it is a simple flow of ordinary daily life without pursuing any formal shape or structural unity. The effect of experiencing minimalism, aleatory, tonality, harmony and sound is intensified by rhythmic contrast of comic gesture at the beginning and the poly-rhythmic gesture in the middle, and the experimental sound in the last part. This work evinces an interest in the strong rhythmic gesture of the composer, and is a product of modernism, which is possible because of today's electro-domineering environmental culture.

La beauté indiscrète d'une note violette (1995)

Jorge Antunes.

This piece, only 2 minutes long, was produced at the Studio Charybde of GMEB, in France, in May 1995. It is a tone colour development of a note E (Mi). After Jorge Antunes Chromophonic Theory, the octaves of this notes matches the colour violet. Therefore, the mini-composition is a kind of tone colour-time picture, where different

nuances ad violet shades blend, unfold end evolves in time. Other than Pro-Tools software for the edition, the composer used Sample-Cell software for the construction of tone-colour transformations, and also the Sound Synthesis technic which Antunes call slipping harmonics, where the synthetic-evolved sound of the spectrum, comes to be a changing time web. The junction of E notes by a female voice, joins an erotic character to the little musical speech.

Atari Etude (1986)

Mladen Milicevic

The only music materials used in ATARI ETUDE are 132 different glissandi, defined within a range of ten octaves. Thanks to FORMULA's ability to handle incredibly fast playing speeds, an additional sound dimension of the glissandi could be utilized. Sounds generated by the Yamaha's TG77 played at fast speeds changed their original identity. The transition from synthesized sounds to the "speed modulated" sounds was the basic constructive element in structuring this minimal piece.

Diaphane (1995)

Gerald Eckert

Diaphane (Diaphanous) for 2-track tape was composed in 1995 at the ICEM (Institute for Computer Music and Electronic Media) at the Folkwang-Hochschule in Essen/ Germany. The title (cf. diaphan - diaphanous) is to be understood as a concept. - A stratum which is in itself complex and has been composed using various means is overlaid by several different strata - or expressed as an association: a surface changes its form due to the simultaneous appearance of different-colored lights refracted by a prism. The result is the overlapping of two different kinds of structures comparable to the interference of two pieces of film laid over each other. This happens in "Diaphane" at carefully chosen points which, temporally, are uniquely related.

This work was composed using various kinds of technology. The "concrete" sound material was won from the sounds of percussion, speech and machines and was digitally revised. The sound structure was created with the language Csound.

Tierra y Sol II (1996)

Ricardo Dal Farra

With "Tierra y sol" I am trying to reflect not only the sonorities from the Andes mountains, but also the pace, the mood, the different way of changes, the hopes (or non-hopes), the times of the people's vital cycle there.

All sonic materials heard on this piece were derived from ancient Andes' woodwind instruments like quenas and mohoceños; cross-culture musical instruments like charangos, and even the classical guitar; and from the voice of a folk singer, still living in the mountains.

The original version of "Tierra y sol" was commissioned by the International Computer Music Association (1996 ICMA Commission Award). This new version was edited and remixed in 1997.

Onset/Offset (1996)

Pete Stollery

My previous tape piece Altered Images was concerned with the dual interpretation of the word "image" on both aesthetic and sonic levels; Onset/Offset is concerned, even more than before, with exploiting the interplay between the original "meaning" of sound objects and their spectro-morphological characteristics. Thus, there are many recognizable sounds in this piece which can, and should, be perceived on both levels - the sound of a key in a lock is on one level refers to the action of unlocking a door, but on another, is also interesting as a pure sound in itself. Onset/Offset was realized in the Electroacoustic Music Studios at Northern College, Aberdeen and at the University of Birmingham in April 1996. It received an Honorable Mention in the Stockholm Electronic Arts Award, 1996.

MAN-MACHINE CONCERTS

El Poder de lo Invisible (1996)

Carlos Cerana

El poder de lo invisible (The power of the invisible, 1996) represents a confluence of two different cultures far apart in time and space. The practice of T'ai-Chi Ch'uan was developed by Taoist monks of China, as a way to harmonize man with the Universe by balancing the energies of body, emotions and mind. The Lightning is a spatial MIDI controller: two infrared transmitters in the forearms of the performer allow a sensor to detect their position and displacements. A software developed in MAX uses the gestures as control factors for a synthesizer. The ancient form of the T'ai-Chi --rendered by Felicia Tracogna-- is translated to sounds by means of present technology: an audible manifestation of Supreme Unity.

Velocity Studies IV: Flutter (1994)

Allen Strange

This is the most recent in a series of compositions for instruments and electronic/computer sounds. Like all its predecessors the music is virtuosic in gesture and based on an active metaphor that permeates the composition, in this case "flutter." In this case the "fluttering" refers to the music of the legendary jazz saxophonist, Charlie "Yardbird" Parker. The composition is structured as a series of seven character variations made from "Parkerish" things. In some cases there are very obvious variations and quotes from famous Parker tunes. In other instances characteristic Parker improvisatory phrases are quoted and expanded and large sections are based on scalar patterns indigenous to Parker's performances. As this work is a tribute to the master improviser it was obvious to me that some of the content should be left to the spontaneous skills of the performer.

Caxambulismo (1996)

Robert Willey

"Caxambulismo" is an interactive improvisational environment written in MAX using a Buchla Lightning controller. It was made for Mauricio Loureiro, and when the wand is attached to his clarinet the vertical movements of the instrument controls the pitches played by the computer, while the horizontal movements regulate the timbres. Vertical motions of his leg to which the second wand is attached to his leg control the volume of the synthesizer, while the horizontal axis regulates the duration of the notes played.

Klang/Clan (1996)

Jorge Sad

From its beginning the Colectivo de Creación Sonora was thought as a place of exploration and research of traditional instruments' sound possibilities and their capability of melting with electroacoustical sounds. The group tends to appear as a living synthesizer, where the source of sound events is often hard to be identified.

The CCS looks for a way of working which incorporates the players intuition and hearing as a structural part of music creation. Sound objects achieved this way are the consecution of a common elaboration process instead of the product of an isolated being.

The members of the Colectivo de Creación Sonora are:

Juliana Moreno: Flute, Enrique Entenza: Bandoneon, Germán Meira: Electric Guitar, Martín Devoto: Cello, Jorge Sad: Composition and Conduction.

The CCS was born in 1994 and had its first performance during the Week of Music and Electroacoustical Means '94 at The Centro Cultural Recoleta. It has been chosen to take part of the 6th. International Symposium of Electronic Arts (ISEA 95) at Montreal. The piece Klang/Clan was commissioned by the Fundación Música y Tecnología and was premiered at the Centro Cultural Recoleta on November 1st., 1996.

The Persistence of Memory (1997)

Ciaran Hope

"The Persistence of Memory" is based on the painting of the same name, by Salvador Dali which was painted in 1931. This painting is one of Dali's most memorable Surrealist works. One hot August afternoon, in 1931, Dali came upon one of his most stunning paranoid critical hallucinations. Upon taking a pencil, and sliding it under a bit of Camembert cheese, which had become softer and runnier than usual in the summer heat, Dali was inspired with the idea for the melting watches. These are one symbol that are commonly associated with Dali's Surrealism, where he literally uses them to show the irrelevance of time. The piece is based on the note E, and like the picture itself, everything is 'suggested'. It is a one movement piece, containing an introduction, a finale, and four thematic sections. The watches are all described in thematic sections, while references are made concurrently to other features of the painting.

Névoas e Cristais

JÔNATAS MANZOLLI
NÚCLEO INTERDISCIPLINAR DE COMUNICAÇÃO SONORA (NICS)
UNIVERSIDADE ESTADUAL DE CAMPINAS (UNICAMP)
Jonatas@dsif.fee.unicamp.br

RESUMO

"Névoas e Cristais" (Portuguese for clouds and chrystals) is an interactive piece for computer and Vibraphon. The sonic clouds are produced by accumulation of fast and short sounds and punctuation of chords and clusters generate fragmented sonic chrystals. The computer's role is to follow the performer's actions playing a second voice part. This enhances the Vibraphon's sonic features enlarging its pitch range and sound colour. A software written in LISP processes a MIDI buffer to produce the sonic output in real time. This composition is the result of a research in Gesture Interfaces developed by the author at the Interdisciplinary Nucleus for Sound Studies (NICS-UNICAMP).

INTRODUÇÃO

Esta composição é derivada do trabalho de pesquisa em Interfaces Gestuais desenvolvido por **(MANZOLLI, 1990, 95).** Neste obra o estudo está centrado na criação de algoritmos interativos para emitir resposta em tempo real a estímulos gerados por instrumentos controlados via MIDI.

Em "Névoas e Cristais" usou-se um Vibrafone acústico que produz eventos MIDI: um sensor piezo-elétrico fixado em cada uma de suas teclas, possibilita que o computador armazene cada nota executada. A versão original da obra, foi realizada em trabalho de *atelier*, desta forma pode-se testar uma série de efeitos e verificar a eficácia dos mesmos. A peça foi gravada para um CD produzido e executado pelo percussionista André Juarez, somente com obras para Vibrafone.

LISP & BUFFER CIRCULARES

A idéia básica do programa desenvolvido para "Névoas & Cristais" é criar efeitos sonoros de duas naturezas: nuvens sonoras produzidas por aglomerados de notas rápidas e curtas e cristais sonoros gerados por clusters e acordes duplicados ou desdobrados pelo computador.

O programa foi escrito em KC-Lisp, um idioma de LISP desenvolvido no Institute for Sonology **(BERG, 1992).** Esta versão possibilita que um *buffer* de informação (MIDI IN) seja codificado como uma LISTA e o processamento da mesma possa ser enviado para porta MIDI OUT em tempo real.

Uma série de funções modifica o *buffer* de entrada para produzir o *output* do sistema. A natureza destas funções é bastante simples para evitar que o processamento de rotinas complexas prejudique a performance, em tempo real, do sistema.

O *buffer* tem uma estrutura circular. Armazena MIDI e envia informação para processamento assim que é solicitado, ou seja, a estratégia é *First In First Out* (FIFO). Existem dois apontadores: *Apontador de Leitura (AL) e Apontador de Escrita (AE)* que não trabalham sincronizados. A escrita e a leitura são feitas mediante demanda de *input* gerada pelo músico e de processamento pelo computador. Quanto mais notas o músico for capaz de produzir, mais informação haverá para ser processada e *vice-versa*. Após o armazemamento no *buffer,* as notas passam por uma série de transformações diferentes: sorteio de notas graves e agudas em torno de uma nota retirada do buffer, duplicação de notas em oitavas ou arpejos rápidos através da leitura de todo o conteúdo do buffer.

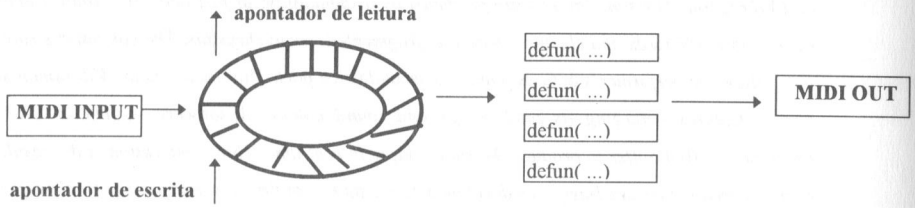


Figura 1.0 - Diagrama simplificado da estrutura do programa de Névoas e Cristais.

SEÇÕES DE NÉVOAS & CRISTAIS

SOLO: SEÇÃO II

Esta seção foi escrita como síntese da peça. Nela introduz-se as idéias da obra e principalmente a sonoridade do vibrafone.

CRISTAIS: SEÇÕES I, III, V

Estas três seções personificam a sonoridade dos cristais. O processamento computacional foi no sentido de produzir notas curtas nas regiões grave e aguda para colorir/iluminar e desdobrar a linha executada pelo instrumento solo.

Seção I - como se estivesse vindo de longe

O trecho inicia-se e desenvolve-se num crescendo constante. O tempo é acelerado gradualmente junto com o aumento da densidade das notas. Este efeito é obtido através da velocidade do executante que é usada pelo computador para processar o **AL** *do buffer:* a medida em que há mais notas no *Buffer*, mais output MIDI é produzido pelo sistema. Esta taxa é controlada pela cálculo da diferença entre **AL e AE**.

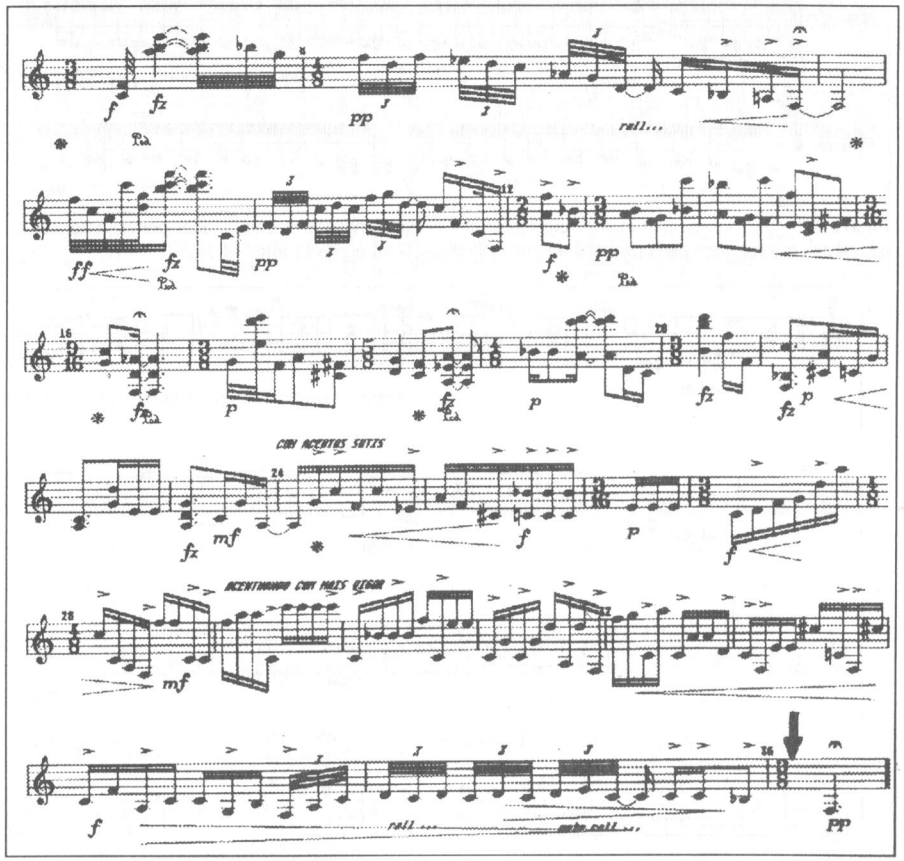


Figura 2.0 - Partitura da seção II de "Névoas e Cristais", solo instrumental amplificado sem interação com o computador.

Seção III - Diálogo

Cria-se um processo de imitação entre a linha executado pelo instrumento e a resposta do sistema. O pulso rítmico mantém-se constante e a dificuldade de execução é manter o tempo deste diálogo. O computador usa a última nota tocada pelo instrumento para disparar o envio de um número de notas pré-determinado para porta MIDI OUT.

Seção V - ressonância de acordes

Trecho lento onde a interação é estabelecida através do desdobramento dos acordes descritos na partitura. O resultado são blocos sonoros com densidade vertical variada. O computador sorteia um número "N" de notas entre (4-8) e oitavas entre (C0,C1,C2, C7,C8,C9) e desdobra as "N" notas do buffer nas respectivas oitavas.

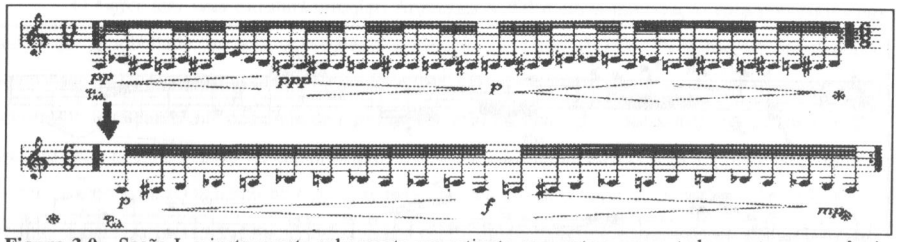


Figura 3.0 - Seção I, o instrumento solo repete um ostinato enquanto o computador pontua a sequência com notas nas regiões aguda (*MIDI note number* entre 72-96) e grave (entre 48-24).

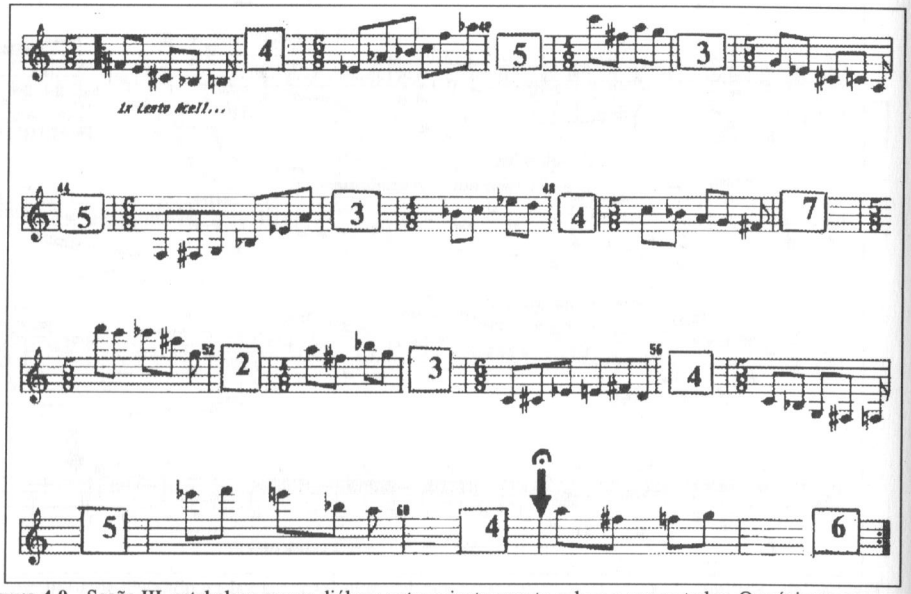


Figura 4.0 - Seção III, estabelece-se um diálogo entre o instrumento solo e o computador. O músico toca as notas escritas e o computador responde o número de notas indicado nos retângulos. As notas produzidas pelo computador são retiradas do buffer circular.

NÉVOAS: SEÇÕES IV, VI

Nestas seções a sonoridade é no sentido de criar-se nuvens de notas. Este resultado é obtido através de procedimentos que duplicam notas, criam arpejos rápidos ou aceleram a leitura do *buffer* circular.

Seção IV - gestos interativos

Esta seção é a mais livre de todas, é executada como se o instrumentista pulverizasse livremente o espaço com massas sonoras. A partitura indica o sentido do gesto e a região sonora; a altura, duração e intensidade são indeterminadas. Cada quatro notas tocadas pelo músico disparam uma leitura simultânea do *buffer* fazendo com que o computador produza trajetórias complexas.

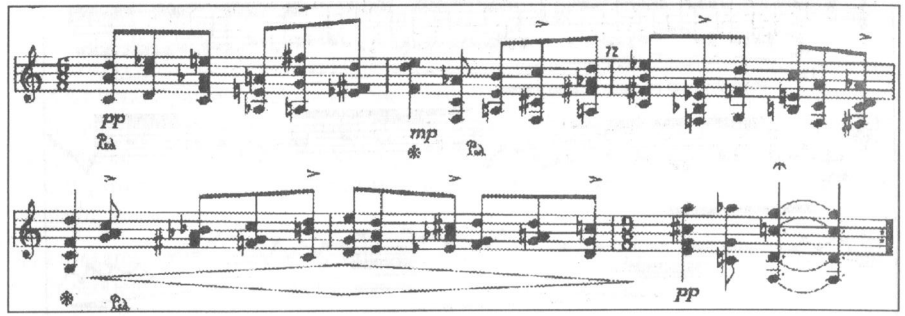


Figura 5.0 - Seção V, uma seqüencia de acordes colorida com notas em regiões agudas e graves, cada acorde é desdobrado aleatóriamente em oitavas.

Seção VI - criando massa sonora

O mesmo processo da Seção I é usado no início deste trecho. Todavia, a medida em que o executante aumenta sua velocidade o computador quadruplica o número de notas enviadas para porta MIDI. Esta interação é levado aos limites de tempo de resposta da máquina, o que cria acúmulo de notas que são dispararadas desordenadamente e em blocos.

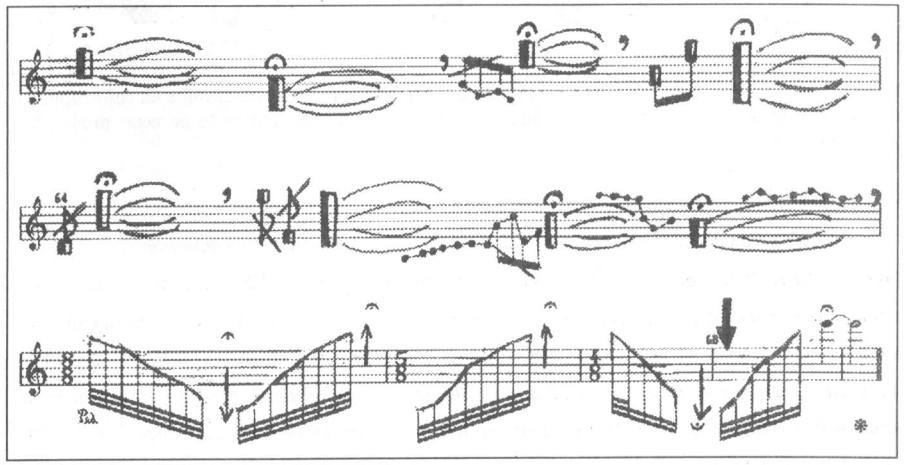


Figura 6.0 - Seção IV, neste trecho o instrumentista dispara trajetórias complexas produzidas através da leitura rápida de todo o conteúdo do buffer.

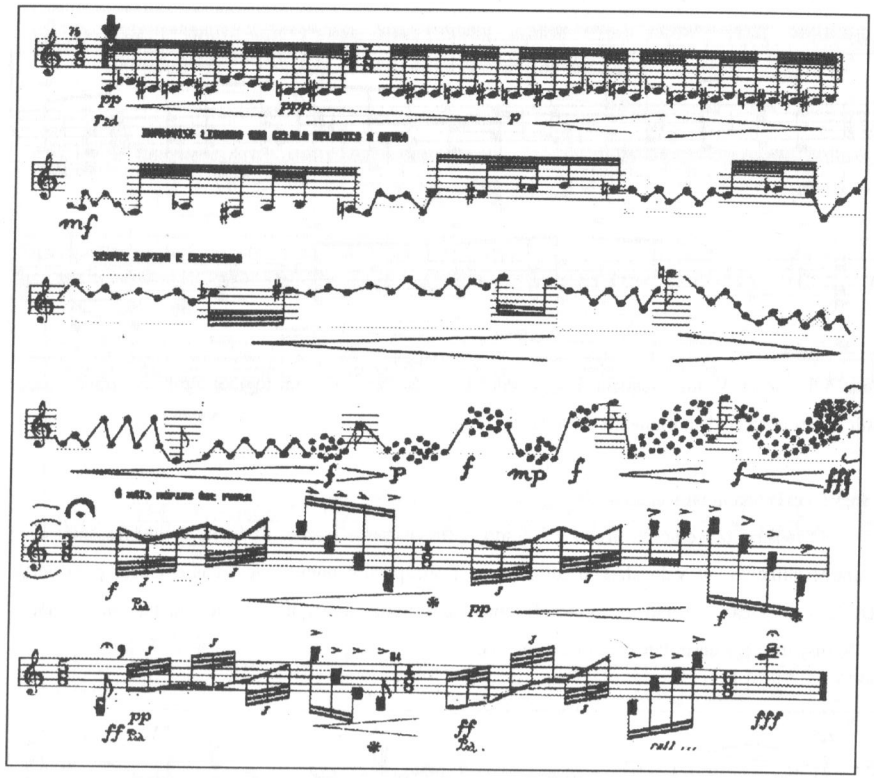


Figura 7.0 - Seção VI, inicia-se com a mesma sonoridade da Seção I e se desenvolve na construção de massas sonoras geradas pela execução rápida do instrumentista e pela duplicação de notas produzidas pelo computador.

CONCLUSÃO

"Névoas e Cristais" pode ser executado também com a Luva Interativa desenvolvida pelo autor **(MANZOLLI, 1995)** ou outro instrumento que produza eventos MIDI. É possível usar duas configurações para diferentes instrumentos: configuração solo - um executante toca um instrumento MIDI como um Yamaha Dysklavier, WX11 Wind Controller, um Vibrafone MIDI ou uma Luva Interativa, configuração dupla - dois executantes tocam dois controladores MIDI diferentes onde as seções solo e cristais (I,II, III e V) ficam para um instrumento de teclado e as seções névoas (IV e VI) para o outro instrumento.

BIBLIOGRAFIA

BERG, P. (1992). *A Brief Introduction to LISP.* Institute for Sonology, The Hague.

MANZOLLI, J. (1991). *BODI Interface.* CEM Bulletin: Centrum voor Electronische Muziek - Holanda, Vol 2 (2) 8-9.

MANZOLLI, J. (1995). *The Development of a Gesture Interfaces' Laboratory.* Anais do II Simpósio Brasileiro de Computação e Música, XV Congresso da Sociedade Brasileira de Computação - Canela RS, PG 81-84.

Vox Victæ (1996-1997; in progress)

Didier Guigue

All low-level material (pitches) is generated by a collection of algorithms built with the *Profiles* library patches included in the Ircam=B9s Patchwork Composing Environment. Each algorithm provides a controlled sequence of interpolated pitches, chords or harmonic complexes from B to A (1st part of the piece, titled "Profile to A") or reversely A to B (3rd part, titled "Profiles to B"). The 2d part represents a stationnary moment of the form.

As an example, the global structure of the III part attends the following format (PS = Pitch Sequence; CS = Chord or Harmonic Complexes Sequence):

00=B900=B200	PS 1a
02=B902=B200	Gap (Low B)
02=B916=B200	PS 1b
03=B947=B206	CS 1
04=B959=B220	PS 2a
06=B949=B221	Gap (Low B)
06=B958=B210	PS 2b
08=B918=B215	CS 2
12=B932=B200	End

In PS sections, positive or negative correlations are estabilished between the pitch sequences themselves and: the linear directionalities of both amplitude (crescendo or diminuendo) and sound focusing/defocusing (by a reverb/chorus increment or decrement) the non-linear directionality of tuning rates progression (from 1/8 to 5/4 of tone) the global tempo progression (increasing or decreasing).

Pitch sequences use sampled piano sounds, and chord sequences, complex sounds (harmonic or non-harmonic) made on a Korg WaveStation. Chord sequences are sustained with sampled String orchestra sounds.

About the title

As a whole, this piece refers to the determinist pessimism which goes through all the work of the Brazilian poet Augusto dos Anjos. 'Vox Victiae' is one the poems I choosed as a reference. The II part of the piece includes excerpts of this and other poems by this autor. The way I worked with frequently obscured or defocused pianistic timbres, the handling of time and periodicity of events, the ineluctable determinism of interpolation as a time process, transfer to the musical domain my own reading of the poet's obsessions

Duration: ca 24'

Instrumentation: Sound Generators (Korg WaveStation and sampler) on DAT tape.

El Poder de lo Invisible (1996)

Carlos Cerana

(See abstract on page 195 in man-machine concerts)

ON-LINE CONCERTS

Three-Threaded Invention (1997)

Aluizio Arcela

(For a full description of this on-line concert--which has been written for one sonic client and three graphic servers--see the homonymous paper on page 171)

Lexikon-Sonate (1992-97)

K@rlheinz Essl

"Lexikon-Sonate" is a work-in-progress which was started in 1992. Instead of being a composition in which the structure is fixed by notation, it manifests itself as a computer program that composes the piece - or, more precisely: an excerpt of a virtually endless piano piece - in real time. Lexikon-Sonate lacks two characteristics of a traditional piano piece:

1) there is no pre-composed text to be interpreted, and

2) there is no need for an interpreter.

Instead, the instructions for playing the piano - the indication "which key should be pressed how quickly and held down for how long" - are directly generated by a computer program and transmitted immediately to a player piano (or a MIDI synthesizer) which executes them.

The title "Lexikon-Sonate" refers to the Lexikon-Roman, written in 1968-70 by the Austrian-Slovakian author Andreas Okopenko. This novel appears to be one of the very first literary HyperTexts, several years before this term was introduced by Ted Nelson. This novel - "a sentimental journey to a meeting of exporters in Druden" (subtitle) - consists of several hundred small chapters which were brought into alphabetical order. By reference arrows as in a lexicon the reader could make her own investigations through the multiple nested web structure of the text. Instead of presenting a sequential text with a predefined direction of reading, Okopenko provides a structure of possibilities, which challenges the reader to become a creator of her own version of this novel.

Originally, "Lexikon-Sonate" was conceived as a musical commentary to an electronic implementation of Okopenko's Lexikon-Roman, carried out by the interdisciplinary group "Libraries of the Mind". But soon afterwards it started its own life due to its manifold ramifications, becoming an outstanding example in the domain of algorithmic composition.

Up-to-now, "Lexikon-Sonate" consists of 24 music-generation modules (structure generators) which are related in a very complex way as a musical HyperText. Each module generates a specific characteristic musical output as a result of the compositional strategy that has been applied. A module represents an abstract model of a specific musical behaviour. It does not contain any pre-organized musical material, but a formal description of it and the methods how it is being processed.

These modules are structural re-implementations of piano gestures obtained by analysis of piano music from Johann Sebastian Bach, Beethoven, Schoenberg, Webern, Boulez, Stockhausen and Cecil Taylor. They will never appear as verbal quotation (because none of this gestures has been "sampled"), but mainly as "allusion". Furthermore, they are open and generic enough so that different modules playing at the same time can intermingle, creating unforeseeable meta-structures.

The idea of autopoiesis - material organizing itself due to certain constraints - plays an important rule. By using a lot of different random generators which are controlling each other (which - according to serial thinking - form a scale between a completely deterministic and a completely chaotic behaviour), always new variants of the same model are generated. Variants that may differ dramatically from each other, though they are always perceptable as "inheritances" of the given structural model. Seen in this light, "Lexikon-Sonate" can be perceived rather as a meta-composition which enables the unfolding of piano music than a fixed work.

The underlying program was written in MAX (Puckette & Zicarelli, (c) 1990-1996 Opcode Systems Inc./ IRCAM), an interactive graphical programming environment for multimedia, music, and MIDI, running on a Macintosh computer. It draws from a large library of musical functions, compositional techniques, and algorithmic strategies which I have developed over the past few years: the "Realtime Composition Library" for MAX.

COMPOSIÇÃO ALGORÍTMICA
DE PROCESSOS ELEMENTARES À CONSTRUÇÃO DE TIMBRES

JÔNATAS MANZOLLI
Núcleo Interdisciplinar de Comunicação Sonora (Nics), Unicamp

ABSTRACT

This workshop is to introduce basic methods of Algorithmic Composition. It starts on Stochastic Methods moving to algorithmic methods for Timbral Design. Models to be used on macro and micro sound structures will be presented. It discusses also basic data structures and functions to control music events in real time. Two programs created by the author called Interasom e Morphsom.will be used to produce sound examples. It is plained to an interdisciplinary group, to make the participants to develop their own view of how to use algorithmic tools on Computer Music.

O ESPAÇO COMO DIMENSÃO EXPLORÁVEL NA MÚSICA

RODOLFO CAESAR
LaMuT, Escola de Música, UFRJ

RESUMO

Dividido em segmentos teóricos e práticos, o tutorial esclarecerá as diversas noções de espaço na música, e proporá exercícios visando a exploração do mesmo como uma dimensão poética e técnica.

Index of Authors

INDEX OF AUTHORS