Computer Assisted Composition today.

Gérard Assayag

Music Representation Team

Ircam, 1 Place Stravinsky F-75004 Paris, France

Assayag@ircam.fr

http://www.ircam.fr/equipes/repmus

1st SYMPOSIUM ON MUSIC AND COMPUTERS. "Applications on Contemporary Music Creation: Esthetic and Technical Aspects". Corfu, 23-25 October 1998.

The first computer experiments on musical composition were carried in the middle of the Fifties. In 1956 R. C. Pinkerton suggested a stochastic composition system which he called the " banal tunemaker ". He used 39 existing folk melodies, as a material that he submitted to random choices by launching dices. This work, as well as an implementation of Mozart's dice game by D. A. Caplin in 1955, was a source of inspiration for J. Cohen and J. Sowa who developed soon after their so-called " Machine to Compose Music ", a computer program, which used in much the same way a set of simple piano pieces.

The machine of Olson and Belar, built at the beginning of the fifties and described in a paper published in 1961, was not really a computer, but a set of electronic circuits comprising a subsystem for sound generation and another one for stochastic composition.

The methodology followed for this first set of experiments may be described in three stages:

- Probabilistic analysis of a corpus of existing music
- Random generation using Markov tables derived from the first step
- (human) choice among the results

One had to wait for L. Hiller however to actually talk about computerized composition. With L. Isaacson, Hiller created in 1957 the first original musical piece made with a computer: " Illiac Suite for String Quartet ". Hiller defended the idea of a " subtractive " approach, using the language of information and system theory that was quite in the mood at the time: "... the process of musical composition can be characterized by the extraction of order from a chaotic multitude of available possibilities... ". The experiments that were led for the Illiac Suite were not so far from Pinkerton's paradigm; however Hiller proposed several innovations. The basic material for the Illiac suite is generated dynamically and in great quantity by the use of the Monte-Carlo algorithm. The latter products numbers which Hiller used to codify various musical parameters, as pitches, dynamics, rhythmical groups, and even instrumental techniques. These parameters were then subjected to a set of compositional rules (inspired by Fux's works on Palestrina). As a difference with the first works mentioned, where the choice was made by looking into tables, here the rules determine the validity of the material. The rules were implemented by using the technique known as "Markov chains". Let us give an example to illustrate the use of this technique in the musical field: suppose we take an existing melody and we build a table in which, for each note, we calculate the probabilities that it is followed by any other note of the dodecaphonic scale. Once the table has been built, we can produce various melodies while respecting the established probabilities.

In a similar way, we can carry the experiment by creating the table and stuffing it with coefficients without needing to analyse a given melody. This example shows a simple use of a Markov Chain. Indeed, the example takes only the immediate predecessor of each new note into account. One speaks then, of a Markov Chain of degree 1. Hiller’s rules used relations of degree $N > 1$. Indeed, the higher the degree, the better the control.

Hiller’s research was a major breakthrough as it opened a new perspective for musical engineering, even if the interest of the artistic result itself may be discussed. It initiated the practice of algorithmic composition, which is still alive, especially in the United States.

The works undertaken by P. Barbaud and R. Blanchard in France at the beginning of the Sixties also hold a significant place in the history of Computer Assisted Composition [Barb68]. " 7! ", written in 1960, is acknowledged as the first computer music piece in France. Just as Hiller, Barbaud uses stochastic techniques, arguing on the fact that music oscillates between order and disorder. However he goes further in the formalization of the musical process in order to translate his ideas in the language of the machines. The theoretical ground on which Barbaud builds his system is the set theory, which he uses as a basis for the analysis of the tonal language. He defines for example the set of pitches as Z/12, the set of remainders modulo 12, and the operation of transposition as the addition in Z/12. Other actors of the tonal language like scales and chords are reduced to sets or sets of sets. Automatic composition, for Barbaud, is the result of the application of various types of rules on data expressed in the form of sets. The rules are specified by finite-state automata or stochastic matrix. Using this tools, Barbaud may generate counterpoint with harmonization and control, up to a certain point, the stylistic imitation. In this set-theoretical vision, however, the problem of musical time remains problematic, because the only way to speak about time is by the succession of events, which implies a strictly linear and irreversible time.

The artistic step of Barbaud is important because it includes a general formalization of music theory. Other precursors, like Philippot or Riotte, made it possible for this French school to occupy an important place in the genesis of Computer Assisted Composition (CAC).

All the works exposed up to now are linked to automatic or algorithmic composition. A prior formalization makes it possible to program an automaton whose output is considered, without final improvement, as the musical result. The basic argument is the division of the composition act in its technical and creative aspects. It is then possible for an algorithm to reduce the technical difficulties and facilitate creation. This separation is however questionable. In addressing the issue of sharing the compositional process between machine and man, G. Koenig [Koen71] advanced a step forward with its programs Project1 and Project2. Koenig’s programs calculate musical structures, starting from a fixed specification of their global shape. Thus the composer gives some parameter lists describing the durations, the tempi, the chords, etc and the program computes the periodic or aperiodic distribution of theses elements to produce the composition.

The musical work here is the result of some symbolic combinations (independent of the physical phenomena) and of a deterministic specification at the level of the musical form. The specification of the form is given a priori and is not controllable afterwards.

Algorithmic composition as described here still has a significant number of followers. Composers like D. Cope or C. Ames, among others, undertake researches based on the idea to establish musical parameter fields on which one imposes automatic control structures whose execution generates the musical piece. Cope evolved recently to a model of databases containing a great quantity of elementary musical gestures that capture elements of style. Algorithmics focus then on the recombination of this " genetic " material according to predetermined forms by solving local articulation problems and by imposing the necessary local transformations.

Cope produced interesting simulations in the style of Bach, Mozart, or Beethoven; but Cope is also a composer. In this case, the problem for him is to define the atoms of his own musical style. There is here the implicit assumption that musical creation consists of the recombination and working out of preexisting cognitive elements representing the non-formalizable part, the absolute originality of a creator, in other words, the style. If this database-oriented approach makes sense for musicology, it seems marked by a too naive idealism as far as contemporary creation is concerned. It denies the idea of invention for which it substitutes that of combinative discovery of the musical " self ", and, by there, seems not very likely to reach real innovation.

Of course in this historical survey of CAC we have to mention the work of I. Xenakis [Xena81] who based its music in a completely original way on the principle of indeterminism. The naturalness with which the music of Xenakis can make use of the computer comes from the fact that the language of his ideas corresponds to the language of the machines. In the various writings of Xenakis the desire to develop a language to handle sound events that would be derived from mathematics, logic and physics is obvious. For this composer, the application of serialism to other compositional parameters than pitches degenerates in the loss of coherence in the polyphonic discourse. Taking note of this " death of the articulated musical discourse ", he chooses to assume a non-syntactic paradigm of musical organization. The primitive elements in his works are not any more the notes then, but clouds of sounds whose development in time is controlled by probability distributions. Composition for Xenakis is held in the filling of a two-dimensional time-frequency space by sets of sound primitives distributed in various areas with various densities.

To finish this first historical stage we will speak about the software Musicomp (Music Simulator Interpreter for Compositional Procedures), which is perhaps the first software designed for assistance to the composition, in opposition to automatic composition. Musicomp was written by Robert Baker a round 1963 with the expertise of L. Hiller and can be seen like as a collection of tools for solving problems specific to musical composition, in the form of sub-routines in the FORTRAN language. Musicomp includes three families of routines: generation routines (mainly Markov Chains), serial and geometrical routines called modifiers and a set of selection rules inspired from traditional harmony.

The three principal musical experiments carried out with Musicomp were: the resolution of problems of rhythmic organization for percussion; the generation of serial music using the model of "Structures pour deux pianos" by Boulez and in general the research of vertical and horizontal structures in tempered scales [Hill69].

Let us quote finally works by B. Truax and S. Pope who tried in the Seventies and Eighties to exploit the contribution of Artificial Intelligence [BAL92] by proposing representations inspired from graph and language theory, and started to use modern programming environments such as SmallTalk.

After this pioneer period, CAC suffered from the considerable development of digital audio technologies. Massively attracting means and people, tempting by its immediate rendering of a new sound world, researches in digital sound synthesis and processing also gave a more scientific status to computer music and perhaps rang the bell for the likened " composer-engineer " character who had been so often associated to former works. This powerful attraction towards the world of digital sound created the same deficiency in terms of creativity as the one noticed in the world of digital image synthesis and processing: these digital images, attractive demonstrations of technological power, hardly showed a relevant thinking on the conditions of emergence of a new artistic language.

In the field of musical sound, some famous exceptions must of course be mentioned: Jean-Claude Risset and John Chowning for instance have maintained the tradition of the composer-researcher and have known how to gain profit from their original scientific results in order to work out a musical language integrating sound synthesis without any artistic compromise.

After a phase of sleep, the " modern " CAC has emerged since the middle of the Eighties. This revival is conditioned by a certain number of factors: the availability of personal computers; progress in graphical interfaces; creation of standards in inter-machine communication, among which, since 1983, the MIDI standard (Music Instrument Digital Interfaces); and progresses in programming languages, the most significant factor in our point of view. In the same way that the choice of a programming language influences the programmer, it plays a role in the formalization of a musical idea. Indeed, the use of a language can suggest expressions that would not be favored in another context.

Advances in the field of programming languages was thus fundamental for CAC as the use of a programming language forces the musician to reflect on the process of formalization and avoids him to consider the computer as a black box which imposes its choices to him. The programming languages offer the composer an enormous freedom of decision, in exchange of a certain effort in formalization and design.

This contemporary CAC have been gradually defined by the contributions of people like S. Pope [Pope91] who proposed several systems based on the Smalltalk language, with the possibility for the composer to define its own objects and methods and to benefit from a high level graphical interface, E. Taube [Taub91] who developed at Stanford a program for composition by patterns called " Common Music ", F. Pachet who puts at work objects and constraints in MusES [Pach94]. These environments are characterized by high-level programming languages; modern programming concepts like objects and constraints, and sophisticated graphical interfaces. These systems are opened, which means that the composer may extend them if he accepts to get involved into some computer sciences.

The contribution of Ircam (Institut de Recherche et de Coordination Acoustique Musique) to the emergence of CAC deserves a special treatment because it is one of the rare places which has been implied institutionally and with perseverance since about fifteen years in the defense of these ideas. Since the beginning of the Eighties several successive teams worked on the difficulty to represent and handle musical structures and knowledge in a compositional perspective.

Formes [RoCo85] was maybe the first CAC environment to be carried out at Ircam. It was conceived by Xavier Rodet and Pierre Cointe and was implemented in VLisp between 1982 and 1985. Formes is an environment for Synthesis and Musical Composition that conceives the sound as a logical production. However this type of application is only one particular case of the great possibilities that this environment offers.

Formes is based on the idea to represent the musical objects by software actors, and the control structures by a message sending mechanism. This language is directed towards the composition and the scheduling of the temporal objects. The central concept is that of process. The structure of a process is defined by a set of rules, a monitor, an environment and a set of child processes. The behavior of a process is defined by the messages that it receives: a process can fall asleep, awake, wait or be synchronized. Any process has a duration defined a priori, without it being necessarily explicit; the duration of a process corresponds to the time during which it is active. The process executes its rules at the time when it becomes active.

In general, activating a process comes down to launching a sound synthesis procedure. The children of a process must satisfy certain conditions; mainly their duration must be contained in the duration of the father-process. The monitor of a process indicates the mode according to which its children are organized, in sequence or in parallel.

A program in Forms is initiated by the sending of an activation message to a root process. The execution of the program consists of the repetition at successive times of the following steps: update of the rules belonging to the computation tree, then execution of these rules. The computation tree at time T is a list of rules belonging to the processes that are active at this moment, ordered according to the position of each process in the hierarchy.

Integration between synthesis and composition, problematized for the first time in Formes, remains a current problem. Unfortunately the major characteristic of Forms constitutes its weakness from our point of view; we think indeed that continuous and irreversible time, necessary for sound synthesis, is not the better paradigm for music composition in general. Note however that, in Formes, one can speak of time, which is not the case in MAX, the well-known real time sound processing by Miller Puckette and David Zicarelli, nor in its descendants.

PreForm, implemented by Lee Boyton and Jacques Duthen in 1987, was an attempt to equip Formes with a graphical interface on the Macintosh and to make it compatible with the Midi world. One of the principal applications implemented on PreForm was Esquisse. Conceived by

P. F. Baisnee and J Duthen with the assistance of a group of musicians among which was Tristan Murail. Esquisse starts from the composer's need for multidimensional musical objects as well as for functions to generate such objects or to transform them. These functions were classified according to different types of musical knowledge: intervals, spectral harmony, interpolations and the like. Though very simple Esquisse was one of the first CAC toolbox with a real expertise coming from the actual actors in contemporary music and it remains very much used today.

The environment Crime [AsCa85], written in Lisp by Gerard Assayag in collaboration with composer Claudy Malherbe, was the first attempt to carry out a general environment in which the user could handle general musical formalisms with results displayed in the form of a musical score using traditional notation. Crime provided the composers with formal languages enabling them to define arbitrarily complex rhythmic, harmonic, and polyphonic structures. Crime also included the first psycho-acoustical models to appear in a CAC environment, namely the Terhardt analysis for pitch salience and virtual fundamental extraction from a complex sound spectrum. This was used for a lot of musical works at the time, e.g. works by Malherbe, Stroppa, Benjamin, Lindberg, Saariaho.

One of the first software adapted to the visualization of compositional syntax and processes was Carla [Cour93]. Written in prolog II by Francis Courtot in 1989-1990 Carla is a graphical interface for logical programming. Carla starts from the principle that "... if it is certain that it is impossible to formalize all musical syntaxes in a universal representation, it must be possible to formalize a language that makes it possible for each composer to define his own set of syntaxes ".

The Carla environment consisted of: a set of basic types and a set of heuristic associated with each type; a model and a graphical interface for formalizing the relations between types; and a logical programming graphical environment. There are two classes of types in Carla: primitive types, defined by using a first order logical language, and complex types. A primitive type is defined by a set of attributes including a value domain. Complex types are built starting from already existing types using 4 building tools: *Ho for construction in sequence; *ve for construction in parallel; * nup for the union of attributes and *union for the union of types.

The composer can define a semantic structure by associating a semantic value with each type and by establishing relations in the form of a conceptual graph. Carla produces a semantic value by default for each type, which facilitates the task of the composer. The graphical interface makes it possible to examine the conceptual graph, where the nodes correspond to the types and the edges are binary relations between types.

The Music Representations team of Ircam, founded in 1992, profited from all the work exposed above in its attempt to build a coherent research thread clearly identified under the name of Computer Assisted Composition, with a particular accent on the concept of "écriture" (an untranslatable French term which is the union of instrumental score writing and musical thought in general — we will use here the term *instrumental writing* or simply *writing*). We try to define here the general framework in which we situate our work.

The composition is present, in the state of project or realization, in all the sectors of computer music research; consequently, why try to constitute Computer Assisted Composition as an autonomous discipline? We propose this brief reply: the specificity of instrumental writing.

Instrumental writing seems to us a field of study which is at the same time precise and open: it constitutes a still unequalled model of adequacy between combinatorial systems of operations on sets of symbols on one hand and a sound universe having its own rules of perceptive and cognitive operation on the other hand. Indeed, the notation operates a coherent link between these two worlds. By there, it is also open to integration of new sound materials - extensions of the instrumental world, synthesis and transformation of sound. Instrumental writing brings with itself an almost ideal combination of formalization, codification, notation, and relation to the physical and perceptive world. We will thus employ the term Computer Assisted Composition while privileging, among all possible interpretations, the one that approaches at most the idea of instrumental writing. CAC systems will have to be able to bring an effective help in the specific case of instrumental writing; they will have to constitute a coherent link between the latter and new (synthetic) sound fields. They will be able in the long term to constitute good models for the control of pure synthesis. We now structure our conceptual framework around three axes: language, notation and perception.

- Language. First of all, it is necessary for us to put in adequacy two activities of a symbolic and combinatorial nature: on a side, the search for new elements of musical language and new ways of structuring them; on the other side, the specification and interactive exploitation of algorithms allowing to actualize these ideas and to explore the technical and artistic consequences involved by them.

- Notation. To build structures, to express computations has utility only insofar as data and results can be interpreted and represented in musically significant dimensions, such as pitches, duration, intensity, or timbre, if we limit ourselves to traditional categories. To provide visual and audio interfaces which improve this interpretation is then an imperative need. We give a great importance to traditional and extended musical notation. Notation acts ideally in a CAC environment not only as a materialization of information going around in the system but also as a medium where formal inventiveness lives. For this reason, notation should be provided with the same flexibility, the same opening (in terms of extensibility and programmability) than the programming language itself. Notation should in the long term constitute the natural environment of experimentation. The levels of language and notation will then tend to merge from the point of view of the user.

- Perception. Handling compositional material is eminently combinatorial. All the work could remain at the stage of formal speculation if a solid link were not established with perception. Several principles can be applied for this purpose: constitution of audio tests starting from the musical outputs, connection of the CAC software to analysis and synthesis tools, or integration of acoustic and psycho-acoustic models in the system itself. Such experiments were recently carried out in the PatchWork environment with models for virtual fundamental and chord roughness computation.

Although many important advances have been recently achieved in the visual expressiveness of CAC languages, the coherent representation of the language, notation, perception trinity still remains an ideal to be reached. In particular, the level of the notation will have to tend to the same modularity as that of the programming language in order constitute a programmable and customizable interface for the musical constraints and knowledge. In order to achieve this ideal, the original concepts developed in the field of visual programming will have to find their counterpart in notation. Thus the composer will program with a complete freedom the functional calculation which will determine the contours of his field of musical experimentation, the constraints which will print a structure into it, the degrees of freedom which will allow a navigation in the knowledge space thus made up. Finally he will be able to choose to mask all the stages of development by defining a *potential score* where he will superimpose display of the results, interactive controls, input of the musical parameters. This potential score will act as a kind of white sheet " informed " by a hierarchy of constraints leading from the constraints of elementary musical consistence up to those, of higher level, expressing the stylistics characteristics that are personal to the composer.

Crucial to the idea of potential score is the concept of computer model of a musical structure, more powerful and more general than that of a musical draft ("esquisse" in french) used formerly. This concept is at the heart of current research in the field of music analysis as well as in CAC. A. Riotte and M. Mesnage have thus produced several models for the Twentieth Century repertoire, among which a model for the "First Piece for String Quartet by" Stravinsky or another one for the "Variations for Piano opus 27" by Webern [MeRio88-89]. But what does one mean exactly here by model? In general, a model is a formal device which, giving account, at least partially, of the characteristics of a physical process, allows an experimental simulation of that process for the purpose of checking, observing, or producing similar processes. Models used in sound synthesis, like the physical models, additive synthesis or frequency modulation work that way. Stylistic simulation, such as "Bach" chorals by K. Ebcioglu, quite fits with this definition, since it is capable to produce an indefinite number of musical instances obeying the laws of a style. But the models of scores by Mesnage and Riotte constitute an extreme case insofar as they deal entirely with the restitution of a single object. They however do not constitute a simple description of the latter. They generalize it insofar as they substitute a collection of formal mechanisms to it, a collection whose particular parameterization will provide the final object. It is then permissible to consider, by testing other sets of parameters, the generation of alternatives to the analyzed text. A similar approach was employed recently by Jean-Pierre Balpe in the field of literature. By this juggling act consisting in the exploitation of the ambiguity of the term "model" (the model is the genetic source of the piece, the piece constitutes the model intended to be imitated during simulation) modern analysis tends to play in the category of creation; and the analyst using scientific tools is not any more very different from the composer confronted with the same tools [Chem92-94].

It is in their action on the model that the two attitudes diverge however. In the analytical case, the successive refining of the model tend to minimize the number of parameters -- the ideal case being their complete disappearance, the values of the parameters of a level of formalization being generated by a formalism of higher level. Thus, it is not so much the experimentation on the model -- the simulation -- that prevails, than its progressive refinement since the stage of the paraphrase until that of the explanation.

For the composer, on the contrary, only the productive experimentation counts. The model is then handled according to two methods: on the one hand the objects which it produces serve as structured musical materials which can be put aside progressively while experimenting. On the other hand, the observation of these elements can lead to the calling into question of the subjacent musical theory, which will be altered consequently, leading to the development of a new model. One then finds the traditional concept of simulation as validation of the theory, with this reservation (and it is a fundamental difference with the scientific posture): the reference phenomenon which should lead the validating comparison does not exist except in the state of an ideal in the composer's imaginary. To the generalizing virtues of the scientific model, the composer, using the computer, adds a teleonomic perspective, aiming at a singular if not single musical work. That is not the least paradox in the articulation between music and sciences.

The use of computers tends to unify the two instances, explanatory and generative, of the model. Indeed, a computer program is initially a " text ", expressing a network of relations with the conventions of a formal language. It gives place during its execution to the deployment in time of a process whose form is regulated by these relations. It thus reproduces, by a kind of "mise en abyme", the dialectics between model and simulation. The computer program, implementing a compositional or analytical computation, thus occupies an ideal position, insofar as it exhibits its logical components in an understandable form. It is this important step that was crossed with the Visual Programming paradigm by which one substitutes diagrammatic charts to the formal language mentioned above.

Because of the very great diversity of esthetical and formal (or anti-formal) models which coexists in contemporary music, one cannot imagine any more a CAC environment as a rigid application offering a finished collection of procedures for generation and transformation. On the contrary, we conceive such an environment as a specialized computer language that composers will use to build their own musical universe. Of course, the idea is not to provide them with a traditional language, whose control requires a great technical expertise, but a language arranged especially for their needs. This leads us to reflect on the various existing programming models, as well as on the interfaces, preferably graphical, intuitive, which make it possible to control this programming, and on the representations, internal and external, of the musical structures, which will be built and transformed using this programming. The ideal is thus a language that encapsulates in a consubstantial way the concept of notation, notation of the result (a musical score) but also notation of the process leading to this result (a visual program).

Several experimental environments for music production were carried out according to the principles reported above in the Music Representation team, in particular, PatchWork and OpenMusic. The PatchWork language [Laur96] developed at Ircam by M. Laurson, J. Duthen, C. Rueda, C. Agon and G. Assayag, was the first stage of the visual integration mentioned above. It was used and expertized by several important composers or musicologists (T. Murail, B. Ferneyough, C. Malherbe, G. Grisey, M. fano among others) who gave considerable feedback.

Recent research in our team led then to the development of the OpenMusic language [AADR98] by G. Assayag, C. Agon, C. Rueda. Before approaching this language, recall that our problem is not the performance in real time. The substance that our models will handle is writing ("écriture"), taken as a universe of forms, structures and relations.

Once this objective pointed out, the question of the notation emerges immediately. Indeed, who talks about writing talks about notation. Let us risk a definition. Notation is a device with two faces: on one hand a concrete and sensitive face made of signs that are materialized on the sheet or on the screen; on the other hand an abstract face, a set of relations that constrain the use of these signs and give them meaning. Music writing is the dynamic process of using notation to format structures and relations of higher level. The first of these higher structuring levels is directly discernible in the concrete substance of the notation : for example a repetition. By analogy with the language, one will qualify this level of syntactical. The second level of structuring is not directly perceivable : it is necessary for that to imagine in the sound space a virtual execution of the score. For example, the evolution of a harmonic color. There is the need here for a beyond-the-notation universe that proceeds from our knowledge and our imaginary. Still by analogy we propose to qualify this level of semantic.

With the computer, the musical score is only one aspect among others. Indeed, the composer in front of his screen handles various objects. Some of these objects are computational objects. This is not new in itself: there always has been some calculation in music. But now, the degree of interaction is such, between computed objects, literal fragments of music materials injected into the system and formal structures, that the concept of score tends to fade out and leave the place to the concept of dynamic model of a score, what we called above a potential score.

It is then on all these levels that notation intervenes. Notation of the programs computing musical materials. Notation of the results of these programs, in mathematical form (e.g. curves) or musical forms (e.g. notes). Notation of the formal structures which articulate these materials. Notation of the programs which take as input these formal structures and again transform them. One enters an infinite recursion which the creator decides to terminate at some point by gauging the level of complexity he wishes (or he is able) to reach and control. In the case of visual programming, the computer program becomes itself a notation since it is a logical diagram.

We have tried with OpenMusic to achieve a seamless integration of these various levels of notation intervening in the musical design.

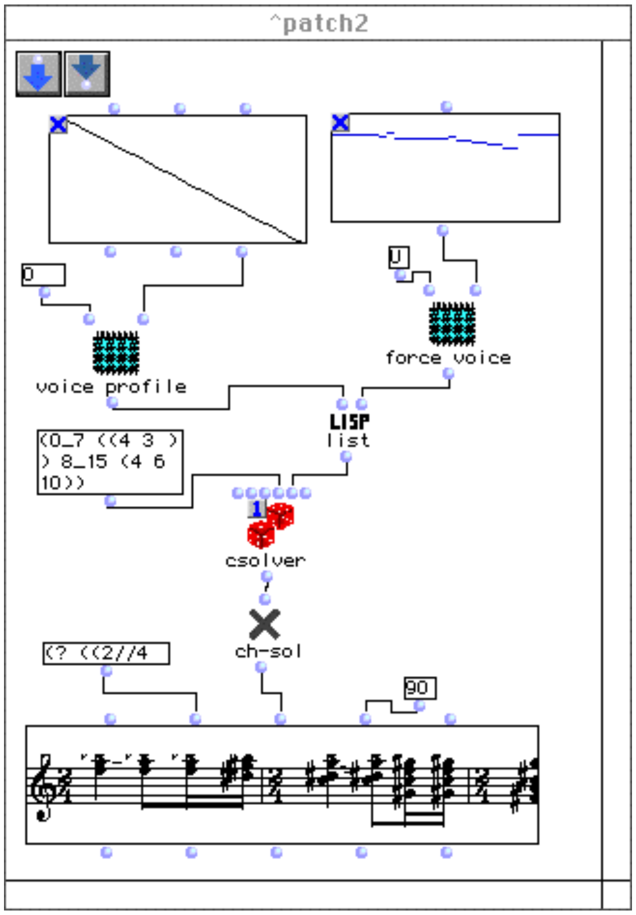
In OpenMusic as in PatchWork, the computing unit is the function. A function is materialized by an icon, which has inputs and outputs. The icons are interconnected to form a patch. The patch is the unit of program. In its turn a patch may be collapsed into an icon, thus masking its internal complexity. The collapsed patch becomes then an atom of computation in another patch. This the first recursive aspect, which one will use to show or hide complexity. Programming then becomes a graphical art by which one will try to make the syntactical and semantic aspects obvious.

As a difference to PatchWork, functions are polymorphic [Cast98]. A function is then to be seen as a set of methods, each of which carries out a computation dependent on the type of the objects that it treats. Thus, the function "addition ", as applied to numbers, will calculate their sum ; as applied to two musical voices, it will operate their rhythmic fusion, if this is what we want. There is a considerable advantage here which is to support abstraction. The same icon will be used in both cases, and in both cases an eyesight at the visual program will provide a semantic indication on the type of operation concerned. One thus passes from the concept of computation to the concept of class of computations by an artifice of notation based on visual identity.

In addition to functions the concept of object has been introduced [Cast98]. An object is an instance, an actualization, of another object called a class. A class is an abstract model necessary to manufacture concrete instances. Examples of classes are *integer*, *set*, *harmonic structure*, etc. A class may have subclasses: for example the class *chord* is a subclass of *harmonic structure*. The objects created directly by the user or resulting from a computation can be materialized : they literally appear at the surface of the patch, in the forms of new icons, which become new sources of value or information storage places. An object in OpenMusic can thus be a simple number, a text or a twenty-two-voice polyphony.

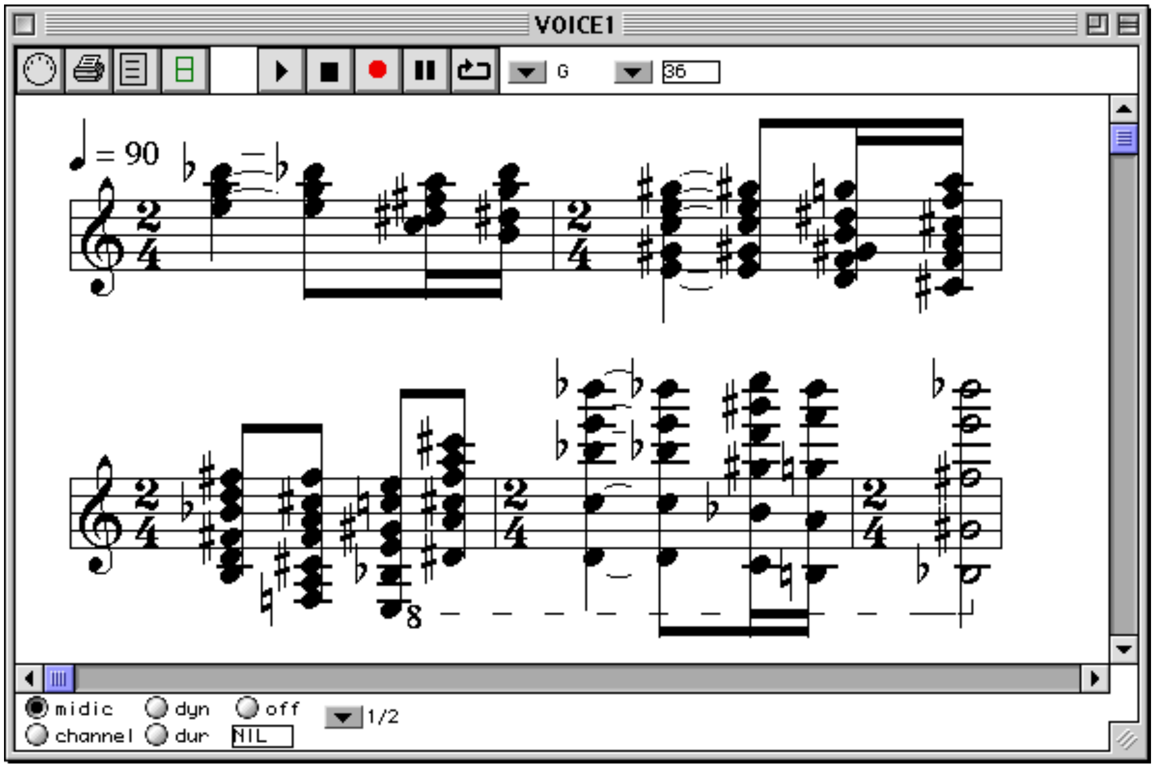
Objects, including the patches themselves, may be "immersed" in a *maquette*, which is a kind of graphical surface whose horizontal dimension represents time. Thus we create, by superposition and concatenation, the basic articulations between musical objects. A maquette may be immersed in an other maquette and articulated in time with other objects. Thus, by recursion, the successive levels of the formal organization will be embedded. A maquette may be immersed in a patch, to become in its turn an atom of computation : the musical structure it represents becomes a source of value, a simple parameter for a higher level computation. Once again it is up to the creator to gauge the desired complexity, but the system itself does not impose any limit.

It is clear that maquettes, patches, are indeed notations. In addition they may be recursively embedded one into another. The contribution of computer science, at this stage, is not only quantitative any more but also qualitative. Notation enters into a "mise en abyme".



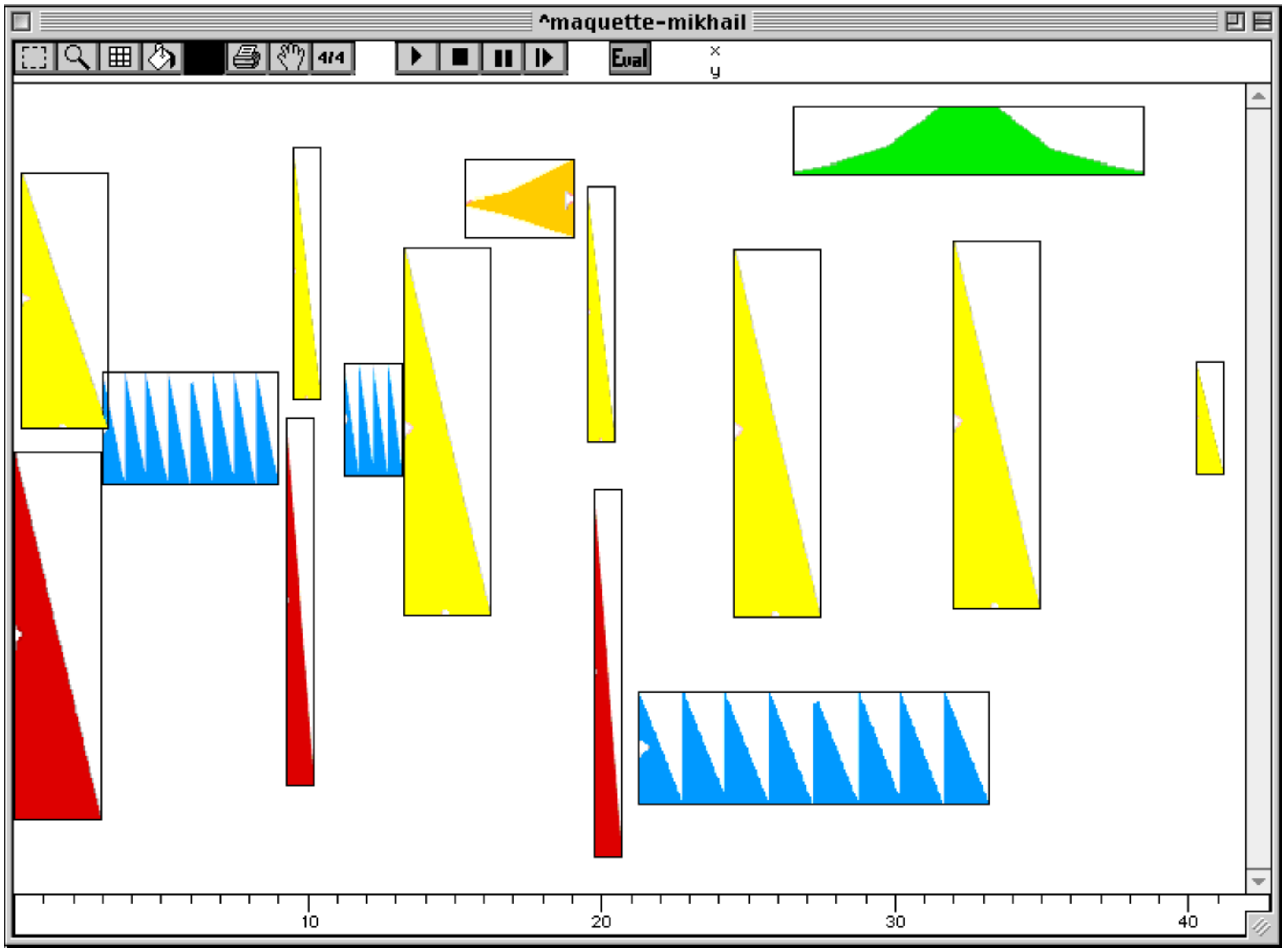
Picture 1

In picture 1 we show an example of an OpenMusic patch (a visual program) in which one harmonizes the beginning of Syrinx by Debussy. This fragment is illustrated in the top-right part of the window in Midi notation (it is actually a Midi file recovered on the Internet and dropped on the surface of the patch). This small melodic fragment will be imposed to the soprano voice. The curve at the top, drawn by hand in a function editor, is a melodic profile that we want to impose to the bass voice. Thus the bass will always go downward. The formula (0_7 (4 3) 8_15 (4 6 10)) is an interval constraint on the chords. The first eight chords will contain only successive major and minor thirds and the last eight chords will contain major thirds, augmented fourths and tenth. Several other constraints relating to the profile of ambitus and density of the chords were masked for the sake of simplicity. The module csolver, a constraint propagation engine developed by C. Rueda [RuVa97] [RLBA98], combines the constraints and computes all the harmonic sequences obeying the rules. One can see in picture 2 the beginning of a possible solution.



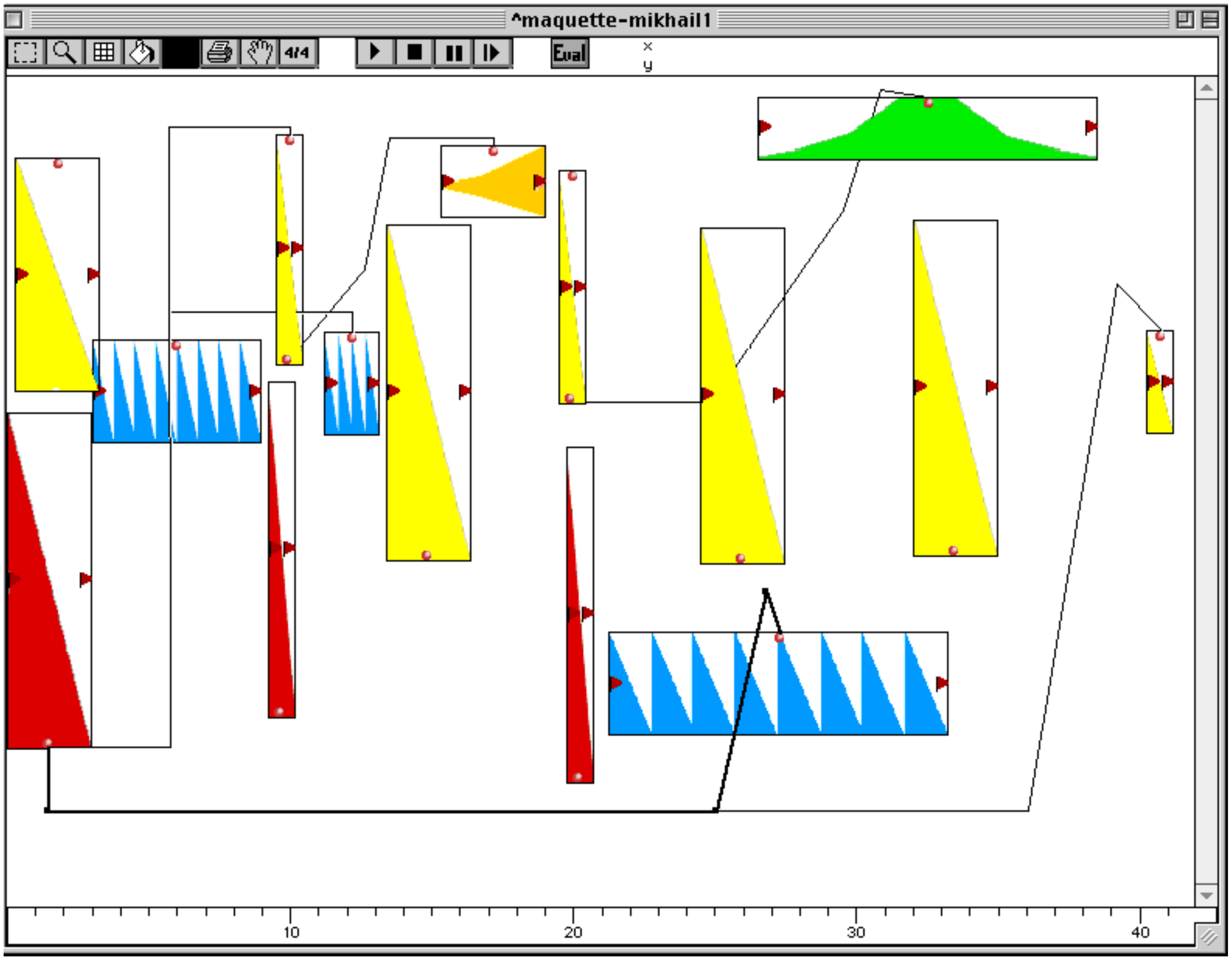
Picture 2.

In maquettes, the structures of hierarchical embedding and temporal scheduling [Pratt92], in other words, the form, have an explicit visual representation. In the draft work for piano presented in picture 3 and realized by composer Mikhail Malt, a certain number of temporal blocks were dropped on the surface of the maquette.



Picture 3

Their position corresponds to dates in absolute time. Their horizontal extension corresponds to their duration. Their vertical extension represents the intensity. Drawings were superimposed on these blocks with an aim of elementary musical semiotics. Thus, the triangles correspond to struck piano chords whose resonance decrease quickly. The saw-tooth blue drawings are ostinato of repeated chords. The horizontal triangle and the bell shape represent fast trills whose intensity grow or decrease continuously while following the drawing form.

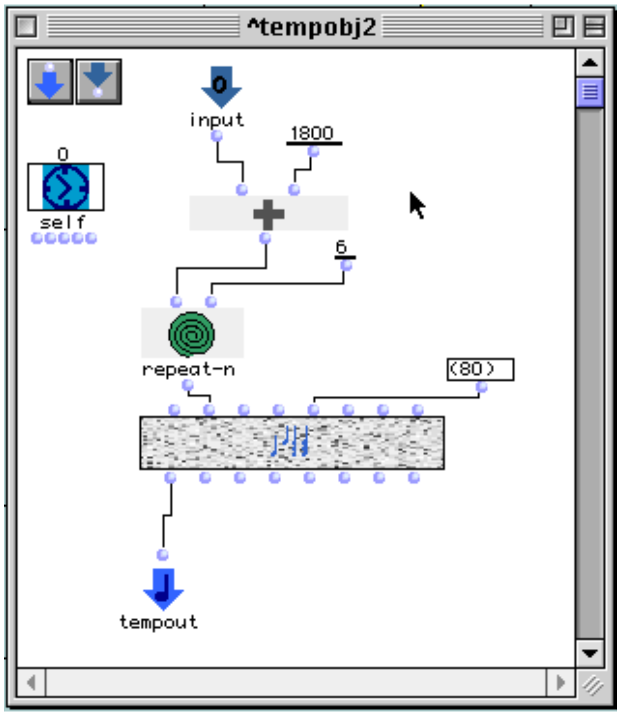


Picture 4

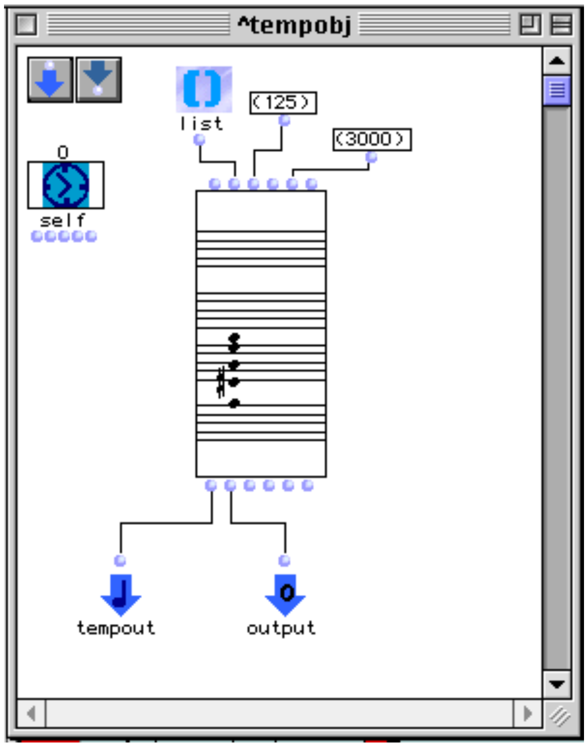
In picture 4, connections between temporal blocks were revealed. They are carrying another level of musical semantics. One can see here that the blocks are deduced one from another by functional computations. For example, the third blue (starting from the left) ostinato is derived from the first red chord. If one " opens " this ostinato (picture 5), one reaches a third level of the musical organization, that could be qualified of syntactic : it is indeed in this place that musical basic materials will become syntagmatic constructions by horizontal deployment. Without being versed in the mysteries of visual programming, one can easily understand that the chord which enters this patch by the arrow named "input" is transposed by an eleventh interval (18 semitones, 1800 cents), then repeated 6 times, and sent in a chord sequence construction module.

Let us return now to the red chord in bottom-left, which provides itself as a parameter to the blue ostinato described above, and let us open it (picture 6). Here algorithmics are reduced to the simpler expression since only the literal definition of a chord is found there, entered by hand by the composer. We reach here the fourth and last level of organization, that of rough musical material.

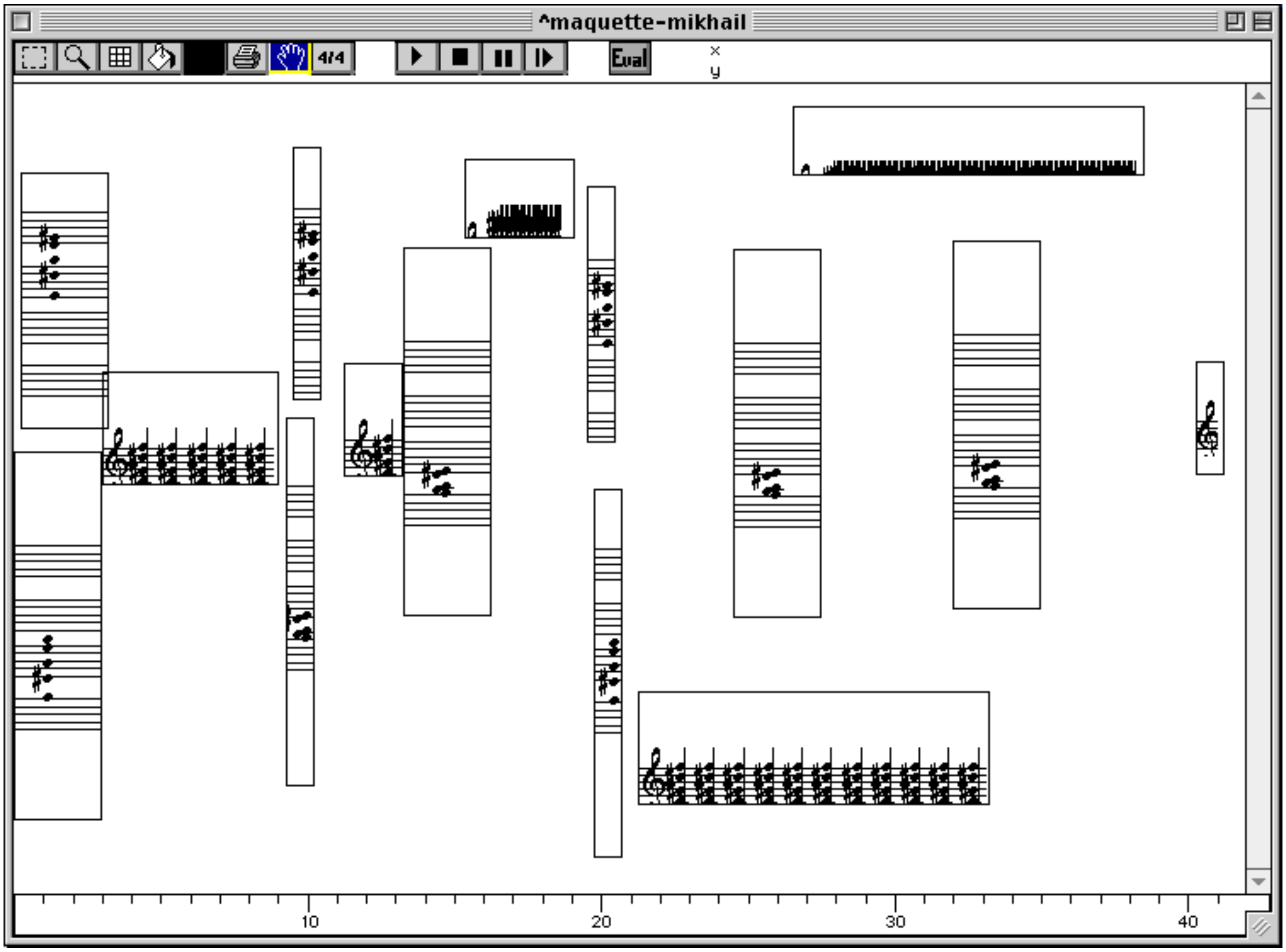
To finish, let us change the nature of the display using a command that switches to musical notation (picture 7).



Picture 5



Picture 6



Picture 7

In this completely new tool that we call a maquette we have access in a fluid way to four interdependent levels of musical organization:

- The static level of the overall musical form, with the possibility to indicate arbitrary visual semiotics markers.
- The dynamic level of the overall musical form, namely the functional relations which bind the blocks. This level is carrying semantic indications.
- The syntactic level, i.e. the computational methods according to which the musical discourse is built within each block
- The level of basic musical materials.

These four dimensions of the musical organization are as many interrelated logical levels of structuring. The major innovation here is to materialize this interrelation in order to provide the composer with an interactive control over it. This control leads to new possibilities of combinative experiments:

- Recombination on the level of the form: the blocks are rearranged, dilated or compressed in time. The other logical dimensions remain unchanged
- Modification of the functional relations (the connections): the trajectories of musical data inside the maquette are then changed without modifying the overall form.
- Modifications of the algorithms contained inside the blocks. It is the syntactical form of the discourse that changes, by preserving the formal pace and the high level temporal organization.
- Modifications in the musical material (change of the pitches in the red chord in our example). The form is preserved in its static and dynamic dimensions, the syntactic structure is preserved, but the sound substance is changed (the harmonic color in our example).

These four experimental methods may obviously be combined.

CAC is very much used in Europe today and starts to diffuse back to the Anglo-American sphere, closing a cycle that began there 40 years ago. Composers as different as Tristan Murail and Gérard Grisey, who make use of it specially for the production of harmony and rhythm, Brian Ferneyhough, which discovered there a challenge corresponding to his taste for extreme complexity, Claudy Malherbe, who considers a tool accorded with his ideal of construction of new models for each of his works, Magnus Lindberg, which aims to achieve vertical control from the great form down to the syntactic elements.

Young Creators like Joshua Fineberg, Mikhail Malt, Fausto Romitelli, and others set up new directions that would not have been even thinkable without CAC. OpenMusic was born from confrontation with all these creators, who have each a different vision of the role of technology. Our experiment of collaboration between scientists and creators taught us that CAC had deeply overturned practices and methods in musical composition. Well beyond a quantitative profit, it operates a qualitative calling into question of the methods used in creation.

CAC puts into a relative perspective some techniques, which were thought complex and were only complicated. When a computer quickly computes all the combinatorial alternatives resulting from a given formalism, for example the serial formalism, the musicological value does not come anymore simply from this formal principle, but from its real adequacy to perception. Correlatively, the computer makes it possible to explore experimental fields of a real complexity and which were inaccessible before, such as the timbre field in its spectral representation.

CAC provides the experimental environment that makes it possible to subject a very great number of musical instances resulting from a formalism to the test of musical quality. More, it authorizes the experimentation on formalisms themselves, which can be tested and in return modified or given up if they do not fulfill their promises.

To the romantic concepts of uniqueness of the musical work and of its creator, it tends to substitute the idea of a model for a work and of co-operation between creators and scientists.

This is where the more revolutionary and indeed the more polemic aspect of CAC resides.

To end with, CAC completely renews the idea of notation, which becomes dynamic, including the genetic mechanisms of the work. Scores become potential scores, and constitute in a way their own analytical descriptions, which is the seed for an upcoming outbreak in contemporary musicology.

Gérard Assayag

October 18, 1998.

References

[AADR98] Carlos Agon, Gérard Assayag, Olivier Delerue, Camilo Rueda. *Objects, Time and Constraints in OpenMusic*. Proceedings of the ICMC 98, Ann Arbor, Michigan, 1998.

[AsCa85] G. Assayag, M. Castellengo, C. Malherbe. *Functional integration of complex instrumental sounds in music writing*. proc. ICMC 85. 1985.

[Bal92] Balaban et al, *Understanding Music with AI.*, Balaban, Ebcioglu et Laske Ed., MIT Press, Cambridge MA, 1992.

[Barb68] Pierre Barbaud. La musique discipline scientifique. Dunod, Paris, 1968.

[Cast98] Castagna Giuseppe. *Foundations of Object-oriented programming*. Workshop at ETAPS '98, Lisboa, 1998.

[Chem92] M. Chemillier, *Structure et méthode algébriques en informatique musicale*. Thèse de doctorat. LITP 90-4, Université Paris VI, 1990.

[Chem94] Marc Chemillier. *Analysis and Computer Reconstruction of a*

Musical Fragment of György Ligeti's Melodien. ISMM Bucarest (Romania), 1994.

[Cour93] Francis Courtot. *Entre le décomposé et l'incomposable*. Les cahiers de l'Ircam 1(3) : La composition assistée par ordinateur 1(3). Paris juin 1993.

[Hill69] Lejaren Hiller. *Music composed with computers*. In Music by computers, Heinz von Foerster col. J. Wiley & sons Ed. New York, 1969.

[Koen71] G. M. Kœnig. *Emploi des programmes d'ordinateur dans la creation musicale*. Dans Revue Musicale No. 268-269, Paris, 1971.

[Laur96] Laurson M. *PATCHWORK: A Visual Programming Language and some Musical Applications*. PhD Thesis, Sibelius Academy, Helsinki, 1996.

[MeRio88] M. Mesnage, A. Riotte, *Un modèle informatique d'une pièce de Stravinsky*, in Analyse Musicale, 10. Paris, 1988, pp. 51-67.

[MeRio89] M. Mesnage, A. Riotte, *Les variations pour piano opus 27: approche cellulaire barraquéenne et analyse assistée par ordinateur*, in Analyse Musicale, 14. Paris, 1989, pp 51-66.

[Pach94] Pachet, François. *The MusES system: an environment for experimenting with knowledge representation techniques in tonal harmony*. First Brazilian Symposium on Computer Music, SBC&M '94, August 3-4, Caxambu, Minas Gerais, Brazil, pp. 195-201, 1994.

[Pope91] Stephen Pope. *Introduction to MODE : The Musical Object Development Environment*. S. T. Pope ed. MIT press. Boston, 1991.

[Pratt92] *V. Pratt, The Duality of Time and Information*. Proceedings of CONCUR'92. Springer-Verlag, New York, 1992.

[RoCo85] Xavier Rodet et Pierre Cointe. *FORMES Composition et ordonnancement de processus*. Rapports de recherche No 36, Ircam, Paris, 1985.

[RLBA98] Camilo Rueda, Mikael Laurson, Georges Bloch, Gérard Assayag. *Integrating Constraint Programming in Visual Musical Composition Languages,* Workshop W4, Constraint techniques for artistic applications, ECAI '98, Brighton, August 1998.

[RuVa97] Camilo Rueda, Frank Valencia. *Improving forward checking with delayed evaluation*. In proceedings of CLEI'97, Santiago, Chile, 1997.

[Taub91] H. Taube. *Common music: A music composition language in Common Lisp and CLOS*. Computer Music Journal, 15(2), Summer 1991. MIT Press.

[Xena81] I. Xenakis. *Musiques formelles*. Stock Musique, Paris, 1981.