TEMPORAL CONTROL OVER SOUND SYNTHESIS PROCESSES

Jean Bresson *Carlos Agon*

Ircam – Music Representations Team
Paris, France

ABSTRACT

This article addresses the problem of the representation of time in computer-assisted sound composition. We try to point out the specific temporal characteristics of sound synthesis processes, in order to propose solutions for a compositional approach using symbolic models and representations.

1. INTRODUCTION

The use of computers for music composition allows the formalisation and experimentations on compositional processes using symbolic computing models. Programs thus become a part of the music representation. Visual programming interfaces such as OpenMusic [1] were developed in order to make the creation of these programs easier, and more accessible for musician users.

Concurrently, digital sound synthesis is another important revolution introduced in music by the use of computers; but the lack of abstraction and symbolical representations in sound synthesis applications remains an obstacle for the creation of rich compositional models.

In this article, we will concentrate on the temporal aspect of sound synthesis programming and address this problem from the Computer-Assisted Composition (CAC) standpoint. Indeed, we think that Computer-Assisted Composition might propose some solutions in order to conciliate sound synthesis and music writing.

We present works carried out in OpenMusic, especially with the *maquette* interface, which model has been reformulated. This model integrates visual programming and temporal control in a coherent and generic way, which, we hope, allows to develop sound synthesis processes in a compositional context.

Section 2 will outline some reflections about time in relation to sound synthesis, and section 3 will cite some related works. In sections 4 and 5, we present the visual programming framework developed in OpenMusic with the maquette features and concepts. Finally, section 6 will give some examples of this system in sound synthesis applications.

2. TIME AND SOUND SYNTHESIS

2.1. Temporal Scales

The control of sound synthesis for music composition involves the temporal parameter at different levels. The first is the organization of sound objects in a temporal axis. This aspect of the temporal control is not specific to sound synthesis and can be brought together with the temporal formalisms used for music composition in general (see [6], [18], [14]). These different formalisms of time representation include absolute models, relative/hierarchical models, functional models, logical models, constraints-based models, etc.

On the other hand, since the earlier works on sound synthesis, time has been identified as an important parameter of timbre. In 1957, while developing a modern musical thought which now integrates timbre as a full-fledged and promising compositional field, K. Stockhausen yet defined timbre as "the result of a temporal structure" [26]. Later, notably with the works of M. Mathews and J.-C. Risset on synthetic timbres [22] [19], temporal phenomena such as attack transients, or relations between spectral parameters' temporal evolutions were pointed out as fundamental descriptors in the perception of the musical sound timbres. Since then, time was intensively used inside sound synthesis programs, and the control of the independent or related evolutions of synthesis parameters became a compositional challenge [23].

This structuring aspect of time in sounds carried our attention on some specific temporal properties. Synthesis processes deal with time at the microscopic level of the sound samples. Indeed, a synthesis program generates the values of a digital waveform that represents a continuous acoustic vibration. The sampling rate of this acoustic phenomenon must be as high as possible in order to reach high audio quality, i.e. to simulate the continuity of the acoustic waveforms. Supposing that continuity can be assumed from the point where perception does not distinguish discrete elements anymore, the temporal problem for the control of sound synthesis can thus be expressed in terms of continuity vs. discrete paradigms (see also [13]). If we want to keep cautious with these epistemological concepts, we can correlate the discrete paradigm with the domain where the objects (*events*) manipulated for creating compound structures keep having an appreciable significance, while in the domain of continuity, the objects of composition will not be delimited and independently discernable anymore.

2.2. Synthesizers

The different time paradigms can be identified in software synthesizers. "Event-based" synthesizers (e.g. Csound [8]) are "naturally" polyphonic, and respond to punctual events and commands. They provide a logical organization and communication between several synthesis modules.

On the other hand, "continuous" synthesizers (e.g. CHANT [25]) are based on "phrases" and produce a sound result by computing the system state at each moment. In this case, elaborated transitions (e.g. interpolations) and continuous manipulations are easier to control.

Evidently this distinction is actually not so restrictive: continuous phenomena can be controlled in the first case, and the second one can also be manipulated in an "event-based" fashion. We will only retain that some software architectures can be more or less adapted and oriented towards one or another particular temporal paradigm [16].

2.3. The Compositional Problem

The nested temporal scales involved in an electronic composition will generally concern simultaneously both temporal domains described above. Structural relations still exist between them, but their inherent characteristics do not necessarily imply the same internal rules.

Events represent the primitive of a discrete conception of sounds. They are elements that can be manipulated at a symbolical level. Nonetheless, continuous phenomena allow to generate complex sonic morphologies, and thus also need to be considered in the control structures. These phenomena can be internal to discrete events (e.g. the expression of an internal movement or transformation), or external (e.g. transitions from an event to another).

The notion of event thus becomes blurred. J. F. Allen in [3] defines an event as "*the way by which agents classify certain patterns of change*". Depending on the desired compositional abstraction, an event can be the beginning of a sound in a large-scale musical structure; it can be a note (an acoustic element with a perceptible pitch and duration), a variation of energy in a spectral region, the beginning of a continuous transition.

Therefore, the compositional control of sound synthesis cannot be restricted in the positioning in time of synthesis events. From the same compositional point of view, however, it must stay at a symbolic level. The problem is thus to establish relations between the linear continuity of time with a symbolic representation that allows its structuration and modelling. In other words, we would need symbolic data able to represent and control continuous temporal objects.

3. RELATED WORKS

Various CAC systems are oriented towards sound synthesis. We propose here a brief overview of a (non-exhaustive) selection of some of them, which present original conceptions of time. More environments exist, that propose different interesting solutions (e.g. [12], [15]). The relevant temporal aspects of the following examples are outlined in order to provide comparison criteria for positioning our works in OpenMusic.

Formes [24] was part of the earliest generations of CAC environments developed at Ircam. Originally designed to be a control interface for the CHANT synthesizer, this textual object-oriented language based on Lisp was one of the first systems integrating composition and sound synthesis. *Formes* proposed a hierarchical processes scheduling system, managed by a general "monitor", and which provided a high-level and continuous control of sound synthesis in time, allowing to program complex temporal situations.

Boxes [7] is a visual environment which proposes a hierarchical temporal organization of musical objects. These object (graphical boxes) contain spectral representations of sounds and are connected to an additive synthesizer. In this framework, temporal constraints can be set between the boxes, which provides a logical time organisation model. However, this temporal control does not get in the internal synthesis processes but rather organize the synthesized objects in time.

In the real-time system *PureData* [20], the *data structure* allows to schedule real time processes in a temporal axis, thanks to a graphical interface and process delaying operations [20]. The control is linear and under restraint of the real-time constraints [28], however the visual representation of time in such a system allows to envisage large scale (possibly hierarchical) organisations of real-time synthesis processes, and to step up to a real-time based score.

Iannix [11] is another real-time system, in which time is approached in an original way. The visual interface is a bi-dimensional temporal space on which the user creates temporal trajectories, which can be lines, curves, circles. Cursors then follow these trajectories with variable speeds and directions, and activate synthesis events when they meet some *trigger* objects. This environment thus allows to organize parallel temporalities in a same space. The OpenGL 3D interface allows to zoom in and out in this space and to simulate a continuous hierarchy for variable-scales composition processes.

4. TEMPORAL ASPECTS IN OPENMUSIC

4.1. About Visual Programs

OpenMusic is a Computer-Assisted Composition environment, which merges visual programs and musical notation [4]. In OpenMusic, a *patch* is a representation of a program. The user writes this program using graphical boxes and connections, and creating a functional expression. Boxes represent functional calls, and connections represent the functional composition of the program. Figure 1 shows an example of patches in OpenMusic. Some special boxes are used in patches to represent musical data structures such as chords sequences or sounds.

The patch maintains a correspondence with a Lisp function: once a patch has been defined, it can thus be used as a function in another patch. This functional call is then also represented by a box, which evaluation consists in applying the function to the values connected to its inputs. Figure 1 illustrates this with a sound synthesis patch and its application in another patch. In this example, the patch requires a sequence of chords and an amplitude envelope as arguments, and returns a synthesized sound (see [10] for sound synthesis possibilities in OpenMusic).

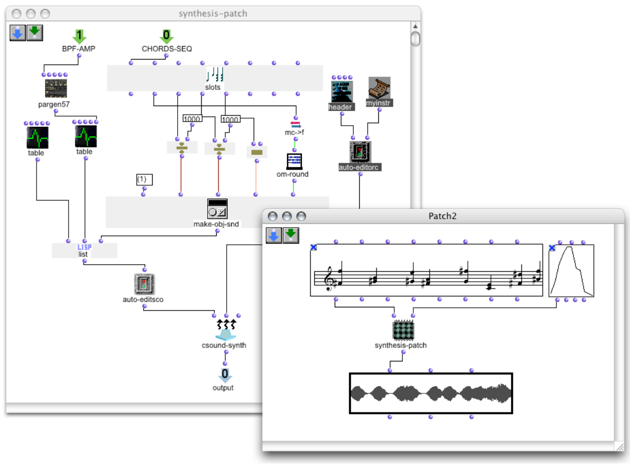


Figure 1. A sound synthesis patch (synthesis-patch) and its application in another patch (patch2).

Multiple occurrences of a patch, represented by boxes in other patches, are thus all references to the same original patch, which function definition remains unaffected. This property presents advantages in terms of function prototyping, and use of abstractions (similarly to the definition of functions in a traditional programming language which allows modifying a function once and affecting all the places where it is used). Abstraction is thus used to reach a high level of control in musical processes.

In contrast with real time systems, the time representation in an OpenMusic patch is that of the calculus: objects are computed following the structure of the functional graph. Musical time is thus represented and controlled as a numerical parameter in the program. Following the distinction made by I. Xenakis in [31], these objects resemble "out-of-time" structures, i.e. structures that have their own internal composition rules. They will be unfolded "in-time" in a later phase, when integrated in a temporal context (e.g. when they are played).

4.2. Temporal Context: Maquettes

A *maquette* is a programming interface with 2 dimensions: the horizontal axis represents time, and the vertical axis is a freely interpreted parameter (called *y*). This bi-dimensional space imposes a temporal context to the objects inside: it represents a way for unfolding musical objects in time.

Foremost, the maquette is a program that allows to define the temporal organization of other programs. These programs are represented in the maquette by rectangles called *temporal-boxes*, whose horizontal position is correlated with an offset, and the extent with a duration (see Figure 2). These temporal-boxes can be connected together, recreating in this way the functional connections between the corresponding programs. A temporal-box can thus *refer* to a patch, or to another maquette (the latter allows to constitute hierarchical temporal structures). It can also refer to a simple object, which is equivalent to the particular case of a patch representing a constant function.

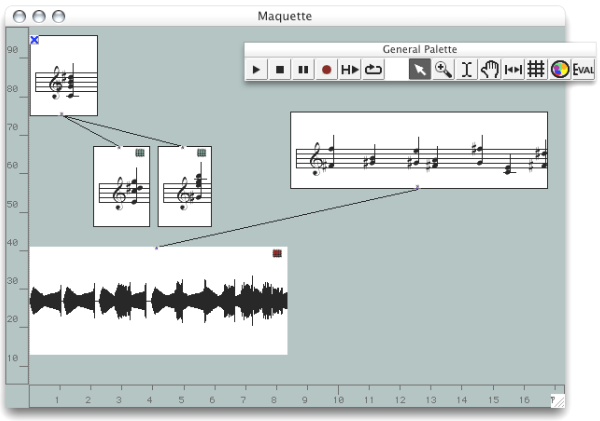


Figure 2. A maquette window with temporal boxes and functional relations.

In Figure 2, the bottom-left temporal-box refers to a patch similar to the one of Figure 1, which generates a sound. Actually, double-clicking this temporal-box opens the patch editor of Figure 4. The specific properties of such a patch will be discussed in section 4.3.

In addition, the maquette can be contained in a patch, and constructed algorithmically. In this case, the musical objects and their temporal offsets are given as input parameters of the maquette box (see Figure 3).

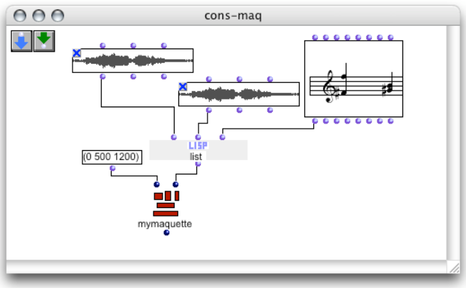


Figure 3. A maquette constructed in a patch.

Interested readers can find more descriptions about the maquette implementation in [1], [5]. In [10], we were considering using this interface as a support for developing sound synthesis models. We propose here a renewed version of the maquette, improved for hosting such processes integrating time and sound synthesis (see section 5 and 6).

4.3. Evaluation vs. Performance

The maquette integrates "out-of-time" patches in a temporal context in order to build a musical development with them. However this is not the matter of scheduling processes, but rather to incorporate the time parameter in the process. In the maquette model, *evaluation* and *performance* are two separated concepts.

During the computation of the maquette, we call *evaluation* the phase where functional calls are evoked, and the values of the temporal-boxes (the musical results of each individual patch or maquette) are calculated. This phase is the preliminary of another one, related to the *performance*, where these results are collected in order to build a global musical object.

This separation allows create temporal situations without being dependant on the linear time flow (in Figure 2, the sound box depends on data coming from an object that occurs later in the performance time.) It also reflects the distinction between composition and performance times.

In the evaluation phase, the "terminal" temporal-boxes in the functional graph of the maquette are evaluated, which evaluates recursively all the temporal-boxes following the functional order defined by this graph.

In the performance phase, the maquette computes its own musical result by collecting and mixing all the objects resulting from the temporal-boxes, now following the linear temporal order defined by the positions of the temporal-boxes.

A container object is created, called *maquette-object*. It will eventually be sent to MIDI or audio renderers if the maquette is played, or represent the maquette itself in the performance phase when it is included in another maquette.

4.4. Programs in Time

Once it is introduced in a maquette, a patch is thus associated to a temporal-box and has two new characteristics due to its relation to the temporal context (see Figure 4):

- It can access the coordinates and properties (position, offset, size, colour, etc.) of the temporal-box that invokes the process thanks to a "temporal input". In Figure 4, the vertical position (y) of the temporal-box in the maquette is used as a multiplying factor for the amplitude envelope.

- It must provide a special "temporal output", which ports the external representation of the program: the object which is collected in the performance phase defined above. In our example (Figure 4), the *TempOut* box ports a synthesized sound (represented on the bottom-left box in the maquette of Figure 2).

This object computation can thus depend on the self properties of the temporal-box, and on other external data: the temporal program still can have other inputs and outputs allowing it to receive and transfer data to the other objects.

These relations of the musical objects with their external context might illustrate a sentence from P. Boulez, formulated in [9] while talking about the two compositional stages that are composing "*within objects, in order to build them, or from the outside, in order to organize them*", and which specified that "*external criteria can act on internal criteria and modify the objects in order to link them in a coherent development and place them in a formal context*".

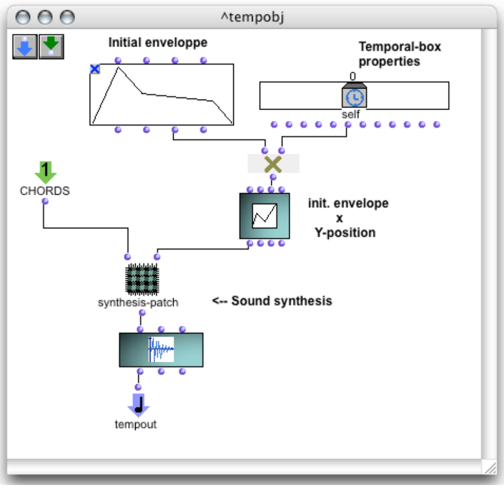


Figure 4. A "temporal patch". This patch corresponds to the bottom-left box of the maquette of Figure 2 and uses the patch of Figure 1.

These temporal characteristics of the patches are optional: a patch that has no temporal output will nonetheless be able to act as a programming element in the evaluation phase of the maquette, without participating to the performance phase.

Including patches in the maquettes enables the use of abstractions and the program reusability. The temporal-boxes can thus constitute "template" objects, that can be duplicated and compute different results depending on their positions and self properties, but still controlled by a unique template program.

5. SYNTHESIS MAQUETTE

In order to obtain an improved temporal control, and to tackle the issues discussed in section 2, we complete the model of the maquette with two specific characteristics, (corresponding to the two phases – evaluation and performance – presented in section 4.3.)

5.1. The Maquette as a Program

Even though we pretend that the maquette is the representation of a program, its temporal specificity implies a particular behaviour. Indeed, the maquette program is based on the temporal properties of the temporal-boxes. However, these properties are susceptible to change: they can be modified by the execution of the program itself (see section 4.4.) For example, a temporal box can have its position (offset) changed depending on a variable parameter. Following the functional analogy, the evaluation of the maquette could therefore constitute a redefinition of the underlying program, which can be problematic when using functional abstractions. The correspondence between the maquette and an abstract Lisp function thus cannot be maintained.

Nonetheless, we introduced the possibility to assign inputs and outputs to the maquettes, so that they can nevertheless be regarded as programs and be abstracted at a functional level.

The external evaluation of a maquette then consists in computing its inputs, the maquette-object, and evaluating the different outputs. Externally, the maquette is then like any other patch whose special temporal output would create the maquette-object.

The temporal hierarchy created by embedding maquettes one in another can then be generalized at this functional level.

In Figure 5, the maquette has 3 inputs and 1 output: the input values are passed to the internal temporal-boxes (one of them refers to another maquette), and the output of one of these temporal-boxes is returned by the output of the maquette.

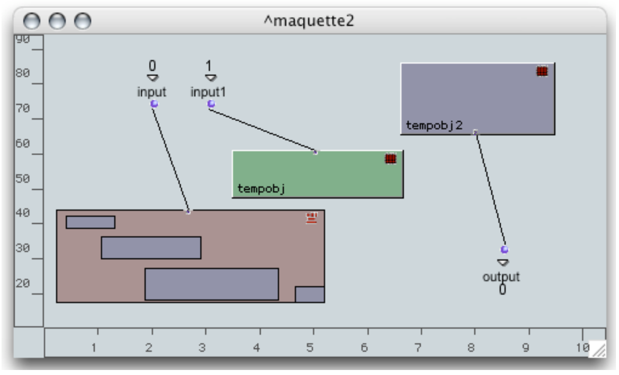


Figure 5. Temporal / functional hierarchies in a maquette.

In a patch, a maquette can then also be regarded as the functional call of the program defined by this maquette (see Figure 6), in contrast with the previous constructive representation (Figure 3). In Figure 6, the inputs of the maquette box correspond to the inputs in the maquette editor of Figure 5. The 2 outputs of the box represent the maquette-object and the output of the maquette editor.

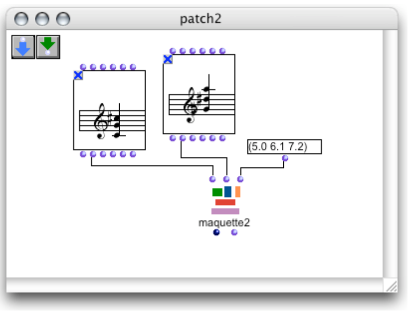


Figure 6. The maquette of Figure 5 in a patch.

However, this functional representation of the maquette still presents a slight limitation. The computation result depends on, and can modify the properties of the internal objects. For that reason, recursion is not allowed; some consistency checking must therefore ensure that the functional abstraction of a maquette is not called inside the self maquette or one of its parents.

5.2. Maquette-Object Computation

Until now, the creation of the maquette-object in what we called the performance phase of the maquette computation was made by mixing the temporal outputs of the patches referred by the temporal-boxes. Such a maquette does not have a real computational control over its own result, as the patches do thanks to their "temporal output" (see section 4.4.)

The new model we propose considers the maquette-object computation as an accessible part of the program, that consists in processing the temporal-boxes and building a musical result depending on their values, positions, properties and relationships.

In sound synthesis applications, the patches and objects in the maquette do not necessarily compute musical objects (sounds) anymore, but can also generate control parameters. The evaluation of the maquette would then consist in a synthesis program, that manipulates all these individual objects, and use them for controlling a high-level sound synthesis process.

In this way, the composer has a hand on the process that creates the musical result of the maquette (the former maquette-object) using the temporal-boxes contained in it. The problem of the transition from discrete control elements (represented by the temporal-boxes) to continuous phenomena can be handled in this framework.

The *synthesis-function* is a new attribute of the maquette. Such function can be a patch or a lisp function, providing this patch or function has one input or argument (representing the list of temporal-boxes) and one output (the computed resultant object). It is supposed to constitute a link between these temporal-boxes and the musical result of the maquette.

This synthesis-function can be assigned to a maquette algorithmically in a patch, or by dragging a patch or function box in a special zone of the maquette window (the bottom-left corner of the window). It is supposed to control the computation of the musical result of the maquette. When no synthesis function is assigned, however, the default maquette-object is computed.

Figure 7 shows a schematic example of the use of a maquette and its synthesis-function. Next section will illustrate with more details some possible cases of use.

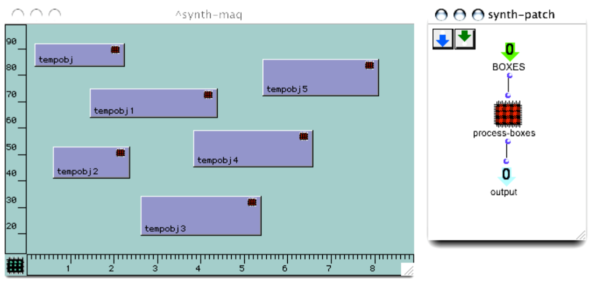


Figure 7. A maquette and its synthesis function. An icon in the bottom-left corner represents the synthesis function. Double-clicking this icon opens the synthesis patch.

The possibility to set a "classical" patch as synthesis-function also allows the use of templates of synthesis processes that can be applied in different contexts.

6. APPLICATIONS FOR SOUND SYNTHESIS

6.1. Templates for Synthesis Events

This first application is an example of control of a Csound program using a maquette. It was inspired by previous works by K. Haddad using the OM2CSound library [17].

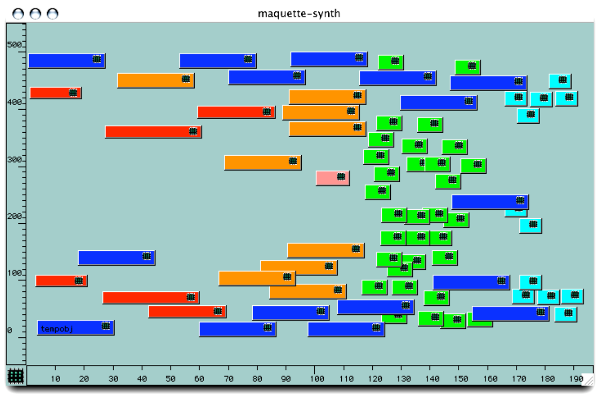


Figure 8. A maquette used for Csound synthesis. In this maquette each temporal box (event) represents a part of the score.

The maquette of Figure 8 is used as a graphical interface for designing the Csound score. Various template patches are used. Each one of them, identified by a particular colour, produces a different kind of data, related to a Csound score event.

The data produced by the temporal-boxes thus does not correspond to "perceptible" objects: they need to be integrated all together in order to create a musical result (a sound).

The *synthesis-function* of this maquette, represented in Figure 9, is a function that converts these values of the temporal-boxes into a Csound score, associates this score to a Csound orchestra, and then runs Csound for computing a sound.

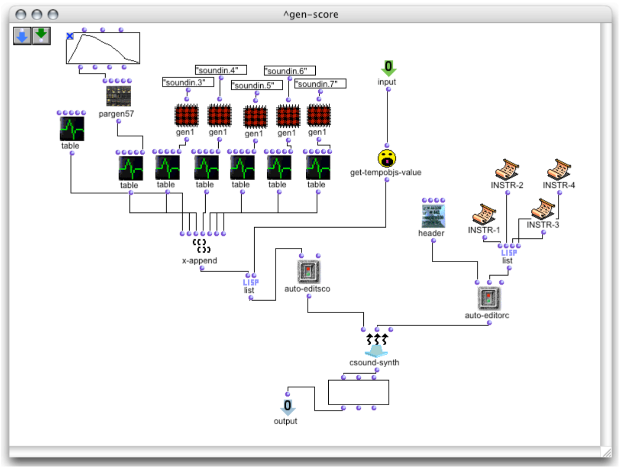


Figure 9. A Csound synthesis patch associated with the maquette of Figure 8.

The visual and interactive control of the synthesis events is thus associated to a global control of the events processing.

If this maquette is used in a higher-level structure, it will (once evaluated) be considered (externally) as the computed sound. Figure 10 shows an example of this maquette used in another maquette next to other sounds. It can also be used in a patch as a sound synthesis function and interface (see Figure 11).

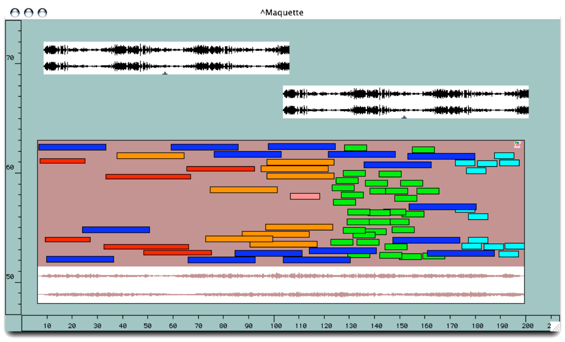


Figure 10. The maquette of Figure 8 in a higher-level maquette. The external representation is the sound.

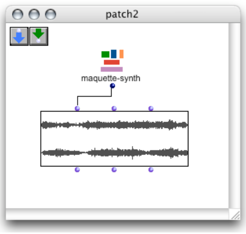


Figure 11. The same maquette (Figure 8) in a patch, as a synthesis function.

6.2. Global Synthesis Performance

In [10], we attempted a first adaptation of the piece *Traiettoria… deviatta* from M. Stroppa [27] into a maquette (see Figure 12.)

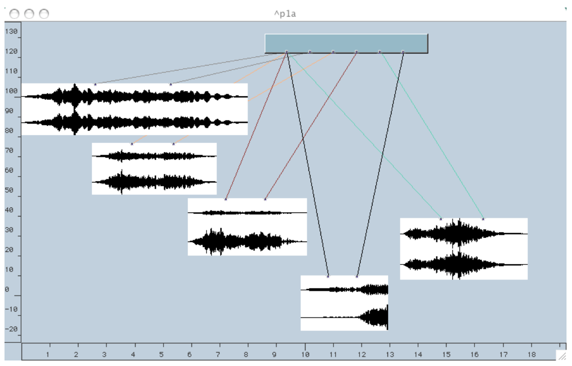


Figure 12. Part of the reconstitution of *Traiettoria... deviatta* in a maquette (extracted from [10]).

In this maquette, the hierarchical feature was used to reach a visual control over the temporal structure of the piece. Each terminal box was a synthesis patch using the OMChroma system [2], and producing a sound. At the time of evaluating the maquette, a multiplicity of synthesis programs were performed sequentially, and the resulting sound was mixed down.

With the synthesis-function, the sound synthesis process is delegated to a higher level: instead of processing separated programs in the maquette, each temporal box returns a matrix of parameters. These synthesis parameters are then collected by the synthesis-function, which triggers the Chroma synthesis program [29].

When using large numbers of such synthesis events, this processing method demonstrates much more efficiency.

Figure 13 shows the new maquette, which uses the synthesis function. The objects represented and manipulated are the control parameters of the sound synthesis; they are closer to the symbolic data involved at a compositional level. The sound object is computed in the late synthesis level, instead of the former maquette-object, which collected the sounds coming from synthesis patches.

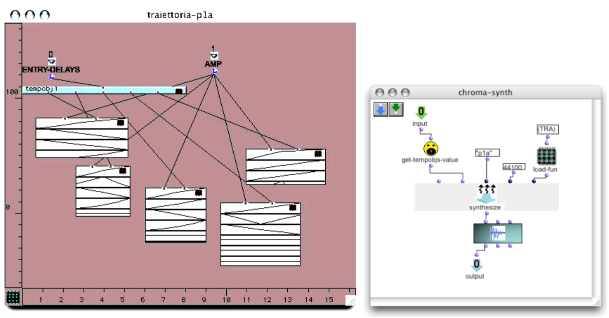


Figure 13. Reconstitution of *Traiettoria... deviatta* in a maquette (2nd version).

This procedure (which consists in computing the synthesis parameters in the evalution phase and running the sound synthesis in the latter performance-related phase) also allows to consider abstract relations between the boxes in time, not expressed as such in the maquette, but built at the time to compute the final sound. This will be illustrated in the following example.

6.3. Creation of Continuous Events

The control of sound synthesis in the maquette allows to systematize the processing of synthesis events in order to create continuous phenomena based on these events. This can be done especially at a global level, i.e. not inside (or between) each individual event. Starting from essential control objects, complex sound descriptions can be computed using continuation or transition processes.

This last example is an application of the maquette for controlling the CHANT synthesizer. This synthesizer is based on continuous phrases described by the states of synthesis modules at different moments.

We will use a CHANT configuration that consists in a single FOF synthesis ("Fonction d'Onde Formatique" – or Formant Wave Function). For setting the synthesizer's parameters, the user must provide the state of a FOF bank (FOB) at different times. This is done by mean of an SDIF file [30] in which each frame contains the FOB state at time *t*. The main parameters to set are the frequency, bandwidth, and amplitude of each FOF. In order to simplify our example, we will consider the frequency parameter only. The maquette of figure 14 contains 3 temporal boxes, each one of them containing a break-point function (BPF) that represents the frequencies of 8 FOF at corresponding times.

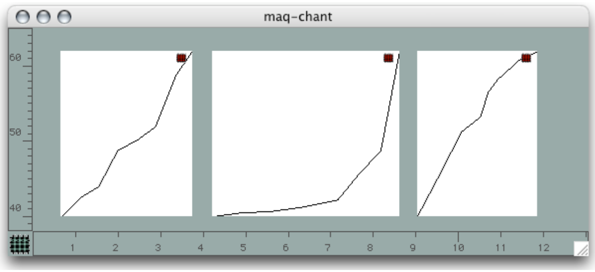
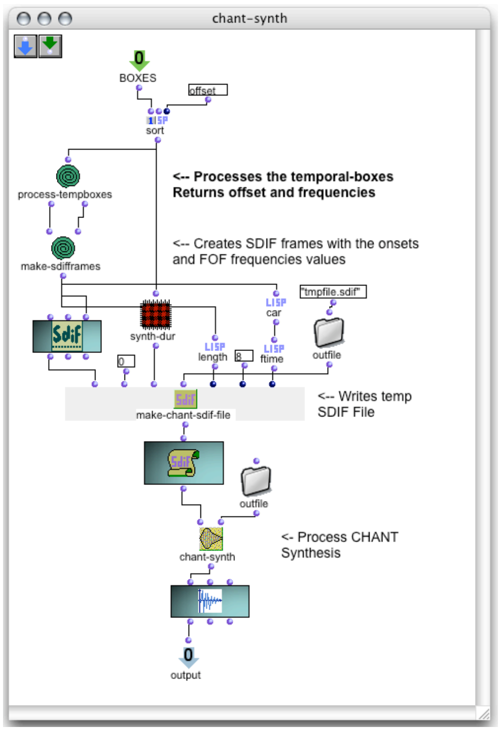


Figure 14. A simple maquette containing 3 control BPF.

In a first version, our synthesis-function will process these values in order to complete the FOF bank data and write its evolution in an SDIF file. The corresponding patch is illustrated in Figure 15.



<-- Processes the temporal-boxes Returns offset and frequencies

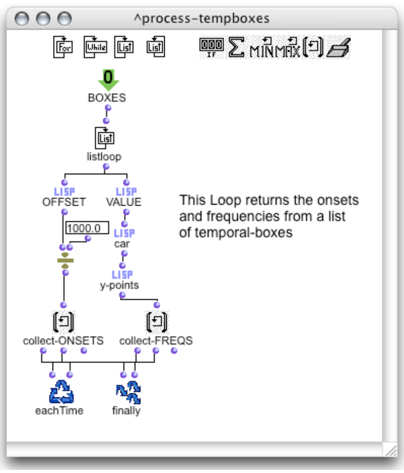
<-- Creates SDIF frames with the onsets and FOF frequencies values

<-- Writes temp SDIF File

<- Process CHANT Synthesis

Figure 15. The CHANT synthesis patch for the maquette of Figure 14.

Figure 16 shows the loop editor for the *process-tempboxes* box of Figure 15.



This Loop returns the onsets and frequencies from a list of temporal-boxes

Figure 16. The "process-tempboxes" loop of Figure 15.

If we use this patch as the synthesis-function of our maquette, 3 FOB frames are created and used to process the CHANT synthesis.

In a second version of the synthesis-function, we replace the *process-tempboxes* loop by another one (represented in Figure 17), that assigns a programmed continuation to the initial FOF elements. Each one of the FOB created will be completed with a period of perturbations (during the extension of the temporal box). Additional SDIF frames are generated, which makes the final FOB evolution more complex (with our 3 boxes and a control rate of 0.1s., some 100 frames are created dynamically.)

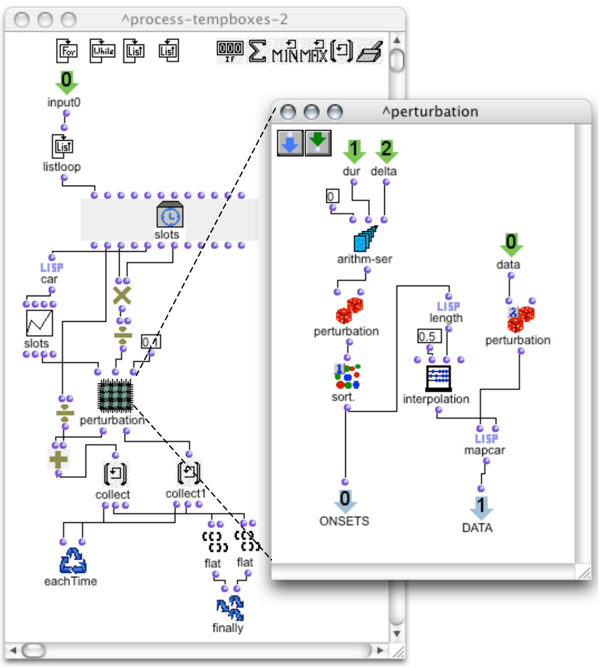


Figure 17. A second version of the "process-tempoxes" loop in the synthesis-function of Figure 15. Additional data are created after each event in order to generate variations.

The Figure 18 shows a comparison of the frequency evolution in the SDIF files generated in the two versions of this example.

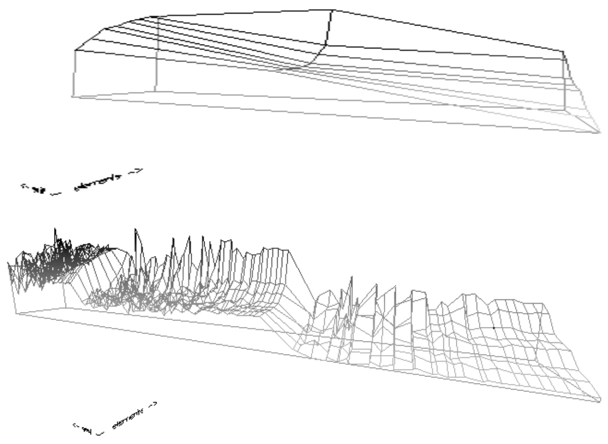


Figure 18. Evolution of the FOF frequencies in the SDIF files generated in the two different examples.

This process thus permitted to construct a complex evolution of the FOF in time (see Figure 18) without having to consider this complexity at the time of setting the initial temporal boxes.

These boxes and their values are the external, visible part of the process, while the synthesis patch is the computing level of this process that integrates the external level and transforms it into a sound.

Both parts have to be defined separately: the maquette constitutes a symbolic interface level chosen by the composer for the control of the entire process.

7. CONCLUSION

While musical editors are insufficient to represent the processing parameters and relations involved by sound synthesis, and programming interfaces cannot integrate a musical and intuitive representation of time, the use of the maquette might be a worthy trade-off for the control of synthesis processes integrating temporal and symbolic aspects.

By mixing a programming environment and a visual control with an arrangement of the musical materials in time, it provides a variable-scaled vision of musical structures. Microscopic time (the time of the sound samples and continuous evolutions), can be handled with macroscopic time (the time of the musical forms), and sometimes be related to it: the internal composition rules of out-of-time objects interact with the high-level temporal organization.

The maquette thus allows to unfold sound synthesis processes in time, but also to integrate time in the synthesis processes. We have seen in the different examples that the synthesis process in the maquette allowed the representation of continuous objects using the discrete representation of both events (in the maquette) and program (in the synthesis-function).

We believe that this system could help composers to get the advantage of the means provided by the CAC environment in order to get to new applications of sound writing and creation.

8. REFERENCES

[1] Agon, C. *OpenMusic: Un Langage Visuel pour la Composition Assistée par Ordinateur*, PhD. Thesis, Université Paris VI, 1998.

[2] Agon, C., Stroppa, M. and Assayag, G. "High Level Musical Control of Sound Synthesis in OpenMusic", *Proceedings of the International Computer Music Conference*, Berlin, Germany, 2000.

[3] Allen, J. F. and Ferguson, G. "Actions and Events in Interval Temporal Logic", *Journal of Logic and Computation*, 4, 5, 1994.

[4] Assayag, G., Agon, C., Fineberg, J. and Hanappe, P. "An Object Oriented Visual Environment for Musical Composition", *Proceedings of the International Computer Music Conference*, Thessaloniki, Greece, 1997.

[5] Assayag, G., Agon, C., Fineberg, J. and Hanappe, P. "Problèmes de notation dans la composition assistée par ordinateur", *Actes des rencontres pluridisciplinaires Musique et Notation*, Lyon (GRAME), France, 1997.

[6] Balaban, M. and Murray, N. "Interleaving Time and Structures", *Computer and Artificial Intelligence*, 17(4), 1998.

[7] Beurivé, A. and Desainte-Catherine, M. "Representing Musical Hierarchies with Constraints", *Musical Constraints Workshop (CP'2001)*, Paphos, Cyprus, 2001.

[8] Boulanger, R. (Ed.) *The Csound Book*, MIT Press, 2000.

[9] Boulez, P. "Timbre and composition – timbre and language.", *Contemporary Music Review*, vol. 2, part. 1, 1987.

[10] Bresson, J., Stroppa, M. and Agon, C. "Symbolic control of Sound Synthesis in Computer Assisted Composition", *Proceedings of the International Computer Music Conference*, Barcelona, Spain, 2005.

[11] Coduys, T. and Ferry, G. "Iannix. Aesthetical/symbolic visualisations for hypermedia composition", *Proceedings of the Sound and Music Computing Conference*, Paris, France, 2004.

[12] Dannenberg, R. B., McAvinney, P. and Rubine, D. "Arctic: A Functional Approach to Real-Time Control", *Computer Music Journal*, vol. 10, n°4, Winter 1986.

[13] Dannenberg, R. B., Desain, P. and Honing, H. "Programming Language Design for Music", in C. Roads et al. (Eds.) *Musical Signal Processing*, Swets and Zeitlinger, 1997.

[14] Desainte-Catherine, M. and Beurivé, A. "Time Modeling for Musical Composition", *Proceedings of the 1st International Conference on Fuzzy Systems and Knowledge Discovery – FSKD'02*, Singapore, 2002.

[15] Eckel, G. and Gonzalez-Arroyo, R. "Musically Salient Control Abstractions for Sound Synthesis", *Proceedings of the International Computer Music Conference*, Aarhus, Denmark, 1994.

[16] Garnett, G. "Music, Signals and Representations: a Survey", in G. De Poli, A. Piccialli, C. Roads (Eds.) *Representations of Musical Signals*, MIT Press, 1991.

[17] Haddad, K. "OpenMusic OM2CSound – Bibliothèque de modules de generation de scores pour Csound", *Ircam software documentation*, M. Battier (Ed.), 1999.

[18] Honing, H. "Issues in the Representation of Time and Structure in Music", *Contemporary Music Review*, 9, 1993.

[19] Mathews, M and Kohut, J. "Electronic simulation of violin resonances" *Journal of the Acoustical Society of America*, vol. 53, no. 6, 1973.

[20] Puckette, M. "Pure Data: another integrated computer music environment", *Proceedings of the Second Intercollege Computer Music Concerts*, Tachikawa, Japan, 1996

[21] Puckette, M. "Using Pd as a Score Language", *Proceedings of the International Computer Music Conference*, Göteborg, Sweden, 2002.

[22] Risset, J.-C. and Mathews, M. "Analysis of instrument tones", *Physics Today* 22 n° 2, 1969.

[23] Risset, J.-C. "Composing sounds, bridging gaps – the musical role of the computer in my music", in *Musik und Technik*, Helga de la Motte-Haber & Rudolf Frisius, ed., Schott, Mainz, 1996.

[24] Rodet, X. and Cointe, P. "Formes: Compostion and Scheduling of Processes", *Computer Music Journal*, vol. 8, no 3, Fall 1984.

[25] Rodet, X., Potard, Y. and Barrière J.-B. "The CHANT project: From the synthesis of the singing voice to synthesis in general", *Computer Music Journal*, vol. 8, no 3, Fall 1984.

[26] Stockhausen, K. "...wie die Zeit vergeht...", *Die Reihe*, n°3, 1957.

[27] Stroppa, M. *Traiettoria* (1982-84), a cycle of three pieces (*Traiettoria… deviata*, *Dialoghi*, *Contrasti*) for piano and computer-generated sounds. Recorded by Wergo: WER 2030-2, 1992.

[28] Stroppa, M. "Live electronics or… live music? Towards a critique of interaction", in *The aesthetics of Live electronics*, M. Battier, Ed., 1994

[29] Stroppa, M. "Paradigms for the high level musical control of digital signal processing", *Proceedings of the COST-G6 Conference on the Digital Audio Effects (DAFX-00)*, Verona, Italy, 2000.

[30] Wright, M., Chaudhary, A., Freed, A., Wessel, D., Rodet, X., Virolle, D., Woehrmann, R. and Serra, X. "New applications of the Sound Description Interchange Format", *Proceedings of the International Computer Music Conference*, Ann Arbor, USA, 1998.

[31] Xenakis, I. *Formalized Music: Thought and Mathematiccs in Composition*, Indiana University Press, 1992.