SOUND WRITING AND REPRESENTATION IN A VISUAL PROGRAMMING FRAMEWORK

Jean Bresson and Carlos Agon
IRCAM - CNRS UMR 9912
Music Representations Research Group

ABSTRACT

This article addresses the issue of the representation and manipulation of sounds in Computer-Aided Composition and presents related works in the OpenMusic visual programming environment.
Keywords – Computer-Aided Composition, Visual Programming, Sound Synthesis.

1. INTRODUCTION

OpenMusic (OM) [1] [2] is a visual programming language for music composition that allows composers to create and experiment on formal computing models of musical structures and processes. It is a visual extension of the LISP/CLOS language improved with data structures and features specialized in the musical processing.
The Computer-Aided Composition (CAC) approach followed by OM puts forward the notion of potential scores; it provides visual representations of music, but also writing supports, that is, places where symbols and objects are organized and thought in order to develop processes leading to musical constructions [3].
Visual programming seems us an interesting approach for this conception of music representation, by offering a great expressiveness together with readable interfaces (see Figure 1).

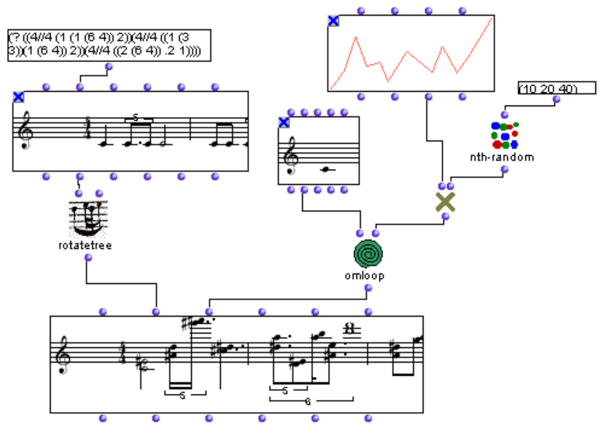


Figure 1. Creating musical structures by visual programming in OpenMusic.

In this framework, we investigate the possibilities to integrate new tools and structures for the manipulation (analysis, synthesis, processing) of sound signals.

Email: jean.bresson@ircam.fr

Contemporary music practices, indeed, increasingly make use of both symbolic computing and sound synthesis techniques, but these two aspects are generally designed and performed independently. We believe that putting them together in a same environment improves the musical expressiveness afforded by the computer for music creation.
We hence deal with new problems which were not that salient when the finality was limited to the building of somehow traditional scores by computing means, and regardless (from a compositional standpoint) of the actual rendering of this score. Sound description data are not the more adapted to symbolic processing and visual programming, due to the new scales and ontology of these musical objects. Musical writing and representation are thus to be thought through a renewed angle, and the programming framework must be adapted in order to integrate and bridge the low-level sound processing data and conformed high-level compositional tools.
This article presents some propositions developed in OpenMusic, regarding data structures (section 2), programming tools (section 3) and high-level writing supports (section 4).

2. DATA STRUCTURES

2.1. High-Level Control and Low-Level Processing

CAC evolves in the symbolic domain, using high-level representations of objects and processes. In the domain of sound synthesis, this is more the question of continuity, of precise, integral descriptions, including sound signals but also all their analytic representations, temporal trajectories and evolutions. The amount of data concerned is also much bigger than when notes are the smallest compositional primitives.
The new compositional data structures must therefore incorporate both a symbolic side for the high-level control and setting of the sound descriptions, and a low-level side where precise subsymbolic signal descriptions [4] are dealt with.
Sound synthesis is generally controlled by some sets of punctual data, envelopes (temporal evolutions of some parameters), or matrices. All these structures, as well as the sound itself, exist among the OM classes, together with tools for manipulating them; they are objects which can be created and transformed in the visual programs, and be part of the symbolic processing framework.

2.2. Matrices: a Link Between Symbolic and Signal Processing

The matrix is an important class in the OpenMusic sound synthesis tools. It can be constituted of various kinds of data and behaviours that describe a sound or a part of it. The joint representation of these data allow to consider them as a single object in the computing flows, and especially to consider relations between them and between their individual evolutions. Figure 2 shows an example of an OM patch in which a sound is specified in the form of a matrix. This matrix is then converted into a sound file through a synthesis function.

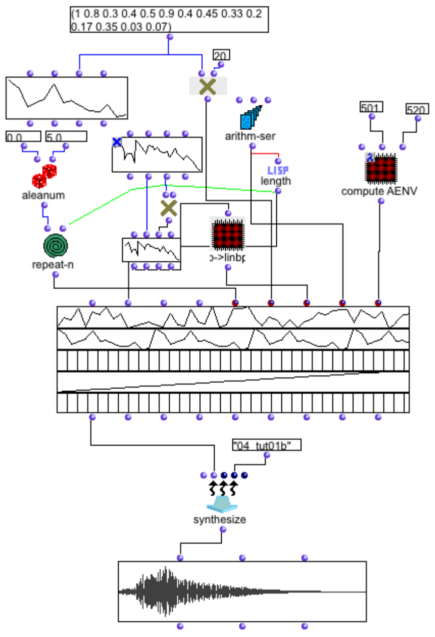


Figure 2. Using an OM matrix for the control of a sound synthesis process.

Matrices can be set by symbolic means. They are made of a set of different parameters which can be specified by objects such as data lists, BPF, or functions. These symbolic data are evaluated and sampled depending on the pre-specified size of the matrix in order to fill the numerical values of the array [5]. The symbolic specification is thus dealt with separately from the numeric values, computed in a latter phase (generally, that of the sound synthesis process). This allows to keep working in the symbolic domain while creating the control objects.

Furthermore, the control data can be transformed and completed dynamically at the time of the synthesis thanks to an optional user defined function for the initial matrix elements processing. Programs and data are thus mixed in this matrix object in order to integrate structural and behavioural properties of the description.

2.3. The SDIF Format: a Generic Support for Sound Descriptions

SDIF (Sound Description Interchange Format) [6] is a standard format for the storage and transfer of sound description data. It is an open and flexible format, based on a matrix data flow, which we proposed to use as a generic support for the concept of sound description in the compositional framework [7]. An SDIF file concentrates in a standardised way the simultaneous temporal evolution of different matrices of data, and allows the design of generic tools for their manipulations. A class represents an SDIF file in OM, and is assorted with tools for its manipulation either in the visual programs or using a dedicated editor (see Figure 3). The use of external files also allows the storage of sound descriptions on the disk, separately from the control level, and thus to prevent the data to use up too much space in the patches and processes.

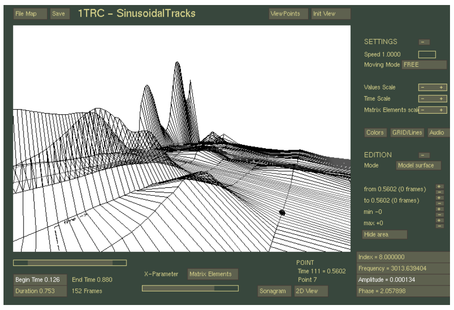


Figure 3. SDIF-Edit [8]: visualization of SDIF sound description files.

3. PROGRAMMING TOOLS

3.1. Low-Level Data Programming

Composers usually do not directly work with low-level data in CAC systems, but rather deal with high-level symbolic structures. However, in order to control the deep structure and rendering of music in such a framework, the low-level data management must be available in it.

Composers must then be provided with tools for a direct interaction with the low-level processing in order to develop creative paths between symbolic and sound synthesis domains, without being restricted by predefined tools.

The MIDI toolkit developed in OM [9] was an example in this perspective: a set of classes and functions representing the basic MIDI structures and operations, completed with compound structures, allowed to bridge MIDI and symbolic objects and to develop visual programs for controlling the low-level MIDI processing, communication and rendering.

The SDIF tools now allow for the similar operations with generalized sound descriptions. Various classes correspond to the SDIF format specification structures (frames, matrices, streams, types, etc.) and to manipulate these data in order to build user-defined structures (see Figure 4), possibly bound to be sent or transferred to external software tools.

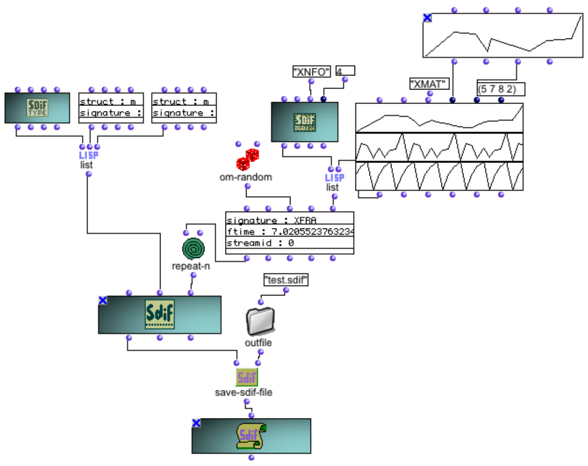


Figure 4. Creating SDIF structures in OpenMusic.

The `SDIFMatrix` class is an extension of the OM matrix presented above, and therefore benefits from the same capabilities for its symbolic setting and processing.

Files (in ASCII or SDIF formats) can be dynamically handled in the visual programs using a file stream reader/writer iteration (see Figure 5).

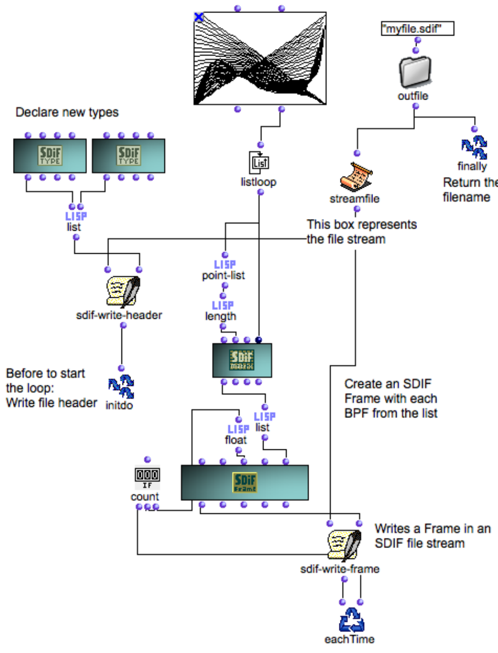


Figure 5. Example of a `file-box` program in OM: iterative writing of SDIF data frames in a file (the file pointer is represented by the `streamfile` box).

3.2. Sound Processing Connection

Csound [10] and Modalys [11] are the most frequently used synthesizers in association with OM. Instruments and scores can be created for them in visual programs [12] [13], and represent the specifications of how will the control data be processed in order to produce the sound.
The sound synthesis, starting from sound descriptions or control data and synthesis patches, is performed by external tools for which OM translate these data in the form of parameters files and/or command lines. Interface functions then allow to run the corresponding programs from the OM patches an to get back the resulting sound files.
A set of functions connected to the underlying audio management system [14] also allow to perform sound processing operations on sound files within the visual programs.
In terms of programming, all these sound processing connection tools allow to incorporate these operations in the functional flow paradigm of the environment. Further descriptions are given in [15].

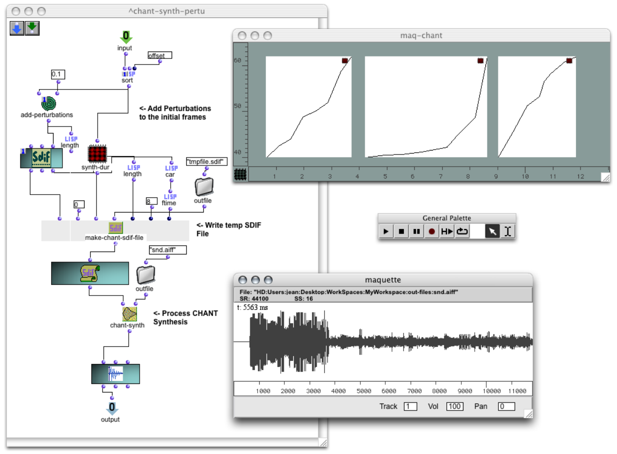
4. SUPPORTS FOR MUSIC WRITING

4.1. Visual Programs

The data structures and programming tools presented above thus allow the composers to create compositional models extended from the high-level symbolic domain down to the signal processing part within the OM environment. The functional abstractions and patch embedding possibilities provided by the programming environment allow the users to define successive abstraction levels, between which they can translate their work in order to compose simultaneously on the musical forms constructions and on the sound synthesis processes.
Other writing/programming supports, bound to be used for the creation and control of sound synthesis processes are currently studied and developed in this scope. We will briefly present two of them in the next sections.

4.2. Temporal Programs: *Maquettes*

The *maquette* [1] is a programming interface provided with a temporal axis which allows to unfold the musical objects and processes in a temporal context. This is a powerful tool that unites temporal and functional properties of a musical structure. In [17], we proposed an extended model providing a control over the program semantics of the maquette.
An example is given in Figure 6: the control structures (three break-point functions) are generated and temporally organised in the maquette (at the right). This temporal form is then processed by an auxiliary program attached to the maquette (at the left), that specifies its functional semantics. This semantics corresponds to a sound synthesis process, so that the result is a sound file, represented in the bottom-right part of the figure.



File: "HD:Users:jean:Desktop:WorkSpaces:MyWorkspace:out-files:snd.aiff"
SR: 44100 SS: 16
t: 5563 ms

Figure 6. Design and control of a sound synthesis process in a maquette.

We thus retrieve here the distinction between the control level and the sound processing level; the maquette being the intermediate representation providing the user with a high-level access to the musical form constituted by the control data, and assorted with a control over the low-level sound processing.
The possibility to embed the maquette one in another additionally enables the construction of hierarchical structures, where the result of the processing of one hierarchical level becomes a control primitive in the upper hierarchical level.

4.3. Multi-modal Scores: *Sheets*

The *sheet* is another type of document which development is currently in progress. It will support the joint representation of heterogeneous objects (voices, sequences, sounds, controllers, etc.) in a same score-based support. The sheet (see Figure 7) is organized in tracks; each one contains a set of sequenced objects, which can be musical objects or control structures, but above all that can proceed from both musical (e.g. scores) or linear (e.g. sounds) time representations. In this case, the linear representation is left down for guarantying the visual alignment of any

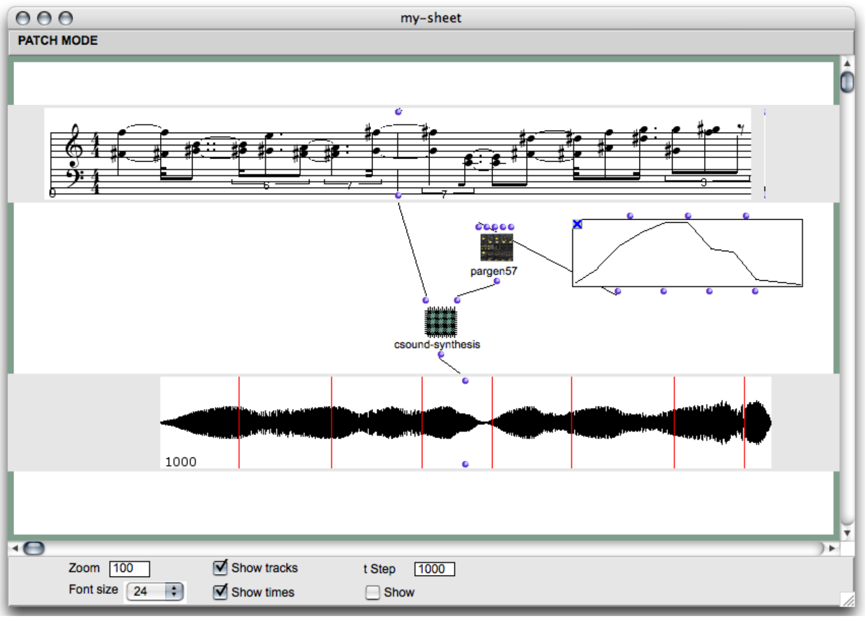


Figure 7. A sheet including two tracks. The sound of track 2 is synthesised using some data from the voice in track 1. Both objects are represented in a common temporal axis: the sound segments (1 second each) are graphically scaled in order to respect the space distortions created by the score display.

simultaneous objects or object elements (e.g. notes in a voice, inflexion points of an envelope, markers in a sound, etc.)
Visual programs can also be developed in the sheet, and incorporate the tracks' musical objects. This score can thus be evaluated, making some parts of it depend on other parts while maintaining a coherent temporal representation.

5. CONCLUSION

In this article we presented some features of the OpenMusic visual programming environment, that allow the users to work with sound related data (analysis data, synthesis parameters, or any personal representation which we group under the term of sound descriptions), and provide a control over the low-level musical processes, traditionally delegated to predefined tools.
We also mentioned interfaces for the high-level composition in this framework, that provide some extended musical representations and writing supports.
As we explained in the introduction, we believe that one of the interesting potentialities of computer music is the availability of both symbolic musical constructions and sound generation and synthesis means within a same tool (the computer). A full control of music in this context thus includes these two levels and must allow for easy and cross connections between them.

6. REFERENCES

[1] C. Agon, *OpenMusic : Un langage visuel pour la composition musicale assiste par ordinateur*, Ph.D. thesis, Université Pierre et Marie Curie (Paris 6), France, 1998.

[2] G. Assayag, C. Agon, J. Fineberg, and P. Hanappe, "An Object Oriented Visual Environment for Musical Composition," in *Proceedings ICMC*, Thessaloniki, Greece, 1997.

[3] G. Assayag, "Computer Assisted Composition today," in *1st symposium on music and computers*, Corfu, 1998.

[4] M. Leman, "Symbolic and subsymbolic description of music," in *Music Processing*, G. Haus, Ed. Oxford University Press, 1993.

[5] C. Agon, M. Stroppa, and G. Assayag, "High Level Control of Sound Synthesis in OpenMusic," in *Proceedings ICMC*, Berlin, Germany, 2000.

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

[7] J. Bresson and C. Agon, "SDIF Sound Description Data Representation and Manipulation in Computer-Assisted Composition," in *Proceedings ICMC*, Miami, USA, 2004.

[8] "SDIF-Edit," http://recherche.ircam.fr/equipes/repmus /bresson/sdifedit/sdifedit.html.

[9] J. Bresson, "OpenMusic MIDI Documentation," Ircam Software Documentation, 2004.

[10] R. Boulanger, Ed., *The Csound Book*, MIT Press, 2000.

[11] N. Ellis, J. Bensoam, and R. Caussé, "Modalys Demonstration," in *Proceedings ICMC*, Barcelona, Spain, 2005.

[12] J. Bresson, M. Stroppa, and C. Agon, "Symbolic Control of Sound Synthesis in Computer-Assisted Composition," in *Proceedings ICMC*, Barcelona, Spain, 2005.

[13] K. Haddad, "OpenMusic OM2CSound," *Ircam Software Documentation*, 1999.

[14] "LibAudioStream," http://libaudiostream.sourceforge.net/.

[15] J. Bresson, "Sound Processing in OpenMusic," in *Proceedings of DAFx-06*, Montreal, Canada, 2006.

[16] M. Wright and A. Freed, "Open SoundControl : A New Protocol for Communicating with Sound Synthesizers," in *Proceedings ICMC*, Thessaloniki, Greece, 1997.

[17] J. Bresson and C. Agon, "Temporal Control over Sound Synthesis Processes," in *Proceedings of Sound and Music Computing Conference - SMC'06*, Marseille, France, 2006.