SOUND PROCESSING IN OPENMUSIC

Jean Bresson

IRCAM - CNRS, Music Representations Team
Paris, France
bresson@ircam.fr

ABSTRACT

This article introduces some new possibilities of audio manipulations and sound processing in the Computer-Aided Composition environment OpenMusic. Interfaces with underlying sound processing systems are described, with an emphasis on the use of the symbolic and visual programming environment for the design of sound computation processes.

1. INTRODUCTION

OpenMusic (OM) [1] is a visual programming language specialized in the symbolic processing of musical objects that provides composers with high-level musical representations. Following the tradition of the IRCAM's Computer-Aided Composition (CAC) systems [2], the main proposal of OM is that of the formalization of the musical structures, which makes it possible the creation of the corresponding computing models. However, the idea of a symbolic manipulation of musical data, close to the compositional concepts, historically tended to make these systems diverge from the sound-related concerns.

The development of contemporary musical practices, which incorporate sounds in the compositional processes, led to a renew interest for bringing together these two different fields. In this context, the OM environment provides possibilities for bridging the compositional and signal processing domains.

In a first step, this connection has been implemented with the automatic generation of scripting or parameter files bound to external processing tools. Various OM libraries were developed in this scope, in relation with different software synthesizers (generation of Csound scores [3], of Diphone [4] scripts, of SuperVP parameter files [5], etc.) Some direct interfaces with these sound processing tools were then created, which allowed an easier experimentation, by launching sound processing tasks directly from the OM visual programs (*patches*) [6]. More than a simple convenience of use, this allows to unite sound processing programs and symbolic musical processes in a same environment and in a common computational flow, which induces improved possibilities for the control and integration of these programs in compositional models (e.g. [7]).

Our current works in OpenMusic are heading towards new tools and interfaces for adapting the computational and symbolic tools to sound processing applications. The articulation between sound signal and musical symbolism in composition is one of the main problematics of this project, which points at exploring the symbolic resources bestowed by computers and CAC in order to describe audio signals and processes.

This article presents some new possibilities of sound manipulations in OpenMusic. The IRCAM's analysis/synthesis tools SuperVP and pm2 are particularly targeted, following the idea of an integrated programming environment where various cycles of the musical materials, from sound to symbolic data, and from symbolic data to sounds, could be developed.

2. NEW AUDIO INTERFACE IN OPENMUSIC

Since version 5.0, the OpenMusic audio architecture is based on the GRAME's LibAudioStream library [8]. This library provided a better control for sounds scheduling and rendering through the concept of sound stream structures. The sound objects now can be individually assigned volume or panoramic values. They are possibly split up into different audio tracks, which allows a global interactive control through a graphical mixing console (Figure 1).

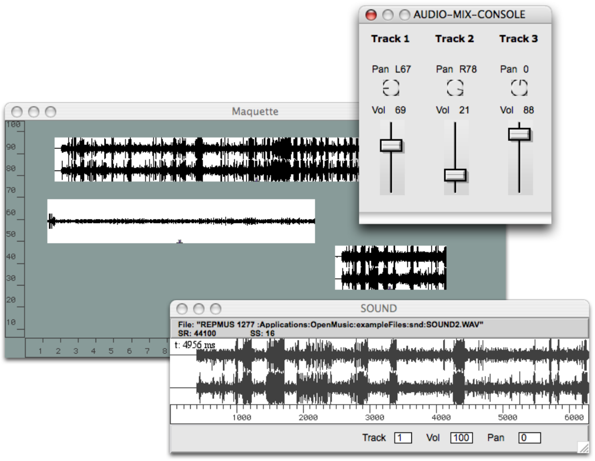


Figure 1: *Audio interface in OM 5.*

3. A COMPOSITIONAL APPROACH OF SOUNDS

3.1. The Digital Sound

Once it is recorded, the sound can become an abstract compositional object , thanks to the "material" support [9]. From this perspective, the digitalization constitutes a major revolution in the realm of the possibilities for a compositional approach of sounds.

An audio file, digital support of a sound, is made of a sequence of samples; each one of them carries a small quantity of information, but the sequence constitutes a whole bigger than the sum of all the individual samples: a stable acoustic waveform that can be observed and manipulated. From a theoretical point of view, this idea can be compared with the symbolic/sub-symbolic distinction formulated in [10]. In this division, sub-symbolic data are void of sense out of their context, while symbolic data are structured and bring semantic information: they have an intrinsic meaning and are straightforward to be used in musical constructions.

3.2. Sound Objects and Symbolic Processing

Considered as a continuous stream, the sound samples constitute a coherent data set (a digital waveform), close to the acoustic reality and to the musical material.

The basic musical manipulations on this sound representation are the cut, mixing, and sequencing, as could be done with a “physical" tape. In the context of visual programming, these operations correspond to functional call boxes on which can be connected sound objects. They are permitted by the LibAudioStream functionalities. Sound streams can thus be created, cut, mixed, and sequenced in visual programs (see Figure 2). The programming tools allow to make algorithmically the “montage" and mixing of sounds, generally “hand-made" using sequencing environments after recording, processing, or synthesizing them.

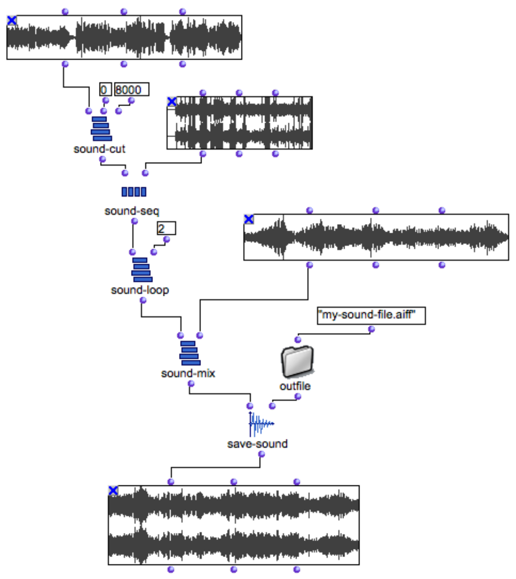


Figure 2: *Manipulations of sounds with the OM audio tools. A section of the first sound is sequenced with the second one and this sequence is repeated two times. The result is then mixed with a third sound.*

3.3. Abstraction of Sound

An abstraction of a sound can be any symbolic data that represent this sound in a compositional purpose. Between the symbolic and sub-symbolic domains, the data extracted from a sound following a scientific or musical demarche constitute the successive steps of abstraction that allow for its compositional manipulation.

The segmentation can be a first method for considering a sound in a symbolic manner, since it identifies primitives for a symbolic processing (the sound segments). The segmentation can be integrated in the sound objects by the use of temporal markers. In OM, markers can be assigned to a sound either by a program (in a patch), or manually inside the sound editor (see Figure 3).

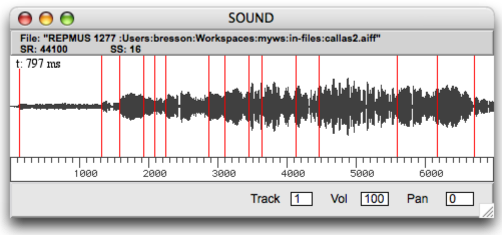


Figure 3: *Editing of temporal markers in an audio file.*

However, the size of the segments that will allow for a symbolic manipulation remains hard to decide. Segments of two samples remain sub-symbolic, while an intelligent segmentation, that discriminates syllables for example, constitutes symbolic data. Between these two limits, the size of a sound analysis window (e.g. 512 samples), or the units of a concatenative sound synthesis algorithm, can be considered either in one or another category.

In another general approach, a set of isolated samples, extracted from a waveform by any formalized way (e.g. downsampling, see Figure 4), can also be regarded as an abstraction of the digital sonic material.

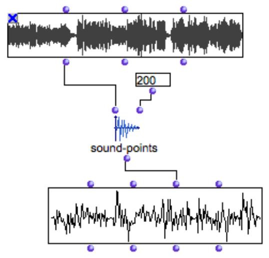


Figure 4: *Extracting sound samples from an audio file with a given sampling rate.*

This method was used for example by K. Haddad in the piece *Adagio for String Quartet*, described in [11].

Sound analysis is another way to get sound representations in compositional processes. This will be detailed in the next section.

4. SOUND ANALYSIS

Sound analysis data are frequently used in computer-assisted composition programs, either for being transformed and dispatched in sound synthesis processes, or for being used as basic compositional material. The SDIF file format [12] allows the different computer music tools and software to share sound descriptions data: a special class (*SDIFFile*) is used to represent it in OpenMusic. This class constitutes an abstract support for sound descriptions [13]; it will generally correspond to an intermediate object in the visual program, used to store the analysis data coming from the sound processing kernels. SDIF file objects can be visualized using SDIF-EDIT [14], and inspected in order to extract the description data .

SuperVP [15] and pm2 are two sound processing tools developed by the Analysis/Synthesis team of IRCAM. They are the processing kernels of the AudioSculpt software [16]. Used autonomously in OM, they provide various types of sound analysis, and return SDIF description files.

4.1. Spectral Analysis

Most of the sound processing algorithms in the SuperVP phase vocoder [17] are based on the short time Fourier transform (STFT), which provides a time/frequency representation of sounds. This analysis function can be called in OpenMusic by the *FFT* box (see Figure 5). The analysis data is then transferred into the OM program via an SDIF file.

For this analysis, the main input data to provide to the *FFT* box (in addition to the sound file to be analyzed) are the FFT size, and the size, shape and step of the STFT analysis window. These parameters should be adjusted depending on the initial sound and the expected characteristics of the analysis data.

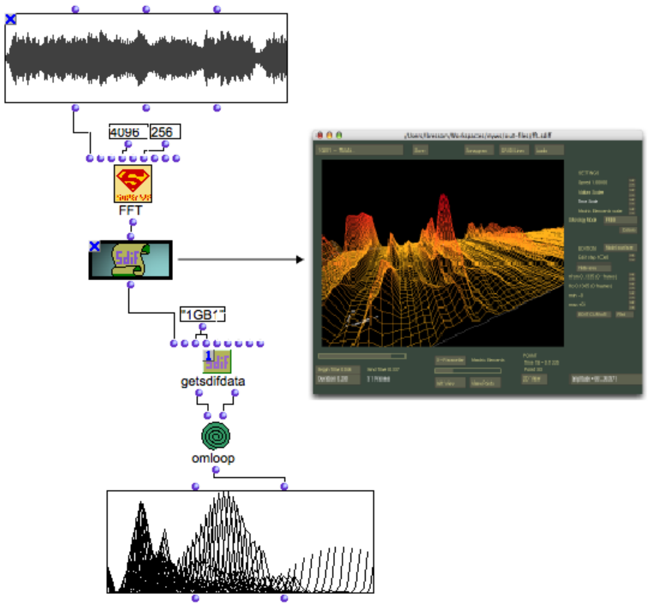


Figure 5: *SuperVP FFT analysis in OpenMusic. The SDIF analysis data can be viewed using* SDIF-EDIT *and extracted in the patch.*

The fundamental frequency estimate (F0) analysis [18] is another description provided by SuperVP and which is likely to be used in compositional and sound processing programs: its direct musical meaning helps musical interpretations and manipulations. In addition to the FFT settings, some supplementary parameters must be set to the *f0-estimate* box (frequency ranges for analysis, noise threshold, smoothing order, etc.) An example will be given in section 7 (Figure 11).

OpenMusic can also invoke the transient detection analysis [19] from SuperVP. This analysis provides an automatic temporal segmentation of the sound files. This segmentation can be useful for correlating sound descriptions to symbolic rhythms or temporal structures, as will be illustrated in the example of Figure 10.

4.2. Additive Analysis

Pm2 is an additive analysis/synthesis software that can process partial tracking analysis of sound files. The partial tracking consists in the detection of the most important partials (sinusoidal components of a sound signal) and the following of their frequency and dynamic evolutions [20].

This analysis can be either harmonic, in which case a fundamental frequency evolution must be provided and the partials will be considered as harmonics of this fundamental frequency, or inharmonic, in which case each partial constitutes an independent component with individual beginning and ending times, frequency and amplitude evolutions.

The results of the pm2 partial tracking analysis are also stored in SDIF files, and can then be converted into a *chord-sequence* object [21], as illustrated in Figure 6.

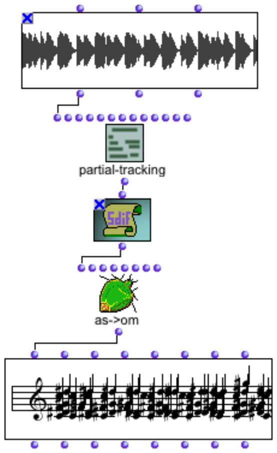


Figure 6: *Partial tracking analysis of a sound and conversion into a chord sequence.*

The *chord-sequence-analysis* is another kind of additive analysis based on a set of temporal markers. These markers can be issued from the sound object (see Figure 3), or being processed by a preliminary transient detection analysis. The additive components are thus averaged frequencial components that can be directly converted into chords or notes.

5. TREATMENTS

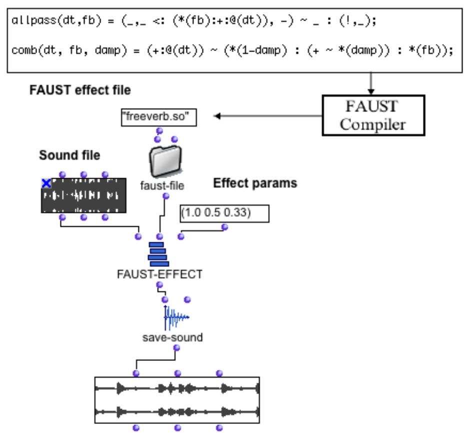
We call treatment the fact of synthesizing a sound by applying a transformation to an initial sound. Such treatments can be applied directly on the waveform representation, or by the analysis/processing/synthesis method.

5.1. Sound Streams Processing

Transforming a sound by processing its samples stream is what is generally called an effect. We currently experiment the integration of such process in OM using Faust [22], a language for the design and compilation of digital sound processing (DSP) objects. The symbolic primitives and syntax of this language make it a suitable tool for the design of sound treatments in a compositional context. The LibAudioStream interface then allows to apply Faust compiled effects to the sound streams (see Figure 7).

5.2. Analysis-driven Processing: SuperVP

The analysis/treatments/synthesis method proposed by SuperVP and AudioSculpt allows for original and powerful manipulations



```
allpass(dt,fb) = (_,_ <: (*(fb):+:@(dt)), -) ~ _ : (!,_);

comb(dt, fb, damp) = (+:@(dt)) ~ (*(1-damp) : (+ ~ *(damp)) : *(fb));
```

Figure 7: *Applying a transformation to a sound using* Faust*. The framed code is a fragment of the* Faust *effect definition file.*

of sounds. Thanks to a treatments sequencer, various kinds of sound transformations can be created and combined, and eventually applied in a processing phase for creating a new sound.

In the OM visual programming framework, the corresponding system is centred on the *supervp-processing* function, a general box for transforming a sound using various combinations of the SuperVP treatments. Each available treatment is represented with another particular box. These boxes include the time stretching, pitch shifting, transposition, band filter, formant filter, breakpoint filter, clipping, and freeze treatments. Each treatement may require some special parameters settings, and can be applied to the entire sound or to a time interval in it. They can be used alone or combined, connected to the general *supervp-processing* box as a list of treatments (see Figure 8).

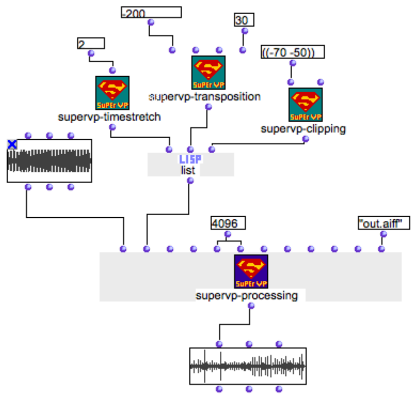


Figure 8: *Applying treatments (time stretch, transposition, and clipping) with the* supervp-processing *box.*

The parameters (e.g. stretching factors, transposition intervals, etc.) can be set with constant values or with time-varying data using imported parameter files or break point functions (BPF). These data can thus come from complex compositional processes implemented in OM. Section 7 will give examples of sound processing patches.

6. SYNTHESIS

Several works have been carried out regarding sound synthesis models and interfaces in OM, most often using Csound [23] which provides a wide range of sound synthesis possibilities. Some interfaces have also been developed with the IRCAM's sound processing tools.

In addition to the treatments processing discussed in the previous section, the *supervp-cross-synthesis* and *supervp-sourcefilter-synthesis* are other boxes that call the corresponding SuperVP functions (see Figure 9).

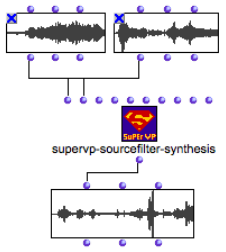


Figure 9: *: Source-Filter synthesis with SuperVP.*

The pm2 additive synthesis, performed from an SDIF additive description file, also exists as a function call box.

The CHANT synthesizer [24] can be used in OM as well, by formatting SDIF parameters files and calling the synthesis kernel. Future works should improve this interface in order to allow for easier and better use of this synthesizer's potentialities.

Finally, the OM-Modalys library allows the design of Modalys physical models synthesis in OM programs [25].

Most of these tools are "direct" interfaces that do not exactly represent the abstract methods generally targeted in CAC applications. They might nonetheless provide a base for the development of higher-level control interfaces.

7. USING OPENMUSIC FOR SOUND PROCESSING

The interest in creating sound processing programs in OM relies on two principal points that we will detail in this section.

7.1. Computation of Processing Parameters

Using sound processing tools in OM allows to set the parameters of the treatments by complex programs and with the symbolic data structures provided by the environment. We will show two examples, where the sound analysis tools presented above are used together with symbolic data structures, for the computaion of some processing parameters.

In the example of Figure 10, a time-stretching factor evolution is computed starting from the attack transients detected in a sound file and from a targeted rhythm: this variable factor will stretch the sound segments in order to make them match with this rhythm.

The example of Figure 11 is another program which transposes a sound using time-varying data coming from the F0 analysis. The transposition data is computed by an operation of symmetry between the F0 and a given pitch, so that the result is a sound with a constant fundamental frequency (the value of the reflection axis), which keeps all the other characteristics (timbre,

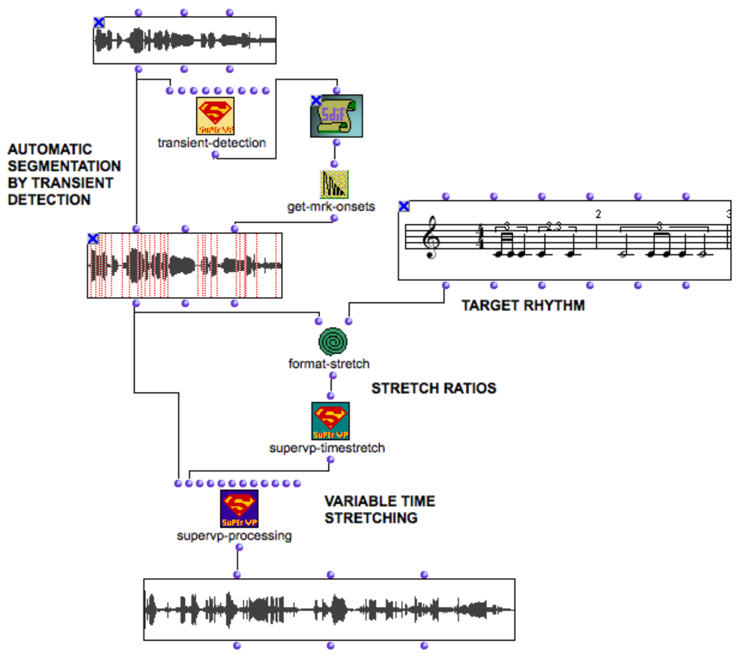


Figure 10: *Variable time stretching: matching sound segments with a rhythm.*

transients, noise, etc.) from the original sound.

In the same perspective, the OM-AS library functions [5], dedicated to the automatic generation of SuperVP treatment parameters files, can also be connected to the corresponding treatment boxes, provinding a symbolic mean to generate these parameters (e.g. computing a sound transposition parameter file starting from a *chord-seq* melody, etc.)

7.2. Iterative Processes

The visual programming tools may also be used for managing multiple processing operations. A sound, or a sound bank, can be processed iteratively using the *omloop* control structure, in order to create another set of sound files depending on fixed or dynamically generated parameters (see Figure 12).

The output sounds can then possibly be arranged in a *maquette* [1] (Figure 12), or directly mixed/sequenced as a single sound file using the audio processing tools discussed in section 3.2.

8. CONCLUSION

We presented the possibilities of an algorithmic and symbolic manipulation of sounds in the OpenMusic visual programming environment, using the audio framework (manipulation of the sound objects) and the IRCAM sound processing tools interfaces (fft analysis, F0 analysis, transient detection, partial tracking analysis, sound treatments, cross-synthesis, etc.) All these tools are available and documented with tutorial patches in the latest OM releases, either included in the OM core distribution (audio features) or in the new OM-SuperVP library (SuperVP and pm2 tools).

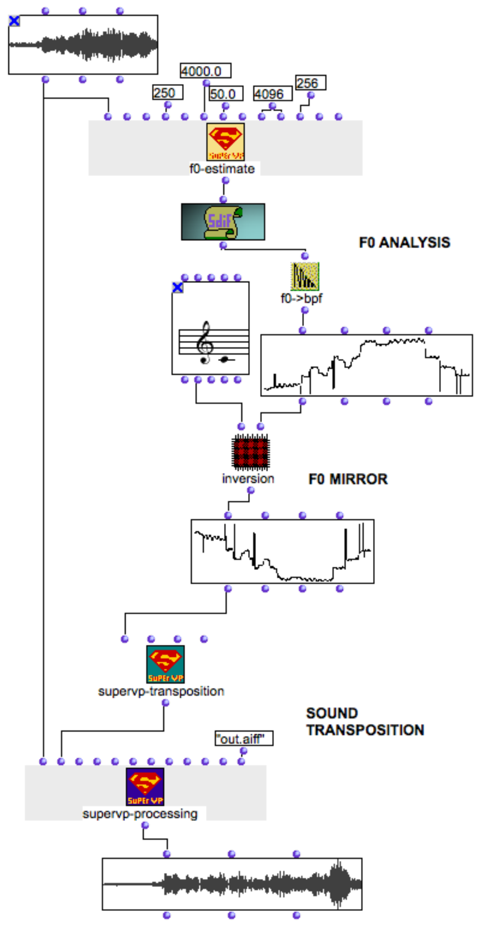


Figure 11: *Fundamental frequency analysis (F0) of a sound and transposition using the mirror of the F0.*

Graphical programming interfacing with sound analysis and processing facilitates sound manipulations in a compositional context. The iterative processes, coupled with visual representations and audio processing tools, allow for efficient and intuitive experimentations. We have in mind that more interfaces with external tools could be added to this framework, which will be completed with higher-level abstractions and assorted with further musical experiences and applications.

9. ACKNOWLEDGEMENTS

We would like to thank Stephane Letz from GRAME for his help and support with the LibAudioStream library. The SuperVP boxes and the examples presented in this article have been created with Jean Lochard from the IRCAM pedagogic department. SuperVP and pm2 are currently developed and maintained by Axel Röbel in the IRCAM Analysis/Synthesis team.

10. REFERENCES

[1] C. Agon, “OpenMusic: Un langage visuel pour la composition assistée par ordinateur,” Ph.D. dissertation, University of Paris 6, 1998.

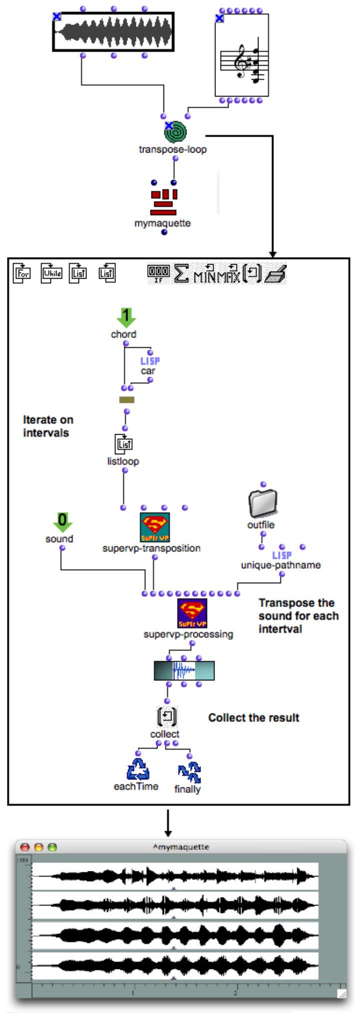


Figure 12: *Iterative creation and superposition of sounds by transposing a sound file following the intervals of a chord.*

[2] G. Assayag, “Computer-assisted composition today,” in *First Symposium on Music and Computers*, Corfu, 1998.

[3] K. Haddad, “OpenMusic OM2CSound,” *Ircam Software Documentation*, 1999.

[4] X. Rodet and A. Lefèvre, “The Diphone program: New features, new synthesis engines and experience of musical use,” in *Proc. Int. Comp. Music Conf. (ICMC’97),* Thessaloniki, Greece, 1997, pp. 418–421.

[5] H. Tutschku, “OpenMusic OM-AS Library,” *Ircam Software Documentation*, 1998.

[6] J. Bresson, M. Stroppa, and C. Agon, “Symbolic control of sound synthesis in computer assisted composition,” in *Proc. Int. Comp. Music Conf. (ICMC’05),* Barcelona, Spain, 2005, pp. 303–306.

[7] C. Agon, M. Stroppa, and G. Assayag, “High level musical control of sound synthesis in openmusic,” in *Proc. Int. Comp. Music Conf. (ICMC’00),* Berlin, Germany, 2000, pp. 332–335.

[8] LibAudioStream, Retrieved June 29th, 2006, [Online] http://libAudioStream.sourceforge.net/.

[9] P. Schaeffer, *Traité des Objets Musicaux*. Paris: Seuil, 1966.

[10] M. Leman, “Symbolic and subsymbolic description of music,” in *Music Processing*, G. Haus, Ed. Oxford University Press, 1993, pp. 119–164.

[11] K. Haddad, “Timesculpt in OpenMusic,” in *The OM Composer’s Book*, C. Agon, G. Assayag, and J. Bresson, Eds. Ircam – Delatour, 2006, pp. 45–62.

[12] Ircam, “Sound Description Interchange Format,” Retrieved June 29th, 2006, [Online] http://www.ircam.fr/sdif/.

[13] J. Bresson and C. Agon, “SDIF sound description data representation and manipulation in computer assisted composition,” in *Proc. Int. Comp. Music Conf. (ICMC’04),* Miami, USA, 2004, pp. 520–527.

[14] Ircam, “SDIF-Edit: Visualization of SDIF sound description files,” 2001, [Online] http://recherche.ircam.fr/equipes/repmus/bresson/sdifedit/sdifedit.html.

[15] P. Depalle and G. Poirot, “A modular system for analysis, processing and synthesis of sound signals,” in *Proc. Int. Comp. Music Conf. (ICMC’91),* Montréal, Canada, 1991, pp. 161–164.

[16] N. Bogaards and A. Röbel, “An interface for analysis-driven sound processing,” in *119th Conv. Audio Eng. Soc.,* New York, USA, 2004.

[17] M. Dolson, “The phase vocoder: A tutorial,” *Computer Music J.*, vol. 10, no. 4, pp. 14–27, 1986.

[18] B. Doval and X. Rodet, “Estimation of fundamental frequency of musical sound signals,” in *Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP’91),* Toronto, Canada, 1991, pp. 3657–3660.

[19] A. Röbel, “Transient detection and preservation in the phase vocoder,” in *Proc. Int. Comp. Music Conf. (ICMC’03),* Singapore, 2003, pp. 247–250.

[20] R. J. McAulay and T. F. Quatieri, “Speech analysis/synthesis based on a sinusoidal representation,” *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 34, no. 4, pp. 744–754, 1986.

[21] P. Hanappe and G. Assayag, “Intégration des représentations temps/fréquence et des représentations musicales symboliques,” in *Recherches et applications en informatique musicale*, M. Chemillier and F. Pachet, Eds. Hermès, 1998, pp. 199–207.

[22] Y. Orlarey, D. Fober, and S. Letz, “Syntactical and semantical aspects of Faust,” *Soft Computing*, vol. 8, no. 9, pp. 623–632, 2004.

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

[24] X. Rodet, Y. Potard, and J.-B. Barrière, “The CHANT project: From the synthesis of the singing voice to synthesis in general,” *Computer Music J.*, vol. 8, no. 3, pp. 15–31, 1984.

[25] N. Ellis, J. Bensoam, and R. Caussé, “Modalys demonstration,” in *Proc. Int. Comp. Music Conf. (ICMC’05),* Barcelona, Spain, 2005, pp. 101–102.