OpenMusic

Visual Programming Environment for Music Composition, Analysis and Research

Jean Bresson
STMS: IRCAM-CNRS-UPMC
1, place I. Stravinsky
75004 Paris, France
bresson@ircam.fr

Carlos Agon
STMS: IRCAM-CNRS-UPMC
1, place I. Stravinsky
75004 Paris, France
agon@ircam.fr

Gérard Assayag
STMS: IRCAM-CNRS-UPMC
1, place I. Stravinsky
75004 Paris, France
assayag@ircam.fr

ABSTRACT

OpenMusic is an open source environment dedicated to music composition. The core of this environment is a full-featured visual programming language based on Common Lisp and CLOS (Common Lisp Object System) allowing to design processes for the generation or manipulation of musical material. This language can also be used for general purpose visual programming and other (possibly extra-musical) applications.

Categories and Subject Descriptors

D.1.7 [**Software**]: Programming Techniques—*Visual Programming*; J.5 [**Computer Applications**]: Arts and Humanities—*Music*

General Terms

Design, Languages

Keywords

Visual Programming, Music, Computer-Aided Composition.

1. COMPUTER-AIDED COMPOSITION

OpenMusic (OM) is an environment designed for music composition in the tradition of what is conventionally called *computer-aided composition* in the contemporary music and computer music research communities [9]. The original purpose of computer-aided composition research was to provide composers with means to develop musical ideas and models using the computer: In this context, creating compositional tools in the form of standard applications, as is the case with most commercial music software, turned out to be too restrictive from a creative point of view. In order to better integrate the artistic thought and give musicians full access to the expressive and computational power of computer tools and formalisms, computer-aided composition environments needed to be more "programmable" and would actually better be *programming languages* than simple programs.

A number of computer music systems have therefore been developed during the past 20 years, with more or less explicit programming language features [22, 21, 29]. Some of them are open source and quite widely used software [27, 30].

While most "musical programming languages" principally deal with signal processing and sound synthesis, an original approach adopted by the IRCAM Music Representation team in the late 80s had the particularity to focus on symbolic musical structures and processes, that is, on compositional aspects traditionally ignored or carried out outside computer environments [11]. In this context and as far as they relate to formalisation and/or calculus, the design of compositional processes using programming languages and environments allows to better understand, explore and develop these processes than with other traditional or more specific media and applications. Fewer of such "symbolic" programming/computer-aided composition environments exist. Most of them are based on Common Lisp or other Lisp dialects (see for instance [33]).

OpenMusic [12] is one of the few visual programming environments existing to date for symbolic music processing (see also [26]). Initiated in 1997, this open source project derived from the PatchWork environment [25] and constitutes a complete visual language including powerful programming features, mostly inherited and adapted from Common Lisp, its underlying implementation language, and CLOS (Common Lisp Object System [24]).

2. BASIC DESCRIPTION

Visual programs in OpenMusic are created in *patch* editors, and are mainly compound of boxes and connections (see Figure 1). Each box represents a functional element in the program: generally, a function (e.g. *om-random*, *repeat-n*, *bpf-sample...* in Figure 1) or a class factory (boxes generating instances of a given class—e.g. the curve at the top or the score object at the bottom of Figure 1). The contents of the factory boxes can be edited thanks to specific graphical editors like score editors, sound editors, break-points function or 2D/3D editors etc., which leaves a significant freedom to the user regarding the algorithmic vs. manual/intuitive parts of his/her work.

The boxes in OpenMusic visual programs have inputs and outputs (represented by small round inlets and outlets respectively at the top and at the bottom of the box icons) which allow to connect them together: at *evaluation*, a box performs a call to its internal functional reference using the

MM'11, November 28–December 1, 2011, Scottsdale, Arizona, USA.

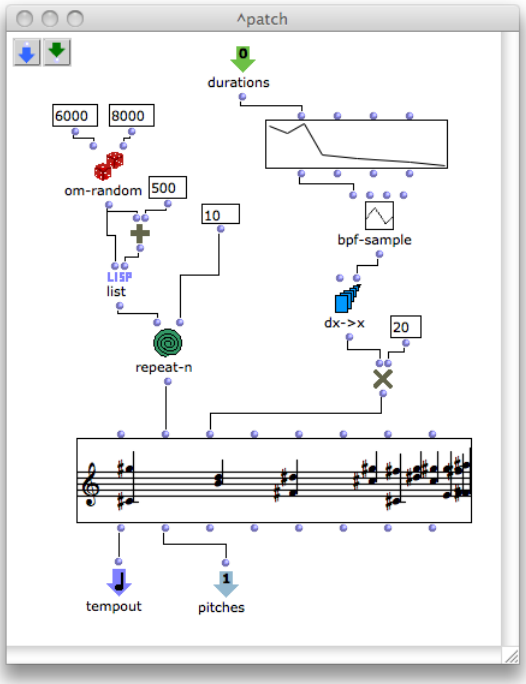

Figure 1: A *patch* or basic visual program editor in OpenMusic.

result of the boxes connected to its inputs as parameters (or arguments). A recursive "bottom-up" function call therefore occurs in the visual program graph, which corresponds to the execution of a program in a very similar way as would be interpreted and evaluated a Lisp expression.[1]

The boxes can actually refer either to in-built OpenMusic functions or classes, or to programming elements (functions, programs, classes) designed by the user graphically or in Lisp. A close and transparent relation exists between the visual and underlying text-based programming environments, which makes it possible to use together any kind of element in visual or Lisp programs.

The musical focus of the environment also led the authors to design advanced programming interfaces including a temporal dimension [1]. In the *maquette* (a "temporal" extension of the OpenMusic *patch*), time is considered as a structural dimension in both the visual program layout and execution, which allows to develop compositional processes integrated in an overall temporal context.

3. VISUAL PROGRAMMING FEATURES

OpenMusic provides a visual semantics implementing most functional and object-oriented features available in Common Lisp and CLOS. Abstraction, application, iterations and control structures are basics of most modern programming languages, although not always straightforward to represent in a visual language. Recursion, higher-level programming are examples of more advanced concepts also implemented in OpenMusic, which proved to be useful and pertinent in compositional situations and musical problems solving. Implementation details on these programming aspects are given in [19].

Visual object-oriented programming is another specificity of OpenMusic [1]. It is possible to define and instantiate classes graphically, but also to benefit from other CLOS features such as multiple inheritance or generic function and method definition, including multiple dispatch and standard method combination systems. The CLOS meta-object protocol (MOP) is another powerful feature making the basic language elements used in the Lisp program design (functions, methods, classes, etc.) instance of meta-object classes which can be manipulated, modified or extended at runtime by the same programs. The OpenMusic visual MOP described in [3] is an original extension of this system to the visual programming language.

The programming features and possibilities mentioned in this section often go far beyond the use of music composers, and extend the scope of the environment to a general new approach to programming using graphical interfaces.

4. A PLATFORM FOR COMPUTER MUSIC RESEARCH

Research in varied computer music areas have been carried out in or using OpenMusic, generally in the context of Master's or PhD theses, or in other types of institutional projects. Music notation and editors [17], quantification and representation of rhythmic structures [23, 5], style modeling and pattern recognition [10, 28], constraint programming and solving systems [31, 34] or sound synthesis and representation in music [6, 14] are example of such areas and projects. OpenMusic is also an important platform for computer-aided music analysis, allowing to carry out experiments and modeling processes leading to a new conception of "computational musicology" [8, 7, 2].

The output of these projects, as well as other more specific works carried out by composers, are generally integrated in the OpenMusic environment or made available to the user community as external, dynamically loadable libraries.

The latest developments in OpenMusic particularly aimed to extend the scope of its applications toward signal processing, for instance with a number of new libraries and tools dedicated to sound analysis, synthesis, and to the manipulation of audio and other low-level description data using the symbolic visual programming framework [16, 13, 18]. Spatial sound and the conception of new ways to represent and generate sound in space and/or using spatial rendering technologies is another currently active area of research and development in the environment [32, 20].

Different works have also been carried out using OpenMusic for extra-musical purposes. A recent example is the *Pixels* project, a library for the generation and processing of pixel arrays combined algorithmically to create pictures and graphics (see Figure 2).[2]

[1]The execution model in OpenMusic is called "demand-driven", as opposed to the "data-driven" model generally implemented in similar visual (graph-based) programming languages.

[2]*Pixels* has been developed and used for *Skyline*, by Shanta Rao (installation with computer-generated video, *Belleville Biennale / Nuit Blanche*, Paris, 2010).

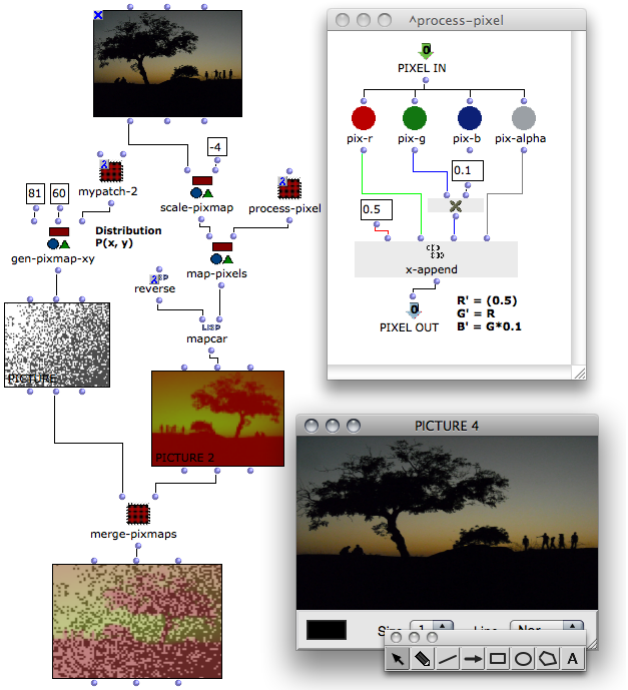

Figure 2: Algorithmic generation and processing of pixel arrays and in OpenMusic with the *Pixels* library (An example of extra-musical application).

5. APPLICATIONS AND USERS

The references provided in the previous section aim to illustrate the diversity of applications of OpenMusic in computer music research. The *OM Composer's Books* [4] can provide a relatively complete and rich overview of real musical applications of the visual programming environment by composers.

OpenMusic is taught in composition classes in many different institution around the world, such as the national conservatoires of Paris (CNSMDP) or Lyon in France, but also different Musikschulen in Germany (Stuttgart, Berlin) and universities in North America (Departments of Music in Columbia, Harvard, Stanford universities, UC San Diego, Université de Montréal, McGill University...) It is also used as a support in computer music or visual programming classes and workshops given at IRCAM or in several Master's program in French universities.

OpenMusic was also used as underlying kernel for the design and development of *Musique Lab 2* [15], an environment dedicated to music education now used and distributed by the French Ministry of Education.

6. REFERENCES

[1] C. Agon. *OpenMusic : Un langage visuel pour la composition musicale assistée par ordinateur*. PhD thesis, Université Pierre et Marie Curie, Paris, France, 1998.

[2] C. Agon, M. Andreatta, G. Assayag, and S. Schaub. Formal Aspects of Iannis Xenakis' "Symbolic Music": A Computer-Aided Exploration of Compositional Processes. *Journal of New Music Research*, 33(2), 2004.

[3] C. Agon and G. Assayag. OM: A Graphical Extension of CLOS using the MOP. In *Proceedings of ICL'03*, New York, USA, 2003.

[4] C. Agon, G. Assayag, and J. Bresson, editors. *The OM Composer's Book (2 volumes)*. Editions Delatour / IRCAM, 2006-2008.

[5] C. Agon, K. Haddad, and G. Assayag. Representation and Rendering of Rhythmic Structures. In *WedelMusic*, Darmstadt, Germany, 2002.

[6] C. Agon, M. Stroppa, and G. Assayag. High Level Musical Control of Sound Synthesis in OpenMusic. In *Proceedings of the International Computer Music Conference*, Berlin, Germany, 2003.

[7] M. Andreatta and C. Agon. Implementing Algebraic Methods in OpenMusic. In *Proceedings of the International Computer Music Conference*, Singaphore, 2003.

[8] M. Andreatta, T. Noll, C. Agon, and G. Assayag. The Geometrical Groove: Rhythmic Canons between Theory, Implementation and Musical Experiments. In *Actes des Journées d'Informatique Musicale*, Bourges, France, 2001.

[9] G. Assayag. Computer Assisted Composition today. In *1st symposium on music and computers*, Corfu, Greece, 1998.

[10] G. Assayag, S. Dubnov, O. Lartillot, and G. Bejerano. Using Machine-Learning Methods for Musical Style Modeling. *Computer*, 36(10), 2003.

[11] G. Assayag and C. Rueda. The Music Representation Project at IRCAM. In *Proceedings of the International Computer Music Conference*, Tokyo, Japan, 1998.

[12] G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue. Computer Assisted Composition at IRCAM: From PatchWork to OpenMusic. *Computer Music Journal*, 23(3), 1999.

[13] J. Bresson. Sound Processing in OpenMusic. In *Proceedings of the International Conference on Digital Audio Effects*, Montréal, QC, Canada, 2006.

[14] J. Bresson. *La synthèse sonore en composition musicale assistée par ordinateur : Modélisation et écriture du son*. PhD thesis, Université Pierre et Marie Curie, Paris, France, 2007.

[15] J. Bresson. ML-Maquette / Musique Lab 2. In *Proceedings of the International Computer Music Conference*, New York City / Stony Brook, USA, 2010.

[16] J. Bresson and C. Agon. Musical Representation of Sound in Computer-Aided Composition : A Visual Programming Framework. *Journal of New Music Research*, 36(4), 2007.

[17] J. Bresson and C. Agon. Scores, Programs and Time Representations: The Sheet Object in OpenMusic. *Computer Music Journal*, 32(4), 2008.

[18] J. Bresson and C. Agon. Processing Sound and Music Description Data Using OpenMusic. In *Proceedings of the International Computer Music Conference*, New York City / Stony Brook, USA, 2010.

[19] J. Bresson, C. Agon, and G. Assayag. Visual Lisp/CLOS Programming in OpenMusic. *Higher-Order and Symbolic Computation*, 22(1), 2009.

[20] J. Bresson and M. Schumacher. Representation and Interchange of Sound Spatialization Data for

Compositional Applications. In *Proceedings of the International Computer Music Conference*, Huddersfield, UK, 2011.

[21] R. B. Dannenberg, P. Desain, and H. Honing. Programming Language Design for Music. In C. Roads, S. T. Pope, A. Piccialli, and G. DePoli, editors, *Musical Signal Processing*. Swets and Zeitlinger, 1997.

[22] R. B. Dannenberg, P. McAvinney, and D. Rubine. Arctic : A Functional Language for Real-Time Systems. *Computer Music Journal*, 10(4), 1986.

[23] O. Delerue, G. Assayag, and C. Agon. Etude et réalisation d'opérateurs rythmiques dans OpenMusic, un environnement de programmation appliqué à la composition musicale. In *Actes des Journées d'Informatique Musicale*, La Londe les Maures, France, 1998.

[24] R. P. Gabriel, J. L. White, and D. G. Bobrow. CLOS: Integration Object-oriented and Functional Programming. *Communications of the ACM*, 34(9), 1991.

[25] M. Laurson and J. Duthen. Patchwork, a Graphic Language in PreForm. In *Proceedings of the International Computer Music Conference*, Ohio State University, USA, 1989.

[26] M. Laurson and M. Kuuskankare. PWGL: A Novel Visual Language Based on Common Lisp, CLOS, and OpenGL. In *Proceedings of the International Computer Music Conference*, Gothenburg, Sweden, 2002.

[27] J. McCartney. Rethinking the Computer Music Language: SuperCollider. *Computer Music Journal*, 26(4), 1996.

[28] B. Meudic. *Détermination automatique de la pulsation de la métrique et des motifs musicaux dans des interprétations à tempo variable d œuvres polyphoniques*. PhD thesis, Université Pierre et Marie Curie, Paris, France, 2004.

[29] M. Puckette. Combining Event and Signal Processing in the MAX Graphical Programming Environment. *Computer Music Journal*, 15(3), 1991.

[30] M. Puckette. Pure Data: Another Integrated Computer Music Environment. In *Proceedings of the Second Intercollege Computer Music Concerts*, Tachikawa, Japan, 1996.

[31] C. Rueda, M. Laurson, G. Bloch, and G. Assayag. Integrating Constraint Programming in Visual Musical Composition Languages. In *Proceedings of the European Conference on Artificial Intelligence*, Brighton, UK, 1998.

[32] M. Schumacher and J. Bresson. Spatial Sound Synthesis in Computer-Aided Composition. *Organised Sound*, 15(3), 2010.

[33] H. Taube. Common Music: A Music Composition Language in Common Lisp and CLOS. *Computer Music Journal*, 15(2), 1991.

[34] C. Truchet, G. Assayag, and P. Codognet. Visual and Adaptive Constraint Programming in Music. In *Proceedings of the International Computer Music Conference*, La Habana, Cuba, 2001.

APPENDIX

A. BUILD AND COMPILATION

OpenMusic is a Lisp-based system which requires an underlying runtime Lisp environment. It currently uses the *LispWorks*[3] Common Lisp implementation, which provides good graphical and GUI toolkits on which most of the low-level graphical aspects of the visual language rely.

The OpenMusic package therefore contains a pre-built executable running on MacOS X or Windows operating systems, *and* the Lisp sources of the software. These sources can easily be loaded in LispWorks (a free—limited—personal edition is available on the LispWorks website). All instructions are detailed in the OpenMusic webpage.[4]

This is generally not necessary, though, since OpenMusic itself embeds a Lisp interpreter and interface, so that is is possible to edit, (re)load and evaluate the sources or additional Lisp code and commands from within the running environment.

The limitations of using the OpenMusic in-built Lisp rather than LispWorks are the impossibility to generate and pack a new executable image, to compile Lisp files (they are only *evaluated*, hence the code is generally less efficient) and the absence of the LispWorks IDE tools. The OpenMusic executable, on the other hand, is easier to launch and to use for standard (non-programmer) users.

B. LICENCE AND DISTRIBUTION

OpenMusic is an open source project which sources are distributed under the GNU Public License (GPL). They can be downloaded from the OpenMusic website and are also distributed with the software itself. Making the sources available was for us an opportunity to raise collaborations and contributions, principally for the realization of specialized packages or libraries. As mentioned in this paper, OpenMusic modules are linked to the distributed sources so that it is possible to track and edit them and therefore to dynamically modify or extend the environment.

IRCAM owns professional LispWorks licenses, which allows us to build and distribute a compiled and packed application to the OpenMusic users. Since the beginning of this year, this released application is available for free and can also be downloaded from the website,[5] which we hope will consolidate and develop our user community.

IRCAM also commercially distributes a set of specialized external libraries for OpenMusic.

C. DOCUMENTATION

A set of documentation resources for OpenMusic is maintained and published online. This documentation features a user manual, a quick-start tutorial including a series of videos, tutorials and a number of additional resources. All pages and documents of interest are accessible from the OpenMusic website.[6]

A developer documentation also allows programmers to get into the OpenMusic architecture and provides the basics of programming in the environment.

[3] http://www.lispworks.com
[4] http://repmus.ircam.fr/openmusic/sources
[5] http://repmus.ircam.fr/openmusic/download
[6] http://repmus.ircam.fr/openmusic/documents