Spatial Structures Programming for Music

Jean Bresson
UMR STMS: Ircam-CNRS-UPMC
Paris, France
Email: jean.bresson@ircam.fr

***Abstract*—We survey works and applications of visual programming for the control of sound spatialization in the OpenMusic computer-aided composition environment. Several salient aspects of the control and spatialization processes are described, such as the high-level description of spatial sound scenes, their unfolding in musical or computational time flows, as well as the concept of spatial sound synthesis.**

I. Introduction

Considering space in music is an long-lasting concern which has been particularly put forward during the last 50 years with the development of computer and audio technologies [19]. After pioneering works on early analogue systems by composers such as K. Stockhausen or E. Varèse, the interest for composers to integrate sound spatialization in their work significantly increased as digital technologies for spatial audio progressed and made it more accessible [27], [30]. With this technology, instrumental music performances can be enhanced by new techniques for sound diffusion in concert halls, and sounds in electro-acoustic works can be spatially composed in real or virtual rooms.

With the term *spatialization* we refer to the localisation and movements of sound sources, but also to their orientation or directivity patterns, to the acoustic properties of a room, or to any other more or less abstract elements related to the diffusion of sound sources and the inclusion of space as a musical parameter in a compositional context [5]. Examples of spatial audio technologies available today range from multi-speakers amplitude panning [33], which can now be controlled at high rate and resolution for large numbers of sound sources and channels, to more advanced systems and hardware equipments such as higher-order ambisonics [14] or wave-field synthesis [6].

Our interest here, however, is not exactly in spatial audio but in its control and integration in compositional frameworks.

Digital audio workstation plug-ins for sound spatialization are commonplace today, and are used to control the virtual localisation of sounds during sound mixing or montages. Powerful spatialization tools are available as well for music and sound processing environments like Max/MSP [21], [34]. Interestingly, recent efforts are also being done to provide high-level interfaces independent from the low-level rendering technologies [18], thereby permitting users to focus on control (and further, compositional) issues. However, systems for creating (composing) spatial structures, possibly used in a subsequent phase to parametrize the aforementioned control tools, are fewer (see for instance [23], [31], [41]).

The works we present take place in the OpenMusic visual programming environment and in the general context of computer-aided music composition. After a quick presentation of this background, we will try to underline the interest and possibilities offered by the integration of sound spatialization in high-level compositional frameworks, and describe some tools and concepts developed recently for this purpose.

II. OpenMusic, Visual Programming and Computer-Aided Music Composition

Visual programming is a relatively widespread approach to computer-aided composition in contemporary music [3]. OpenMusic (OM) is a visual programming language designed for music and used by composers, musicologists or computer music researchers to create or transform musical material [4], [11]. Programs in OM (also called *patches*) are represented and manipulated as directed acyclic graphs made of boxes and connections. A patch represents a functional expression which can be constructed and evaluated on-demand in order to produce or transform musical material (Figures 1 and 3 are examples of OM patches). OM is implemented in Common Lisp and OM patches have therefore a direct correspondence to Lisp expressions. Conversely, any Lisp code or process can be integrated in an OM patch in a straightforward way. The main programming paradigm in OM is therefore functional (most of the functional constructs available in Lisp can be implemented in visual programs, such as higher-order, recursive function, etc. [10]) although object-oriented features are also available in the visual language.

OM also contains a number of domain-specific data structures and associated graphical editors, allowing to visualize and manipulate the data involved in the visual programs and thereby providing an important input and feedback between the user/programmer and the program being created. It may be important, however, to differentiate functional environments such as OM from real-time (highly interactive) environments more commonly used in music (e.g. [32]). In OM in principle, no temporal control or external clock influences the patch execution, which is a static (declarative), and globally "out-of-time" description of a musical structure, generally including time as an autonomous and unconstrained dimension.

Advanced formalisms and compositional processes can therefore be carried out in OM in order to generate material or experiment with computation and computer modelling in diverse musical contexts [1], [9].

III. Spatial Concepts in Computer-Aided Composition

There exist several and varied ways in which spatial attributes can be linked to musical aspects of a work in a musical programming environment like OpenMusic. A first example, which actually does not deal with spatialization, is for instance given in [20] where the author uses spatial relations coming from the transcription and processing of architectural sketches and maps to produce musical parameters.

Conversely, and more related to our present concern, any kind of musical or extra-musical data can be processed in OpenMusic and generate attributes or parameters for a spatialization process. This approach has been experimented in the late nineties for instance to control the MusicSpace constraint-based spatialization system [16] with the *OpenSpace* project [17], or the Ircam *Spatialisateur* [21] with *OMSpat* [26]. The main interest here is that spatialization parameters (mostly, the position and movements of the sound sources in space) can be precisely controlled and set in accordance and relation to the other parameters of a compositional process, and therefore of the created musical material. A pioneering work in this respect, carried out in OpenMusic, was B. Ferneyhough's piece *Stellæ for failed times* [25].

Usually two different approaches can be observed in the control of sound spatialization. The first one focuses on the sound sources distribution among the different channels or speakers. This approach is adapted (and generally specific) to particular rooms and speaker set-ups: the composer "thinks" in terms of these speakers and devises how sound sources are allocated, and possibly move among them [40]. A second approach rather focuses on perception, that is, on positions where sound sources shall be located, and spatialization processors work at rendering this perceptual idea in the actual room and with a given speakers set-up. In this case, composers think in term of absolute positions and trajectories. In principle (but rarely in fact) this approach in the compositional domain can be independent of the room and set-up. The recent works carried out in OM, presented in the following sections, mostly focus on the latter approach, describing spatial sound scenes and processes in terms of positions and trajectories (although they do not invalidate the former one).[1] This idea has also been extended to the micro-level of sounds with the concept of *spatial sound synthesis* described in Section VIII.

IV. Spatial Scene Description in Visual Programs

Spatial sound scenes[2] are represented in OM visual programs by matrices, where one dimension stands for the different sources, and the orthogonal dimension represents the parameters used for a given spatialization process (these parameters may change depending on the application and spatialization technique, although they often include at least 2D or 3D position information). These matrices are included and instantiated in functional programs as described in section II (see Figure 1–a). The generative processes can be arbitrarily complex and involve varied formalisms and programming concepts (iterative processing, randomization, higher-order, etc.—see Figure 1–b).

[1]These two approaches are not completely incompatible, and most observed applications and practices actually partly involve both the "abstract" spatial thinking and a consideration of particular targeted spatialization systems.

[2]We call "spatial sound scene" a set of sound sources associated to positions, trajectories, directivity patterns and other space- or acoustic-related features.

V. Time and Trajectories

Time is a fundamental parameter in music and therefore needs a specific consideration in spatialization processes as well. Every parameter is subject to possible changes or continuous evolution. In particular, positions of the sound sources are often rather considered in terms of such evolution: in this case, spatial coordinates will not be represented by values but with sampled curves (determined in the generative processes either point-wise or as functional specification), or contained at a higher level as "trajectory" objects.

The *trajectory* objects in OM aim at providing self-contained representations for the evolution of spatial (3D) positions, as well as visualization and editing features. Each point of a trajectory has 3 Cartesian coordinates and an optional time-tag allowing to map the position to a temporal referential. These time-tags do not necessarily represent absolute time values and can possibly be modified (or generated) by additional processing (scaling, delays, sampling...) The trajectories generated in Figure 1–b, for instance, do not include specific time information and will be unfolded depending on the onsets and durations specified for their respective sources in the matrix scene representation. The trajectories also have some lazy properties through a number of parameters allowing to define how time unfolding is to be computed, given the explicit timed-points and some rules, for instance respecting a constant speed (hence depending on the distance between successive points), or assuming a constant time interval between the same successive points (and independently of their relative distance). Late sampling (or re-sampling) is also often useful in order to keep compositional specifications relatively small and easily controllable (using reduced sets of "control points") and to convert them to smoother movements and evolutions at rendering time.[3]

VI. Control of Spatial Rendering: OM-Spat

Due to a certain level of abstraction maintained in the specification of the spatial sound scenes during the early compositional stages, in principle no excessive specific knowledge is required about the rendering process, which is most often carried out using external software systems.[4]

The SDIF file format [39] is used as a standard to encode the spatial scene descriptions created in OM and transfer them

[3]The *path* object in [23] provides similar features for the specification of trajectories in Common Lisp Music [36].

[4]This "stratified" vision [29] is not always completely realistic in fact, since elements of the rendering techniques often need to be specified and incorporated at the control level.

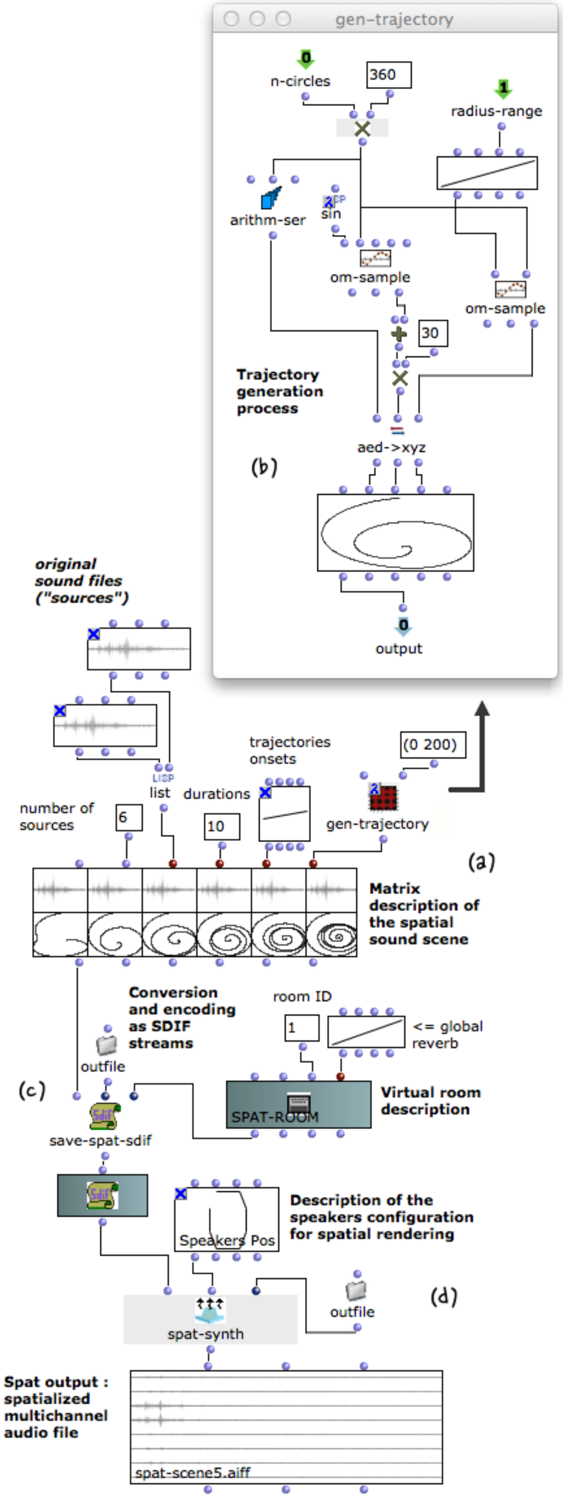

Fig. 1. Representation and rendering of a spatial sound scene in OpenMusic. The matrix representation of the scene (a) is instantiated from a set of sound sources and spatialization parameters provided either "literally" (e.g. durations, onsets...) or as functional specification (e.g. the trajectory generation process (b) provided as higher-order function). It is eventually encoded as an SDIF file (c), and here rendered to a multi-channel audio file (d).

to external software and rendering environments.[5] SDIF is a "polyphonic", stream-based format allowing to encode timed frames containing any kind of data structured as matrices. Each matrix (identified by a type) contains a number of components described at the time of the containing frame by a set of field values (e.g. x, y, z in the case of spatial positions). Several matrices of different types can coexist in common frames, and several frame streams can coexist in a same SDIF description file. SDIF is therefore quite well adapted to render the OM spatial sound scenes (see Figure 1–c): source descriptions can be can be interpreted in terms of such flat and precisely timed streams describing the evolution of their different spatial parameters (see [12]).

Specific matrix types have been defined in SDIF corresponding to the main control parameters of the Ircam *Spatialisateur*. A command line rendering tool developed from this software (*Spat renderer*[6]) allows to generate spatialized multichannel audio files from the SDIF descriptions created in OM (see Figure 1–d).

As shown in Figure 1, room descriptions can also be addressed in the compositional and spatialization processes: the *Spatialisateur* provides powerful perceptual room modelling features, to which can be "attached" the different sources.[7][8]

Both SDIF file conversion and *Spat* rendering features described in this section are available in the OpenMusic *OM-Spat* library.

VII. Data Streaming and Real-Time Integration

In the present musical and technological context, sound spatialization is either performed as a completely off-line process (most often, in the case of pure electronic music), or in real time during concerts and performances. In the latter case a lower degree of abstraction and complexity is affordable for the description, control and rendering of spatial sounds.

The streaming of the SDIF-formatted data generated in OM (hence already "flattened", timed and ordered) is envisaged as one solution in the integration of spatial data, prepared beforehand in the computer-aided composition environment, in external real-time processes.

Spat-SDIF-Player is a standalone application developed for this purpose (see Figure 2–a). Implemented in Max/MSP [32] using the MuBu buffering library [35], this application loads SDIF spatial description files and provides standard playback controls as well as additional features such as stream (i.e. source) selection, looping or speed control. Note that this player does not produce any sound but broadcasts control messages via UDP. The messages are formatted in OSC [44] and respect the SpatDIF specification [28] so that any

[5] SDIF is also used to interchange data between the OM compositional environment and external sound rendering or other lower-level processing tools [8].

[6] *Spat renderer* by Thibaut Carpentier, Ircam.

[7] Several simultaneous "virtual rooms" can coexist and be superimposed in a same spatial sound scene, attached to different sound sources.

[8] In the SDIF description, separate frame streams are used for the room parameters and their possible evolution through time, and a "room ID" attribute allow to match the sources to one of the (possibly multiple) rooms.

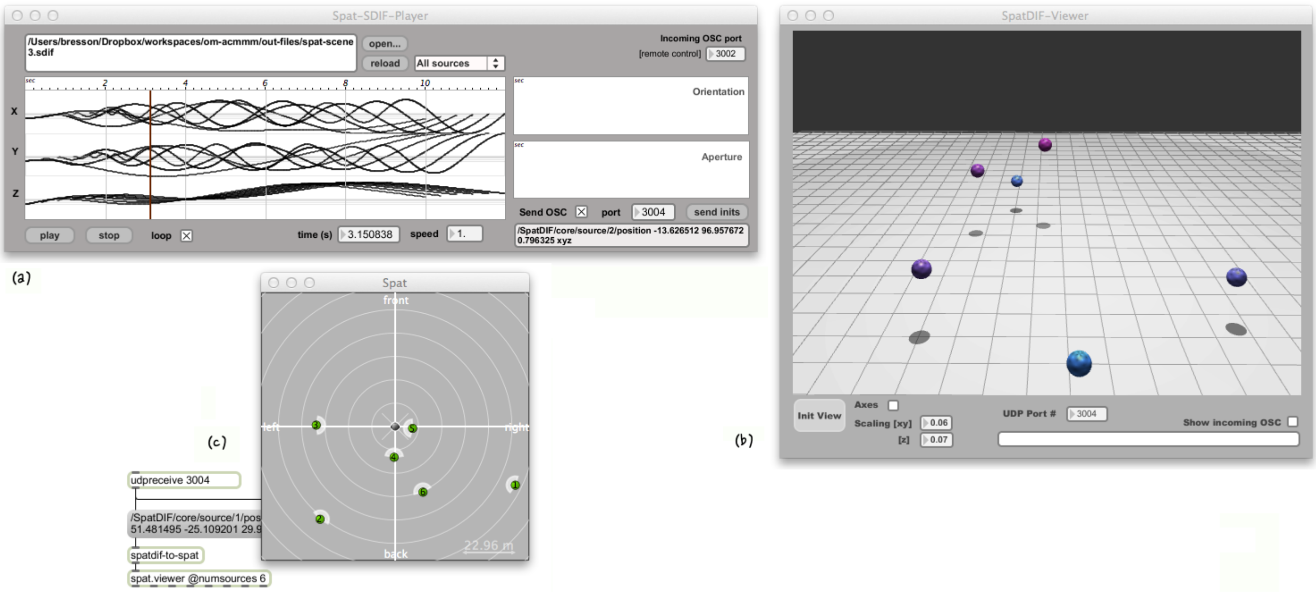

Fig. 2. Streaming of SDIF spatial descriptions using *Spat-SDIF-Player* (a). Control messages are be received and interpreted by the *SpatDIF-Viewer* 3D renderer (b) and by the *Spat* library tools in Max/MSP (c).

rendering system compliant with this format can interpret and eventually render the spatial descriptions accordingly. Figure 2 shows two such examples: one (*SpatDIF-Viewer*, Figure 2–b) is a visualization tool rendering SpatDIF messages in a 3D display, and the second one is a Max/MSP patch receiving the same data and converting them into control messages for the *spat.oper* object in the *Spat 4* library (Figure 2–c).

This networking protocol provides interesting flexibility and inter-operability between applications and tools for spatialization. It is however limited as messages may be numerous (for instance in case of high sample rates and/or when numerous spatialization parameters are involved), and are necessarily sent sequentially. In these cases, the simultaneous control of multiple sound sources (which is not a limitation at the level of the specification of the spatial sound scenes in OM, for instance) can raise synchronization issues, delays, or other undesired behaviours.

VIII. Spatial Sound Synthesis

An interesting concept developed and implemented in the context of these works on sound spatialization is the one of *spatial sound synthesis* [37]. Generalizing similar ideas developed with the spatialization of spectral or sinusoidal decompositions [22], [42], or with granular and swarm-based models [24], [43], the concept of spatial sound synthesis consists in addressing spatial issues at the very same level as the sound synthesis itself.

Sound synthesis processes can be represented and controlled in OpenMusic using matrix objects (in a similar way as is done with spatial sound scenes—see Section IV). In the *OMChroma* library [2], these matrices describe parameter values corresponding to a given sound synthesis "instrument", for a number of synthesis components (or virtual instances of this instrument). This representation in OM visual programs is eventually converted to textual code compiled into a sound file by the Csound synthesizer [7]. Typically, from a simple digital oscillator implemented in Csound and used as the "synthesis instrument", an *OMChroma* matrix would allow to describe and store the frequency, amplitude, duration and possible additional parameters of this instrument for an arbitrary (and possibly important) number of components, hence implementing a powerful "additive synthesis" control and rendering process (see Figure 3–a).

Synthesis processes in *OMChroma* can be extended to sound spatialization by devising appropriate Csound instruments considering sound sources as one of the inputs of the synthesis, and the multichannel output as its result. As with the matrices described in section IV, spatialization processes developed using *OMChroma* can therefore make for unlimited polyphony (number of sources), and provide an important diversity in the rendering techniques thanks to the numerous spatialization tools and programming possibilities available in the Csound language. The *OMPrisma* library, developed by Marlon Schumacher at CIRMMT (McGill University) is an extension of *OMChroma* providing a rich set of such spatialization options to be used in complement (or combination—see below) to the *OMChroma* synthesis objects [38].

More interestingly, it is possible to develop both the synthesis and spatialization processes in a same DSP instrument, and thereby to build arbitrarily complex sound structures including spatial descriptions for every single component of the sound synthesis. In the "additive synthesis" example given above, for instance, one could imagine to assign a specific trajectory to every single "partial" of the sound (or atomic sinusoidal component, hence considered as an individual sound source). Unique and innovative sound textures can therefore be designed establishing strong relations between the sound synthesis parameters and corresponding spatial morphologies.

In order to allow for unconstrained combination between the different sound synthesis techniques provided in *OMChroma* with the spatial rendering tools implemented in *OMPrisma*, a "merging" protocol has been defined between synthesis and spatialization objects (and subsequently between corresponding Csound source code). This dynamic integration provides a high degree of flexibility for the experimentation on the different spatialization and synthesis techniques combinations; Any sound synthesis object can be connected and merged to any spatialization object in order to perform a spatial sound synthesis process integrating both parameters and attributes (see Figure 3).

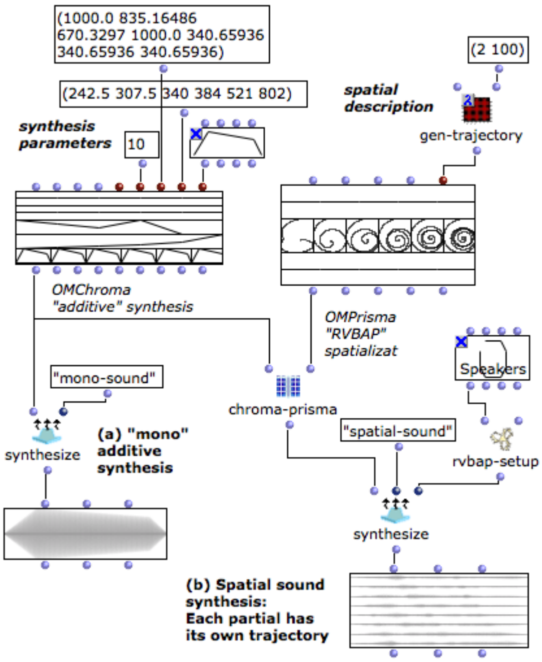

Fig. 3. Spatial sound synthesis in OM with *OMChroma/OMPrisma*.

IX. Conclusion: Toward Extended Computational Paradigms for Spatialization ?

Sound spatialization is now a widespread concern in electroacoustic and contemporary music, and a major issue in computer music research and development. The technological advanced of the recent years opened a world of possibilities, and many musical venues and research institutions are now equipped with high-quality spatial rendering facilities.[9]

Compositional concerns and research currently emerge on top of these technologies, as show for instance the different tools and projects presented in this article, in OpenMusic and more generally in the computer-aided composition domain. An interesting point here is the fact that the spatialization processes can now be thought and carried out inside or in a close relation to the compositional processes and corresponding formalized approaches.

In this regard, the connection to real-time frameworks is probably still a critical—and interesting—aspect: while most of the signal processing tools render spatial sounds in real-time, their integration with compositional inputs and specifications is not straightforward (the works presented in Section VII are preliminary attempts in this direction). More and more frequently as high-resolution systems get developed and available, the integration of relatively complex sound structures and spatial control can raise computational issues which can be solved by merging off-line generation of compositional inputs, using dedicated computational paradigms and environments, to reactive (real-time) spatial sound rendering systems.

In this respect and in the continuation of the different past projects dealing with spatial concerns in computer-aided composition, interesting new directions could involve extended constraint-based approaches to spatial structures, or specific formal frameworks such as qualitative spatial reasoning [13]. By including spatial concepts in the programs generating and processing data, spatial computing [15] could also be a promising approach to the control of spatialization processes and to cope with the general issue of composing spatial structures using computer processes.

Acknowledgements

Significant parts of the works presented in this paper have been carried out in collaboration with Marlon Schumacher at Ircam and at CIRMMT, McGill University. The author would also like to acknowledge Thibaut Carpentier for his collaboration on the *Spat* rendering and on the definition the interchange formats used in this project.

References

[1] C. Agon, G. Assayag and J. Bresson (Eds.), *The OM Composers Book. 1*, Editions Delatour/Ircam, 2006.

[2] C. Agon, J. Bresson and M. Stroppa, "OMChroma: Compositional Control of Sound Synthesis." *Computer Music Journal*, 35(2), 2011.

[3] G. Assayag, "Visual Programming in Music." *Proceedings of the International Computer Music Conference*, Banff Canada, 1995.

[4] 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.

[5] M. A.J. Baalman, "Spatial Composition Techniques and Sound Spatialization Technologies." *Organised Sound*, 15(3), 2010.

[6] A. J. Berkhout, D. de Vries, D. and P. Vogel, "Acoustic Control by Wave Field Synthesis." *Journal of the Acoustical Society of America*, 93, 1993.

[7] R. Boulanger (Ed.) *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing,and Programming*, The MIT Press, 2000.

[8] J. Bresson and C. Agon, "SDIF Sound Description Data Representation and Manipulation in Computer-Assisted Composition." *Proceedings of the International Computer Music Conference*, Miami, USA, 2004.

[9] J. Bresson, C. Agon and G. Assayag (Eds.), *The OM Composers Book. 2*, Editions Delatour/Ircam, 2008.

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

[9]Notable example of concert venues and research facilities providing high-quality spatialization environments include Ircam's *Espace de projection* with its WFS and higher-order ambisonics systems, the *Acousmonium* at the GRM (Paris), the Birmingham ElectroAcoustic Sound Theatre (*BEAST*), the *Allosphere* at UC Santa Barbara, the ZKM *Klangdom* in Karlsruhe, the *Sonic Lab* in Belfast, and many others.

[11] J. Bresson, C. Agon and G. Assayag, “OpenMusic. Visual Programming Environment for Music Composition, Analysis and Research.” *Proceedings of ACM MultiMedia*, Scottsdale, USA, 2011.
[12] J. Bresson and M. Schumacher, “Representation and Interchange of Sound Spatialization Data for Compositional Applications.” *Proceedings of the Inter- national Computer Music Conference*, Huddersfield, UK, 2011.
[13] A. G. Cohn and J. Renz, “Qualitative Spatial Reasoning,” in F. Harmelen, V. Lifschitz and B. Porter (Eds.), *Handbook of Knowledge Representation*, Elsevier, 2007.
[14] J. Daniel, *Représentation de champs acoustiques, applications à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimedia*, PhD Thesis, Université Pierre et Marie Curie, Paris 6, France.
[15] A. De Hon, J.-L. Giavitto and F. Gruau, *Computing Media and Languages for Space-Oriented Computation*, Dagsthul Seminar Proceedings, Dagsthul, 2006.
[16] O. Delerue, *Spatialisation du son et programmation par contraintes : Le système MusicSpace*. PhD Thesis, Université Pierre et Marie Curie, Paris 6, France, 2004.
[17] O. Delerue and C. Agon, “OpenMusic + MusicSpace = OpenSpace.” *Actes des Journées d’Informatique Musicale*, Issy-les-moulineaux, France, 1999.
[18] M. Geier, M., J. Ahrens and S. Spors, “The SoundScape Renderer: A Unified Spatial Audio Reproduction Framework for Arbitrary Rendering Methods.” *AES* 124^{th} *Convention*, Amsterdam, The Netherlands, 2008.
[19] M. A. Harley, *Space and Spatialization in Contemporary Music: History and Analysis, Ideas and Implementations.* PhD Thesis, McGill University, Montreal, Canada, 1994.
[20] C. Jaksjø, “Architecture as Virtual Music (Re-Actualizing *Zonnestraal*),” in J. Bresson, C. Agon and G. Assayag (Eds.), *The OM Composers Book. 2*, Editions Delatour/Ircam, 2008.
[21] L.-M. Jot and O. Warusfel, “A Real-Time Spatial Sound Processor for Music and Virtual Reality Applications.” *Proceedings of International Computer Music Conference*, Banff, Canada, 1995.
[22] D. Kim-Boyle, “Spectral Spatialization - An Overview.” *Proceedings of the International Computer Music Conference*, Belfast, Ireland, 2008.
[23] F. Lopez-Lezcano, “A Dynamic Spatial Locator ugen for CLM.” *Proceedings of the Sound and Music Computing Conference*, Berlin, Germany, 2008.
[24] A. McLeran, C. Roads, B. L. Sturm, J. J. Shynk, “Granular Sound Spatialization Using Dictionary-Based Methods.” *Proceedings of the Sound and Music Computing Conference*, Berlin, Germany, 2008.
[25] G. Nouno, “Some Considerations on Brian Ferneyhough’s Musical Language through His Use of CAC. Part II – Spatialization as a Musical Parameter,” in J. Bresson, C. Agon and G. Assayag (Eds.), *The OM Composers Book. 2*, Editions Delatour/Ircam, 2008.
[26] G. Nouno and C. Agon, “Contrôle de la spatialisation comme paramètre musical.” *Actes des Journées dInformatique Musicale*, Marseille, France, 2002.
[27] F. Otondo, “Contemporary Trends in the Use of Space in Electroacoustic Music.” *Organised Sound*, 13(1), 2008.
[28] N. Peters, S. Ferguson and S. McAdams, “Towards a Spatial Sound Description Interchange Format (SpatDIF).” *Canadian Acoustics*, 35(3), 2007.
[29] N. Peters, T. Lossius, J. Schacher, P. Baltazar, C. Bascou and T. Place, “A Stratified Approach for Sound Spatialization.” *Proceedings of the Sound and Music Computing Conference*, Porto, Portugal, 2009.
[30] N. Peters, G. Marentakis, S. McAdams, “Current Technologies and Compositional Practices for Spatialization: A Qualitative and Quantitative Analysis.” *Computer Music Journal*, 35(1), 2011.
[31] L. Pottier, “Dynamical Spatialisation of sound. HOLOPHON: a graphical and algorithmical editor for $\Sigma 1$. *Proceedings of the International Conference on Digital Audio Effects*, Barcelona, Spain, 2008.
[32] M. Puckette, “Combining Event and Signal in the MAX Graphical Programming Environment.” *Computer Music Journal*, 15(3), 1991.
[33] V. Pulkki, “Virtual sound source positioning using vector base amplitude panning.” *Journal of the Audio Engineering Society*, 45(6), 1997.
[34] J. C. Schacher and P. Kocher, “Ambisonics spatialisation Tools for Max/MSP.” *Proceedings of the International Computer Music Conference*, New Orleans, USA, 2006.
[35] N. Schnell, A. Röbel, D. Schwarz, G. Peeters and R. Borghesi, “MuBu & Friends - Assembling Tools for Content Based Real-Time Interactive Audio Processing in Max/MSP.” *Proceedings of the International Computer Music Conference*, Montreal, QC, Canada, 2009.
[36] B. Schottstaedt, “CLM: Music V Meets Common Lisp.” *Computer Music Journal*, 18(2), 1994.
[37] M. Schumacher and J. Bresson, “Spatial Sound Synthesis in Computer-Aided Composition.” *Organised Sound*, 15(3), 2010.
[38] M. Schumacher and J. Bresson, “Compositional Control of Periphonic Sound Spatialization.” *2nd International Symposium on Ambisonics and Spherical Acoustics*, Paris, France, 2010.
[39] D. Schwartz and M. Wright, “Extensions and Applications of the SDIF Sound Description Interchange Format.” *Proceedings of the International Computer Music Conference*, Berlin, Germany, 2000.
[40] E. Stefani and K. Lauke, “Music, Space and Theatre: Site-Specific Approaches to Multichannel Spatialization.” *Organised Sound*, 15(3), 2010.
[41] T. Todoroff, C. Traube and J.M. Ledent, “NeXTSTEP Graphical Interfaces to Control Sound Processing and Spatialization Instruments.” *Proceedings of the International Computer Music Conference*, Thessaloniki, Greece, 1997.
[42] D. Topper, M. Burtner and S. Serafin, “Spatio-Operational Spectral (S.O.S.) Synthesis.” *Proceedings of the International Conference on Digital Audio Effects*, Hamburg, Germany, 2002.
[43] S. Wilson, “Spatial Swarm Granulation.” *Proceedings of the International Computer Music Conference*, Belfast, Ireland, 2008.
[44] M. Wright, “Open Sound Control: An Enabling Technology for Musical Networking.” *Organised Sound*, 10(3), 2005.