MusicSpace: a Constraint-Based Control System for Music Spatialization

François **Pachet**, Olivier **Delerue**
SONY CSL Paris, 6, rue Amyot 75005, Paris, FRANCE
Tel: (33) 1 44 08 05 16, Fax: (33) 1 45 87 87 50, E-mail: pachet@csl.sony.fr

Abstract

Common graphical interfaces for 3D sound spatialization systems are usually based on a direct mapping of sound sources to icons. The result is that only one sound source can be moved at a time by the user. To make these interfaces more useable, we introduce *constraints*, as relations between sound sources which should always be satisfied. These constraints are enforced by a constraint solver which operates in real time. We illustrate the system - MusicSpace - on a number of situations like listening, mixing and composing.

1. Music Spatialization

Music spatialization has long been an intensive object of study in computer music research. Most of the work so far has concentrated in building software to simulate acoustic environments for existing sound signals. These techniques typically exploit difference of amplitude in sound channels, delays between sound channels to account for interaural distances, and sound filtering techniques such as reverberation to recreate impressions of distance (e.g. [4]). These spatialization techniques are mostly used for building virtual reality environments, such as [2], [5]. However, letting users change spatialization arbitrarily induces the risk that the original properties of the configuration of sound sources are no longer preserved. We propose a system in which user may change the positions of sound sources, while ensuring that spatializations are always "correct" in some precisely defined sense.

2. MusicSpace

MusicSpace is not a spatialization system *per se*, but rather an interface for producing high level commands to a spatializer. The basic idea in MusicSpace is to represent graphically sound sources in a window, as well as an avatar that represents the listener itself. In this window, the user may either move its avatar around, or move the instruments icons. The relative position of sound sources to the listener's avatar determine the overall mixing of the music, according to simple geometrical rules mapping distances to volume and panoramic controls (see Figure 1). The real time mixing of sound sources is then performed by sending appropriate commands from MusicSpace, to whatever spatialization system is connected to it, such as a mixing console, a Midi Spatializer, or a more sophisticated spatialization system such as [4].

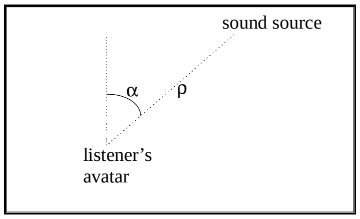

Figure 1. Volume of sound_sourcei = f(distance(graphical-objecti, listener_avatar)). f is a function mapping distance to Midi volume (from 0 to 127). Stereo position of sound source i = g(angle(graphical_Objecti, listener_avatar)), where angle is computed relatively to the vertical segment crossing the listener's avatar, and g is a function mapping angles to Midi panoramic positions (0 to 127).

In this context, MusicSpace is seen as a command generator for an arbitrary spatialization system (see Figure 2).

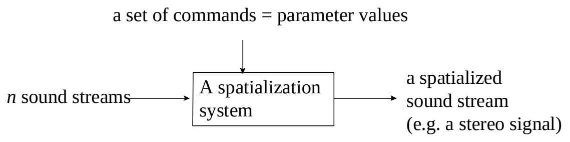
a set of commands = parameter values
n sound streams
A spatialization system
a spatialized sound stream (e.g. a stereo signal)

Figure 2. A spatializer module.

3. Mixing Consistency

The problem with allowing users to change the configuration of sound sources, and hence, the mixing, is that they do not have the knowledge required to produce coherent, nice-sounding mixings. Indeed, the knowledge of the sound engineer is difficult to explicit and to represent. Its basic actions are actions on controls such as faders and knobs. However, mixing also involves higher level actions that can be defined as compositions of atomic actions. For instance, sound engineers may want to ensure that the overall energy level of the recording always lies between reasonable boundaries. Conversely, several sound sources may be logically dependent. For instance, the rhythm section may consist in the bass track, the guitar track and the drum track. Another typical mixing action is to assign boundaries to instruments or groups of instruments so that they always remain within a given spatial range. The consequence of these actions is that sound levels are not set *independently of each another*. Typically, when a fader is raised, another one, (or a group of other faders) will be lowered.

We propose to encode this type of knowledge on sound spatialization as *constraints*, which are interpreted in real time by an efficient constraint propagation algorithm, integrated in MusicSpace.

3.1 Constraints for Interactive Systems

Constraints are relations that should always be satisfied. Constraints are stated declaratively by the designer, thereby avoiding him to program complex algorithms. Constraint propagation algorithms are particularly relevant for building reactive systems typically for layout management of graphical interfaces [3].

3.2 Constraints and Mixing Consistency

We defined a set of constraints appropriate for specifying "interesting" relations between sound sources. Each sound source is represented by a point, i.e. two integer variables (one for each coordinate): $p_i = \{x_i, y_i\}$ with $x_i, y_i \in [1, 1000]$ (a typical screen). An additional variable l represents the position of the listener's avatar, itself consisting of two integer variables: $l = \{x_l, y_l\}$ with $x_l, y_l \in [1, 1000]$.

Most of the constraints on mixing involve a collection of sound sources and the listener. We describe here the most useful ones.

- Constant Energy Level

This constraint states that the energy level between several sound sources should be kept constant. According to our model of sound mixing, this constraint can be stated between variables *pi, i = 1, .., n* as follows: $\prod_{i=1}^{n} \|p_i - l\| = Cte$. Intuitively, it means that when one source is moved toward the listener, the other sources should be "pushed away", and vice-versa. The constant value on the right-hand side of the constraint is determined by the current values of p_i and l when the constraint is set. Note that this constraint is non linear, and not functional (except in the case of two sources).

- Constant Angular Offset

This constraint is the angular equivalent of the preceding one. It expresses that the spatial configuration of sound sources should be preserved, i.e. that the angle between two objects and the listener should remain constant. It can be stated between variables p_1 and p_2 as: $(p_1, \widehat{l}, p_2) = Cte$. It is easily generalized to a collection of objects $p_1, \ldots, p_i \ldots, p_n$.

- Constant Distance Ratio

The constraint states that two or more objects should remain in a constant distance ratio to the listener:

$$\|p_1 - l\| = \alpha_{1,2} \|p_2 - l\|$$

- Radial Limits of Sound Sources

This constraint allows to impose radial limits in the possible regions of sound sources. These limits are defined by circles whose center is the listener's avatar (see **Erreur ! Source du renvoi introuvable.**).

$\|p_i - l\| \geq \alpha_{\inf}$ (lower limit), $\|p_i - l\| \leq \alpha_{\sup}$ (upper limit)

- Grouping constraint

This constraint states that a set of n sound sources should remain grouped, i.e. that the distances between the objects should remain constant (independently of the listener's avatar position):

$\forall i, j \leq n : \left(x_i - x_j\right) = Ctx_{i,j}$ and $\left(y_i - y_j\right) = Cty_{i,j}$

Other typical constraints include symbolic constraints, holding on non geographical variables. For instance, an "Incompatibility constraint" imposes that only one source should be audible at a time: the closest source only is heard, the others are muted. Another complex constraint is the "Equalizing constraint", which states that the frequency ratio of the overall mixing should remain within the range of an equalizer. For instance, the global frequency spectrum of the sound should be flat.

3.3 Constraint algorithm

The examples of constraints given above show that the constraints have the following properties:

- the constraints are not linear. For instance, the constant energy level (between two or more sources) is not linear. This prohibits the use of simplex-derived algorithms.
- The constraints are not all functional. For instance, geometrical limits of sound sources are typically inequality constraints.
- The constraints quickly induce cycles. For instance, a simple configuration with two sources linked by a constant energy level constraint and a constant angular offset constraint already yields a cyclic constraint graph.

There is no general algorithm, to our knowledge, which handles non linear, non functional constraints with cycles. We designed a propagation algorithm which implements only a part of our requirements, but with predictable and reactive behavior. This algorithm is based on a simple propagation scheme, and allows to handle functional constraints, inequality constraints. It handles cycles simply by checking conflicts. An important property of the algorithm is that new constraint classes may be added easily, by defining the set of propagation procedures ([6]).

3.4 The interface

The interface for setting constraints is straightforward: each constraint is represented by a button, and constraints are set by first selecting the graphical objects to be constrained, and then clicking on the appropriate constraint. Constraints themselves are represented by a small ball linked to the constrained objects by lines.

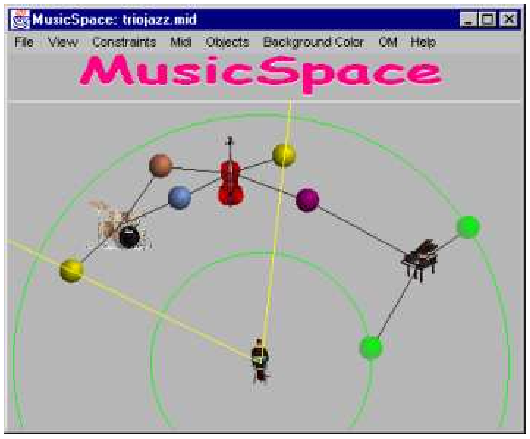

Figure 3. The MusicSpace interface for setting constraints.

Figure 3 displays a typical configuration of sound source for a Jazz trio. The following constraints have been set:

- The bass and drum sound sources are linked by a "constant distance ratio" constraint, which ensures that they remain grouped, distance wise.
- The piano is linked with the rhythm section by a "balance" constraint. This ensures that the total level between the piano and the rhythms section is constant.
- The piano is limited in its movement by a "distance max" and a "distance min" constraint. This ensures that the piano is always heard.
- The drum is forced to remain in an angular area by two "angle constraints". This ensures that the drum is always more or less in the middle of the panoramic range.

Starting from the initial situation of Figure 3, the user moves the piano closer to his avatar. The constraint system is then triggered, and the other sound sources are moved to satisfy the constraint set.

4. Applications

The core MusicSpace system consists of a graphical interface for representing sound sources, and a library of constraints for establishing relations between sources, through graphical links (represented as balls of various colors). This basic system is general enough to be applied in a number of situations where high level user actions may be transformed into sets of lower levels parameter settings. We review here some of the situations where MusicSpace has been applied successfully.

4.1 Interactive listening

MusicSpace is used primarily as a player for midi files. In this scheme Sound sources represent tracks of a midifile, hence instruments. The system parses the midi file, recognizes its tracks and reorganizes it if necessary so that each track represents a different instrument. The user can then set constraints on the instruments (as illustrated in Figure 3), and play the file, while moving instruments.

4.2 Interface for mixing console

MusicSpace has been used for controlling various spatialization systems, such as Ircam's spat, as well as mixing consoles, such as the Yamaha O2R. In this last, scheme, sound sources represents tracks of the console. Constraints allows to produce mixing in real time, which are impossible to produce by hand. In the case of Ircam's spat, specific constraints allow to set relations holding on other parameters than distance and pan, such as directivity (see Figures 4 and 5).

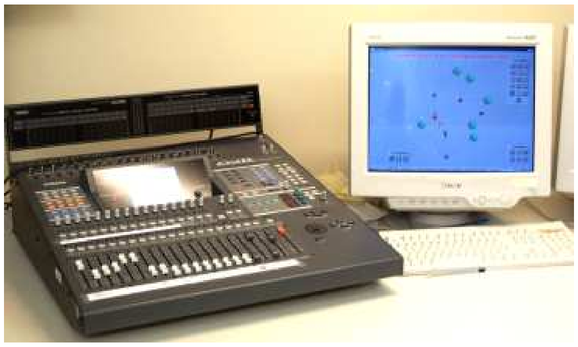

Figure 4. MusicSpace controlling tracks 1 to 6 of an O2R mixing console

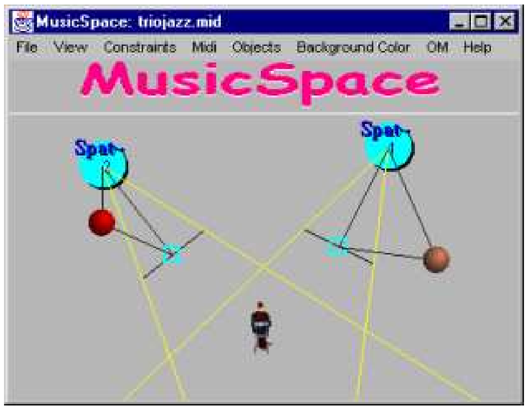

Figure 5. MusicSpace controlling two modules of the Ircam's Spatialisateur. Each module can define its own orientation ; the way the orientation evolves depends on its corresponding source object and the constraint that is set in between.

4.3 Composition tool

MusicSpace has been used for producing music. In this case, sound sources represent autonomous *musical processes*. These processes are defined using a composition languages. We conducted two such experiments. One with the Lisp-based OpenMusic language [1]; another one with the Java-based MusES language. These languages allow to create complex musical processes, and endow them with spatialization capabilities in an homogeneous way. For instance, we show in Figure 6 a patch in OpenMusic that generates a set of commands for MusicSpace. Connections between OpenMusic and MusicSpace are established via the Midi interface.

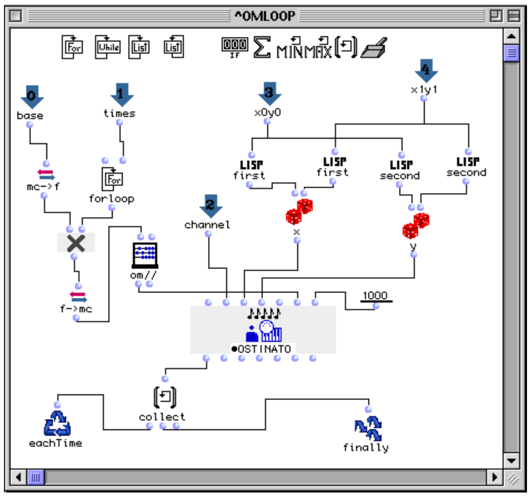

Figure 6. A patch in OpenMusic delivering complex data for MusicSpace.

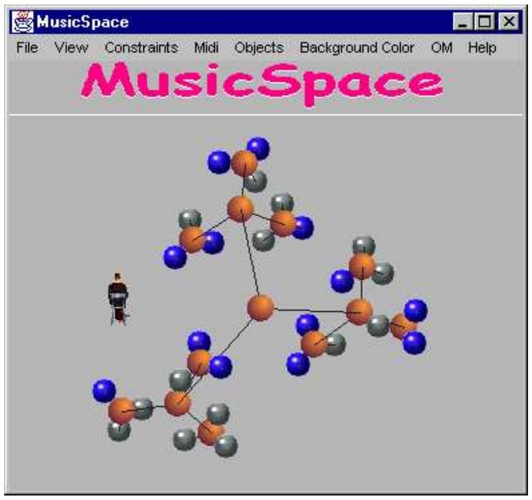

Figure 7. Result in MusicSpace of the evaluation of the patch described in Figure 6.

Figure 7 shows the result of the evaluation of the previous patch in MusicSpace : a set of musical objects and constraints arranged according to a fractal algorithm.

5. Future work

MusicSpace provides a high level command language for moving groups of related sound sources, and may be used to control arbitrary spatialization systems. MusicSpace was connected successfully to a Midi Spatialization system for playing midi files, to a midi-controlled audio mixing console for mixing multi-track recordings, as well as to Ircam's spatialization system [4].

In fact MusicSpace has applications also outside the field of spatialization. MusicSpace can be used for any situation where:

1) Streams of real time data can be controlled by discrete parameters (e.g. streams of audio sources controlled by distance, pan, directivity, etc.),
2) Relations between these parameters can be expressed as constraints or combinations of constraints.

Such situations occur frequently in music composition, sound synthesis, and real time control. We have sketched some of them here. Other applications in progress concerns the automatic animation of sound sources (e.g. defining sources which revolve automatically around other sources, or which move through a path itself defined with constraints).

MusicSpace and related information can be obtained at http://www.csl.sony.fr/MusicSpace

6. References

[1] Delerue O., Agon C., "Open Music + Music Space = Open Space", Proceedings of the Journées d'Informatique Musicale JIM 99, Issy-les-Moulineaux, 1999.

[2] Eckel G., "Exploring Musical Space by Means of Virtual Architecture", Proceedings of the 8th International Symposium on Electronic Art, School of the Art Institute of Chicago, 1997.

[3] Hower W., Graf W. H., "a Bibliographical Survey of Constraint-Based Approaches to CAD, Graphics, Layout, Visualization, and related topics", Knowledge-Based Systems, Elsevier, vol. 9, n. 7, pp. 449-464, 1996.

[4] Jot J.-M., Warusfel O., "A Real-Time Spatial Sound Processor for Music and Virtual Reality Applications", Proceedings of ICMC, 1995.

[5] Lea R., Matsuda K., Myashita K., *Java for 3D and VRML worlds*, New Riders Publishing, 1996.

[6] Pachet F., Delerue O., "A Temporal Constraint-Based Music Spatializer", ACM Multimedia Conference, Bristol, 1998.