Constraint-Based Spatialization

François Pachet
Olivier Delerue
Sony Computer Science Laboratories - Paris.
pachet{delerue}@csl.sony.fr http://www.csl.sony.fr

Abstract

This paper describes an application of constraint programming to interfaces for audio mixing. MidiSpace is an interface representing each sound source of a musical piece as a graphical icon, as well as an object corresponding to the listener in a window. MidiSpace is coupled to a spatialization system so that moving graphical objects modifies the audio mixing of the musical piece according to the respective position of the sound sources to the avatar. We further introduce a constraint-based mechanism which allows to maintain consistency in the overall mixing. Constraints represent properties of related sound sources, which should always remain true, and may be stated by the user through the interface. When an object is moved, a constraint solver uses the constraints to propagate changes. We describe the library of currently designed constraints, and propose an extension of the system to handle reproduction systems with multiple loudspeakers.

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 systems that simulate acoustic environments for existing sound signals. These works are based on techniques allowing to recreate impression of sound localization using a limited number of loudspeakers.

For instance, The *Spatialisateur IRCAM* [4] is a virtual acoustic processor that allows to define the sound scene as a set of perceptive factors such as azimuth, elevation and orientation angles of sound sources relatively to the listener. This processor can adapt itself to any sound reproduction configuration, such as headphones, pairs of loudspeakers, or collections of loudspeaker. Other commercial systems with similar features have recently been introduced on the market, such as Roland *RSS*, the *Spatializer* (Spatializer Audio Labs) or Q-Sound labs's *Q-Sound*, which builds extended stereophonic images. This tendency to propose integrated technology to produce 3D sound is further reflected, for instance, by Microsoft's DirectX API now integrating 3D audio.

Spatialization techniques have mostly been used to *enhance* existing interfaces or systems such as the Cave or *CyberStage* [2] [3]. Conversely, we are interested in building interfaces for controlling spatialization *per se*. The main applications are 1) high level interfaces for mixing devices, and 2) interfaces for future multi track listening devices. In this context, the main concern is to maintain some sort of consistency of musical pieces, while allowing the user to navigate freely in a control space.

We will first describe our system MidiSpace, which precisely allows users to control in real time spatialization of sound sources, without any restriction. In Section 3, we will show how to add some semantics to limit the range of user actions in a meaningful way. Finally in Section 4 we will show how to extend the system to handle sound reproduction systems with several loudspeakers in an homogeneous way.

2 The MidiSpace System

2.1 The basic system

MidiSpace is an interface for controlling an arbitrary spatialization system. The basic idea is to represent graphically sound sources in an editor, as well as an avatar that represents the listener itself. In this editor, the user may either move its avatar around, or move the instruments themselves. The relative position of sound sources and the listener's avatar determine the overall mixing of the music, according to simple mapping functions, as illustrated in *Figure 1*. The 2D interface of MidiSpace is represented in *Figure 2*. The real time mixing of sound sources is realized by sending Midi volume and panoramic messages to the spatialization system.

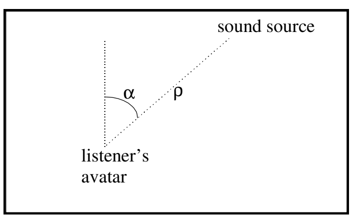

Figure 1. Volume of $source_i$ = f(distance($graph\_object_i$, listener_avatar)). f is a function mapping distance to Midi volume.

> Stereo position of $source_i$ = g(angle($graph\_object_i$, 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.

An application of MidiSpace is the control of mixing consoles (e.g. the Yamaha 02R). Indeed, a mixing console can be seen as a simple spatialization system.

3 Introducing Mixing Consistency with Constraints

There are two main problems with the basic system described above:

1) users can move only one object at a time, which may be cumbersome in a lot of situations

2) users have in some sense too much control: the configuration of sound sources may be freely edited by the user, which may not have the required knowledge to ensure that the overall sound is not distorted, let alone "pleasant".

To solve these problems, we introduce constraints in the systems. Constraints allow to ensure some sort of consistency in the mixing, and also to raise the level of the communication with the system.

3.1 Mixing Consistency

Mixing involves a set of actions that can often be defined as compositions of atomic actions. For instance, sound engineers use knowledge on the *energy* of the signal to ensure that it always lies between reasonable boundaries. One effect of this property is that sound levels are usually not set independently of one another. Typically, when a fader is raised, another one, (or a group of other faders) should be lowered. 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, and all these tracks may have to be edited together. Other typical mixing action is to assign *boundaries* to instruments or groups of instruments, and so forth.

The main idea of MidiSpace is to encode this type of knowledge on sound spatialization as *constraints*, which are interpreted in real time by a constraint propagation algorithm.

3.2 Constraints and Mixing Consistency

Constraints are defined by relations holding on variables. We will first describe the variables needed, and then the relations.

3.2.1 MidiSpace Variables

In MidiSpace, the variables are the following. First there are as many variables as sound sources on the interface. More precisely, each sound source is represented by a point p_i, i.e. two integer variables (one for each coordinate): p_i, where $p_i = \{x_i, y_i\}$ with x_i, $y_i \in [1, 1000]$ (a typical screen).

Moreover, there is one variable representing the position of the listener's avatar, itself consisting of two integer coordinate variables: l, where $l = \{x_l, y_l\}$ with $x_l, y_l \in [1, 1000]$.

3.2.2 MidiSpace Constraints

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

The simplest constraint is the constraint stating that the energy level between several sound sources ($i = 1, .., n$) 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, when one source is moved toward the listener, the other sources should be "pushed away".

- Constant Angular Offset

This constraint is the angular equivalent of the preceding one. It expresses that the spatial organization between 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 follows:

$$(p_1, \widehat{l}, p_2) = Cte$$

- 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 (as represented graphically in *Figure2*).

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

- 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):

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

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, such as [1].

- 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 simple propagation algorithm which implements only a part of our requirements, but with predictable and reactive behavior. The current algorithm we use is based on a simple propagation scheme, and allows to handle functional constraints, inequality constraints. It handles cycles simply by checking conflicts. It is described in [5].

Basically each constraint requires two specifications:

1) an invariant, defined in terms of the positions of the sources and the listener. This invariant represents a relation which should always be true.

2) a propagation method, which specifies the choices made, in case of indeterminacy. This method specifies how the invariant should be satisfied when one of the variables in the relation sees its value changed, either by the user, or by the propagation algorithm itself.

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 button. Constraints themselves are represented by a small ball, whose color depends on the constraint's type, linked to the constrained objects by lines. Some constraints have specific behavior, such as "limit constraints", showing a circle centered on the listener's avatar to display their scope (*Figure2*).

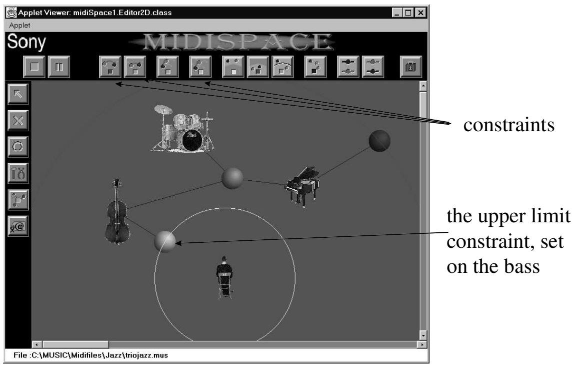

Figure 2. The 2D MidiSpace interface for specifying mixing constraints.

4 Multi Loudspeakers Systems

In this section, we focus on the application of MidiSpace to handle arbitrary configurations of loudspeakers. We will first introduce a useful constraint which allows to represent pairs of coupled sound sources, and which can be seen as an approximation of a pair of loudspeakers. We will then revisit the MidiSpace system by stressing on two important underlying assumptions regarding the status of the sound reproduction system. We then propose to handle multiple loudspeakers by introducing a "trick": represent loudspeakers as constraineable objects in the interface.

4.1 The "Stereo Pair" Constraint

This constraint allows the representation of tightly coupled sources, such as the microphones of an XY pair. Its representation requires the use of a "virtual" object that defines the azimuth of the pair : this object is then used as a handle to the whole pair.

This constraint ensures that both sources remain symmetrically opposed with respect to the handle which can be done by combining a constraint on angles with a constraint on distances, using the handle as a reference position.

Specific choices in the propagation method of this constraint are made in order to provide several controls on the parameters of the pair. As a result, when a source is moved, the constraint propagates the change to the other source, rather than to the handle : this allows to adjust the stereo angle of the pair. On the other hand, moving the handle propagates to both sources, and changes the overall angular position of the pair, as well as its global level.

Thus, this constraint can be used for representing a pair of loudspeaker, e.g. two sound sources corresponding to left and right channels of a stereo signal.

However, it is not sufficient for handling multi loudspeaker configurations.

4.2 Introducing Multi Loudspeakers

MidiSpace in its basic form is based on two strong assumptions. First, it is assumed that the loudspeakers are positioned at fixed places, and second, that the listener (the real one, not its avatar) is ideally centered between the loudspeakers. In the case of stereo configurations, these assumptions are most often approximately satisfied. However, in the case of multi loudspeakers configurations, this is no longer true.

MidiSpace is designed to control any spatialization system. In particular, Ircam's spatializer is interesting because it can handle arbitrary configurations of loudspeakers. Indeed, one of the original characteristic of Ircam's spatializer is to separate clearly the layer in charge of the spatialization from the layer in charge of the actual sound reproduction system.
In principle, therefore, it is possible to use MidiSpace in a multi-loudspeaker setting. This would require two interfaces:

1) MidiSpace's current interface to control the relative positions of sound sources
2) an interface to set the parameters corresponding to the actual sound reproduction.

However, this may be cumbersome to use and control, since it requires two indirections. Instead, we propose minimal extensions of MidiSpace to handle directly the configuration of the sound reproduction system. In particular we propose to represent in the same interface the sound sources, as well as the loudspeakers. This "trick" allows to have only one interface instead of two. Additionally, it allows to create interesting mixings which would be impossible to do otherwise. To implement this, we need two things: a redefinition of the mapping functions to handle several loudspeakers, and 2) a global constraint to ensure that the loudspeakers always remain "coherent".

4.3 Mapping Functions for Several Loudspeakers

Taking into account multi loudspeaker outputs induces a major change in the "sendSpatMsg" method of each sound source. In the case of a stereo configuration, a sound source outputs a signal s that will be amplified by both speakers, according to the sound source position. If the coordinates of the source are represented in a polar system as (ρ, θ) the right amplification is expressed as a function of these coordinates: $RightAmp = f(\rho,\theta)$ as well as the left amplification: $LeftAmp = g(\rho,\theta)$

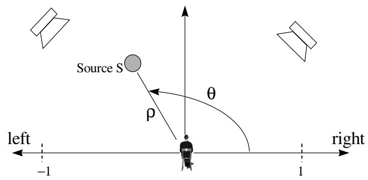

Figure 3. Polar coordinates of a sound source

In order to keep a constant energy level when the sound source moves along a constant radius, these functions must verify: $\forall \theta, f^2(\rho,\theta) + g^2(\rho,\theta) = Cste$. Moreover, for symmetrical reasons, we need: $\forall \rho,\theta: f(\rho,\theta) = g(\rho,\pi-\theta)$. A solution to this problem is: $f(\rho,\theta) = (1-\rho)\cos(\theta/2)$ and $g(\rho,\theta) = (1-\rho)\sin(\theta/2)$ with $\theta \in [0,\pi]$ and $\rho \in [0,1]$

This model can be extended in the case of multi loudspeaker outputs, by representing the positions of each loudspeaker within the same coordinates system:

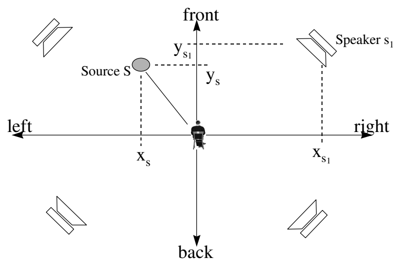

Figure 4. Representing the loudspeakers within the coordinates system

Amplifications values are then computed as a combination of two transfer functions mapping front/back and left/right axis.

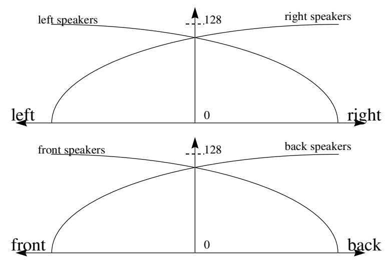

Figure 5. Transfer functions for front/back and left/right axis

Eventually, the "sendSpatMsg" method of a sound source will result in computing its amplification values for each loudspeaker and send them to the spatialization device.

4.4 Coherence of Loudspeakers

The loudspeaker "objects" in MidiSpace represent particular objects which are in fixed positions. To ensure that the interface always reflects this state, we add a constraint on all loudspeakers.

This constraint ensures a constant position of "virtual" loudspeakers with respect to the listener avatar. It corresponds exactly to the "Grouping" constraint defined in 3.2, with the following additions:

- The listener is considered as a constrained object (so that the group constraint can be set)

- The loudspeakers cannot be moved explicitly by the user (so that the listener is not forced to move by the constraint).

5 Conclusion

We have described MidiSpace, a system to control spatialization systems through an intuitive interface. Constraints are added to MidiSpace to ensure consistency of mixings. We described an extension of MidiSpace to handle configurations of multiple speakers, based on the idea of representing loudspeakers as sound sources in an homogeneous way, together with specialized constraints. The resulting system allows to control precisely the spatialization system, and to create novel but realistic mixings.

6 References

[1] Borning A. Lin R., Marriott K., "Constraints for the web", Proceedings of ACM Multimedia Conference, Seattle, pp. 173-181, 1997.

[2] Dai P., Eckel G., Göbel M., Hasenbrink F., Lalioti V., Lechner U., Strassner J., Tramberend H., Wesche G., "Virtual Spaces: VR Projection System Technologies and Applications", Tutorial Notes, Eurographics '97, Budapest, 1997.

[3] 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.

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

[5] Pachet F., Delerue O., "MidiSpace: A Temporal Constraint-Based Music Spatializer", Proceeings of ACM Multimedia, Bristol, 1998.