In Roads, C., editor, Sound in Space, Santa Barbara, 2000. CREATE

MusicSpace goes Audio

François **Pachet**, Olivier **Delerue,** Peter **Hanappe**
SONY CSL Paris, 6, rue Amyot 75005, Paris, FRANCE
E-mail: musicspace@csl.sony.fr

Abstract

The MusicSpace project aims at providing high-level user control on music spatialization, i.e. the position of sound sources and the position of the listener's avatar. This is done by introducing a constraint system in a graphical user interface representing the sound sources, and connected to a spatializer. The constraint system allows to express various sorts of properties on configuration of sound sources. When the user moves one source - through the interface or via a control language - the constraint system is activated and tries to satisfy the constraints that may have been violated. A first Midi version of the MusicSpace has already been designed and proved very successful. We describe here a second version of MusicSpace which now handles full-fledged multi-track audio files. We report on the design of the system and preliminary experiments.

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. [3]). These spatialization techniques are mostly used for building virtual reality environments, such as [1], [4]. 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 an interface for producing high level commands to a spatializer [6]. 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. 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 [3].

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 have proposed 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. 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 [2].

3.1 Constraints and Mixing Consistency

We defined a set of constraints appropriate for specifying "interesting" relations between sound sources. 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. Intuitively, it means that when one source is moved toward the listener, the other sources should be "pushed away", and vice-versa.

- 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.

- Constant Distance Ratio

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

- 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.

- 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).

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.

3.2 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.
- The constraints are not all functional. For instance, geometrical limits of sound sources are typically inequality constraints.
- The constraints 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 behaviour. 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 [5].

3.3 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.

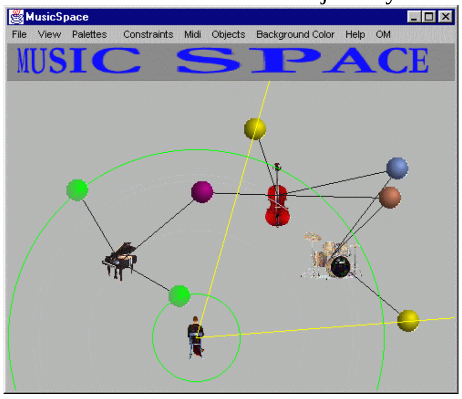

Figure 3. The MusicSpace interface for setting constraints.

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" 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. The Audio Version

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 [3]. In all these cases though, MusicSpace was used as a mere control system, needing a remote controlleable spatialization system.

We recently built a specific Dynamic Link Library (dll) for PCs which allows MusicSpace to control Microsoft *DirectX* 3D sound buffers. This dll of MusicSpace-audio

basically provides a connection between any Java application and DirectX, by converting DirectX ‘s API C++ types into simple types (such as integers) that can be handled by Java.
Although DirectX may arguably not be the more accurate spatialization system around, this extension has a number of benefits.
First, DirectX provides parameters for describing 3D sound sources which can be constrained using MusicSpace. For instance, a *DirectX* sound source is endowed with an orientation, a directivity and even a Doppler parameter. An “orientation” constraint has been designed and included in the constraint library of MusicSpace. This constraint states that two sound source should always “face” each other: when one source is moved, the orientation of the two sources move accordingly.
Second, DirectX allows to handle lots of sound sources in real time. This is useful for mixing complex symphonic music, which have often dozens of related sound sources.
Lastly, the presence of DirectX on a number of PC makes MusicSpace easily useable to a wide audience.

5. Applications of MusicSpace

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] 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.
[2] 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.
[3] Jot J.-M., Warusfel O., “A Real-Time Spatial Sound Processor for Music and Virtual Reality Applications”, Proceedings of ICMC, 1995.
[4] Lea R., Matsuda K., Myashita K., *Java for 3D and VRML worlds*, New Riders Publishing, 1996.
[5] Pachet F., Delerue O., “A Temporal Constraint-Based Music Spatializer”, ACM Multimedia Conference, Bristol, 1998.
[6] Pachet F., Delerue O., “MusicSpace, a constraint-based control system for music spatialization”, ICMC, Beigin (China), 1999.
[7] Pachet F., Delerue O., “Constraint-Based Spatialization”, First DAFX Workshop, Barcelona, 1998.