First International Conference on Virtual Worlds, Springer Verlag Lecture Notes in Computer Science 1434, pp. 298-307, 1998.

A Mixed 2D/3D Interface for Music Spatialization

François Pachet[1], Olivier Delerue[1]

[1] SONY CSL Paris, 6, rue Amyot, 75005 Paris, France
Email: pachet{delerue}@csl.sony.fr

Abstract. We propose a system for controlling in real time the localisation of sound sources. The system, called MidiSpace, is a real time spatializer of Midi music. We raise the issue of which interface is the most adapted for using MidiSpace. Two interfaces are proposed: a 2D interface for controlling the position of sound sources with a global view of the musical setup, and a 3D/VRML interface for moving the listener's avatar. We report on the design of these interfaces and their respective advantages, and conclude on the need for a mixed interface for spatialization.

Active Listening

We believe that listening environments of the future can be greatly enhanced by integrating relevant models of musical perception into musical listening devices, provided we can develop appropriate software technology to exploit them. This is the basis of the research conducted on "Active listening" at Sony Computer Science Laboratory, Paris. Active Listening refers to the idea that listeners can be given some degree of control on the music they listen to, that give the possibility of proposing different musical perceptions on a piece of music, by opposition to traditional listening, in which the musical media is played passively by some neutral device. The objective is both to increase the musical comfort of listeners, and, when possible, to provide listeners with smoother paths to new music (music they do not know, or do not like). These control parameters create implicitly control spaces in which musical pieces can be listened to in various ways. Active listening is thus related to the notion of *Open Form* in composition [8] but differs by two aspects: 1) we seek to create listening environments for *existing* music repertoires, rather than creating environments for composition or free musical exploration (such as *PatchWork* [11], *OpenMusic* [2], or *CommonMusic* [18]), and 2) we aim at creating environments in which the variations always preserve the original semantics of the music, at least when this semantics can be defined precisely.

The first parameter which comes to mind when thinking about user control on music is the spatialization of sound sources. In this paper we study the implications of giving users the possibility to change dynamically the mixing of sound sources. In te next section, we review previous approaches in computer-controlled sound spatialization, and then propose a basic environment for controlling music spatialization, called MidiSpace. We then describe a simple 2D interface for controlling sound sources, and then describe a VRML interface which gives users a more realistic view on the music heard. We compare the two approaches and argue in favor of a mixed solution integrating both interfaces. The last section describes the overall design and implementation of the resulting system.

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 results in psychoacoustics that allow to model the perception of sound sources by the human hear using a limited number of perceptive parameters [4]. These models have led to techniques allowing to recreate impression of sound localization using a limited number of loudspeakers. These techniques typically exploit differences 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 and of spatial volume.

For instance, The *Spatialisateur IRCAM* [10] 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) which allows to produce a stereo 3D signal from an 8-track input signal controlled by joysticks, or Q-Sound labs's *Q-Sound*, which builds extended stereophonic image using similar techniques. This tendency to propose integrated technology to produce 3D sound is further reflected, for instance, by Microsoft's DirectX API now integrating 3D audio.

These sound spatialization techniques and systems are mostly used for building various virtual reality environments, such as the Cave [5] or *CyberStage* [6], [8]. Recently, sound spatialization has also been included in limited ways in various 3D environments such as *Community Place*'s implementation of VRML [12], ET++ [1], or proprietary, low-cost infrastructures [3].

Based on these works, we are interested in exploiting spatialization capabilities for building richer listening environments. In this paper, we concentrate on the interface issue, i.e. how to give average listeners the possibility of exploiting sound source spatialization in a natural, easy way. We will first describe our basic spatialization system *MidiSpace*, which precisely allows user to control in real time the spatialization of sound sources. Then we describe two interfaces for MidiSpace, and compare their respective advantages.

First International Conference on Virtual Worlds, Springer Verlag Lecture Notes in Computer Science 1434, pp. 298-307, 1998.

The Basic MidiSpace System

MidiSpace is a system that gives listeners control on music spatialization. We first outline the characteristics of midi-based spatialization before describing the system.

Midi-Based Spatialization

MidiSpace is a real time player of Midi files which allows users to control in real time the localization of sound sources through a graphical interface (extensions to audio are not discussed in this paper). MidiSpace takes as input arbitrary Midi files [9]. The basic idea in MidiSpace is to represent graphically sound sources in a virtual space, as well as an avatar that represents the listener itself. Through an appropriate 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 geometrical rules illustrated in Fig. 1. The real time mixing of sound sources is realized by sending Midi volume and panoramic messages.

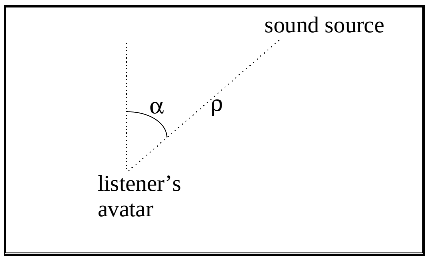

Fig. 1. Volume of sound_source$_i$ = f(distance(graphical-object$_i$, listener_avatar)). f is a function mapping distance to Midi volume (from 0 to 127). Stereo position of sound source i = g(angle(graphical_Object$_i$, listener_avatar)), where angle is computed relatively to the vertical segment crossing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

It is important to understand here the role of Midi in this project. On the one hand, there are strong limitations of using Midi for spatialization *per se*. In particular, using Midi panoramic and volume control changes messages for spatializing sounds does not allow to reach the same level of realism than when using other techniques (delays between channels, digital signal processing techniques, etc.), since we exploit only difference in amplitude in sound channels to create spatialization effects. However, this limitation is not important in our context for two reasons : 1) this Midi-based technique still allows to achieve a reasonable impression of sound spatialization which is enough to validate our ideas in user interface and control, and 2) more sophisticated techniques for spatialization can be added in MidiSpace, independently of its architecture.

First International Conference on Virtual Worlds, Springer Verlag Lecture Notes in Computer Science 1434, pp. 298-307, 1998.

We will now describe the interfaces for MidiSpace: first, a 2D interface, which provides a global view on the musical setting, and allows to move sound sources around. Then we describe a VRML interface and discuss its relevance for music listening. We conclude on the interest of a mixed approach.

The 2D Interface of MidiSpace

In the 2D interface of MidiSpace, each sound source is represented by a graphical object, as well as the listener's avatar (see Fig. 2.). The user may basically play a midi file (with usual tape recorder-like controls), and move objects around. When an object is moved, the spatializer is called with the new position of the object, and the mixing of sound sources is recomputed accordingly. Other features allow to mute sound sources, or select them as "solo".

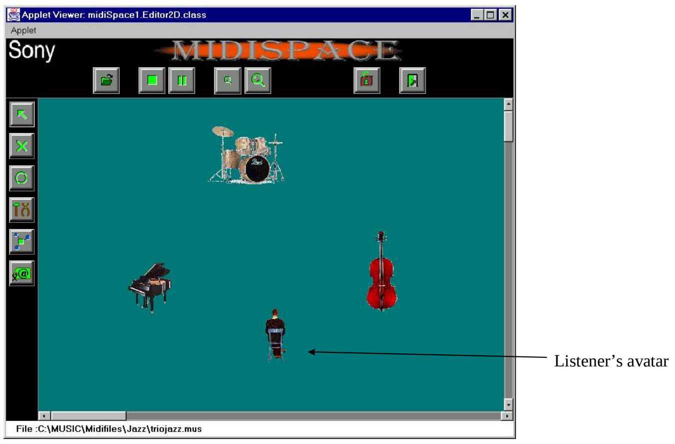
Applet Viewer: midiSpace1.Editor2D.class
Applet
Sony
MIDISPACE
Listener's avatar
File :C:\MUSIC\Midifiles\Jazz\triojazz.mus

Fig. 2. The 2D Interface of MidiSpace. Here, a tune by Bill Evans (Jazz trio) is being performed

In an initial version, we allowed *both* sound sources and the avatar to be moved. However this was confusing for users. Indeed, moving sound sources amounts to changing the intrinsic mixing of the piece. For instance, moving the piano away will changes the relationship between the piano sound and the rhythmic part. Moving the avatar simply amounts to changing the mixing in global way, but respects the relationships between the sound sources. The effect is quite different since in the second case the structure of the music is modified.

The interface provides a global view on the musical setup, which is very convenient to edit the musical setting. However, there is no impression of musical immersion in the musical piece : the interface is basically a means for editing the piece, not to explore it.

First International Conference on Virtual Worlds, Springer Verlag Lecture Notes in Computer Science 1434, pp. 298-307, 1998.

The VRML Interface for navigating in MidiSpace

Second, we have experimented with interfaces for navigating in the musical space. Several works addressed the issue of navigating in virtual worlds with spatialized sounds. The most spectacular are probably the Cave system [5] or CyberStage [8], in which the user is immersed in a fully-equipped room with surrounding images and sound. Although the resulting systems are usually very realistic, their cost and availability are still prohibitive.

Instead, we chose to experiment with affordable, wide-spread technology. A 3D version of MidiSpace in VRML has been built (see Fig. 3.), in which the VRML code is automatically generated from the 2D interface and the Midi parser. The idea is to create a VRML world in which the user may freely navigate using the standard VRML commands, while listening to the musical piece. Each instrument is represented by a VRML object, and the spatialization is computed from the user current viewpoint. In this interface, the only thing the user can do is move around using standard commands; sound sources cannot be moved.

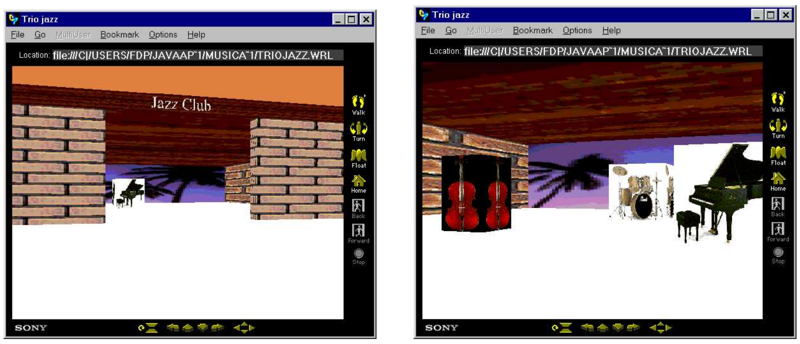
Trio jazz
File Go MultiUser Bookmark Options Help
Location: file:///C|/USERS/FDP/JAVAAP~1/MUSICA~1/TRIOJAZZ.WRL
Jazz Club
Walk
Turn
Float
Home
Back
Forward
Stop
SONY

Fig. 3. MidiSpace/VRML on the Jazz trio, corresponding to the 2D Interface of Fig. 2. On the left, before entering, on the right, inside the club.

Although the result is more exciting and stimulating for users than the 2D interface, it is not yet fully satisfying because the interface gives too little information on the overall configuration of instruments, which is a crucial parameter for spatialization. When the user gets close to an instrument, she loses the sense of her position in the musical set up (e.g. the jazz club, see Fig. 3.). Of course, this problem is a general problem with VRML interfaces, and is not specific to MidiSpace, but in the context of a listening environment, it is particularly important to provide a visualisation which is consistent with the music being played. This consistency is difficult to achieve with a 3D virtual interface on a simple screen.

First International Conference on Virtual Worlds, Springer Verlag Lecture Notes in Computer Science 1434, pp. 298-307, 1998.

The Mixed interface

Based on experiments with users on the two interfaces, we concluded on the interest of combining them for obtaining an optimal satisfaction on user control. The 2D interface is used for *editing* purposes, i.e. moving sound sources. Moving the avatar is not possible in this interface. The VRML interface is used for *exploration*, in a passive mode, to visualize the musical setting in 3D, and move the avatar around. Moving objects is not possible.

The communication between the two interfaces is realized through the VRML/Java communication scheme, and ensures that when the avatar is moved in the VRML interface, the graphical object of the 2D interface is moved accordingly. The overall architecture of MidiSpace is illustrated in Fig. 4.

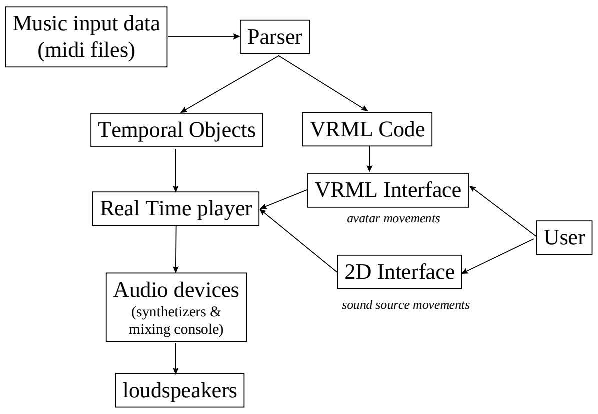

Fig. 4. The architecture of MidiSpace User Interaction.

Implementation

The implementation of the MidiSpace Spatializer consists in 1) translating Midi information into a set of objects within a temporal framework, 2) scheduling these temporal objects using a real time scheduler, 3) the interfaces.

The Parser

The Parser task is to transform the information contained in the Midi file into a unified temporal structure. The temporal framework we use is described in [18], an object-oriented, interval-based representation of temporal objects. In this framework, each temporal object is represented by a class, which inherits the basic functionalities from a root superclass *TemporalObject*.

One main issue the Parser must address comes from the way Midi files are organized according to the General Midi specifications. Mixing is realized by sending volume and panoramic Midi messages. These messages are global for a given Midi channel. One must therefore ensure that each instrument appears on a distinct Midi channel. In practice, this is not always the case, since Midi tracks can contain events on different channels. The first task is to sort the events and create logical melodies for each instrument. This is realized by analysing program change messages, which assign Midi channels to instruments, thereby segmenting the musical structure. The second task is to create higher level musical structures from the basic musical objects (i.e. notes). The Midi information is organized into notes, grouped in melodies. Each melody contains only the notes for a single instrument. The total piece is represented as a collection of melodies. A dispatch algorithm ensures that, at a given time, only one instrument is playing on a given Midi channel.

Scheduling temporal objects

The scheduling of MidiSpace objects uses *MidiShare* [14], a real time Midi operating system, with a Java API. *MidiShare* provides the basic functionality to schedule asynchronously, in real time, Midi events, from Java programs, with 1 millisecond accuracy. More details on the scheduling are given in [7].

MidiSpace Interfaces

The 2D interface uses the standard Java *awt* library, and follows a straightforward object-oriented interface design. The VRML interface is generated automatically from the Parser. More precisely, the Parser generates a file, which contains basically 1) the information on the global setup, 2) description of individual sound sources, corresponding to the various sound tracks identified, and 3) routing expressions to a spatializer object, which is defined as a Java script, using the VRML/Java communication scheme [12]. An excerpt of the VRML code is shown in Fig. 5.

```
WorldInfo {title "Trio jazz"}

# Various global settings
NavigationInfo {speed 2  type  [ "WALK" ]}
# The viewpoint from which the spatialization is computed
DEF USERVIEWPOINT Viewpoint {position -13 0 45}

# The definition of the musical setting (here, a Jazz
Club)
…
```

```
# the Label
Transform {
  translation -2 4.7 21
  children [
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 1 1 1}}
      geometry Text {
        string "Jazz Club"}}]}

# The sound sources, as identified by the Parser in
General Midi
DEF PIANO Transform {
  children [
    Shape {
      appearance Appearance {
        texture ImageTexture {
          url     "piano.jpg"
          repeatS FALSE
          repeatT FALSE}
        textureTransform TextureTransform {}
      }
      geometry Box {
        size 3 3 3}}]}

DEF DRUMS Transform { …}
DEF BASS Transform {…}

# The Java script for handling movements and
spatialization
DEF MY_SCRIPT Script {
 url "MusicScript.class"
 field SFString midiFileName
"http://intweb.csl.sony.fr/~demo/trio.mid"
 field SFNode channel10 USE DRUMS
 field SFNode channel2 USE BASS
 field SFNode channel3 USE PIANO
 eventIn  SFVec3f    posChanged
 eventIn  SFRotation orientation
 eventOut SFRotation keepRight
 eventOut SFVec3f    keepPosition}

“ The routing of VRML messages to the Java script
```

First International Conference on Virtual Worlds, Springer Verlag Lecture Notes in Computer Science 1434, pp. 298-307, 1998.

```
ROUTE DETECTOR.position_changed    TO MY_SCRIPT.posChanged
ROUTE    MY_SCRIPT.keepPosition                          TO
USERVIEWPOINT.set_position
ROUTE            DETECTOR.orientation_changed            TO
MY_SCRIPT.orientation
ROUTE   MY_SCRIPT.keepRight                              TO
USERVIEWPOINT.set_orientation
```

Fig. 5. The generated VRML code for describing the musical setting from a given midi file

Conclusion, Future works

The MidiSpace system shows that it is possible to give some degree of freedom to users in sound spatialization, through an intuitive graphical interface. We argue that a unique interface is not appropriate for both moving sound sources and avatars, while giving users a realistic feeling of immersion. In the case of spatialization, although they appear at first to be similar user actions, moving sound sources and moving the avatar bear significantly different semantics, and we concluded that allowing these two operations in the same interface is confusing. We propose an approach combining a standard 2D interface, appropriate for moving sound sources and editing the setup, and a VRML interface appropriate for moving avatars and exploring the musical piece. The prototype built so far validates our approach, and more systematic testing with users is in progress.

Future work remains to be done in three main directions. First we are currently adding a *constraint-based* mechanism for maintaining consistency in sound source positions [15]. This mechanism allows users to navigate in a restricted space, which will always ensure that some mixing properties of the music are satisfied. Second, an audio version of MidiSpace is in progress, to 1) enlarge the repertoire of available music material to virtually all recorded music, and 2) improve the quality of the spatialization, using more advanced techniques such as the ones sketched in the introduction of this paper. Finally, an *annotation* format is currently being designed to represent information on the content of musical piece, to enrich the interaction. These annotations typically represent information on the structure of the piece, its harmonic characteristics, and other kinds of temporal information [17]. These extensions are in progress, and remain compatible with our choice for user interface design proposed here.

References

1. Ackermann P., *Developing object-oriented multimedia software*, Dpunkt, Heidelberg, 1996.

First International Conference on Virtual Worlds, Springer Verlag Lecture Notes in Computer Science 1434, pp. 298-307, 1998.

2. Assayag G., Agon C., Fineberg, J., Hanappe P., "An Object Oriented Visual Environment For Musical Composition", Proceedings of the International Computer Music Conference, pp. 364-367, Thessaloniki, 1997.
3. Burgess D.A., "Techniques for low-cost spatial audio", ACM Fifth Annual Symposium on User Interface Software and Technology (UIST '92), Monterey, November 1992.
4. Chowning, J. (1971), "The simulation of moving sound sources", *JAES*, vol. 19, n. 1, p. 2-6.
5. Cruz-Neira, C., Leight, J., Papka, M., Barnes, C., Cohen, S.M., Das, S., Engelmann, R., Hudson, R., Roy, T., Siegel, L., Vasilakis, C., DeFanti, T.A., Sandin, D.J., "Scientists in Wonderland: A Report on Visualization Applications in the CAVE Virtual Reality Environment", Proc. IEEE Symposium on Research Frontiers in VR, pp. 59-66, 1993.
6. 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, 75 pages.
7. Delerue O., Pachet F., "MidiSpace, un spatialisateur Midi temps réel", Cinquièmes Journées d'Informatique Musicale, Toulon, 1998.
8. 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.
9. IMA, "MIDI musical instrument digital interface specification 1.0", Los Angeles, International MIDI Association, 1983.
10. Jot J.-M., Warusfel O. "A Real-Time Spatial Sound Processor for Music and Virtual Reality Applications", Proceedings of International Computer Music Conference, September 1995.
11. Laurson M., Duthen J., "PatchWork, a graphical language in PreForm", Proceedings of the International Computer Music Conference, San Francisco,172-175, 1989.
12. Lea R., Matsuda K., Myashita K., Java for 3D and VRML worlds, New Riders Publishing, 1996.
13. Meyer L., *Emotions and meaning in music*, University of Chicago Press, 1956.
14. Orlarey Y., Lequay H. "MidiShare: a real time multi-tasks software module for Midi applications", Proceedings of the ICMC, 1989, ICMA, San Francisco.
15. Pachet, F. Delerue, O. A Temporal Constraint-Based Music Spatialization system, submitted to ACM MultiMedia 98, 1998.
16. Pachet F., Ramalho G., Carrive J. "Representing temporal musical objects and reasoning in the MusES system", *Journal of New Music Research*, vol. 25, n. 3, pp. 252-275, 1996.
17. Pachet, F. Delerue, O. "Annotations for Real Time Music Spatialization", International Workshop on knowledge Representation for interactive Multimedia Systems, KRIMS-II workshop, Trento, Italy, 1998.
18. Taube H., "Common Music: A Music Composition Language in Common Lisp and CLOS", *Computer Music Journal*, vol. 15, n° 2, 21-32, 1991.