Proceedings of International Workshop on Knowledge Representation for Interactive Multimedia Systems, Trento, Italy, 1998.

Annotations for Real Time Music Spatialization

François Pachet, Olivier Delerue
SONY CSL Paris, 6, rue Amyot, 75005 Paris, France
Email: pachet{delerue}@csl.sony.fr

Abstract

We are interested in developing multimedia technology for enriching the listening experience of average listeners. One main issue we focus on is the design and construction of software systems in which users interact with music in various ways while maintaining as much as possible the semantics of the original music. In this context, we develop a research activity concerning music spatialization. We propose a system called MidiSpace, in which users may listen to music while controlling in real time the localization and spatialization of sound sources, through a simple interface. In this paper we show how to represent various information about the musical content as annotations, and how to exploit these annotations to enrich the interaction with the user. We propose a format for these annotations, and an implementation of an interactive real time player. We further introduce a constraint propagation mechanism that ensure the spatialization always satisfies some mixing properties. We finally show that these constraints can themselves be represented as annotations in our framework.

1. 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 (Eckel, 1997) 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* (Laurson & Duthen, 1989), *OpenMusic* (Assayag, 1997), or *CommonMusic* (Taube, 1991)), 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. We will first review previous approaches in computer-controlled sound spatialization, and propose a basic environment for controlling music spatialization, called MidiSpace. We will then show how to enrich this basic system by adding annotations that give various semantic information on the music, and on the mixing of sound sources. A first kind of annotation is representation of music structure, i.e. how music is segmented into parts. A higher level of information relates to harmony, ie. the various chords, and tonal keys of the music. Finally, we introduce a level of annotation representing constraints on musical sources. These annotations are all represented in a unified format, handled in real time by the spatializer.

2. 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 (Chowning, 1971). These models have led to techniques allowing to recreate impression of sound localization using a limited number of loudspeakers. 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.

For instance, The *Spatialisateur IRCAM* (Jot & Warusfel, 1995) 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 (DirectX, 1998).

These sound spatialization techniques and systems are mostly used for building various virtual reality environments, such as the Cave (Cruz-Neira, 1993) or *CyberStage* (Dai et al 97), (Eckel, 97). Recently, sound spatialization has also been included in limited ways in various 3D environments such as *Community Place*'s implementation of VRML (Lea et al., 1996), ET++ (Ackermann, 1996) or proprietary, low-cost infrastructures (Burgess, 1992).

Based on these works, we are interested building interactive listening environments in which users may modify the spatialization of sound sources. A main point of concern is to allow the user to explore a control space while maintaining some sort of *consistency* of musical pieces. We will first describe our basic system MidiSpace, which precisely allows user to control in real time spatialization of sound sources, without any restriction. In the next sections, we will show how to add some semantics to limit the range of user actions in a meaningful way using annotations.

3. The Basic MidiSpace System

MidiSpace is a real time player of Midi files which allows users to control in real time the localization of sound sources through a 2D interface (extensions to audio and 3D are discussed in 7). MidiSpace takes as input arbitrary Midi files (IMA, 1983). The basic idea in MidiSpace 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 geometrical rules illustrated in Figure 1. The 2D interface of MidiSpace is represented in Figure 2. Additional features such as muting sources are provided but not discussed here. The real time mixing of sound sources is realized by sending Midi volume and panoramic messages.

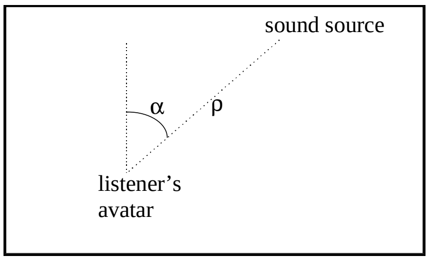


Figure 1. Volume of sound_source$_i$ = f(distance(graphical-object$_i$, listener_avatar)). f is a function mapping distance to Midi volume (from 0 to 127). Stereo position of sound source i = g(angle(graphical_Object$_i$, listener_avatar)), where angle is computed relatively to the vertical segment crossing 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 research. 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 recreate spatialization. However, this limitation is not important 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, and 2) more sophisticated techniques for spatialization can be added in MidiSpace, independently of its architecture.

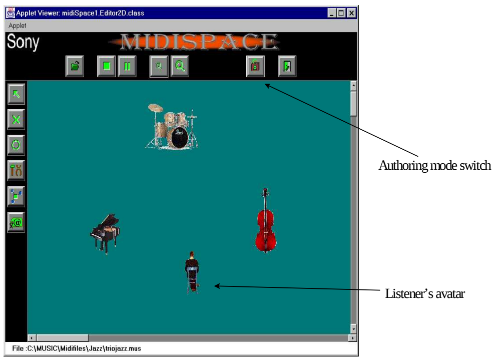

Applet Viewer: midiSpace1.Editor2D.class
Applet
Sony
MIDISPACE
Authoring mode switch
Listener's avatar
File :C:\MUSIC\Midifiles\Jazz\triojazz.mus

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

4. THE TEMPORAL ASPECT OF MIDISPACE INTERACTION: ANNOTATIONS

4.1 ANNOTATIONS AND REPRESENTATION OF CONTENT

The need for representing and exploiting representation of content in multimedia systems is now widely acknowledged. The Mpeg7 project for instance aims at standardizing content representation of multimedia documents for future multimedia applications. Other standards in use or in progress are more dedicated to musical information, such as SMDL or HyTime. However, these formats are not primarily designed for real time applications.

In MidiSpace, we developed a simple format for representing annotations on musical pieces, that can be interpreted in real time to influence the spatialization. Examples of annotations useful for spatialization are :

- the structure of the musical piece (how a piece is divided into various segments such as introduction, chorus, coda, etc.),
- harmonic information (e.g. the chord sequence associated with the music),
- analytical information (e.g. the underlying keys or tonalities),
- movements of sound sources,
- etc.

4.2 THE ANNOTATION FORMAT

Our format is based on a time-tagged attribute/value representation. The time information is expressed in musical beats, and is therefore independent of the tempo.
The annotation file is divided into blocks. Each block represents a particular type of annotation, such as structure, harmony, movements, etc. A key word indicates what type of information is represented in the block (e.g. structure, harmony, movements, etc.). In order to increase the readability of blocks, we introduce an organization of the temporal structure into parts. Each part represents a temporal segment, and is named with a label. The whole piece can then be described using these labels (see Figure 3).

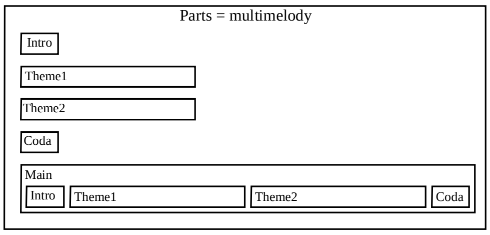


Figure 3. The temporal structure of a musical piece.

Each block consists in three syntactic parts:

- Declaration of temporal parts and associated labels. The parts are associated with durations.
- Declaration of the sequencing of parts. Each part is associated with its corresponding start time, the duration information is given in the declaration part.
- Actual content information of each part. The syntax of this information depends on the type of annotation.

Examples of annotations are given in the following sections.

4.2.1 The Temporal Structure as Annotation

Representing the structure of a musical piece is important in our context of interactive, real time spatialization. Indeed, the structure of the piece can influence the localization of sound sources. For instance during a chorus, there is a main instrument which plays a central role, and can therefore be brought in foreground. Similarly, during the ending, and depending on the type of piece, the instruments can fade away. Exploiting this information requires two things:

1) representing and accessing the temporal information on the structure
2) interpreting this information to modify the spatializer accordingly in real time

Our format allows to represent this temporal information. For instance, the structure of the piece in Figure 2 is the following:

```
[structure]
labels   // durations are between parenthesis
            intro {8}
         theme {12}
         theme2 {12}
         chorus {12}
         coda{12}
         ending{2}
Info
intro    type=Introduction instr=piano
theme    type=Exposition instr=piano
theme2   type=Exposition instr=piano
chorus   type=Chorus instr=piano
coda     type=Coda instr=piano
ending   type=Exposition instr=piano
…
Global  // start time are between parenthesis
intro (1) theme (9) theme2 (27) chorus (39) theme
(51) coda (63) ending (75)
```

Figure 4. Structure of a musical piece. Start time and durations are expressed in musical beats.

Thanks to this information, the mixing of a musical piece can take the musical structure into account, by emphasizing instruments accordingly. The precise interpretation of each type of musical segment (i.e. chorus, intro, ending, etc.) is dependent on the structure type. The implementation is described in section **Erreur ! Source du renvoi introuvable.**.

4.2.2 Harmony

The harmonic information about a musical piece gives a precise account on the tonal structure of the piece (i.e. which keys underlie the music). In our active listening context this information may be very useful to enrich the interaction between a user and the musical piece. Knowing precisely, at each moment of the piece, which notes are out of tune for instance, may help novice users to explore dissonant pieces, as proposed in (Delerue & Pachet, 1998b).
This harmonic information may be represented as a chord sequence, as illustrated in Figure 5. Chords are represented using a format derived from the chord format of SMDL (SMDL, 1995; Sloan, 1993), and adapted for tonal popular music (Jazz, rock, pop) tunes. Chord sequences are divided into parts (such as A, B, C), and the overall structure of the chord sequence is described in terms of these parts (e.g. AABA). This harmonic information is typically used for analytical process, such as harmonic analysis (Pachet, 1998).

```
 [harmony]
labels   // durations are between parenthesis
            intro{8}
            A {12}
            B {12}
            ending{2}
Info
intro = Bb maj7 | Bb maj7| Bb maj7| Bb maj7| Bb
maj7| Bb maj7| Bb maj7| Bb maj7
A= Bb maj7| Eb maj7| D min7| G 7| C min| G 7|
C min| F 7| C min7| C min maj7| C.min.7|F7
B= Bb maj7| Eb maj7| D min7| G 7| C min| G 7|
C min| F 7| B min7| E 7| C min7| F 7
ending = Bb maj7 | Bb maj7
Global  // start time are between parenthesis
intro (1) A (9) A (27) B (39) A (51) B (63) ending
(75)
```

Figure 5. A harmonic annotation. The chord sequence itself uses a proprietary chord format

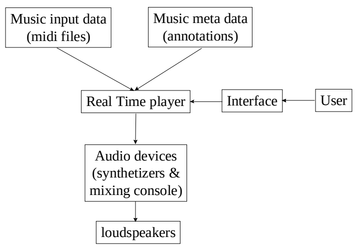
4.2.3 Movement of sound sources

The interactive nature of MidiSpace makes it possible to record automatically movements of sound sources by the user. These recordings may easily be represented as annotations, as illustrated in Figure 6. A specific editor of temporal structures (not represented here) allows to edit manually the information once recorded. Note that for movements of sound sources, there is no need for a temporal structure. Therefore, by default, only one label is declared (*Default*).

[movements]
labels *// durations are between parenthesis*
Default{77}
Info
Default =
{startBeat=7.692 duration=0 instr=bass x=127 y=329
startBeat=7.698 duration=0 instr=bass x=128 y=329
startBeat=7.881 duration=0 instr=bass x=131 y=329
startBeat=8.067 duration=0 instr=piano x=100 y=50
startBeat=8.214 duration=0 instr=piano x=97 y=52
startBeat=8.463 duration=0 instr=piano x=96 y=54
startBeat=8.631 duration=0 instr=piano x=95 y=55
startBeat=9.36 duration=0 instr=bass x=131 y=329
startBeat=9.531 duration=0 instr=bass x=135 y=329…}
Global *// start time are between parenthesis*
Default(1)

Figure 6. Temporal information representing movements of musical sources are represented as annotations.

The architecture of MidiSpace is illustrated in Figure 7. It mainly consists in a real time player which takes as input a musical data (a Midi file in the current version), and an annotation file containing all the annotations pertaining to the musical piece.
The existence of annotations makes it necessary to differentiate between two modes in MidiSpace: an *authoring* mode, in which the user may create annotations, and a *listening* mode, in which the annotation are used for spatialization. The only difference between the two modes is a facility to record, while playing, the user actions which generate an annotation file.


Music input data (midi files)
Music meta data (annotations)
Real Time player
Interface
User
Audio devices (synthetizers & mixing console)
loudspeakers

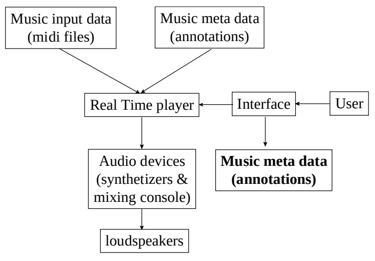

Music input data (midi files)
Music meta data (annotations)
Real Time player
Interface
User
Audio devices (synthetizers & mixing console)
Music meta data (annotations)
loudspeakers

Figure 7. Above, the architecture of MidiSpace in the listening mode. Below, the architecture in authoring mode: the main output is an annotation file, which can be used in the listening mode by users.

The next section introduces yet another kind of semantic information we introduced in MidiSpace: mixing consistency.

5. INTRODUCING MIXING CONSISTENCY IN MIDISPACE

The notion of musical semantics in general has been extensively debated (see e.g. Meyer, 1956). Without committing to a particular general semantic theory of music, we believe that it is possible to define a reasonable and pragmatic notion of musical semantics in the context of interactive systems, based on the properties of the navigational spaces that may be enforced automatically. In the context of spatialization, this semantics may be represented as a set of properties imposed on the overall mixing, which may in turn be represented as constraints between the sound sources and the listener's avatar. Moreover, we propose to represent these constraints as annotations, in order to produce dynamically evolving navigational spaces.

5.1 MIXING CONSISTENCY

The knowledge of the sound engineer is difficult to explicit and even more to represent as a whole. Its basic actions are modifications of mixing consoles controls, such as faders. However, mixing involves a set of actions that can often be defined as compositions of these atomic actions. For instance, sound engineers use knowledge on sound energy to ensure that the overall energy level of the recording 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. Other typical mixing action is to assign boundaries to instruments or groups of instruments, and so forth.
These remarks led us to introduce a consistency mechanism in MidiSpace, that precisely allows to represent composite mixing actions as constraints.

5.2 CONSTRAINTS AND MIXING CONSISTENCY

Constraints may be defined as relations between objects that should always be satisfied. Constraints are interesting because they are stated declaratively by the programmer, thereby avoiding him to program complex algorithms. Constraint techniques are traditionally divided into two categories: constraint satisfaction algorithms (CSP) and constraint propagation algorithms. CSP are used mostly for solving complex combinatorial problems, and are particularly efficient on finite domains, but are usually not usable for reactive systems, which makes them unsuitable in our context. Constraint propagation algorithms are particularly relevant for building reactive systems (see e.g. (Hower & Graf, 1996) for a review), typically for layout management of graphical interfaces, from the pioneer *ThingLab* system (Borning, 1981), to *Kaleidoscope* (Lopez et al., 1994) and more recently *OTI Constraint Solver* (Borning & Freeman-Benson, 1995).
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 remain 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 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.
- Constant Distance Ratio : the constraint states that two or more objects should remain in a constant distance ratio to the listener.
- Radial and Angular Limits of Sound Sources : these constraints allow to impose limits in the possible regions of sound sources. These limits are represented by circles or radial lines whose center is the listener's avatar (as represented graphically in Figure 8).
- Grouping constraint : this constraint states that a set of sound sources should remain grouped, i.e. that the distances between the objects should remain constant (independently of the listener's avatar position) : it might be used to make the distinction between several sets of source.

5.3 CONSTRAINT ALGORITHM

The examples of constraints above bring to evidence major properties of mixing constraints. First, these are not linear, as for instance, the constant energy level constraint. This prohibits the use of simplex-derived algorithms, such as (Borning, 1997). Second, the constraints are not all functional. As an example, geometrical limits of sound sources are typically inequality constraints. Finally, the constraints induce cycles: 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. Indigo (Borning et al., 1996) is an algorithm for functional constraints with inequalities, but does not handle cycles. Conversely, cycle solvers such as Purple (linear constraints) and DeepPurple for linear inequalities (Borning & Freeman-Benson, 1998), do not handle non linear constraints. The general solution as proposed in the literature consists in using hybrid algorithms such as Detail or UltraViolet. However, these algorithms add a considerable level of complexity: they are difficult to implement and tune, and may have unexpected behavior (Borning et al., 1996). We designed an ad hoc algorithm, described in (Pachet & Delerue, 1998), which enforces non linear, non dataflow constraints, with inequalities, and without constraint hierarchies.

The interface for setting constraints is illustrated in Figure 8 : 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 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", which show a circle centered on the listener's avatar to display their scope.

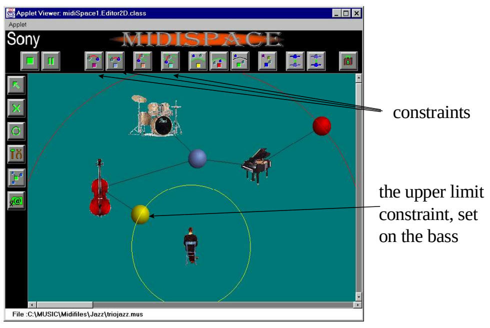

Applet Viewer: midiSpace1.Editor2D.class
Applet
Sony
MIDISPACE
constraints
the upper limit constraint, set on the bass
File :C:\MUSIC\Midifiles\Jazz\triojazz.mus

Figure 8. The MidiSpace authoring interface for specifying mixing constraints

5.4 CONSTRAINTS AS ANNOTATIONS

The final step in MidiSpace is to allow constraint sets to be dynamically created in time, to reflect changes in music. A simple way to do so is by considering constraints as particular annotations. This requires two additions to MidiSpace : 1) a format for expressing constraints compatible with our annotation format, and 2) a scheme for adding and removing constraints in real time during listening. For the moment, we experimented with a simple approach in which the format of constraints is similar to the format for other annotations, i.e. a time tag, followed by a constraint type (out of a predetermined number of constraint types), and parameters when needed. In this scheme, constraints are represented as temporal objects with a start time and a duration (see Figure 9). It is to be noted that since constraints are considered as fully-fledged objects, they also can be moved by the user, and the movements may in turn be considered as temporal annotations.

```
[constraints]
startBeat=7.692                    duration=2000
type=CtEnergyLevel instr=bass, drums
startBeat=7.698                    duration=10000
type=CtEnergyLevel instr=bass, piano
startBeat=7.881                    duration=10000
type=CtAntiRelated instr=bass, piano
startBeat=8.067                    duration=20000
type=CtUpperBound  instr=piano  parameter=45
…
```

Figure 9. Constraints as annotations.

The difficult part is the real time handling of the constraint set, and raises two problems:

1) the incrementality of the algorithm. The constraint propagation algorithm should be able to incrementally add or remove constraints, without having to recompute too many variables.

2) the smoothness of the interface. A main concern in building mixing interfaces is that the user should not be lost because of too sudden movements of objects. When a constraint is added, it can be the case that some objects are in positions that violate the constraint. In this case, a natural solution would be to have the objects move smoothly from the old location to the new one. This second issue is not yet handled, and is the subject of current work.

6. IMPLEMENTATION

The implementation of MidiSpace consists in 1) translating Midi information and annotation files into a set of objects within a temporal framework, 2) scheduling these temporal objects using a real time scheduler.
The Parser task is to transform the information contained in the Midi file and in the annotation files into a unified temporal structure. The temporal framework we use is described in (Pachet et al., 1996), 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*. More information are given in (Delerue & Pachet, 1998).
The scheduling of MidiSpace objects is based on Midishare (Orlarey et al. 1989), 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. The main loop of the Midi player consists in 1) creating a task that schedules all events that fall on the current date, and 2) rescheduling the task to the next date. When an event is scheduled, it is sent the method *play(t)*, with the date as parameter. This method is implemented in all subclasses of *PlayableObject*. Note objects implement the *play(t)* method by sending an appropriate Midi message.

The interpretation of annotation is implemented similarly. Defined as particular *playableObjects*, annotations respond with a *play(t)* method during the listening that leads to a special behaviour: instead of triggering a midi output, the method results in a action that can be redefined according to, e.g. user preferences. For instance, as mentioned in section 4.2.1, a chorus annotation will be interpreted as a motion of the lead instrument towards the listener at the beginning of the chorus, followed by a motion towards its original place at the end.

Thanks to this representation, it is straightforward to implement dynamically changing constraint sets, by simply ensuring that constraints implement the *PlayableObject* interface, and the method *play(t)*. This method will simply add the constraint to the current constraint set at the start date, and remove it at the end date.

7. CONCLUSION, FUTURE WORKS

The MidiSpace system shows that it is possible to give users some degree of freedom in sound spatialization, while preserving some semantics on the mixing of sound sources. We showed how to introduce annotations in MidiSpace to 1) represent various type of semantic information on the music being player, and 2) exploit this information in real time to enrich the listening experience.
The prototype built so far validates our approach, but future work remains to be done in several directions. First, we are experimenting with other interfaces for navigation. A 3D version of MidiSpace in VRML is in progress (Pachet & Delerue, 1998b), in which the VRML code is automatically generated from the 2D interface. Although the result is clearly stimulating, it is not yet fully satisfying because the 3D interface gives too little information on the overall configuration of instruments, which is a crucial parameter for spatialization, but this problem is a general problem with 3D interface, and is not specific to MidiSpace. Second, an audio version 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 2.

8. REFERENCES

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

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.

Borning A., Anderson R., Freeman-Benson B., “Indigo: A Local Propagation Algorithm for Inequality Constraints”, *Proceedings of the ACM Symposium on User Interface Software and Technology*, pp. 129-136, 1996.

Borning A., Freeman-Benson, B. “The OTI Constraint Solver : a Constraint Library for Constructing Interactive Graphical User Interfaces”, *Proceedings of the First International Conference on Principles and Practice of Constraint Programming*, pp. 624-628, 1995.

Borning A., Freeman-Benson, B., “Ultraviolet: A Constraint Satisfaction Algorithm for Interactive Graphics”, *Constraints*, Special Issue on Constraints, Graphics, and Visualization, Vol. 3 No. 1, pp. 9-32, April 1998.

Borning A., “The Programming Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory”, *ACM Transactions on Programming Languages and Systems*, 3 (1981), pp. 353-387.

Borning, A. Lin, R., Marriott, K. “Constraints for the web”, *Proceedings of ACM Multimedia Conference*, Seattle, pp. 173-181, 1997.

Burgess D.A., “Techniques for low-cost spatial audio”, *ACM Fifth Annual Symposium on User Interface Software and Technology (UIST '92),* Monterey, November 1992.

Chowing, J. (1971), “The simulation of moving sound sources”, *JAES*, vol. 19, n. 1, p. 2-6.

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.

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.

Delerue O., Pachet F., “MidiSpace, un spatialisateur Midi interactif”, Journées d’Informatique Musicale - Jim 98, Marseille, 1998.

DirectX 1998, Microsoft DirectX Reference Manual, available at: http://www.microsoft.com/ msdownload/directx/dxf/sdk5.0/default.htm

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.

Hosobe H., Matsuoka S. Yonezawa A., “Generalized local propagation: a framework for solving constraint hierarchies”, *Proceedings of CP’ 96*, Boston, August 1996.

Lopez G., Freeman-Benson B., Borning A., “Kaleidoscope: A Constraint Imperative Programming Language”, In *Constraint Programming*, B. Mayoh, E. Tougu, J. Penjam (Eds.), NATO Advanced Science Institute Series, Series F: Computer and System Sciences, Vol 131, Springer-Verlag, 1994, pages 313-329.

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.

IMA, “MIDI musical instrument digital interface specification 1.0”, Los Angeles, International MIDI Association, 1983.

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.

Laurson M., Duthen J., “PatchWork, a graphical language in PreForm”, *Proceedings of the International Computer Music Conference*, San Francisco,172-175, 1989.

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

Meyer L., *Emotions and meaning in music*, University of Chicago Press, 1956.

Orlarey Y., Lequay H. “MidiShare: a real time multi-tasks software module for Midi applications”, *Proceedings of the ICMC*, 1989, ICMA, San Francisco.

Pachet F., Delerue O. “A Temporal Constraint-Based Midi Spatializer”, *ACM Multimedia Conference, Brighton, 1998.*

Pachet F., Delerue O. “A Mixed 2D/3D Interface for Music Spatialization”, *First Virtual Worlds Conference, Paris,* 1998b.

Pachet F., “Computer Analysis of Jazz Chord Sequences. Is Solar a Blues ?”, *Readings in Music and Artificial Intelligence*, Harwood Academic Publishers, to appear, 1998.

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.

Sloan Donald, “Aspects of Music Representation in Hytime/SMDL”, *Computer Music Journal*, Cambridge, MA, MIT Press, 17:4, Winter 1993.

SMDL, Draft International Standard, ISO/IEC CD 10743, 1995.

Taube H., “Common Music: A Music Composition Language in Common Lisp and CLOS”, *Computer Music Journal*, vol. 15, n° 2, 21-32, 1991.

Vander Zanden Brad, “An incremental algorithm for satisfying hierarchies of multi-way dataflow constraints”, *ACM Transactions on Programming Languages and Systems*, 18(1):30-72, 1996.