ON-THE-FLY MULTI TRACK MIXING

François Pachet, Olivier Delerue

Sony Computer Science Laboratory
6, rue Amyot
75005 Paris - France
PH: +33 1 44 08 05 16 FAX: +33 1 45 87 87 50
e-mail: pachet@csl.sony.fr, delerue@csl.sony.fr

Abstract - We propose a system for performing on-the-fly mixing of audio sources, to produce spatialized sound. We introduce to this aim a constraint system linked to 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, the constraint system is activated and attempts to enforce the constraints that may have been violated. The resulting system allows to perform mixing on-the-fly, while letting the user choose between several listening viewpoints in a coherent manner. We report on the design of the system and an audio implementation using Microsoft DirectX API.

I. INTRODUCTION

Mixing is a crucial process on which depends the quality of the music reproduced. It is common place that good music is much more than the simple juxtaposition of individual tracks.

In the traditional music production chain, mixing can be seen as a clearly defined process, which consists in reducing multi track data into some multi channel format. It is during mixing that composers or sound engineers determine the nature and quality of the music that listeners will eventually hear. The role of the rest of the music reproduction chain is simply to ensure that the listener will actually listen to the mix as designed in the mixing phase. This scheduling of tasks has two major consequences. First what happens between mixing and actual listening is out of control from the sound engineer/composer. Secondly - and conversely - listeners are not taken into account, and in particular have no control over the mixing. This paper introduces the concept of on-the-fly mixing, which precisely addresses these two issues: ensuring that the sound engineer/composers intentions are always preserved in the music reproduction and allowing some degree of user control.

Moreover we believe that the listening experience can be highly enhanced by providing users with some control over the spatialization of the sound sources. However, changing spatialization arbitrarily induces the risk that important properties of the configuration of sound sources are no longer preserved. We propose a system in which 1) the properties of mixing can be specified in an abstract way by the sound engineer, and 2) these properties are enforced automatically when the user changes the positions of sound sources. This system - MusicSpace is based on a constraint language and a constraint solver, and is designed to control an arbitrary audio spatializer.

Most of the work in audio spatialization has concentrated so far in building software to produce so-called multi-channel sound, from a set of audio tracks. These spatialization techniques are mostly used for building virtual reality environments, such as (1) or (5). Moreover, these systems usually provide low level controls on the spatialization that may not be relevant to inexperienced users. More precisely, most of the existing spatialization system give access to their elementary parameters such as the sound source position along three axes, source orientation. We investigate the use of these techniques to provide music listeners high-level control on the localization of sound sources, thereby enhancing the music experience by providing implicitly various listening viewpoints on a given music piece. We report on the design of MusicSpace and on experiments conducted using various spatializers.

II. THE MUSICSPACE SYSTEM

Our system, MusicSpace, allows to perform and control on-the-fly-mixing for a given set of sound sources. The basic idea in MusicSpace is to represent graphically sound sources in a window, as well as an avatar that corresponds to the listener. In this window, the user may either move the instruments or his avatar around. The overall mixing of the music is determined by mapping positions in the interface to positions in the auditory scene and is performed by sending appropriate commands from MusicSpace, to whatever spatialization system connected to it, such as a mixing console, an integrated audio spatializer or a more sophisticated spatialization system (4).

Figure 1: Three sources from a jazz trio example

Figure 1 describes our system's interface while mixing a jazz trio example: one can see the avatar as well as the three sound sources (piano, bass and drums) that compose the jazz trio. The user can move its avatar closer or further to any of the sound sources, or move around the sound sources themselves and perceive the corresponding spatialization changes in the resulting mix.

Moving the sound sources or the avatar around creates potentially a wide range of possible variations over the resulting mix. This leads naturally to the notion of music rendering, when the resulting mix reaches a coherent and precisely defined style.

Music Rendering

The most basic application of our system consists in providing the listeners with a predefined collection of configurations for a given set of sound sources. Each configuration defines the position of the sound sources according to the avatar and possibly their orientation and directivity as well as muting information. These configurations - called renderings - allow to create different styles of mixing, such as the original mixing of the standard distributed CD version, but also for instance an *a capella* or *unplugged* style.

Figure 2: an *a capella* rendering of a popular music title

Figure 2 shows an "*a capella*" rendering example of a popular music title. To achieve the *a capella* style, all the instruments yielding some harmonic content are muted. The various voice tracks (lead singer, backing vocals) are kept and located close to the listener. To avoid a dry mix, we also include some drums and bass, but locate them a bit further from the listener. Several other renderings can be created using this same set of sound sources, such as an "unplugged" version or a "techno" version, as described below.

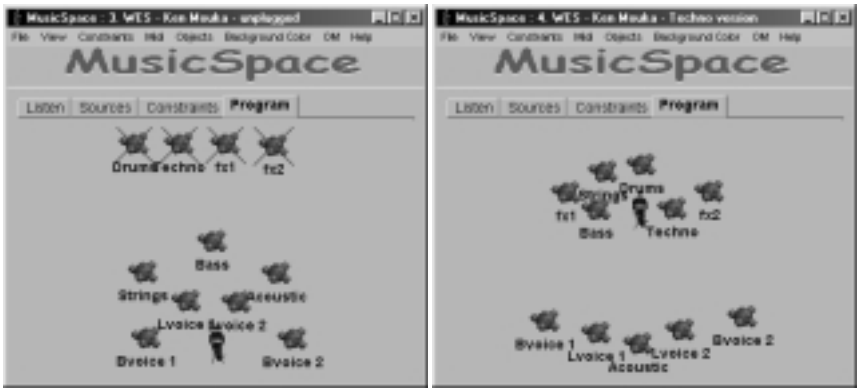

Figure 3: an *unplugged* version (left) and a *techno* version (right) of the same music title

Figure 3 illustrates two different renderings that were created using the same sound sources as in the example shown on Figure 2. On the left is an *unplugged* version of the piece: this rendering consists in muting the electronic sound sources to concentrate only on the acoustic part of the sources. Thus, the voice tracks were arranged around the avatar, along with the strings, bass guitar and acoustic tracks.

Opposite is the *techno* version (Figure 3, right), which put the emphasis on the electronic and rhythmical sources. The various voices and acoustic sources are kept in order to maintain the coherence of the result, but they are drawn to the rear speakers so that the listener does not focuses his attention on them.

The renderings presented in the this section are static configuration of sound sources, in the sense that their position do not change in time. We now turn to dynamic situations where the user is able to influence mixing, e.g. by adjusting slightly the position of some sound sources, thereby bringing some spatialization changes to the proposed rendering while preserving its style.

Constraints for Mixing

The problem with allowing users to change the configuration of sound sources, and hence, the mixing, is that they do not have the knowledge to produce coherent mixings. Indeed, the knowledge of the sound engineer is necessary to transform high-level commands into sets of atomic parameter changes. For instance, a sound engineer may want to reinforce the presence of a given instrument, say the piano. Meanwhile, he also has to ensure that the overall energy level of the recording remains between reasonable boundaries. Additionally, several sound sources may be logically dependent, such as the bass, drum and guitar tracks composing the rhythm section. Finally, the composer may have given instructions that some instruments should always remain within a given spatial range. The initial desire to bring an instrument closer will therefore result in a complex set of atomic actions on the parameters of the mixing table or spatialization.

To be accessible to inexperienced users, our system needs to be able to interpret one elementary user action (such as "bring the piano closer" for instance) into its corresponding meaning and thus induce the complementary atomic actions that preserve the consistency of the result. To achieve this goal, we built up the system with a set of information on the mixing, specially designed for the set of sound sources to be mixed.

We illustrate this idea with the jazz trio example as show in Figure 1: although this example is very simple as it brings into play only a small number of sound sources, one can yet express several important properties that should be satisfied in order to preserve a coherent mixing result. For instance :

- The notion of trio should be kept, i.e. the three instruments can not be handled independently from each other,
- All instruments should be hearable, i.e. not too far from the listener,
- The drums and bass represent the *rhythm section* , and therefore should be somehow related,
- The piano is the soloist instrument,
- One should be able to distinguish clearly between the soloist and its accompaniment, the rhythm section.

We propose to encode this type of knowledge as *constraints*, which are relations that should always be satisfied, interpreted in real time by an efficient constraint propagation algorithm (3). An important property of these constraints is that they are stated declaratively by the sound engineer or the composer. Since they can easily be added or removed from the constraint graph, the whole set of constraints can be defined incrementally, along with the testing phase. These constraints are represented in the interface by a round icon, whose color identifies the type of constraint it represents. Line segments represent the connections between each constraint and its constrained sound sources.

Figure 4: setting constraints on the jazz trio example

Figure 4 shows the jazz trio example of the Figure 1 to which we added a set of constraints that ensure that any user modification of the position of the sources will result in a consistent mixing. Particularly, these constraints were set up in order to achieve the properties that were mentioned previously: the bass and drums are related with a constant angular offset constraint and a constant distance ratio constraint. As a result, the balance between bass and drums remains constant and the two instruments remain in the same region of space. These two instruments form the rhythm section that, due to two angular limit constraints, has to remain in the right part of the auditory scene. This rhythm section was opposed to the soloist – the piano – using a constant distance product constraint: when the user drags the piano closer to its avatar, the rhythm section is moved further – keeping its balance constant – so that the overall sound volume does not change. Finally two distance limit constraints are attached to the piano: an upper limit that ensures the piano can always be heard and a lower limit so that the piano is not too loud.

When the user moves a sound source, a constraint propagation algorithm is triggered: all the constraints attached to this source are notified and try – and only if necessary – to send perturbation to the other constrained sound sources in order to balance the relation they are representing. In turn, the sources notify the constraints of there new position. The constraint graph is thus examined until the constraints relations are all satisfied again or a contradiction is raised either by a limit that is being violated or by two contradictory constraints referring to the same sources. According to the result of the propagation place, the user action can be accepted in which case all the sources positions are updated or refused : the modified source remains at its original position.

We defined a set of constraints appropriate for specifying “interesting” relations between sound sources including for instance a Constant Energy Level constraint, a “Constant Angular Offset” constraint, or a “Radial Limits of Sound Sources” constraint. Moreover, a number of specific “animated” constraints have been defined : these constraints allows to specify a number of elementary automatic movements such as rotations, translations and random walks, that can in turn be combined or constrained to produce some more complex trajectories.

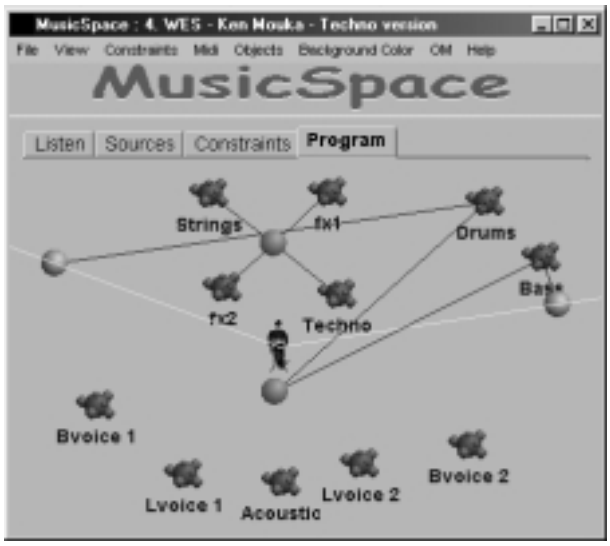

Figure 5: the techno version rendering enhanced with animated constraints

Figure 5 illustrates this point by describing the techno rendering as shown on Figure 3 (right), to which we added to ‘rotating constraints’. The first one is set on the strings, sound effects 1 and 2 and the techno sound sources. As a result these sources fade in and out progressively as they get closer and further to the listener. The second rotating constraint was set on the rhythm section : the drums and bass sound sources. As the center of this rotation is almost located at the same place as the listener’s avatar, the overall volume of the rhythm section does not change. However, these sound sources will bounce periodically between the left and the right part of the frontal auditory scene, according to the two angles limits that were set up on the drums and the bass.

On the technical point of view, our constraint propagation algorithm handles non linear, non functional constraints with cycles, and reaches our requirements with predictable and reactive behavior. Its implementation allows to

define new constraints type easily and can thus be extended simply. More information on the constraint algorithm can be found in (7).

High Level Handles

Moving around sound sources does not fit to end users needs. Moreover, when a large number of sound sources is being used, the system, although constrained, can become confusing and meaningless for inexperienced users.

To address this problem, we introduce higher level control objects that we call *handles*. These are meant to represent an important feature of the mixer that makes sense to end users. These handles encapsulate a group of sound sources and their related constraints into a single interface object. These handles are implemented by so-called “one way constraints”, a lightweight extension of the basic constraint solver. Thanks to handles, the user may easily change the overall mixing dynamically. Several handles may coexist in a given configuration, providing the user a set of coherent alternatives to the traditionally imposed unique mixing.

Figure 6 illustrates our interface in its two main operating modes: on the left is the listen mode, intended for end users in which we represent only those high level parameters. Six different handles have been designed for this particular popular music title : a handle lets the user adjust the acoustic part of the sound sources, another the synthetic instruments, as well as one for the drums and for the voices. The “plug” handle represents a balance control between the acoustic and the synthetic parts: bringing the “plug” handle closer to the listener will enhance the synthetic part and give less importance to acoustic instruments, and vice versa. Finally, a “volume” handle allows to change the position of all sound sources simultaneously in a proportional manner.

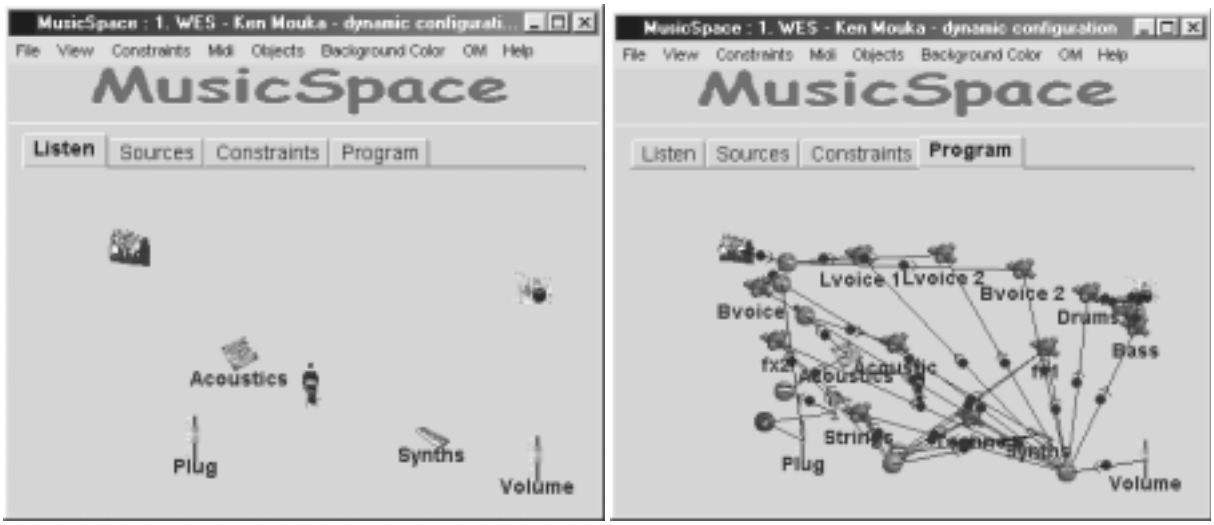

Figure 6: the MusicSpace system while in the ‘listen’ mode (left) or in the ‘program’ mode (right).

This example makes an extensive use of the constraint system to build the connections between the sound sources and the handles. The right part of Figure 6 displays the interface of our system when in its “program” mode. Since this mode was typically designed to create the set of constraints and handles that best fit a given number of sound sources, all the elements for the spatialization are represented: handles, sound sources, constraints and one way constraints.

In a same way, for the jazz trio example, a “piano” handle would replace the piano itself in order to encapsulate all the atomic actions that have to be taken when the piano is move closer to the listener or further.

III. IMPLEMENTATION AND EXPERIMENTS

This section describes the general architecture of the system and the set of experiments we conducted, using various spatialization devices. The two main components of the system are its graphical interface and constraint solver. Both are independent to the spatialization system connected to perform the audio processing and that, therefore, can be easily replaced.

Input of the system are user actions, performed in most cases using the mouse or the keyboard. Output is a set of real-time spatialization commands that are transmitted to the spatialization system using a specific communication protocol.

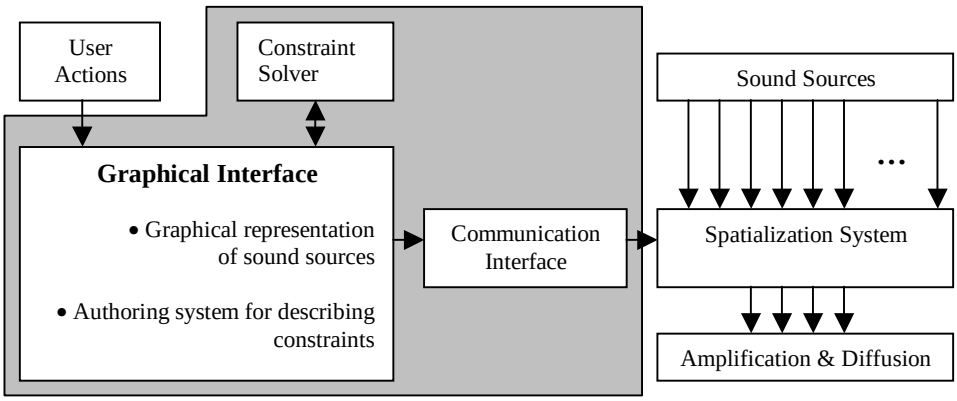

Figure 7: general architecture of the system

The generic architecture of the system as shown in Figure 7 is then adapted according to the spatialization system that is being used: for instance, the communication protocol will be based on a MIDI interface when using an external mixer as the spatialization system. This Midi implementation is also in used when controlling the Ircam Spatialisateur. For the full audio version (see next section), a general scheme for designing the organization of sound sources and constraints was added to the system in order to be able to predefine mixing configurations for a given set of sound sources.

The original version of our system was performing midi spatialization. In this case, the system holds two additional components, a Midi parser and a Midi player:

- The Midi parser reconstructs sound sources from a midi file: indeed the track organization that can be found in a Midifile reflects more the composer's work organization than a sound source organization and the parser may need to split Midifile tracks or, on the contrary, combine tracks in order to recreate a sound source oriented configuration.
- The Midi player handles the scheduling of midi events.

Midi spatialization is performed by a specific communication interface that sends midi control change messages on controllers number 7 – volume – and 10 – panoramic – to the synthesizer. Despite of its poor accuracy, this preliminary version allowed to conduct the first experiments with the constraint algorithm. More information concerning the Midi implementation of the system and its components can be found in (6). All the midi communications performed by our system are processed by the Midishare Midi environment (2).

Later, a set of experiment using various spatialization systems where conducted in order to access more accurate sound spatialization rendering and parameters. We now discuss these experiments, targeting various application of out system, from end-users to professional.

Controlling a mixer

First, MusicSpace was connected to a midi-controlled audio mixing console. Midi communication was used between the two systems. In this case, each channel of the mixing console is assimilated to a sound source in the MusicSpace interface and is thereby represented individually with an icon. We experienced this setup in two different loudspeakers setup.

When mixing in the traditional stereo configuration, distance and angular position with respect to the avatar in the graphical interface are mapped to the volume and panoramic parameters of the corresponding console channel. However, our system allows also to use the mixer for different loudspeaker configurations. For instance, when mixing for a quadraphonic loudspeaker setup, by making use of individual auxiliary buses to perform the spatialization : switching from stereo to quadraphonic mixing is performed only by changing the set of spatialization commands that are sent from the interface to the mixer.

Figure 8: the MusicSpace system connected to a mixing console

We show that our system allows to extend the possibilities of the mixer in two different ways. First moving the avatar in the interface induces on the mixer a change of parameters that is too complex to be performed directly by the sound engineer: not only the number of controlled parameters is often too high to be controlled by a single person in real time, but also, the parameters involved may not all be accessible simultaneously from the hardware interface of the console. Secondly the channels of the mixing console obey to the constraints set up on their corresponding objects in the interface : for instance, setting up a “constant distance ratio” constraint between a number of channel will have a similar action as grouping these channels directly on the console. However, using different type of constraints in the same context will induce a variety of behavior of the console’s channels that could not be defined directly from the mixing console. For instance, if a set of channels of the mixer are related in the interface with an opposition constraint, moving up the level of one of these channels will decrease proportionally the volume of the other channels.

When connected to a mixer, the MusicSpace system can received process user actions coming either from the mouse of from the mixer. In this case, particular care is taken to avoid processing the result of an output (a change in a channel slider position for instance) as a new input to the system and thus entering a feedback loop.

Connection to the Ircam’s Spatialisateur

Among existing spatialization systems, the Ircam’s Spatialisateur (4) is one of the most accurate in terms of sound source positioning and room acoustics. Besides the precision of its audio rendering, it provides a number of spatialization parameters that are not accessible in other systems such as the sound source orientation or directivity.

These specific parameters can be easily represented in the interface and constrained just as sound sources objects. As a result, one can efficiently describe how a sound source orientation for instance should evolve when the source itself is moved around: a specific constraint can ensure a sound source is always oriented toward the listener, whereas another one will ensure that the source will be always oriented in an absolute direction, independently to the

listener's position. The Ircam's Spatialisateur affords a Midi interface, so it can run on a separate machine, and be controlled by Midi, to alleviate the cpu.

Full integrated audio version

The latest version of MusicSpace is an integrated audio version, based on the Microsoft DirectX API to perform sound spatialization. DirectX provides parameters for describing 3D sound sources that can be constrained, such as the source position, orientation, directivity and Doppler. As with other spatialization systems, the MusicSpace constraint library allows to constrain any of these parameters.

The implementation of this audio version implied the creation of 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 (MusicSpace). Our dll also handles the streaming of audio files to the DirectX buffers, by implementing timers and managing explicitly buffers under the windows 98 platform.

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.

One important issue in our audio version of the project concerns data size and access timing, i.e. to the audio files to be spatialized. The current performance of hard disks allow to read a large number of audio tracks independently. However, these tracks require a lot of disc space: a typical music example lasts three and a half minutes and is composed of about 10 independent mono tracks: the required space for such a title is more than 200 megabytes.

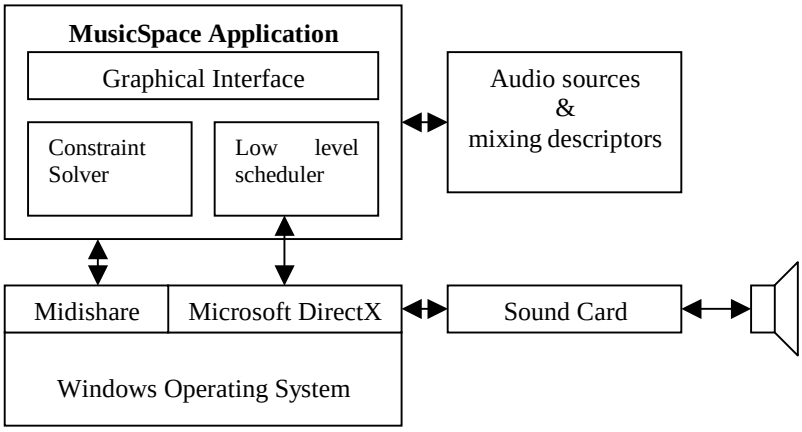

Figure 9: the MusicSpace audio version implementation

Experiments conducted with other type of storage, such as CD-Rom for instance showed serious limitations: reading independently a large number of tracks from a CD-Rom is currently not possible. The solution we propose consists in interlacing the different audio tracks in a single file: the reading head does not have to jump continuously from one position to another to deliver the samples for each track, and the samples are read continuously. The WAV format supports multi-track interlaced files. However, this solution brings also a number of important limitations. First, since each track has to be read, muting a track will not release any CPU resource. Second, the synchronization between each track has to be fixed once for all: it is not possible to offset one track according to another for instance.

Eventually, each track has to be read at the same speed or sample rate. This is a limitation since it prevents from using the DirectX Doppler effect for instance, which is implemented by shifting slightly the reading speed of a sound file according to the speed and direction of the source with respect to the listener. If these limitations apply in specific and experimental applications, they do not block our goals: the number of tracks for a music title can be fixed in advance and there are no reason why the offset between the tracks should be modified.

Our audio version of the system runs on a standard personal computer platform with the Windows 98 environment. Experiments were driven on a multimedia personal computer, equipped with a Creative Sound Blaster Live sound card and outputting to a quadraphonic speaker system. These experiments with DirectX showed that a minimum of 20 sound sources can be easily processed in real time.

IV. CONCLUSION

Today, the technical limitations that prevented from processing on-the-fly mixing on end-users systems have been surpassed: backend devices for listening music are more and more designed for a multipoint diffusion system such as the 5.1 surround format. Consequently large capacity storage for multi track recordings have been designed such as the Sony SACD or DVD. Within this new end-user applicative slot, we propose a system, MusicSpace, that allows to control on-the-fly-mixing in a consistent way: it brings more freedom to listener by providing them with the possibility to either choose between different rendering of a given music title or to control directly and dynamically the spatialization of the sound sources of this music title. Moreover MusicSpace make these parameters available to anyone, and especially to non expert users: firstly by ensuring that a number of properties of the mixing, expressed either by the composer or a sound engineer, will be always verified. Secondly by hiding the original low-level parameters of the spatialization system in use and presenting high level control parameters whose meaning can be understood by everyone. Finally, MusicSpace has applications also outside the field of spatialization. MusicSpace can be used for any situation where:

- Streams of real time data can be controlled by discrete parameters (e.g. streams of audio sources controlled by distance, pan, directivity, etc.),
- 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).

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) Dominique Fober, Stéphane Letz, Yann Orlarey, « Midishare joins the Open Source Softwares », in Proceedings of the 1999 International Computer Music Conference.

(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) François Pachet, Olivier Delerue, « MusicSpace: a Constraint-Based Control System for Music spatialization », in Proceedings of the 1999 International Computer Music Conference

(7) Pachet F., Delerue O., “A Temporal Constraint-Based Music Spatializer”, ACM Multimedia Conference, Bristol, 1998.