Dynamic Audio Mixing

François Pachet (1), pachet@csl.sony.fr
Olivier Delerue (1), delerue@csl.sony.fr
Peter Hanappe (2), peter@hanappe.com

(1) SONY CSL, 6 rue Amyot, 75005 Paris, France
(2) 115, rue du Chemin Vert, 75011 Paris, France

Introduction

3D sound is a powerful technology which allows to give a realistic impression of localization of sound sources. This technology is now mature enough to produce a fine-grained spatialization on a set of audio sound sources in real time. However, the are serious limitations in the way this technology is used in current practice. Firstly, composers and sound engineers have great difficulty in mastering this technology to produce 3D sound pieces, because the corresponding controls are too low-level and intricated. As a cobnsequence, the potential of this technology is not fully exploited. In particular, listeners are still considered as passive receivers, and do not have any control on the spatialization of the pieces they listen to.

In this paper, we introduce the notion of dynamic mixing, which addresses these two problems: 1) provide composers and sound engineers with a powerful paradigm to compose easily musical pieces in space and time, while 2) allow listeners some degree of control on the spatialization of the music they listen to.

Previous work on this project (see [3]) introduced a novel means for controlling the spatialization of sound sources based on the constraint paradigm. The system that was presented targeted a midi output or midi based communication with mixing devices. Recent developments of the project lead to a full audio version that overcomes the restrictions imposed by midi based music. We discuss in the later part of this paper the technical issues concerning the implementation of the audio extension.

Dynamic Mixing ?

We believe that the listening experience can be highly improved by postponing the mixing process at the latest possible time in the music listening chain, that is during listening. Instead of delivering the music in the traditional ready-to-use mixed form, designed for an imposed reproduction set-up (stereo, Dolby DTS,…), the key idea of dynamic mixing is to deliver independent musical tracks that are mixed or spatialized together at listening time, and according to the diffusion set-up.
To do so, we attach to the audio tracks a set of instructions describing how the musical tracks should be mixed and what are the important relations to be maintained between the sound sources. Thus, beyond its adaptability to the diffusion system, on-the-fly mixing brings also more freedom to listeners: since several such mixing descriptions can be provided for a single music piece, the listener can choose between several renderings of the piece to emphasize specific musical dimensions, or to fit with its particular taste.

Musical Rendering

Having access to the individual tracks of a given music title allows to create several arrangements of the same set of sound source. The first possibility is of course to recreate the original mixing of the standard distributed CD version. It is also possible to define alternative configurations of sound sources, as illustrated below.

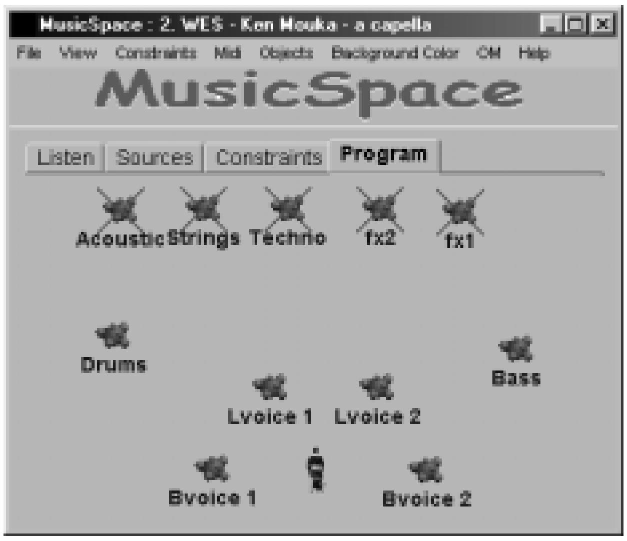

Figure 1: An "a capella" rendering of a musical piece

Figure 1 shows an "*a capella*" rendering example of a 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 animated mix, as described below.

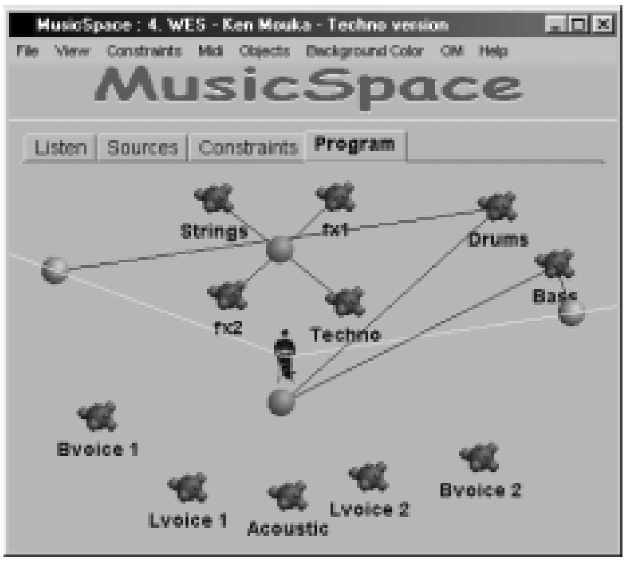
MusicSpace : 4. WES - Ken Mouka - Techno version
File View Constraints Midi Objects Background Color OM Help
MusicSpace
Listen Sources Constraints Program
Strings
fx1
Drums
Bass
fx2
Techno
Bvoice 1
Lvoice 1
Acoustic
Lvoice 2
Bvoice 2

Figure 2: A techno version with animated constraints

Figure 2 displays a "techno" rendering of the same music title : emphasis is put on the synthetic and rhythmic instruments that are located to the front in the auditory scene. To maintain consistency in the result, we kept the voices tracks as well as the acoustic instruments, but locate them in the back so they do not draw all the listener's attention.
Animated constraints were used for this rendering, to bring variety to the resulting mix. The strings, sound effects and techno tracks are related together by a rotating constraint so that emphasis is put periodically on each of them as they come closer to the listener. Drums and bass tracks are also related with a rotating constraint but some angle limit constraints force their movement to oscillate alternatively between left and right sides.

High level handles

To enhance the usability of the system, we further introduce the notion of user "handle", which encapsulates 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.

In the example shown on Figure 3, the sound sources are not shown any more: the user has only access to a set of proposed handles that were created specially for the music title. In this case, 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.

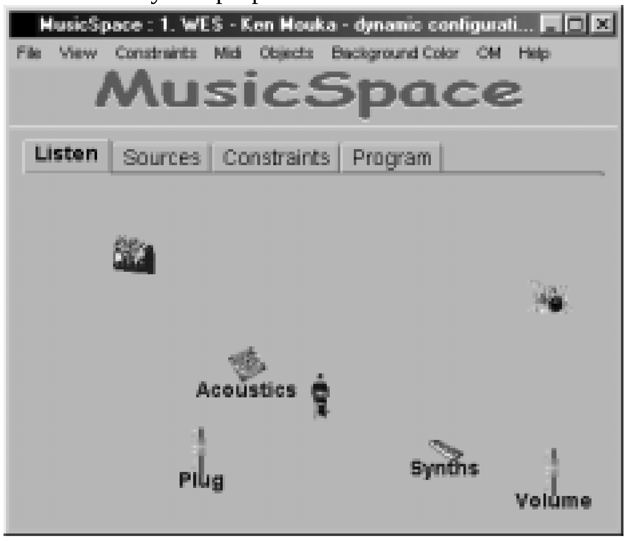
MusicSpace : 1. WES - Ken Mouka - dynamic configurati...
File View Constraints Midi Objects Background Color OM Help
MusicSpace
Listen Sources Constraints Program
Acoustics
Plug
Synths
Volume

Figure 3: A Dynamic Configuration of the piece; "Listen" mode

The example show in Figure 3 makes an extensive use of the constraint system to build the connections between the sound sources (such as represented on Figure 1) and the handles. Figure 4 displays the interface of our system when it is in "program" mode. In this mode all the elements for the spatialization are represented: handles, sound sources, constraints and one way constraints.

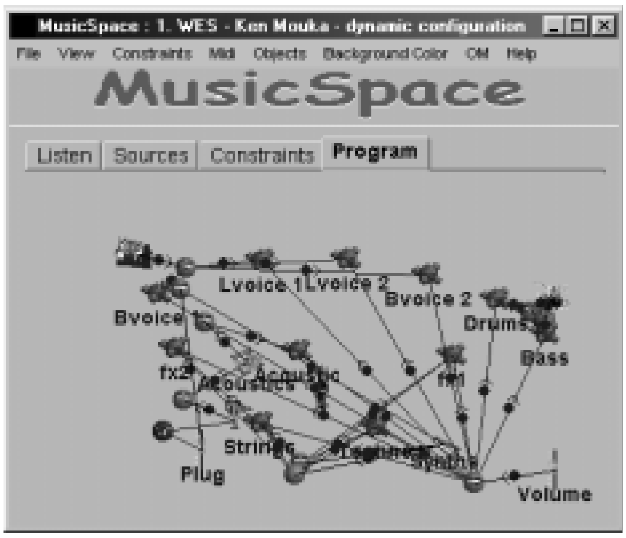
MusicSpace : 1. WES - Ken Mouka - dynamic configuration
File View Constraints Midi Objects Background Color OM Help
MusicSpace
Listen Sources Constraints Program
Lvoice 1
Lvoice 2
Bvoice 2
Bvoice 1
Drums
Bass
fx2
Acoustics
Acoustic
fx1
Strings
Techno
Synths
Plug
Volume

Figure 4: The dynamic configuration; "Program" mode

Technical Issues

Most of the properties of the MusicSpace system concerning its interface and the constraint solver have been discussed in [3] and [4] . We discuss here only the technical issues concerning the audio version.

The system presented here has two main modules: 1) a control system, which generates high level spatialization commands, and 2) a spatialization module, which performs the real time spatialization and mixing of audio sources. The control system is implemented using the Midishare operating system [2] and a Java-based constraint solver and interface. The spatialization module is an interface to the underlying operating system (Microsoft DirectX [1]) and the sound card. This module takes in charge the real time streaming of audio files as well as the conversion of data types between java (interface) and C++ (spatialization module). We describe here a connection from the control system to the spatialization system. This connection is realized by implementing a low level scheduler which manages the various buffers of the sound card (see Figure 5).

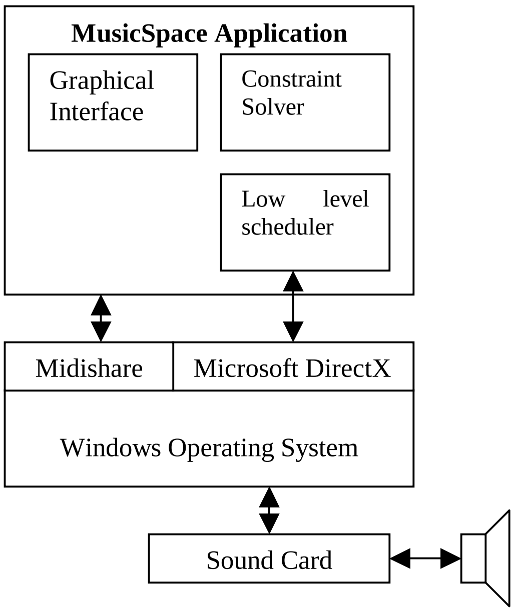

Figure 5: System Architecture

Our system runs on a personal computer platform running Windows 98. Experiments were driven on a multimedia personal computer, equipped with a Creative Sound Blaster Live sound card and outputting to a quadraphonic speaker system: up to 20 individual monophonic sound files can be successfully spatialized in real-time.

Synchronization

Dynamix mixing yields a synchronization issue between the two tasks that write and read from/to the 3D-sound buffer. The reading task is handled by the spatialization system (i.e. DirectX) and our application needs to fill 'in-time' this buffer with the necessary samples.

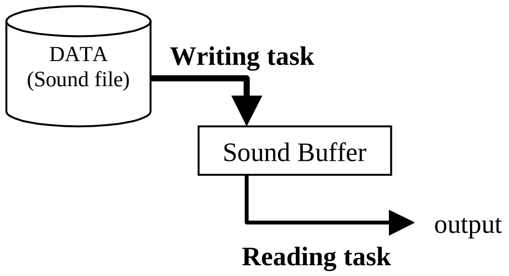

Figure 6: Synchronizing the writing and reading tasks

To achieve a correct synchronization between the audio streaming task (reading the sound files) and the audio output, the standard technique consists in using notification events on the position of the reading head in the buffer (see Figure 7). In this implementation, the reading task notifies the writing task when the reading position has gone over a certain point.

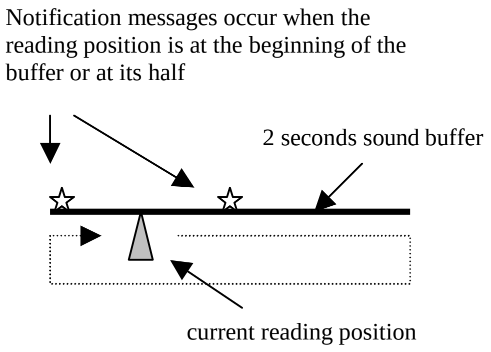
Notification messages occur when the reading position is at the beginning of the buffer or at its half
2 seconds sound buffer
current reading position

Figure 7 : the streaming model

The sound buffer is thus split in two halves and when a notification event is received, the writing task clears and replaces samples for the half of the buffer that is not currently being read.

However, to access these notification events, the buffers have to be handled by the operating system. As a result they cannot benefit from the hardware acceleration features and for instance use the quadraphonic output of the sound card.

The solution we choose consists in creating "static" 3D audio secondary buffers in DirectX. These buffers are physically located in the sound card memory and thus can take advantage of its 3D acceleration features. Since in this case the notification events are not available any more, we replaced them by a "waitable timer" that wakes up the writing task every second. The writing task then polls the reading task to get its current position and updates the samples already read. Since this timer was introduced in the Windows version 98 and NT4, the system cannot be used under Windows 95.

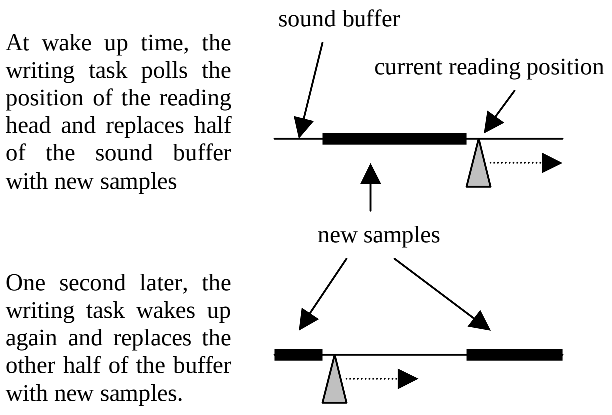
At wake up time, the writing task polls the position of the reading head and replaces half of the sound buffer with new samples
sound buffer
current reading position
new samples
One second later, the writing task wakes up again and replaces the other half of the buffer with new samples.

Figure 8: the "timer" model

Each buffer requires 2 seconds of memory within the sound card: this represents less than 200 k for a 16 bits mono sample recorded at a 44100 Hz sample frequency. Actual sound cards internal memory can contain up to 32 megabytes so the number of tracks the system can process in real time is not limited by memory issues.

Access Timing

One important issue in our audio version of the project concerns data 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.

External supports such as CD-Rom are not as flexible as hard-disks: 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 (see Figure 9): 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, 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 any more 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.

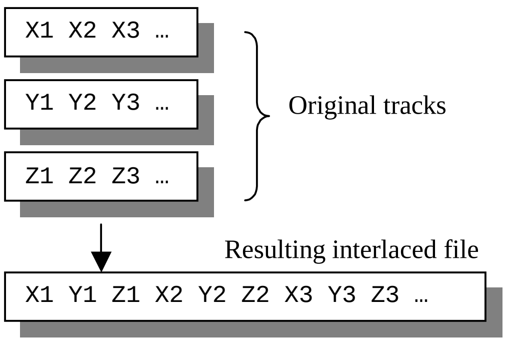

Figure 9: interlacing three tracks

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.

Conclusion

We have introduced the concept of dynamic mixing, and an implementation system – MusicSpace. Dynamic mixing fits naturally with the trend in new music standardization processes such as Mpeg4 and Mpeg7: dynamic mixing constraints are natural metadata. Additionally, the idea of reconstructing on-the-fly the musical piece conforms to the notion of scene description of Mpeg 4. Current work focuses on the design of a fully Mpeg7 compatible system.

References

[1] DirectX : online information http://msdn.microsoft.com/directx/ (home site of the API, download and documentation) and http://www.directx.com/ for programming issues

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

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

[4] François Pachet, Olivier Delerue, « A Temporal Constraint-Based Music Spatializer », in Proceedings of the 1998 ACM Multimedia Conference, Bristol 1998.