From the visualization of sound to real-time sonification: different prototypes in the Max/MSP/Jitter environment

Anne Sedes, Benoît Courribet et Jean-Baptiste Thiébaut
Centre de recherche en informatique et Création Musicale (CICM),

Université de Paris VIII - MSH Paris Nord – France
{asedes,bcourribet,jbthiebaut}@mshparisnord.org

Introduction
We consider the interest of visualizing sound in the context of visual software interfaces, as well as in the interactive arts. Hence, we may imagine a visual space in two dimensions, presented as an overview or in a subjective view, understood as the control space of different dimensions of a sound space. Sound object may be represented as spheres, or color region may visualize parametric variation zone. A controller, as a keyboard mouse, or any joystick may then allow exploring sensitive variation of sound and music.

Beyond scientific visual representations in one hand, and imaginary, poetic and literary illustrations on the other hand, new approaches of sound visualization are emerging due to our relation with the computer science tools, evolving on the mode of emulation and transduction between the man and the machine. In this context, the temptation of real time sonification of image produced by visualization of sound is emerging....

1. About the visualization of sound

We are going to describe several approaches and strategies generally used in order to give a visual representation of a sound or a musical work. We have decided to sort these approaches considering the way the representations are linked formally to the music itself. By “linked formally”, we mean that the visual and audio parts are sharing data in a way or another. In the following section, we will not be discussing the goals that may lead to these strategies, but rather describing their “modus operandi”.

First, we can think about the visual representation of a sound analysis or even the audio signal data itself. The result of the sound analysis is simply displayed with only a few possibilities for changing its visual properties (color scales, range…): for example, the waveform of a sound or its spectrum. It is important to notice that it is possible to use either a still image or a series of images in time (a video): displaying a waveform the way an oscilloscope does or STFT (Short-Time Fourier Transform) frames instead of the spectrum of the whole sound. Dynamic images are more suited to render the dynamic dimension (which is fundamental) of sounds: with a still image, it is possible to foresee what is going to happen and to look “back in time” whereas dynamic images refer to the transiency of the sound phenomenon.

Then, still closely linked to the audio signal, we find the so-called “visualizations” in softwares such as Winamp or iTunes: one of the representations described above is used as a raw material then processed with visual filters (blur, geometric deformations, feedback processes…). Most of the time, even if the sounds are really different, the visual results are similar from one to another.

Still dealing with sound analysis, we have the possibility not to display the result of the analysis by itself, but to use this result in order to generate visual data or to modify a pre-existing video. For example, in the Egosound software (Sedes & al. 2003), the amplitude of a sound is controlling the radius of a 3D OpenGL sphere or the saturation of its color. Another example would be the detection of transients into a music work in order to trigger video processing: a pre-existing video would be processed *according to* the music. The central concept of *mapping* rises from this approach: how will we share data between the domains of sound and video, which relationships do make sense? These questions are being extensively studied thanks to the Max/MSP/Jitter environment, which makes it possible to perform both audio and video processing within the same software.

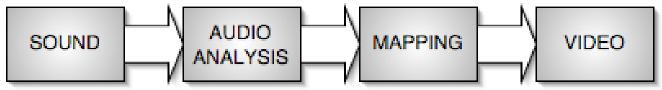


Figure 1 : a sequential approach to sound/visual relations

It is also possible to share data between these two domains without the sequential aspect described previously: global parameters can be sent in parallel to an audio unit and to a video processing unit. The *Egosound* software is a good example of this possibility: the cartesian coordinates of a virtual sound source (we will call them "intentional" parameters) are sent both to an audio spatialization engine and to an OpenGL renderer, so that the software displays what the user would see (the moving sound source) if the sound was visible. Benoît Courribet also used an audio/video granulation engine for his project Isol with percussionist Julien Tardieu: each time a grain of sound was produced, a 3D structure was deformed on the screen.

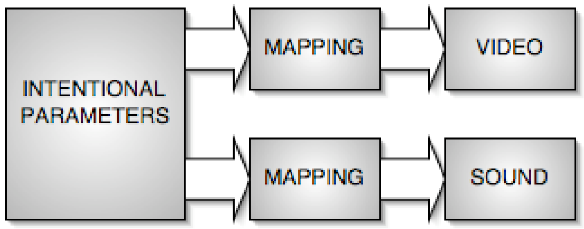


Figure 2 : a parallel approach to sound/visual relations

Finally we can consider many possibilities where no formal link exists between the audio and video parts: the most obvious example in this case is the pop music video clip. Assuming one decides whether or not the video is going to match with the sound characteristics, the results range from short plotted movies with no apparent link to the music (Radiohead's Karmapolice, for example) to extreme synchronization to the music like in Autechre's Gantz Graf video clip. It's interesting to notice that in the last example, it's hard to find out whether or not a formal link exists. Director Alex Rutterford has created this video clip, synchronizing 3D structures frame by frame according to the sonic events. It appears to be a visual annotation of a musical work, in the same way that it's possible to draw colored shapes and pictures on top of a sonogram in the GRM's Acousmographe software: though the sonogram represents a canvas, it is possible for the user to draw whatever shape he wants.

2. visual interface of sound

We have described many ways to give visual representations of sound or music, but it seems like we have focused on either analytic or artistic ways to consider the sound-visual relations. But what we will discuss now is the possibility to create a visual representation of a sound in order to use it as an interface for interactive music composition or audio signal processing.

We have seen previously that we can give what we will call an "intentional" visual representation of a sound by displaying graphically some parameters used for sound generation, audio signal processing, or even structural composition. Now we can think about manipulating this visualization in order to interact with the sound or music. Let's take the example of the MaxMSP's "filtergraph~" object: it's a graphic interface object that displays a magnitude/frequency graph: the frequency response of a filter. It is used together with a biquadratic filter in order to change dynamically the frequency response of the biquadratic filter by manipulating the graph with the mouse. Hence, the graph is both a visual representation of the filtering process (and of the sound itself) and a dynamic processing interface. In the wider domain of real-time computer music, we are now working on a prototypal interface in order to compose interactive sound spaces (we have been discussing these terms in previous articles). This interface consists in a 2D plane where each color represents a parameter. The intensity of the color is linked to the value of this parameter. Each (x,y) pair, where x and y are the coordinates of a point in this plane, corresponds to a pixel and a single value of the parameter. Due to the nature of the "matrix" objects in Jitter, it's possible to have as many different planes (each plane corresponds to a parameter) as desired. It is possible to draw shapes and to use interpolation to smooth the resulting patterns. Anne Sedes and the choreograph Laurence Marthouret work with such an interface to control adaptive sound spaces in real time for the stage.

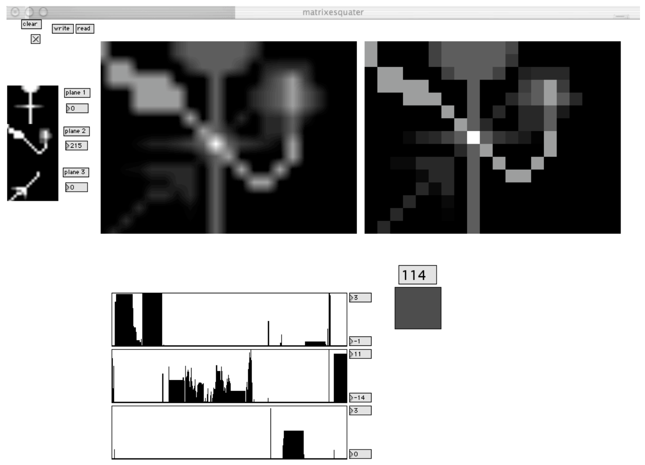


Figure 3 : a prototypal interface for composing sound spaces

3. The *Sonos* software

We will now present the software *Sonos*. The background of this work comes partly from the graphical synthesis (Curtis Roads 1996), which took several forms. First of all, the UPIC system (Unité Polyagogique Informatique du CEMAMu) conceived by Iannis Xenakis and associates researchers in 1977, which has evolved until now as real-time UPIC. In 1993, Vincent Lesbros developed the Phonogramme software. And recently, the software Metasynth (U&I software 1997) which is advertised by : "Paint with sound, compose with light".

3.1. A time/frequency representation

These programs propose a time/frequency representation of the sound and graphical function to draw the sound in time. The conceptual problem is to map a graphical gesture to a musical result, to transpose musical aesthetic consideration to a visual interface. This approach is highly creative but somehow conducts the user to draw a STFT. Influenced by real-time emerging interface like the one we presented earlier, our goal is to provide a more visual control of the sound synthesis.

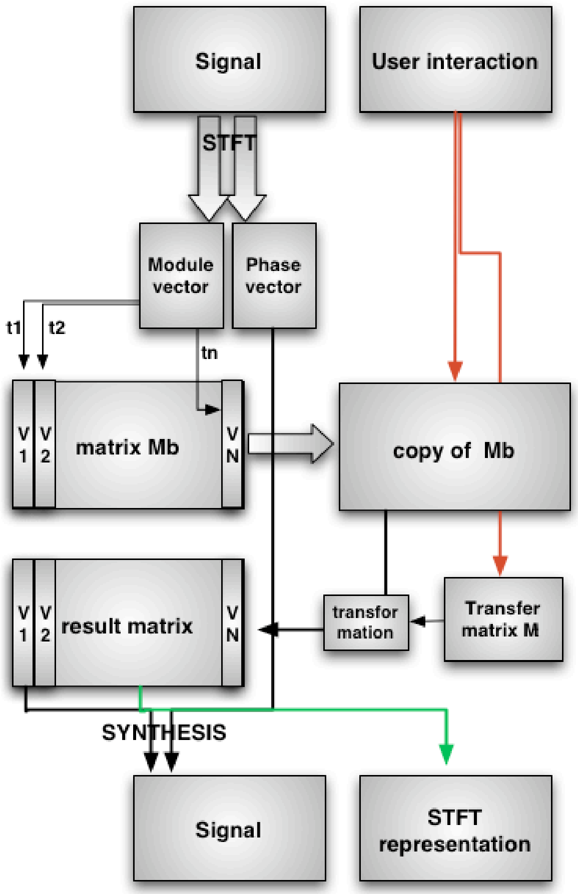


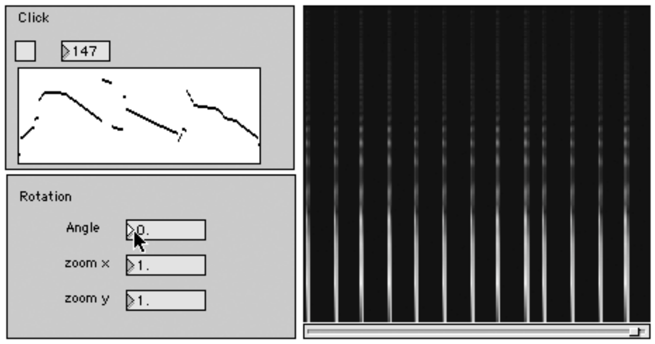
Figure 4 : signal and data flow in the Sonos approach

The aim of this software is to visualize and transform sound in real-time. It uses the STFT representation to visualize the sound and a graphical interface to transform the sound's image. The modified image is then considered as a new STFT and is synthesized to produce a sound. *Sonos* is implemented in the Max/MSP/Jitter environment.

The basic material of the process is the sound itself. The image on which we apply graphical transformation is a STFT representation. We are more dealing with digital effects than with synthesis. This working progress software has two main methods.

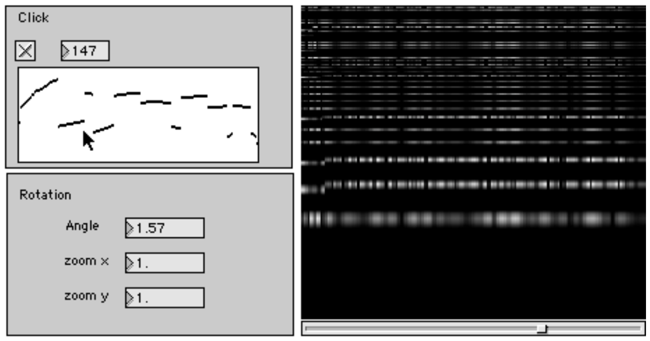
3.2. A first transformation method : dealing with sound like with image

Considering that the sum of *n* vector of the STFT constitute a matrix, *Sonos* apply transformations to a whole set of vector. The basic sound is sequentially written in the matrix. For the display the matrix is resized to a logarithmic view so the user can act on the image the closer to its perception. The operations on the sound are inspired by the manipulation of images in the Jitter environment. Among them we focused on the rotation, zoom, blur and saturation. The rotation of the matrix has a progressive shift effect on the partials which creates a kind of glissando. Its duration depends of the size of the matrix. When the rotation is equal to π/2, the continuous partials of the basic sound become a noisy attack, and the transient attack become a continuous sound. The horizontal zoom-in and zoom-out functions occurs time-stretch effects while the vertical zoom shift the frequency (Luke Dubois 2003).



Click
147
Rotation
Angle 0.
zoom x 1.
zoom y 1.

Figure 5 : STFT representation of a repeated click.



Click
147
Rotation
Angle 1.57
zoom x 1.
zoom y 1.

Figure 6 : Same sound with a rotation of π/2

Some other graphical transformations are implemented in *Sonos* like blur, interpolation or saturation. They have sensible effects on sound.

3.3 Second transformation : graphical transformations using a transfer matrix

The second approach of graphical transformations regards the use of matrixes as control interface. Each plane of a matrix is relied to a transformation, and each pixel stores a value. The user colours the matrix to perform transformations. The horizontal axis represents the time domain, and the vertical axis represents the frequency domain. The drawing matrix is a transfer matrix which is supplied by the basic visualization image and provide a new matrix (i.e. a new sound and a new image). The transfer matrix is sixteen time smaller than the basic matrix. The goal of this reduction is to get a processor gain and to divide the frequency bands into 32 in place of 512. The reduction is applied to the time domain as well. It implies that a transformation is applied to a time band of 371ms. Indeed, each vector corresponds to 11.6 ms of signal, i.e. 512/44100, and 11.6 * 32 = 371.

For the frequency domain, the transfer matrix is scaled according to an exponential law of the form :

$$f(x) = c.2^{k.x}$$

with *c* and *k* constant.

We present here the implementation of two transformation planes. The first transformation consists in a frequency filter. The value stored in each pixel of the plane is a 8-bits value, of a range of [0..255]. This value is scaled to [0..1] and is multiply by the value contain in the bins of the corresponding frequency band. The result, which is the new module of the STFT is displayed in real-time. A cursor shows the current position of the synthesis.

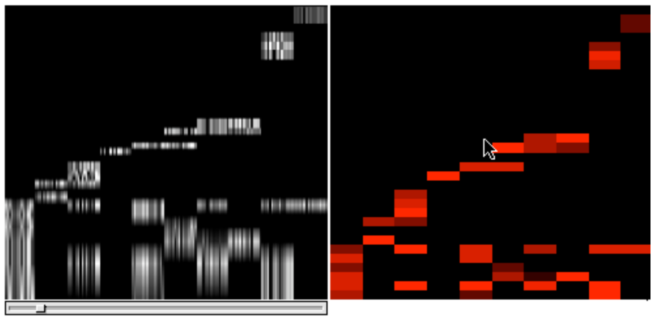


Figure 7 : the right image is the filter interface. The left image is the result of the basic sound filtered by the transfer matrix

The second transformation uses the green plane. It implements a frequency delayed line. The intensity of the color of a bin is also scaled to [0..1]. The value is then interpreted as a delay parameter which allows to repeat a part of the signal corresponding to the colored bin. The delayed sound is then delayed and added to the signal of the following bin.

The transfer matrix supplies the delay values. These values are multiply with a copy of the resulting matrix, which provide a new matrix with the delayed signal. This matrix is then shifted of the size of a bin and added to the result matrix.

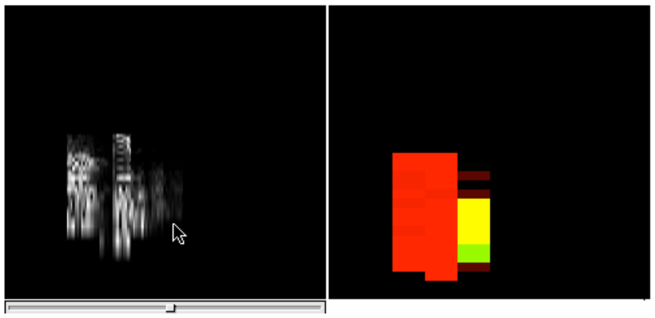


Figure 9 : the right image shows a frequency filter (red pixels) and delayed values (yellow pixels). Close to the cursor on the left image we can see the delayed signal.

More transformations are about to be implemented in *Sonos*.

.

Conclusion

We have evoked here some approaches of the visualization of sound. We have talk about the visualization as an interface for the sound's parameter. The work in progress project *Sonos*, in the continuity of the graphical synthesis, propose to integrate real-time interaction in the synthesis process and in sound transformation. Perspectives in interactive arts and inter-media are numerous. They could also concern the development of interfaces and operational tools for work, sequence and mix sound and multimedia images.

References

Dubois, L. *Jitter_pvoc_2D* (2003). Abstraction in the Max/MSP/Jitter environment.

Lesbros, V. *Phonogramme* (1993). Software.

Raczinski. J.-M., G. Marino and M.-H. Serra. 1991. "New UPIC system demonstration." In B. Alphonceand B. Pennycook, eds. *Proceedings of the 1991 International Computer Music Conference*. San Francisco : International Computer Music Association. pp. 567-570.

Roads, C. *Computer Music Tutorial* (1996). MIT Press.

Sedes, A. , Courribet, B.,Thiebaut J.-B. (2003). "Egosound, an egocentric interactive and real-time approach of sound space". Proceedings of the 6th international conference of Digital Audio Effects DAFX-03, London.

Sedes, A. , Courribet, B.,Thiebaut J.-B., (2003). "Visualisation du sonore avec le logiciel egosound: work in progress". Actes des journées d'informatique musicale 2003, Montbéliard

Sedes, A.□(2003). "Espaces sonores, actes de recherches", éditions musicales transatlantiques, Paris, 2003.

Wenger, E. *Metasynth*. 1997. Software, U&I Software.