ARCHITECTURE AND EXPERIMENTS IN NETWORKED 3D AUDIO/GRAPHIC RENDERING WITH *VIRTUAL CHOREOGRAPHER*

Christian Jacquemin
LIMSI-CNRS & University Paris 11
BP 133, 91403 ORSAY Cedex, France

ABSTRACT

The full integration and synchronization of 3D sound and image requires a data model and an architecture that allow for homogeneous description of geometrical and sonic components, together with mechanisms for distributed and synchronized rendering. First, several modes of sound and graphic combination are examined, and the resulting constraints for the rendering tools are detailed. *Virtual Choreographer* is then described with a particular emphasis on its XML data model for animation of spatialized geometrical and sonic components. An architecture is proposed that relies on the networked connection of *Virtual Choreographer* and a tool for sound synthesis and event synchronization such as *Pure Data* or *Max/MSP + SPAT* for sound spatialization. Last several experiments that involve these tools for different artistic or technical purposes are presented.

1. INTRODUCTION

Convincing combination of sound and image is a key aspect in producing virtual environments. Sound spatialization and 3D graphics real-time rendering are the two base ingredients for building such environments. However, their successful combination can only be achieved if they are correctly synchronized and interconnected: any interaction on one of the media must be immediately and relevantly associated with a corresponding interaction of the other one. In this paper, we focus on the interdependence between musical and graphical rendering in a spatialized and interactive framework.

For the purpose of modeling and rendering interactive 3D audio/graphic scenes, we present *Virtual Choreographer*, an XML language and a browser that combine scene component description, interaction, and interprocess communication. After reviewing works in which music and image are combined with various degrees of dependency, we describe the architecture and the base features of the tool. Then two modes of communication between sound and image are presented through sample environments.

2. SOME TRENDS IN GRAPHICS AND MUSIC COMBINED RENDERING

In order to review some of the works in which sonic and graphical components are combined to produce a sonified visual environment (or a graphical sonic scene), we will classify them by the relative influence of each media on the output. We first consider tools in which graphic output depends on an autonomous musical channel. Then we turn to graphically determined sound productions. Last, we examine environments in which both channels are mixed in a cohesive and holistic scene.

2.1. Music-driven Graphics

The most popular form of music-driven graphics is implemented in visualizers for music players such as *WinAmp*, *iTunes*, or even *Xbox*. These visualizers rely on an analysis of the volume magnitudes of frequency bands and their conversion into graphic parameters. *Zuma* (*3dMaxmedia*) is a visualizer for the professional world that offers a genuine interface to design attractive visual scenes contrary to the preceding tools in which visual scenes are predefined.

More complex dependencies of graphical output on music can be obtained either through more complex audio parameters than audio levels, or through more complex graphical outputs. This is precisely the scope of plug-ins of audio synthesis tools such as *Gem* for *Pure Data* or *Jitter* for *Max/MSP*. Such plug-ins can access any parameter used in the graph for music synthesis, and they also can use complex OpenGL primitives and video sources for elaborated graphic effects. Since *Pure Data* and *Max/MSP* are basically dedicated to music synthesis, they only have simple data structures (graphs and tables). For this reason, the design of complex geometrical models with these two tools is difficult, if not impossible.

Sophisticated geometrical models can be obtained through independent applications with real-time rendering. In the case of simple input parameters, the scope of obtained effects remains restricted. It can however have interesting applications in context-sensitive augmented

reality in which simple correlations between music and graphic rendering is acceptable [11]. Music has been used as input stimulus of more elaborated models such as emotion-sensitive face animation [8] or motion curve modification in choreography inspired models [3]. In *Soundsculpt Toolkit*, complex 3D scenes for Virtual Reality can be entirely generated from synthetic music through a patch bay that converts Midi parameters into geometrical effects [9].

2.2. Graphics-driven Music

Graphical interfaces can be used at different levels of music control. For instance, in interfaces for sound spatialization such as *MidiSpace* [15] or *VirtueString* [4], the graphical interface is used to control interactively the parameters of a virtual sonic scene by modifying the location of the sound sources and the sonic properties of the scene components.

The coupling between the graphical interface and the musical output is even stronger in the case of graphical interfaces for musical composition. The interface has no a priori aesthetic purpose, even though certain abstract or algorithmic designs such as cellular automata may offer interesting graphical schemes. The synthesis engine generally involves complex internal processes that transform the combination of input symbols into sounds.

Graphical interfaces for music composition can be categorized as *symbolic* in the case of *IanniX* in which circles and lines build the core elements [5], or *algorithmic* in the case of *Elody* [12] which relies on functional programming, or Beyls' automata [1] which use genetic algorithms and rewriting systems for music synthesis. Both types of interfaces involve computation, but the composer plays a more important role in symbolic interfaces in which he has more direct control on the output of the composition.

2.3. Audio-Visual Environments

Intermediate between the two preceding classes, audio-visual environments correspond to systems in which musical and graphical expressions are more intricate, and in which there is a bidirectional control between music and graphics. Some of these environments can also be dedicated to musical composition, but contrary to the preceding category, the interface also constitutes a part of the work of art. The notion of *audio-visual performance* introduced by Levin [13] breaks the barriers between graphical and musical designs. It may however be confusing and not profitable to the artist to be confronted to an environment in which an interface (a control device) is mixed with its by-product (the work of art).

Clearly architectured applications require well-identified components and means for controlling their interactions and their synchronous evolution. For these reasons, environments in which sound, image, and user interact must be implemented on appropriate distributed real-time middleware with traceable data exchange between distributed rendering components (such as *Aura* [7]). In addition, the data models must allow for distributed compositing, interactivity on all the media, and animations. MPEG-4 BIFS (*BInary Format for Scene Description*) implements these functionalities and extends VRML'97 for multimedia streaming and encoding. X3D is also an extension of VRML'97 in an XML format with a modular architecture. X3D is primarily dedicated to the modeling of interactive 3D scenes, but also incorporates the description of sound elements. In parallel other efforts are made to enrich X3D with more complete interaction facilities and enhanced audio components.

In the *Contigra* project, three separate data structures are used to describe interactive 3D scenes for the World Wide Web: graphical components, spatialized audio data through *Audio3D* [10], and behavioral data with *Behavior3D* [6]. In our approach described below, three types of data are integrated into a single framework that ensures a strong cohesiveness to the application. This design choice restricts the reusability of the application such as the association of similar control structures with various geometries. It however facilitates the design of interactive scenes by offering a unified framework for interaction with spatialized sound and graphics. We now turn to the description of the language and the browser of *Virtual Choreographer* that we have developed for interactive sound and graphic rendering.

3. A SPATIALIZED MULTIMEDIA INTERACTIVE SYSTEM

Our purpose is to offer a generic system that can be used to design an environment that belongs to any of the three categories presented in the preceding section: one way music-driven graphics or graphics-driven music, or bidirectional audio-visual environments. In such a system, it is expected that graphical and sonic rendering can control each other, and offer multiple modes of synchronization.

3.1. Architecture

In order to ensure distributed and synchronized rendering, we have chosen an Ethernet network as the backbone of this architecture. *Virtual Choreographer*, *Pure Data*, and multiple interfaces are connected through this network and can exchange messages for parameter updates and animation control (see Figure 1). This archi-

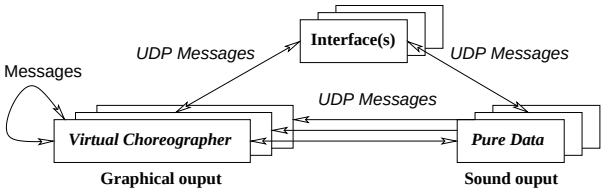

Figure 1. Message passing between *Virtual Choreographer* and *Pure Data*.

tecture allows for separate sound and image rendering on different machines with possibly different OSs.

3.2. Data Structure

Since message passing is also used inside *Virtual Choreographer* for scene updating, the graphical output can be made by compositing multiple scenes rendered on several networked UCs. This a valuable convenience in the case of distributed configurations such as virtual reality environments or live performances. Firstly robustness is ensured by replacing a possibly faulty graphical renderer by another one with a copy of the scene. Secondly heavy scenes can be distributed on several several machines and rendered separately.

Scene graphs, an already-established technique in interactive graphics for spatialized data representation, is the heart of *Virtual Choreographer* data structure, as it is the case for X3D, and previously VRML'97. More generally, *Virtual Choreographer* XML data representation bears many similarities with X3D. Most of it could be easily transcoded from and to X3D syntax. It is not our purpose, in this paper, to go into the details of this representation. A full reference guide on *Virtual Choreographer* syntax can be found on the project's site at *SourceForge* (http://virchor.sourceforge.net), together with downloadable source files and sample audio-visual scenes.

In order to illustrate the way in which audio-visual data can be defined and interactively modified, we will first detail a small sample scene. In the next section, more elaborated projects will be presented without details on their actual implementations. The structure below represents a sonified and textured sphere that is translated by the vector $\vec{v}(5, 0, 0)$ (the interpolator is not active at time 0):

```
<node id="node translation sphere 2">
  <interpolator id="node translation
       sphere 2" type="transformation" size="3">
    <schedule begin="100000" dur=".2"
       repeatCount="1" mode="sinus-periodic"/>
    <transformation id="t2.1" geometry="translation"
       x="5.0" y="0.0" z="0.0"/>
    <transformation id="t2.2" geometry="translation"
       x="5.0" y="10.0" z="0.0"/>
    <transformation id="t2.3" geometry="translation"
       x="5.0" y="0.0" z="0.0"/>
  </interpolator>
  <node id="sphere 2">
    <sound id="sound 1" xlink:href="Sounds/c4.wav"
           type="soundloop" fade_distance="0.0"
           fade_power="0" level="128" source="1"
           head="2001" begin="25" end="100"
           period="10.0" dur="10.0">
    </sound>
    <sphere id="sphere 2 nightsky" radius="0.6">
      <texture encoding="jpeg"
          xlink:href="textures/skyline.jpg" id="sky"/>
    </sphere>
  </node>
</node>
```

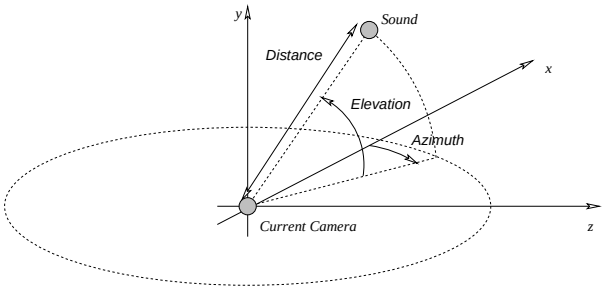

Figure 2. Sound Localization in the Coordinate System of the Current Camera.

When this scene is loaded, it is rendered as a textured sphere, and the visual aspect of the scene is modified according to the user navigation. Since the sphere is sonified, a sound associated with the file (Sounds/c4.wav) is emitted from time 25 to time 100. The sound localization is updated according to the camera displacements in the scene by emitting messages that contain details about the azimuth, elevation, and distance in the coordinate system of the camera (see Figure 2). The emitted messages can be caught by *Max/MSP + SPAT* for real-time spatial sound processing.

3.3. Data Exchange and Interactive Rendering

Event processing is the primary mechanism for scene interaction and animation, a classical solution for interactive graphic environments such as Virtual Reality systems [2]. *Virtual Choreographer* has a single unified application programmer interface (API) for data structure and message definition. The typical structure of a command for message processing is

```
Trigger
(<action> Action_description (Target)+
 </action>)+
```

Events are synchronous prescheduled events or asynchronous commands from the user, from nodes in the local or remote scenes, or from other applications on the

network such as *Pure Data*. Event processing by a node does not differ whether the event is internal or not.

There are mainly two types of actions: XML data structure modifications, and event emissions towards internal or remote nodes or applications. Actions that perform data modifications are made of an identifier (the name of the action), an operator, and a partial XML tag. The result of the action is to modify the values of the attributes of the target nodes that are instantiated in the action.

```
<script id="script sphere">
  <command>
    <trigger type="message_event" value="click"
             state="A" bool_operator="==" />
    <action>
      <set_sound_attribute_value operator="=">
        <sound begin="now" end="(now + 100)"/>
      </set_sound_attribute_value>
      <target type="single_node" value="#sphere 2" />
    </action>
  </command>
  <command>
    <trigger type="keystroke" value="C" />
    <action>
      <set_schedule_attribute_value index="1"
                                    operator="=">
        <schedule begin="(now + 0.1)"/>
      </set_schedule_attribute_value>
      <target type="single_node"
              value="#node transformed sphere 4" />
    </action>
  </command>
  <command>
    <trigger type="time_limit" value="7.0" />
    <action>
      <send_message_udp value="source 0 level 100" />
    </action>
  </command>
</script>
```

The first action above (set_sound_attribute_value) is triggered by the event click. It replaces the begin and end attributes of the sound tag of node sphere 2 by the values of the current time t and $t + 100$, thus starting the sound play for 100 time units.

The second action (set_schedule_attribute_value) triggers the scheduler that controls the interpolated translation of sphere 2 and makes it jump and fall back to its initial location. This action shows how event processing is used to control animations. Any numerical attribute in a tag can be animated through interpolation. There are basically two types of interpolations: finite interpolations that perform smooth transformations from a key structure to the next one, and random walks that moves through small random steps from an initial key structure.

The third action (send_message_udp) is a message broadcast on the network. This event can be received by *Max/MSP + SPAT* for opening a sound source; it is used for controlling sound rendering by the graphic application. Control can be performed in the reverse direction through events or actions emitted by external applications. Figure 3 shows the subpart of a *Pure Data* patch that is used to send an action: the modification of the x coordinate of a transformation tag. In the next section we show how such a bidirectional control can serve the purpose of rich interactive 3D audio-visual animations.

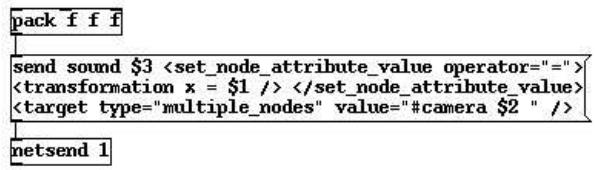

Figure 3. Message from *Pure Data* for *Virtual Choreographer*.

4. SAMPLE AUDIO-VISUAL SCENES

This section reports on two experiments that combine *Virtual Choreographer* as graphical rendering tool and geometrical modeler, and *Pure Data* as an interface and a sound synthesis tool. Data exchange between these applications is performed through a bidirectional UDP connection.

Virtual Choreographer receives from *Pure Data* either events that trigger message passing chains from node to node, or actions that modify part of an XML tag. If this tag is associated with a geometry, it results in dynamic geometrical deformations; if it is associated with a scheduler, it results in animation modification, launching, or interruption; if it is associated with a sound, it results in sound characteristics modifications, etc.

Pure Data or *Max/MSP + SPAT* receive commands that trigger, stop, or modify sound synthesis. The commands that concern the relative position of the sound sources with respect to the current camera are automatically generated. The other commands must be explicitly defined in the scripts of the scene nodes and associated with send_message_udp actions.

4.1. Spatialized Visualizer: *Sonic Wall*

The first environment, called *Sonic Wall*, is the metaphor of a street bordered by two walls made of strips. The architecture of the walls is inspired by the work of Zada Hadid [14]. Walls are made of parallel strips that can be freely curved and oriented by manipulating control points and normals.

In this environment, the viewer perpetually follows this street, and the appearance of the walls is correlated with the ambient music. According to our previous classification, this environment is music-driven graphics. The red, blue, and green components of the wall color are correlated with the amplitudes of high, medium, and bass frequencies. It makes the wall look red at high frequencies,

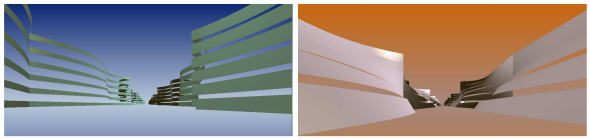

Figure 4. *Sonic Wall*.

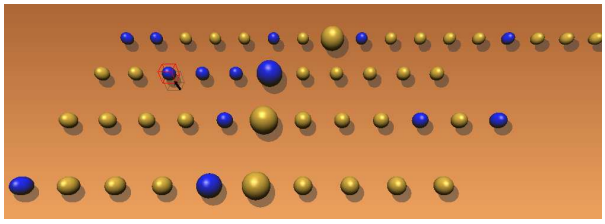

Figure 5. *PolyRythms* Rendering.

and green at low frequencies. The width of the strips depends on the sound amplitude of the whole spectrum. The wall is invisible in silence, it is made of thin strips for low sounds, and it is a continuous surface during loud parts. Last, the whole geometry (curves and normals) is controlled by parameters that are extracted from the different channels of the music pieces. The left part of Figure 4 corresponds to a moderately high bass sound, while the right part is produces by loud treble sounds.

4.2. Sonic and Human Interaction: *PolyRythms*

The second environment, called *PolyRythms*, has a graphical interface made of four rows of spheres (Figure 5). Each sphere is associated with a sound sample so that the sum of the sample lengths in each row is equal to 1.2 seconds (Figure 6). Sound playing begins by triggering the leftmost sphere in each row. When a sphere is triggered for sound playing, an interpolation is started for scaling up the sphere, and a message is sent to *Pure Data* for playing the corresponding sound. When the sound play is over, *Pure Data* sends a message back to the sphere. The sphere launches the scale interpolation that brings it back to its original dimension and sends a message to the next sphere in the row.

The last sphere in the first row is in charge of resynchronizing the first spheres. When it receives the acknowledgement from *Pure Data*, it sends a message to the first sphere of each row as indicated in Figure 7 in the case of two rows. In order to introduce variations in the production of the rhythms, each sphere can be activated or deactivated by passing the mouse over it. When it is deactivated (in blue in Figure 5), the sound is replaced by a silence of exactly the same length. The combination of activated and deactivated spheres results in various rhythmic patterns on the same tempo. Because of

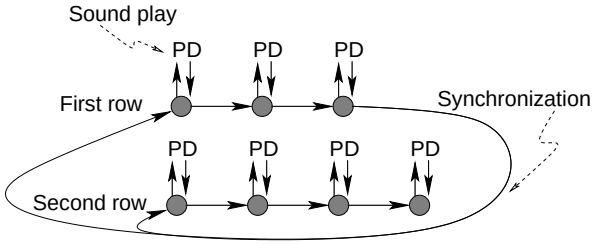

Figure 7. *PolyRythms* Event Passing.

the cross-control of image by sound through sound play acknowledgement and sound by image through activation/deactivation and message passing, this environment belongs to the class of audio-visual environments.

5. PERSPECTIVES

In addition to the preceding examples, other works in progress that involve *Virtual Choreographer* suggest several interesting issues for future developments in audio-visual interactive rendering:

- *Virtual Choreographer* is currently used for enriching the IanniX system for musical composition [5] with a spatialized graphic output. As for any XML data, the XML geometrical data of *Virtual Choreographer* can be transformed through XSL style sheets in order to produce variations in the graphical output. Thus a single generic geometrical model can be used for various styles of graphical outputs and adapt to the style of a composer, to the environment of musical performance...
- *Virtual Choreographer* is also used in a CNRS/EDF-R&D collaboration for information visualization. Numerical outputs are automatically transcoded into geometrical scenes. Sounds are added in order to produce a 3D sound and graphic environment for information browsing. Spatialized sound output in information access is a useful medium because it can draw the attention of the user to pieces of information that are visually hidden.
- Last, 3D models have interesting applications for audio spatialization without graphical output. The 3D model of *Virtual Choreographer* and its spatialized audio rendering is used for cognitive experiments on spatialized sound perception by blind people who move in a spatialized sound landscape. A full 3D geometrical model is required for tracking the position of the user in her environment and processing spatialized interactions.

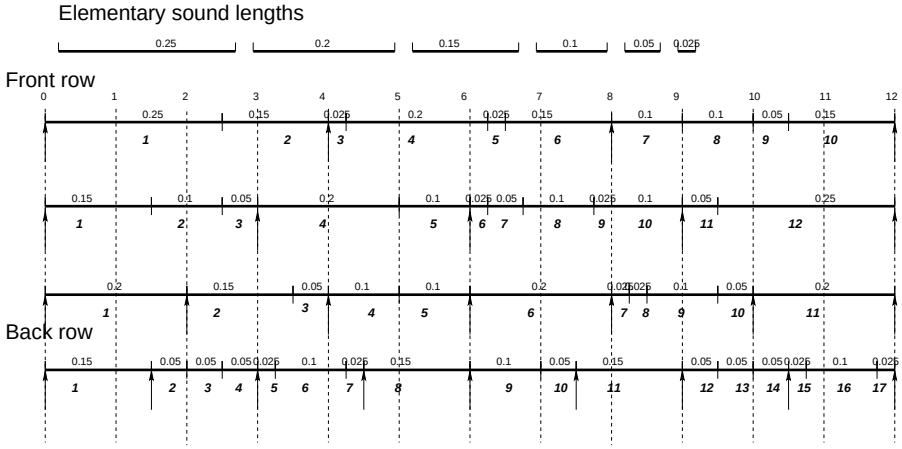

Figure 6. *PolyRythms* Rhythmic Composition.

6. THANKS

Many thanks to Serge Adam (*Quoi de Neuf Docteur*) for his collaboration on *Sonic Walls*, to Alan Blum and Brian Katz for sound spatialization, to Jean-Marc Vezien and Damien Touraine for the *VENISE* project, to Thierry Coduys, Cyrille Henry, and Guillaume Ferry (*La Kitchen*).

7. REFERENCES

[1] Beyls, Peter. Selectionist musical automata integrating explicit instruction and evolutionary algorithms. In *Proceedings of 6th Generative Art Conference, GA*, 2003.

[2] Burdea, G. and P. Coiffet. *Virtual Reality Technology - 2nd Edition*. Wiley, New York, 2003.

[3] Cardle, M., L. Barthe, Stephen Brooks, and P. Robinson. Music-driven motion editing: Local motion transformations guided by music analysis. In *Proceedings of Eurographics UK Chapter - EGUK Conference*. Eurographics, 2002.

[4] Chodura, Hartmut and Arnold Kaup. Virtual music reproduction. In *Proceedings of New Directions in Visual and Audio Expression, Sketches and Applications, SIGGRAPH*, 1999.

[5] Coduys, Thierry, Adrien Lefevre, and Gérard Pape. IanniX. In *Proceedings of 10ème Journées d'Informatique Musicale JIM*, 2003.

[6] Dachselt, Raimund and Enrico Rukzio. BEHAVIOR3D: An XML-based framework for 3D graphics behavior. In *Proceedings Web3D*. W3C, 2003.

[7] Dannenberg, Roger. Aura as a platform for distributed sensing and control. In *Proceedings of Workshop/Symposium on Sensing and Input for Media-centric Systems, SIMS '02*, pp. 49–57. University of California Santa Barbara Center for Research in Electronic Art Technology, 2002.

[8] DiPaola, Steve and Ali Arya. Affective communication remapping in MusicFace system. In *Proceedings of European Conference on Electronic Imaging and the Visual Arts, EVA*, 2004.

[9] Greuel, Christian, Mark T. Bolas, Niko Bolas, and Ian E. McDowall. Sculpting 3D worlds with music: advanced texturing techniques. In Fisher, Scott S., John O. Merritt, and Mark T. Bolas, editors, *Proceedings of Stereoscopic Displays and Virtual Reality Systems III, SPIE Vol. 2653*, pp. 306–315. SPIE, 1996.

[10] Hoffmann, H., R. Dachselt, and K. Meissner. An independent declarative 3D audio format on the basis of XML. In *Proc. ICAD*, 2003.

[11] Johnson, A. and S.K. Semwal. Music as an input device. In *Proceedings of VR Workshop Beyond Wand and Glove Based Interaction*. IEEE, 2004.

[12] Letz, S., Y. Orlarey, and D. Fober. Real time functional languages. In *Proceedings of the International Computer Music Conference*, pp. 549–552. Computer Music Association, 1995.

[13] Levin, Golan. Painterly Interfaces for Audiovisual Performance. Ph.D. diss., MIT Media Lab, 2000.

[14] Noever, Peter, editor. *Zada Hadid architektur//architecture*. Mak/Hatje Kantz, Vienna, 2003.

[15] Pachet, F. and O. Delerue. Annotations for real time music spatialization. In *Proceedings of International Workshop on Knowledge Representation for Interactive Multimedia Systems*, 1998.