Enhancing sound description in VRML

Alexandre Topol, Florent Schaeffer

Centre d’Etudes et de Recherche en Informatique
Conservatoire National des Arts et Métiers
email: {topol, schaeffer}@cnam.fr

Abstract

Sounds in VRML are spatialized (i.e. located in space) but no room effects can be described to make them more realistic. In this paper we present a solution we have implemented for extending the simple sound management within a VRML browser. The first aspect is the description of sound related materials in VRML format using new nodes and fields. We describe the different elements we have introduced for dealing with a 3D perceptual sound simulator. Those adjunctions were made by looking at the likeness and differences of the description and the computation of 3D visual and 3D audio worlds. It appears that only one type of sound source is needed as well as some material properties for the polygons. These material attributes for sounds are added in order to achieve some simple occlusion effects. The second aspect is the implementation of this audio-enhanced VRML browser. We present the tool we have implemented to evaluate this work.

1 Introduction

Visual augmented reality has become since few years a powerful way to enhance the comprehension of digitalized moving pictures of the real world. Card, Robertson, and York (1996) have shown that 3D graphical information can be perceived by the human perceptual system rather than being analyzed by the cognitive processor. We think that the expressive power of an audio augmented reality can also help users to perceive information instead of analysing it. For example, it makes it possible to recreate sound effects at the controller’s console in power plants. With this virtual audio information, special agents called “golden ears” can supervise and control the production process. The most important thing is that they don’t have to go to an unsafe area in order to listen for suspicious sounds.

As stated by Natkin (2000), an audio augmented reality system can also be used to provide synthetic sounds in a real space. In this particular case, users positions are given by three coordinates provided by wearable devices (head mounted for example). These coordinates are then used by a software that is able to compute in real time the binaural sounds to send to users. In such systems, different aspects must be studied carefully :

- how to represent the virtual world,
- how to localize users,
- how to send the audio data,
- how to distribute the computation according to the number of users and to the shape of the virtual world.

On the other hand, 3D sound spatializers can enhance the comprehension of 3D animations (Herder 1997). The standard audio equipment of today’s PC can produce immersive sounds at a real time rate like graphics hardware can accelerate the 3D pipeline that renders scenes. We believe like Tsingos and Gascuel (1997) that sounds are essential to enhance the visual experience and should fit what is seen. However, most of the time, 3D toolkits and file formats do not include an audio description part. When they do, most of them use a poor and unusable set of commands. 3D sounds need to be computed also in terms of objects occlusion. Otherwise, sounds played within a 3D visual environment could produce a negative effect.

In this paper, we present the solutions investigated to solve both problems : for an audio enhancement of 3D scenes and for a representation of a 3D audio world. Based on the likeness of 3D visual and 3D audio synthesis, we studied the use of the VRML syntax (VRML International Standard 1997) to describe a more complete 3D audio perceptual environment. In addition to the simple description of 3D audio worlds, the sound management used within a visual 3D real-time engine, also permits to create more realistic scenes. This union of 3D visual rendering and 3D sound computation enables to describe complete virtual reality worlds and browse through a richer online content than with the standard VRML file format.

In the first section, we will present shortly the related works that offer both visual and audio description. We will also discuss about the position of our own study related to those works and answer to the question : why didn’t we lean on one of those previous work ? The current sound possibilities included in the VRML97 standard and the X3D draft will be examined in the second part. More specifically, we will list what can be done and what can not. In the third section, we will present our propositions of enhancement for the VRML/X3D syntax. Before concluding, we will then present the architecture of our 3D audio-enhanced VRML browser in a fifth section.

2 Model and description choices

Sounds can be generated in different ways. For each of these, parameters and algorithms differ. In this paper, we are studying a possible description of 3D sounds. With an audio scene specification, worlds could be more easily generated and put online. Furthermore, a description of a scene can be used by different browsers and can be reused as a prototype in other scenes. However the description must be closely related to the program that will parse the file and synthesize what must be heard. Hence, all sound parameters used by the sound generator must be included into the specification.

Since this choice is made, we must first determine which technique will be used to compute sounds in order to know which parameters will be useful. In our particular case, the main criteria when choosing the computational model is its real time behaviour. We then evaluate the current known sound file formats to see if they can fit our needs.

2.1 Sound Model

Computing binaural 3D sounds can be done by using different techniques. Basically, two ways exist to render an audio scene :

- with a physical model that computes sounds using the scene's geometry, material properties and physical laws for reverberation (Savioja, *et al.* 1997),
- with a perceptual model that relies on global variables to achieve room effects (Jot 1997).

The first method is closer to what happens in the real world model for noises propagation. Waves, reflection and refraction properties of materials are used to compute sounds. With this technique, sounds heard by an user will depend on the volume definition of the scene and the energy distribution within. Hence, since sounds are computed by waves going from the audio source to the user's ears after some possible reflections, the virtual world must be closed so that waves are not "lost in space". However, physical models require a heavy computation that can not be done in real time. They can be compared to ray tracing techniques and more precisely with radiosity that lies also on an energy distribution algorithms. Some strategies can however be implemented to speed up this model. By observing the physical laws that rule the energy distribution, a static physical model can be used. Simplified dependencies can be established between the room's energy and a direct sound, first reflections and late reverberations. It allows to update dynamically the energy parameters used in the reverberation algorithm in accordance with the sources and the listener's positions. But this method is way to much complicated to describe since it must use parameters to specify the room's energy distribution.

In the second method, the computation of sounds is detached from the shape of the scenes. It is only related to some global variables (like reverberation, dryness of the air, …) that are given for the whole scene. These variables are used to compute the early reflections and the late reverberation that compose the room effects. This method makes it possible to define sound properties for a virtual world and obtain a result that could not be achieved in a corresponding real place. For instance, the scene can be a small rectangular room but the global flags given can correspond to the acoustic of a cathedral. Some validation experimentations made by Warusfel and Cruz-Barney (1995) and also by Marin (1996) have compared real reproduction with a simulated one. They show that it is possible to have a reliable synthesis with these perceptual parameters : localisation of the sound sources, reverberation of the room, …

The sound enhancement we propose for VRML is based on the second model. In a psycho-acoustic model, the room geometry and the materials properties are not used to compute sounds. But we also want to add some simple attributes to the objects composing the virtual world. Those properties are used to filter sounds in some particular cases. However, our model can not be considered as a mix of the physical one and the perceptual one since attributes are not used as parameters for physical laws but only to compute some filtering operations. The shape of rooms will also be used in the same way to attach different room effects within the same scene. We will show how this can be made and we will also describe the different attributes added to *Material* nodes and explain their utility.

To achieve a fast rendering (for visual and/or for audio data), the calculation must be quick enough to meet the real time requirements. Hence, a simple sound description and computation model are important. The psycho-acoustical model is used for that purpose : it is much quicker than the physical one. Our hypothesis is that the human hearing can tolerate unrealistic sound calculations as long as they can be managed in real time. This fact is also widely known for real time 3D rendering. Our goal is to respect the sounds sample rates (users could not bare hearing those sounds otherwise) but ease the calculations with a cheaper method. Sounds will be fluid and realistic-like for users in a real time context.

2.2 Description of sounds

Some file formats could be useful to describe a virtual sound space with the functionalities like the ones given by Natkin (2000). We have naturally studied those file formats and consider the following important points in order to determine if they can fit our needs :

- can a perceptual model be handled with the file format ?
- is it used widely ? Considered as a standard ? Are players freely available ?
- is it easy to describe contents with it ?

MPEG4

In the second version of MPEG4, advanced Audio BIFS (BInary Format for Scene description) can handle sound spatialization. As described by Scheirer, Väänänen, and Huopaniemi (1998), the physical and perceptual model should be supported. The physical model is based on the DIVA Project from Savioja, *et al.* (1996, 1997) from the Helsinki University of Technology. The perceptual approach is directly inspired by the *Spatialisateur* from Jot and Warusfel (1995, 1997) from IRCAM and France Telecom.

MPEG4 BIFS, including audio BIFS and advanced audio BIFS, are audio-visual elements that are organized in a scene graph equivalent to the one described in a VRML97 file. Each element is considered as a node with attributes that are attached by fields. Visual and audio nodes are organized in a graph hierarchy by using various grouping nodes.

In MPEG4, an audio scene graph is attached to the main scene graph by a *Sound* node. Like in VRML, these nodes are subject to transformations and define the source of the sound. The main difference between MPEG4 and VRML is the source description. In VRML, audio sources are simple sounds positioned and oriented in space. In MPEG4, sound sources are defined by a whole subgraph composed with various filtering nodes, effects nodes or mixing nodes. These nodes are able to transform the audio data extracted and decoded from the multiplexed streams before attaching it to a *Sound* node. To make a comparison, VRML can be seen like having the same capabilities than the Microsoft's DirectSound API. MPEG4, in relation to VRML, is more like EAX (Creative's Environmental Audio eXtension) which relies on and extends DirectSound capabilities with an advanced perceptual model.

In MPEG4, the advanced auralization process can be defined with three new nodes :

- *AcousticScene* is a grouping node to define common properties for different objects,
- *AcousticMaterials* enables to attach sound properties like reflection to polygonal surfaces,
- *DirectiveSound* is used to define frequency-dependent and directive sound sources.

The figure 1 shows the difference between the MPEG4 and the VRML scene graphs with sound definitions. One can see that MPEG4 is more sophisticated and that it can handle sound streams whereas VRML handles only sound files.

These additions to the first MPEG4 standard seem to meet our needs. However, MPEG4 is still under development and it is quite hard to find some precise technical information. Furthermore, MPEG4 version 2 resources are rare and only accessible by the members of the expert group. Last but not least, for our purpose, it seems that we do not need all the functionalities integrated in the MPEG4 standard. Also, since it is the first step of our project, we have to define quickly the representation format of the data. We want our representation to be used in virtual worlds and we think, as told previously, that only a simple sound spatialization system is needed. Hence, we do not need reflection capabilities that requires a heavy calculation but instead some higher level material properties like the porosity of a material, or its airlightness. We do however need environment variables to define different room effects.

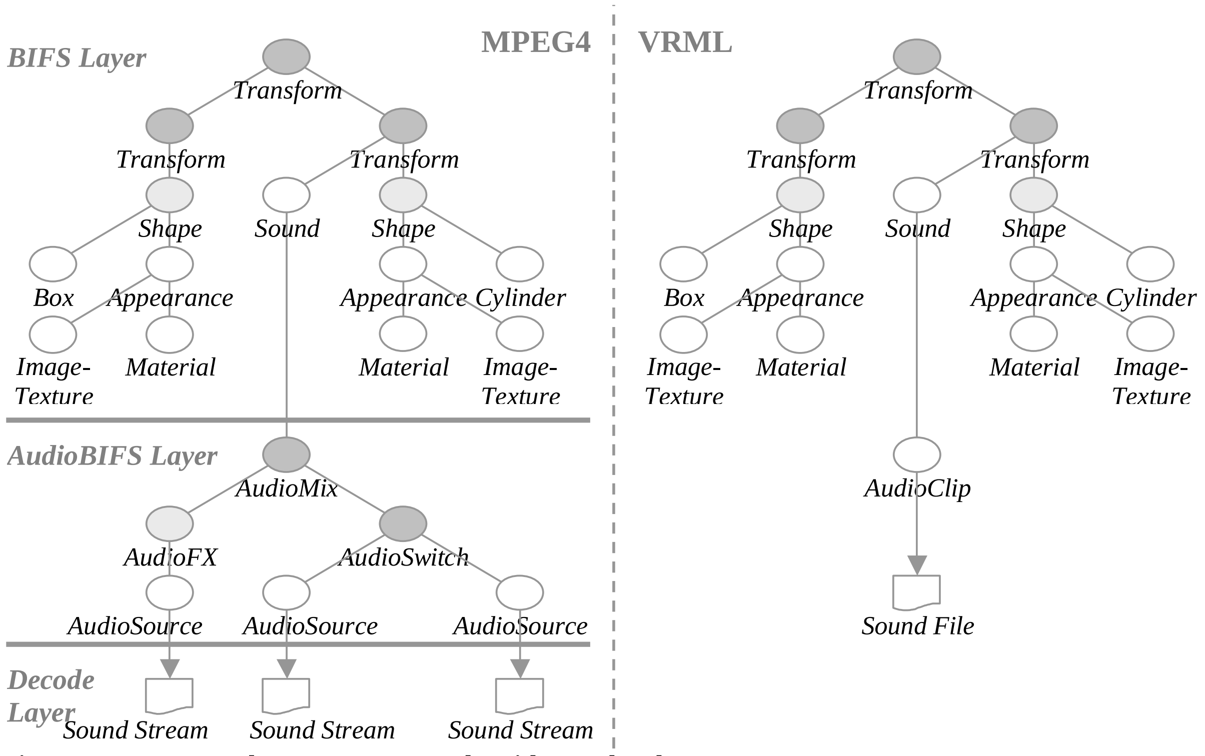

Figure 1 : MPEG4 and VRML scene graphs with sound nodes

X3D

The Web3D Consortium (1999) is also working on the inclusion of a better sound management in VRML200x/X3D. This proposal proves that the current VRML97 specification can not help to describe advanced sound properties. However, the VRML200x/X3D specification for sounds is only a rewriting of the VRML97 standard in XML syntax for the moment. It does not include any new nodes to deal with sounds.

3 Sound management in VRML

Since nothing is really available yet for both previous formats (no resources neither file format specifications), we decided to use the VRML file format to describe sounds. When this project involving Cecile Le Prado and Stéphane Natkin (2000) started a year ago, the MPEG4 version 2 draft with Advanced Audio BIFS was not public. But since it was already a two year old project, we had the hope that it would become an open standard pretty quickly. Now that one more year has past, there is still no evident sign of its transformation into a final specification. Hence, we have studied an alternative solution based on the VRML syntax. Still, we do believe that principles described in the documents we have read are powerful. This is why the extensions to VRML we propose are close to the one described in MPEG4's drafts.

Any other language for describing 3D audio-visual scenes could have been appropriate. However, VRML is the only well specified 3D descriptive language that is also a wide spread standard. Hopefully, it is also the language we have studied for several years on different projects including studies on 3D interface for digital libraries (Cubaud and Topol 2000) and a 3D workbench for XWindow applications (Topol 2000). For these studies we have implemented with the OpenGL API two crossed platform VRML browsers at the Conservatoire National des Arts et Métiers. The first one, written in C language, supplies tools and is optimised to ease the fluid navigation within static scenes. The second, in Java, deals more specifically with 3D animations. In both browsers it is easy to add nodes and/or fields to the VRML parser for our special needs.

3.1 What can be done with the *Sound* node ?

The VRML specification defines three nodes (*Sound*, *AudioClip* and *MovieTexture*) that are used to add sounds to a 3D scene (see figure 1). The *Sound* node is used to attach sounds to the scene. This node specifies the sounds properties like its location, its orientation and two ellipsoids that define the volume's maximum intensity and decay regions. Like the other nodes in VRML scenes, *Sound* nodes are translated, oriented and scaled by *Transform* nodes. This means that sound sources can be positioned and oriented respectively with the *location* and *direction* field of the *Sound* node but can also be moved and rotated by ancestral *Transform* nodes. Hence, it is possible to define some animated sound sources.

The sound stream is provided by an *AudioClip* or a *MovieTexture* node attached to the *source* field of the *Source* node. The *AudioClip* node specifies URLs where to retrieve the sound file and some properties that tell how to play it. The sound files can be either a PCM-encoded sound file or a MIDI file. The same attributes are given for the *MovieTexture* node that makes it possible to play the sound stream of a movie file.

In the current status of the VRML standard, a sound can be spatialized and the user will hear it differently depending the previously given four parameters. The viewpoint's and the sound's orientations are used to compute the binaural sound the user will hear. Hence, if the sound is on the left side of the point of view, the user will hear it much louder in the right ear. The two positions are also used to compute the intensity of the sound. For each sound source, there can be three cases depending on the user's distance to the source :

- When the user is inside the maximum volume ellipsoid, he hears the sound at his maximum intensity.
- When the user is outside the inner ellipsoid but inside the decay ellipsoid, the sound's intensity is computed by using a linear interpolation. The maximum intensity will be played on the inner ellipsoid's bound and the sound will be muted on the bound given by the outer ellipsoid. If the user is half way between those two bounds, the sound's intensity will be decreased by half.
- When the user is out of the outer ellipsoid the sound is not heard (hence, not played).

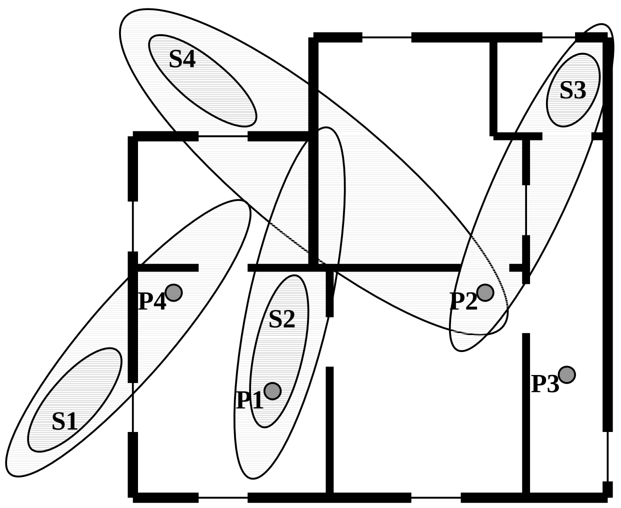

Figure 2 : Sound sources example with VRML files

The figure 2 illustrates the different possibilities that can be encountered when using the VRML sources descriptions and depending on the user's location. The heavy grey ellipsoids gives the positions where the user can hear the corresponding sound source with its maximum intensity. The light grey surface gives the region where the sound's intensity is proportionally reduced in accordance with the user's distance to the source. The next table gives for each user's position P1, P2, P3 and P4 what sound sources S1, S2, S3 and S4 are heard and at approximately which ratio of the full intensity.

User's position	Sound source audible
P1	S2 (100%)
P2	S3 (20%), S4(10%)
P3	–
P4	S1(20%)

One can see that the sound effects are very poor in the VRML model. The first one, described previously, is a volume filtering based on the user's and the sources positions and orientations. A second effect is achieved automatically depending also on the user's position and the sound sources ellipsoids. When an user is in the boundaries of several sound sources they are mixed so that he can hear all of them. For instance, in position P2, the user will hear 20% of the sound attached to S3 and 10% of S4.

3.2 What can not be done ?

The auditive result that can be obtained by using ellipsoids to define the different sounds boundaries is far from what can be heard with a real sound propagation model. For example, in position P2 or P4 the user would certainly hear both S1 and S2 sources in the real world. Another problem is that occlusions are not described. Without occlusions, an user would hear a sound with the same intensity behind a wall or not if its distance to the source is the same while still in the ellipsoid region. Furthermore, depending on the ellipsoids of a given source, an user could hear a sound behind two walls but not in a direct path. For instance, in position P3 the user should hear S3 but maybe not in position P2.

Sounds can stop propagating after a given distance but in every direction, not only in a ellipsoidal region. This ellipsoidal definition of sound's boundaries is not a natural way for specifying regions in which the sound can be heard. It must be replaced by a more realistic and comprehensive description. Sound cones would be much more helpful to compute direct sounds but also the first reflections and the reverberation. With cone-shaped sounds, an user in front of a source without obstacles on the way would probably always hear a sound.

Furthermore, it would be impossible to define ellipsoidal boundaries of sound sources that would match exactly the reverberation and occlusion effects expected. Hence, sounds in VRML are spatialized but reverberations and occlusions can not be implemented. Since in the architecture described by Natkin (2000) we want different sound parameters for each room, we need reverberation information to be attached to a volume and occlusion attributes to polygonal faces. Simple attributes could be added to the visual attributes of the *Material* node in order to specify how sounds are transmitted through the polygons.

What requires further integration in VRML is support for further audio rendering features, such as the simulation of reflected sounds and reverberations. In addition, it is important to provide a means for obstacles to reduce or otherwise interfere with the propagation of sound through the VRML world when they are positioned between the sound source and the observer's position.

4 Proposition of enhancement

For adding sound attributes into the VRML97 file format, one must know how graphics data is specified. By knowing the likeness of 3D audio and 3D graphics, we will be able to add sound nodes and fields with the same semantic than the graphics ones. What must be considered carefully is :

- the types of sound sources needed,
- grouping nodes to affect rooms properties,
- the sound related attributes of polygons.

4.1 Sound sources

A comparative analysis between 3D visual and 3D audio synthesis has been done by Natkin (1998). One of the main differences is the type of sources for lights and sounds. Light rays are much more directional than sound waves. However, in virtual worlds, one can prefer some directional sound sources to achieve some special unrealistic effects. Where point, spot and directional sources are needed for lights in order to achieve different visual effects, we only need one type of sound sources. The sound definition in the current VRML model is not efficient enough. The attenuation of sounds is made arbitrarily by giving the ellipsoidal regions. We prefer a cone definition that can be used to simulate different "shapes" for a sound source, from a realistic one (omnidirectional) to an unrealistic sound-laser (unidirectional). Inner and outer cones will define three zones where the angle-dependant gain will be computed differently : constant in the inner cone and changes over the transitional zone to the value specified outside the zone. The overall attenuation will be computed with the distance of the user relative to the sound source.

The corresponding node that we call *SoundCone* in order to keep a backward compatibility with the previous *Sound* node is defined in VRML syntax with the following fields. The default values for the inner and outer cones is 360 degrees (*i.e.* omni directional sound source). *presence*, *warmth* and *brilliance* are some source perception variables that are used by the *Spatialisateur* (Jot 1997).

```
SoundCone {
  exposedField SFVec3f  location      0 0 0
  exposedField SFVec3f  direction     0 0 1
  exposedField SFFloat  innerAngle    6.2832
  exposedField SFFloat  outerAngle    6.2832
  exposedField SFFloat  outerAngleGain 0.0
  exposedField SFFloat  presence       0.0
  exposedField SFFloat  warmth         0.0
  exposedField SFFloat  brilliance     0.0
  exposedField SFNode   source         NULL
  field        SFBool   spatialized    TRUE
}
```

4.2 Grouping node

Two solutions can be considered for defining the room effects. Late reverberations can be attached to a binding node or to a grouping node. The first case would only enable a single room effect for the whole VRML scene. This would not be helpful to manage a scene like the one described by Natkin (2000) where each room has different sound properties. Even if a scene is composed from different VRML files, each one having a different room effects, only one binding node would be active. A grouping node is much more powerful. Room effects will be defined within a bounding box that includes all nodes attached to the grouping node.

VRML has different grouping nodes, each performing a specific operation on the descendant nodes attached in their *children* field :

- *Anchor* defines the nodes that act as a link to another web content,
- *Billboard* groups nodes that must be automatically oriented toward the user's viewpoint,
- *Collision* specifies the collision properties of its children,
- *Group* contains children nodes without introducing some specific operations,
- *Inline* reads its children nodes in an external url,
- *LOD* defines various level of detail for an object and gives hints to the browser for choosing the appropriate appearance,
- *Switch* traverses zero or one of the nodes specified in the *choice* field,
- *Transform* defines a local coordinate system for its children.

To be consistent with this approach (one grouping node for one operation), we define a new grouping node for attaching a room effect to its children. This node, named *RoomEffect*, will define the perception flags to use while within the bounding box. The other fields (besides the *children* field and the bounding box definition) are the global variables that we took from the spat (Jot and Warusfel 1995). Together, they will help to compute the room effects. The definition of this node in VRML syntax will be :

```
RoomEffect {
  eventIn      MFNode   addChildren
  eventIn      MFNode   removeChildren
  exposedField MFNode   children       []
  exposedField SFBool   doplerEffect   TRUE
  exposedField SFBool   airArbsorbtion TRUE
  exposedField SFFloat  roomPrecence   0.0
  exposedField SFFloat  runningReverb  0.0
  exposedField SFFloat  envelope       0.0
  exposedField SFFloat  lateReverb     0.0
  exposedField SFFloat  heaviness      0.0
  exposedField SFFloat  liveness       0.0
  field        SFVec3f  bboxCenter     0 0 0
  field        SFVec3f  bboxSize      -1 -1 -1
}
```

The bounding box could be computed automatically by the browser to include all children attached to the node. But, this could lead to a scene where bounding boxes could overlap one another if children nodes are not well grouped. To be conform to what is done with the other grouping nodes in the VRML standard, we assume that bounding boxes are given by the scene's author. Like normals in a *IndexedFaceSet*, bounding boxes could be computed automatically if the author does not give them.

4.3 Material properties

Material attributes are only used to specify some simple properties. Mainly, we define the attenuation of a sound's intensity when it encounters the face. But other filters could also be described. For example, which frequencies of a sound would pass through an object. Visual attributes of a material are given for all visible sides. With *IndexedFaceSet*, it is possible to define if both sides are visible or if only one side (given by the normal) is visible. However, when both sides must be drawn, the same visual appearance is used. With the other drawing primitives (*Box*, *Cone*, *Cylinder* and *Sphere*), both sides are always drawn with the same appearance.

For sound attributes, we can not use the same principles since we want different effects depending on the direction of the sounds going through the polygons. This is why we define pairs of the same sound attributes : one for outside-inside direction and one for the opposite one. With *IndexedFaceSet* geometries, the outside region will be the one pointed by the normals. If only one side of the *IndexedFaceSet* must be drawn, the sounds will only be filtered when coming from one side. All computations to determine which 3D primitives intersect the direct sound trajectory are the one used in ray tracing. As it can be seen in the next code sample, a new field *soundFiltering* within the *Shape* node is used to attach the *SoundFiltering* node that gives the sound properties for the shape. The fields *frontIntensity* and *backIntensity* are used to specify the amount of the direct sound's volume that is filtered by the polygons of the shape in both directions. These fields operate in the same way for sounds than the *transparency* field in the *Material* node for shading. A little part of the VRML augmented code that describes the scene in figure 3 is given next :

```
[...]
SoundCone {
  location 0 –10 0
  direction 0 0 -1
  source AudioClip { url “bipbip.wav” }
}
RoomEffect {   # first room
  bboxCenter 0 -8 0
  bboxSize 3 3 2
  children [
    Shape {           # first wall
      geometry IndexedFaceSet { [...] }
      appearance Appearance { [...] }
      soundFiltering SoundFiltering {
        frontIntensity 0.5
        backIntensity 0.8
      }
    }
    Shape {           # second wall
      geometry IndexedFaceSet { [...] }
      appearance Appearance { [...] }
      soundFiltering SoundFiltering {
        frontIntensity 0.5
        backIntensity 0.8
      }
    }
    [ ... ]
  ]
}
RoomEffect {   # second room
  bboxCenter 0 -4 0
  bboxSize 3 5 2
  children [
    [ ... ]
  ]
}
```

The figure 3 shows what can be obtained with these attributes. Each wall decreases the intensity of the emitted sound source S4. The light-grey zones give the positions where the sound is not filtered by walls. The middle grey gives the places where the sound is heard after one wall filtering. The heavy grey shows positions where the sound is filtered by two walls. The white zones are the zones where the sound is filtered by 3 or more walls. The corresponding VRML description of the scene with our adjunctions is also given in the figure 3.

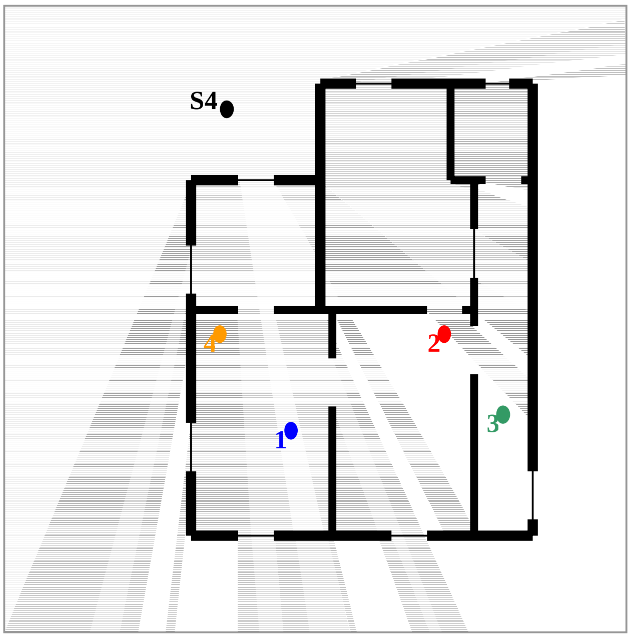

Figure 3 : Example of material absorption

5 Implementing a prototype

For implementing the principles described above, we have used one of the two VRML browsers we have developed. In the chosen one, implemented in C language, the VRML expressions are described with Lex and the grammar is described with Yacc. Adding nodes and attributes to the original grammar is quite easy. All recognized VRML nodes are then translated in OpenGL commands and rendered to show the visual result.

For sounds, we have used an API called OpenAL where AL stands for Audio Library. The commands are close to the ones used with the Graphic Library OpenGL. OpenAL supports sources positioning, reverberation, decay and doppler effects. OpenAL commands are not affected by the OpenGL transformations. Sounds location and orientation as well as the listener ones will have to be computed in the global coordinate system. For the listener’s position this is not really difficult since in the VRML file (with the *ViewPoint* node for instance) the observer is located relative to the global coordinate system. The problem comes from the sources position. In VRML files, *Sound* nodes and our added *SoundCone* node can be affected by previous *Transform* nodes. Hence, the location and orientation of

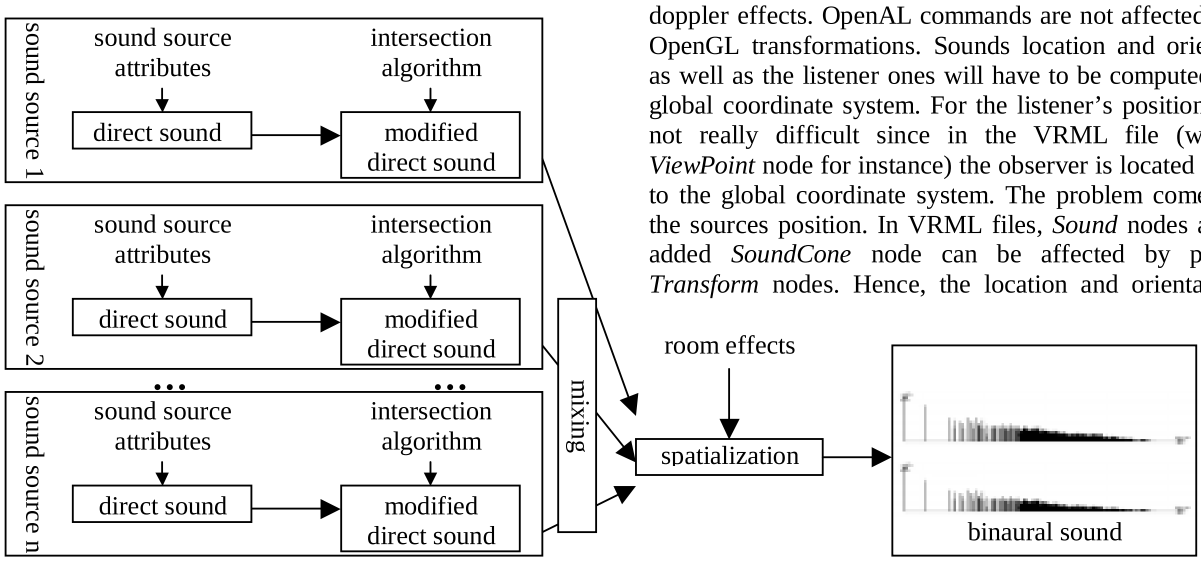

Figure 4 : Binaural sound production

sound sources are given in a local coordinate system. But we use the current transform matrix in a sound node to translate coordinates from the local system to the global one.

Occlusion effects are computed with a raytracing-like algorithm. All polygons intersecting the ray going from the source's position to the viewpoint's position are used to filter the direct sound. This computation has to be done whenever the sound sources, objects or the viewpoint are moved. Since at least a normal is given for each polygon, it is easy, by comparing the normal and the "sound ray", to choose from the *frontIntensity* and the *backIntensity* fields which one to use.

The production of binaural sounds is described in the figure 4. Room effects are limited with OpenAL for the moment. For a better "rendering" of sounds, we should use the *Spatialisateur*. With our second VRML browser written in Java, it should be possible to use the Spat module from the JMax software also written in Java (JMax). We will investigate this solution in a future work.

6 Conclusion – Future work

We have added some new nodes to the VRML syntax in order to describe all the perceptual spatialization process. Until now, sound sources were located and played in 3D but room effects were not described in VRML files. In adjunctions to this perceptual method we have added some shape properties for sound filtering. These simple sound behaviours, even if they are far from being realistic, work well for filtering direct sounds. Since the room effects are computed with these direct sounds, our simple model can add some realism to scenes.

Our extended VRML language can be used in the two different ways we were aiming. First, for a purely aural navigation, it can be useful to simulate in real time what users can hear. The routing of events from a Java script to the sound attributes can achieve some interesting effects like travelling automatically in an audio world or sharing moving audio resources with others. This "rendering" of 3D audio will be very useful for audio augmented reality system and particularly in art installations. The second aspect is for a mixed audio and visual worlds. 3D sounds within 3D scenes are interesting for games but also video and audio conferencing and power plant maintenance. In every applications where sounds must be located relatively to objects in a scene, the easy description will be helpful.

In a multi-user context, aural interactions between participants could be added too. Since VRML does not make it possible to specify these kinds of interactions, this must be done with Java scripts. The principle for attaching sounds to users is the same than for attaching a visual appearance. Like in network games, each user has to tell the others where he is and how he looks and sounds like. When connecting to a shared world, the user's avatar described with the VRML syntax is sent to a server managing the scene. A first Java script must handle this initialisation process. Each user must also send its position and orientation in the scene and its new appearance (if it has changed). This can be done each time the user moves. For rendering the scene including other users avatars, a request asking the other users information is made periodically to the server. This operation is done by a second Java script that retrieves these information and modifies the VRML scene graph in accordance to what the server has sent back. This modified scene graph will be rendered visually and aurally to reflect the other users moves.

We shall now work on adding some sound behaviours to objects. In this paper we explained how objects could be used to modify the sounds perceived. We also believe that it is interesting to study how sounds can modify objects attributes. Like sounds can turn on lights in the real world, they could be also used in virtual worlds to activate some events to start some animations.

Note for readers : the implementation of the principles described in this paper is accessible on our web site at the following URL :
http://cedric.cnam.fr/~topol/icmc2001

References

Card S., Robertson G., York W., The WebBook and the Web Forager : An Information Workspace for the World-Wide-Web. in *Proc. of ACM CHI'96*. Vancouver, April 13-18, 1996.

Cubaud P., Topol A., A VRML-based user interface for an online digitalized antiquarian collection. Proc. of *Web3D'2001*, Paderborn, Germany, Feb. 2001.

Herder J., Tools and Widgets for Spatial Sound Authoring, Proc. Of *Compugraphic'97*, Vilamoura, Portugal, December 1997.

Huopaniemi J., Savioja L. and Takala T., DIVA virtual audio reality system, P*roc. of ICAD'96*, Palo Alto, California, Nov. 1996.

JMax, http://www.ircam.fr/equipes/temps-reel/jmax/

Jot J.-M., Warusfel O., Spat~ : A spatial processor for musicians and sound engineers. *Proc. Int. Conf. on Acoustics and Musical Research*. Ferrara, 1995.

Jot J.-M., Efficient models for reverberation and distance rendering in computer music and virtual audio reality, P*roc. of ICMC'97*, September 1997.

Marin M., *Etude de la localisation en prise et restitution pour la téléconférence de qualité*, PhD Thesis, University of Mine, Le Mans, October 1996.

Natkin S., Objects in a Virtual Space: a Comparative Analysis between Image and Sound Spatial Representation and Synthesis. *Proc. of 5th Brazilian Symposium on Computer Music,* Belo Horizonte, Brasil, August 1998.

Natkin S., Mapping a Virtual Sound Space into a Real Visual Space, *Proc. of ICMC2000*, Berlin, September 2000.

Savioja , L., Huopaniemi, J., Lokki, T., and Väänänen, R., Virtual environment simulation - Advances in the DIVA project. *Proc. Of ICAD'97*, Palo Alto, California, USA, Nov. 3-5, 1997.

Scheirer E. D., Väänänen R., Huopaniemi J., AudioBIFS: The MPEG-4 Standard for Effects Processing, *Proc. of DAFX '98*, Barcelona, Nov. 1998.

Topol A., Immersion of XWindow Applications into a 3D Workbench. P*roc. of ACM CHI'2000*, The Hague, Netherlands, April 2000.

Tsingos N., Gascuel J.-D., Soundtracks for Computer Animation : Sound Rendering in Dynamic Environments with Occlusions. *Graphics Interface '97*, Kelowna, May 21-23, 1997.

Virtual Reality Modeling Language - International Standard ISO/IEC 14772-1:1997. http://www.web3d.org/Specifications/VRML97/

Warusfel O., Cruz-Barney F., Validation of a computer simulation environment for room acoustics prediction, *Proc. of the 15th International Conference on Acoustics*, Trondheim, 1995.

Web 3D Consortium, http://www.web3d.org, 1999.