A framework for integrating sound into Virtual Environment interfaces

Hesham Fouad and James K. Hahn
Department of Electrical Engineering and Computer Science
The George Washington University
Washington, DC 20052, USA
hesham@seas.gwu.edu
hahn@seas.gwu.edu

ABSTRACT

Sound represents a largely untapped source of realism in Virtual Environments (VEs). In the real world, sound constantly surrounds us and pulls us into our world. In VEs, sound enhances the immersiveness of the simulation and provides valuable information about the environment. While there has been a growing interest in integrating sound into VE interfaces, current technology has not brought about its widespread use. This, we believe, can be attributed to the lack of proper tools for modeling and rendering the auditory world. We have been investigating some of the problems which we believe are pivotal to the widespread use of sound in VE interfaces. As a result of this work, we have developed the Virtual Audio Server (VAS). VAS is a distributed, real-time spatial sound generation server. It provides high level abstractions for modeling the auditory world and the events which occur in the world. VAS supports both sampled and synthetic sound sources and provides a device independent interface to spatialization hardware. Resource management schemes can easily be integrated into the server to manage the real-time generation of sound. We are currently investigating possible approaches to this important problem.

1. INTRODUCTION

Sound represents a largely untapped source of realism in Virtual Environments (VEs). In the real world, sound constantly surrounds us and pulls us into our world. In VEs, sound has the potential to enhance the immersiveness experienced by the user as well as serve as an important channel of communication within the environment. While there has been a growing interest in integrating sound into VE interfaces, current technology has not brought about its widespread use. This, we believe, can be attributed to the lack of proper tools for modeling and rendering the auditory world.

Tools currently available for integrating spatial sound into VE interfaces provide low level interfaces that are hardware specific. This is understandable since much of the work in auditory feedback has been focused on the spatialization problem and resultant systems are geared towards a specific hardware solution to the problem. Such systems require the programmer to expend a great deal of effort in acquiring intimate knowledge of audio processing hardware devices. Furthermore, current tools place a hard limit on the number of sounds supported, leaving the programmer to handle the contingency of exceeding that limit. Finally, most current systems support only digitally sampled sound sources. While sampled sound sources are appropriate in many instances, synthetic sound sources have advantages for some applications and therefore should also be supported.

Our work has focused on addressing these problems in order to bring auditory feedback closer to mainstream use. The Virtual Audio Server (VAS) was developed as a framework where we could investigate these problems. Our goals at the outset of this work have been as follows:

- Devising high level abstractions for modeling the auditory world, while keeping in mind the integration into existing graphics models.

- Devising a scheme for resource management, thus avoiding a hard limit on the number of sounds supported by the system.
- Hardware independence was a major concern in the design of this system, since many spatialization devices exist and no one device is ideal for all applications.
- We considered remote access an important feature of VAS. It servers as a means of load distribution and can be essential since the machine where the audio output device resides may not be ideal for running the VE simulation.

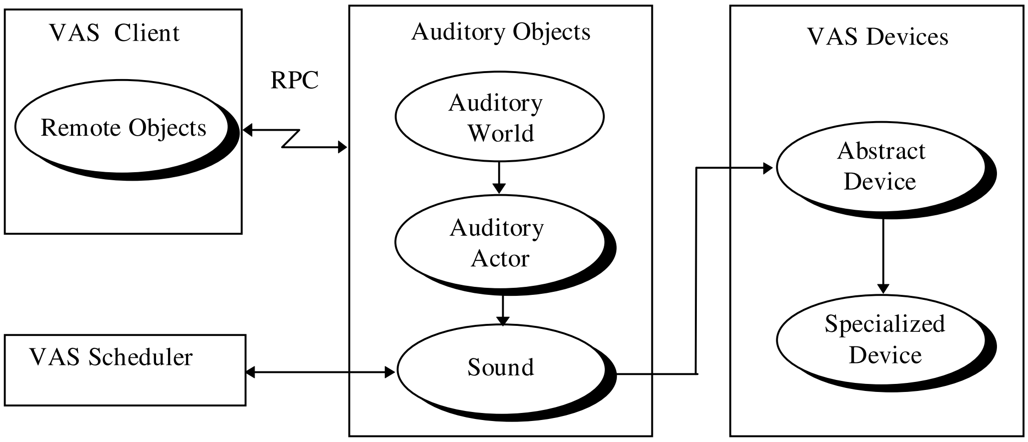

Figure 1 VAS system architecture

2. SYSTEM ARCHITECTURE

The VAS server is partitioned into four functional areas as depicted in fig. 1. The elements comprising the virtual auditory world are modeled using *auditory objects*. Those consist of the *auditory world* which maintains the state of the world and provides access to the other objects in the world. *Auditory actors* model the sound producing entities in the world. They have a sound repertoire consisting of a set of *sound objects* which model the sound sources within the world. Sound objects evaluate sound samples in real-time and write the resultant samples to VAS *device objects*. An instance of a VAS device object is attached to each sound object and provides it with a device independent interface to any spatialization device used by the server. Finally, the VAS *scheduler* manages the real-time evaluation of active sounds in the server.

A client/server architecture facilitates load balancing in VAS by distributing the graphics and audio processing on different machines. VAS's distributed architecture is based on the *Remote objects* used for inter-process communication in the Mach kernel[1]. Each VAS auditory object with which the client interacts has a corresponding remote object on the client's machine. Remote objects communicate with their corresponding auditory objects through a Remote Procedure Call (RPC) interface signaling any state changes due to a client interaction. This approach maintains an object oriented interface to the server while supporting remote operation. Remote objects comprise a C++ class library which the client application uses to interact with the VAS server.

3. MODELING THE AUDITORY WORLD

The problem of creating realistic virtual environments is fundamentally that of creating models of the phenomenon we want to simulate in those environments. Much of the reason sound has not been integrated into more Virtual Environments is that the current models are too low-level and not comprehensive. One of our major concerns in this effort has been the formulation of a more complete, interactive model which provides appropriate abstractions for modeling the auditory world. In devising those abstractions, a great deal of emphasis was placed on exploiting the parallels that exist between the auditory and visual worlds. The resultant model closely parallels many of the constructs used for modeling the visual world. This facilitates the integration process since the programmer is not forced to reconcile between two divergent views of the world.

In the following sections, each of the modeling abstractions in VAS is described along with its correspondence to elements in the visual world.

3.1 Auditory Actors

The most prevalent abstraction in modeling the auditory world is the auditory actor. It represents sound emitting entities in the world. Through the auditory actor's interface, applications can create sounds, control their behavior, and place them in 3D space.

Visual appearance differentiates objects in the visual domain. This appearance is specified by the material properties assigned to those objects as well as the properties of the light sources in the environment and possibly the atmospheric effects present in the environment. In some systems such as Inventor[2] these properties are modeled as first class entities in the world. In the auditory world, objects are not differentiated by how they look but by how they sound. It therefore follows that actors in the auditory world must be assigned aural properties consisting of a set of sounds they produce. In fact, auditory actors are completely defined by their position in 3D space and their sound repertoire.

Any number of sounds can be associated with an auditory actor. They can be controlled individually, or as groups which are arbitrarily specified by the user. The position of the auditory actor determines how the sounds are localized in 3D sonic space. Control of the sounds attached to an auditory actor is achieved through simple primitives for starting, stopping, and pausing sounds as well as modifying sounds' intensities and controlling their parameter space which we discuss later.

The behavior of an auditory actor defines how and when sounds are activated by the actor. We can term this problem the sonic control of an actor. The idea of control carries over directly from the visual realm. The difference being that the behavior manifests itself in the production of sound not motion. Where this control mechanism should reside poses an important problem. If we assume that visual objects in the VE have an active control mechanism associated with them, then introducing a separate active mechanism to control the auditory actor creates a problem of sharing data and maintaining synchronicity between the two mechanisms. We have found that integrating the sonic control of the auditory actor into the motion control mechanism of visual objects is a better approach. However, in instances where the auditory actor is required to exhibit autonomous behavior or where there is no correspondence between an auditory actor and a visual object, a control procedure may supplied upon the instantiation of the actor. This procedure is invoked as a light weight process by the system.

The server communicates events to auditory actors through a message queue associated with each actor. Two types of events are sent to an actor: state change messages signal a change in the play state (whether a sound is playing or stopped) of the sounds attached to an actor. Timing messages are sent to each actor by the server at regular intervals. This implements a global clock for the system which enables synchronized behavior among the actors.

3.2 The Listener

In Virtual Environment simulations, the camera is positioned and oriented based on the position and orientation of the user in virtual space. This is an essential ingredient of immersive environments, that the user can move around the virtual space and "look around" by orienting a viewing device. In modeling the visual world, the camera is given a position and orientation among other properties. In the auditory world we have listeners who are able to move around the world and orient themselves in this world. In order to maintain immersiveness, the active sounds in the environment must attenuate with distance as well as move in 3D sonic space as the listeners move and orient themselves. The methods of actually achieving these effects have been the major focus of audio research in VE. Two general solutions have been suggested. One is to simulate the head related transfer functions (HRTFs) of a listener by convolving a filter with the original signal which mimics the effect that the HRTFs would have on a sound emanating from a particular direction[3]. The summed signals of convolved audio streams are fed into headphones placed on the listener's head. The other approach is to use loudspeakers placed around the user to affect the perceived position of the sound by controlling the amplitude of the signals fed into the individual speakers. This extends stereophonic listening into three dimensions[4]. Each of these approaches has its merits and relative costs. While the problem of spatialization is not directly addressed by our work, maintaining device independence requires that our model support both methods as well as any future spatialization technique.

The listener actor in VAS is derived from the auditory actor by the addition of a head orientation property. VAS clients control the perceived position of sounds in the environment simply by moving and orienting the listener actor along with the camera. The listener actor signals changes in its position and orientation to the spatialization device through a device object attached to it.

Because the listener actor is derived from the auditory actor, sounds can be attached to it and controlled just as they can with the auditory actor. Those can be used to provide important motion and collision cues to the user.

3.3 Auditory Spaces

An important issue that must be addressed is that of modeling the environment in which sounds occur. There is a marked difference between the behavior of sound and light in this regard due to the longer wavelength of sound. Light can only be perceived by the viewer through direct transmission, reflection, or passage through transparent objects. Sound, on the other hand is able to refract around objects. It is therefore more prevalent in the sense that sounds not occurring in the listener's immediate enclosed space may well be heard by the listener. The process of determining the actual sound signal reaching the listener has been termed sound rendering[5].

Given the position of all the active sounds in the world and the position and orientation of the listener actor, a VAS device must render the sounds taking into account distance based attenuation, delay, and listener orientation to produce spatialized sounds corresponding to the active sounds occurring in the world. In order to correctly model sounds occurring in enclosed spaces, the rendering algorithm needs information regarding the geometry of the enclosures in the world. The auditory space object models these enclosures. An auditory space has associated with it the geometry of an enclosure in the form of *barriers*. Measures of reflectivity as well as attenuation are associated with each barrier of an auditory space. The reflectivity of the barriers determines how reverberant an enclosure is, while attenuation determines how easily sounds can penetrate the barrier through diffraction or passage through openings. Because the auditory space object is derived from the auditory actor, sounds can be attached to it. This allows ambient sounds to be associated with enclosures giving them a unique character. Ambient sounds are not spatialized and are therefore relatively low cost.

3.4 Sounds

As mentioned earlier, VAS supports both sampled and synthetic representations of sound. Sampled sounds are digital recordings of sound that are temporally mapped and played back on demand. They have gained popularity because they can be used to easily create complex sounds that would otherwise be difficult or impossible to synthesize. The static nature of their representation, however, limits the control parameters available to modify the characteristics of those sounds.

Synthetic sounds are procedurally defined, one dimensional functions in the temporal domain. They can be arbitrarily parameterized which allows sounds to be synthesized from motion parameters providing important tactile cues to the user for interaction based sounds such as collision and scraping. Parameterization also supports the mapping of data to sound parameters which is important for data sonification.

A procedural definition gives users complete control over the shaping of the sound's timbre. This is both an advantage and a drawback of synthetic sounds. Creating procedures to produce specific sounds is a difficult task. It requires a deep understanding of the sound synthesis process and an intuitive sense of how to achieve desired results. VAS addresses this problem in its representation of synthetic sounds described below. Another drawback of synthetic sounds is that they are computationally expensive. We are currently investigating resource management schemes which attempt to minimize the cost of synthetic sound evaluation through optimization and degradation techniques.

VAS models synthetic sounds using Timbre Trees[6], a functional representation of sound. Timbre Trees model synthetic sounds using a tree structure where internal nodes of the tree are signal processing nodes, while the leaf nodes are signal generating nodes. A digital representation of a sound propagates from the leaf nodes and is processed by the internal nodes of the tree. The sound that emerges at the root node represents the final synthetic sound. Timbre Trees allow for the arbitrary parameterization of any of the nodes in the tree. The semantics of the parameterization is dependent on how a sound is constructed. VAS provides access these parameters through the auditory actor's interface. Constructs are provided for querying the named parameters of a specific sound, and for setting and querying their values.

Parameterization also aids in the simplification of the sound design process. A single Timbre Tree can represent a class of sounds. Specific results can then be achieved by varying a small set of intuitive parameters associated with each class. We have used a Timbre Tree to represent metal chime sounds, for example. The tree was constructed to generate chime sounds based on the dimensions of the chime which are fed into the tree as parameters. Clearly such a representation simplifies the sound design problem from a user's perspective. One of the goals in the design of Timbre Trees was that libraries of such trees would be developed, giving the user a wide choice of parameterizable sounds.

Sound objects model the sound sources within the auditory world. Two types of sound objects: *sampled sound* and *synthetic sound,* model sound sources using digitally sampled and functional representations of sound respectively. Sound objects are implemented in VAS as active objects. Each instance of a sound object has a light weight process associated with it which is responsible for generating the sound. Upon instantiation, a sound object registers an evaluation procedure with the VAS scheduler which in turn spawns a light weight process to generate the sound. Control over this process is completely with the scheduler. Sound objects communicate with the scheduler in order to stop and start the sound generation processes. A shared memory area is used to allow the sound object the necessary access to the sound generation processes' data structures. This scheme was devised in order to facilitate the development of resource management algorithms in the scheduler.

4. VAS DEVICES

VAS device objects provide the system with a generic interface to any spatialization device used by the server. This may be a hardware device connected to the server, or a software process running on the server. An abstract device object provides the auditory objects in the server with a standard interface for interacting with the spatialization device. It provides primitives for opening and closing the device, writing sound samples to the device, positioning sounds in space, and positioning and orienting the listener. New devices are incorporated into VAS by specializing the device object through inheritance. The specialized object implements the primitives specified by the abstract device by interacting with the programming interface supplied with the spatialization hardware, or by directly implementing a spatialization technique. In this fashion the internal workings of the server need never change to support new devices.

As an alternative to hardware-based spatialization, VAS provides a no cost spatialization option using the audio capabilities found in many computers. A specialized device object was derived which uses the four channel output available on most Silicon Graphics computers. Each device object generates four channels of output properly scaled to position the sound to the desired location in virtual sonic space. The spatialization algorithm determines the scaling for each channel. A straight forward approach to spatialization using four speakers would be to linearly scale each channel based on the sound's relative distance from each speaker. This approach, however, suffers from the "hole in the middle" effect; As a sound travels between two speakers, there is a marked falloff of the sound's intensity reaching a trough midway between the speakers. In order to compensate for this, we incorporated an algorithm developed by Moore[7] which uses a non-linear scaling to counteract this effect.

5. RESOURCE MANAGEMENT

The process of evaluating a Timbre Tree involves a preorder traversal of the tree, at each level invoking the operations of the functional node to the results obtained from evaluating the child subtrees of the node. In order to generate a sample stream for playback, the tree must be evaluated once for each sample in the stream. Given that the sampling rate of high quality audio is 44.1 kHz, the tree must be evaluated 44.1k times a second. This places an extremely high demand on the processing hardware. The problem is further exacerbated by the requirement that the system support multiple concurrent sounds.

In a first attempt to address this problem, a multiprocessing scheme has been adopted using a shared memory MIMD machine. In this approach the sample stream is segmented into blocks of time corresponding to 50 ms each. The evaluation routine for each sound calculates a single sample block and writes the resultant samples to its device object. The upper bound on the block size is the minimum perceivable latency between the occurrence of an event in the VE and the onset of the sound corresponding to that event. Because the evaluation routines are executed as light weight processes, the operating system automatically handles the parallelism among them. While this approach enables VAS to generate a small number of synthetic sounds in real-time, it quickly degrades as more sounds are activated, and silent pauses begin to appear in the generated sound.

A more rigorous approach being developed incorporates a real-time scheduling strategy with degradation of the quality of the generated sounds in order to support a large number of concurrent sounds. The main idea of this approach is to prioritize the active sounds in the environment based on an evaluation of which sounds the listener may be attending to. As the load increases, the active sounds are non-uniformly degraded by lowering their sampling frequencies based on their priority. Optimization techniques are also being investigated which would minimize the evaluation time of a Timbre Tree by intelligently pruning subtrees during evaluation.

6. CONCLUSION

The Virtual Audio Server begins to address some of the problems currently limiting the wide spread use of sound in VE interfaces. It provides high level abstractions for modeling the auditory world making the integration of sound into VE interfaces fast and easy. Abstract devices make the system device independent thus able to support any present or future spatialization technique. Synthetic sounds support the mapping of motion parameters to sound as well as data sonification. Finally, resource management techniques being investigated have the potential to facilitate the support of a large number of concurrent sounds in VAS.

7. ACKNOWLEDGMENTS

We would like to thank the staff of the ENEWS program at the Naval Research Laboratory for their continuing support of this work. We would also like to thank the members of the graphics group at The George Washington University, particularly Joe Geigel and Suneil Mishra for their technical contributions to the design of VAS.

8. REFERENCES

1. M. J. Accetta, W. Baron, R. V. Bolosky, D. B. Golub, R. F. Rashid, A. Tevanian, and M. W. Young, *Mach: A new kernel foundation for unix development*, In Proceedings of the Summer Usenix Conference, July, 1986.

2. P. S. Strauss, and R. Carey, *An Object-Oriented 3D Graphics Toolkit, Computer Graphics*, 26(2), pages 341-349, July 1992.

3. S. H. Foster, E. M. Wenzel, and R. M. Taylor, *Real Time Synthesis of Complex Acoustic Environments*, Proc. CHI'91, ACM Conference on Computer Human Interaction, New Orleans, LA, April 1991.

4. J. M. Chowning, *The Simulation of Moving Sound Sources*, Journal of the Audio Eng. Soc., 1970.

5. T. Takala, and J. Hahn, *Sound Rendering*, Computer Graphics, 26(2), pages 211-220, July 1992.

6. J. Hahn, J. Geigel, J. W. Lee, L. Gritz, T. Takala, and S. Mishra, *An Integrated Approach to Sound and Motion*, Journal of Visualization and Computer Animation, Vol. 6, No. 2, pp. 109-123.

7. F. R. Moore, *Elements of Computer Music*, Prentice Hall, Englewood Cliffs, New Jersey, 1990.