Model-based interactive sound for an immersive virtual environment

Robin Bargar*, Insook Choi*, Sumit Das†, Camille Goudeseune*

*Audio Development Group, National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign
†Electronic Visualization Laboratory, EECS, University of Illinois at Chicago
rbargar@ncsa.uiuc.edu

Abstract

We discuss an audio rendering pipeline that provides real-time interactive sound synthesis for virtual environments. Sounds are controlled by computational models including experimental scientific systems. We discuss composition protocols and software architecture for hierarchical control and for synchronization with graphics. Rendering algorithms are presented for producing sound from a physically-based simulation of a chaotic and from higher-dimensional topological structures.

1. Introduction

In this paper we discuss the implementation of a rendering pipeline designed to bring sound synthesis and composition as research components into virtual environments (VE). We find that VE research provides a platform for projects closely related to computer music composition. We also find the VE research community is interested in the potential relevance of composition for their work, and the relevance of their work for composers. We have been developing a software-based sound synthesis and composition protocol to enhance the possibilities of collaboration. This protocol defines a pipeline from computational models to sounds. Along this pipeline we identify endeavors related to computer music including real-time sound synthesis, gesture-based interaction, composition algorithms, physically-based sound production models, and techniques for synchronizing sound with graphical events.

A rendering pipeline encourages composers to consider the entire synthesis process as a composition. Generating and controlling complexity is among the most difficult tasks of computer music, and the pipeline model is valuable for connecting complex systems with rendering engines. Rapid low-level communication from complex model to sound synthesis engine permit the composer to use interactive control of the complex model to control the sound.

2. VE Systems

Virtual environments are multiple-engine computation systems converging toward solutions for immersive human interface. Immersion has been associated with two classes of experience: fictional constructs and feedback constructs.[1] Music listening may be identified primarily with the first class and music performance primarily with the second. In order that sound provide aspects of both classes of experience we embark toward a familiar goal: techniques for generating dynamical audio spectra informed by real-world analysis, using efficient numerical encoding that allows synthesis with interactive variation in real-time.

[1] Fictional constructs involve an observer's "willing suspension of disbelief" that supports the *diegesis* (the narrative world created in literature, theater and cinema). Feedback constructs are "everyday experiences" that an observer constructs by taking actions and observing their consequences through multiple sensory modalities.

Existing computer music systems provide some of these capabilities in specialized hardware. Rather than adopt existing music systems we have focused on the importance of demonstrating that sound synthesis is relevant for general-purpose computing. We want researchers to have immediate access to sound computation in the same language, operating system and control flow that supports standard computing and graphics rendering engines. Therefore our pipeline is written in unix/c/c++ to maintain potential portability and scalability and stay close to graphics architectures and their user communities.

2.1 3D Primitives

The reality image in computer graphics is encoded in an objective 3D embedding space. This objective encoding describes the range of potential subjective views that may be obtained from the image. Actual 3D rendering depends on hardware that is downstream from the objective 3D encoding. We adopt similar subsystems for audio. A primary software language encodes oscillations, envelopes, and sound propagation information, and manages rendering of dry (non-localized) sources. Secondary, scalable display subsystems are used to generate localization and depth cueing. Regardless of rendering hardware, 3D features of the propagation environment are encoded as attributes of the primitive description of a sound.

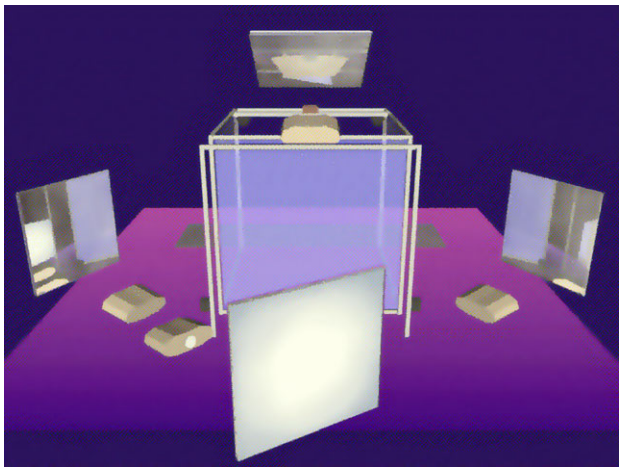

Figure 1. CAVE Automatic Virtual Environment University of Illinois

2.2 The CAVE virtual environment

The primary testbed for our sound system is the CAVE, a surround-sound, surround screen, projection-based VR system designed to convey an unencumbered immersive group experience [DeFanti, 1992]. 3D computer graphics are projected into a 10′×10′×10′ cube composed of display screens that completely surround the viewer (figure 1). A head and hand-tracking system produces the correct stereo perspective and isolates the position and orientation of a 3D input device. CAVE observers do not wear helmets or obstructing gear. Instead they put on lightweight stereo glasses and walk around freely, interacting with virtual objects and with one another. One set of glasses carries a magnetic tracking device that allows the stereo projection reference point to be computed for a mobile point of view. The wand, a 3-D mouse provides 6 degrees of continuous control plus 3 buttons and a pressure-sensitive joystick.

2.3 The CAVE Audio Display System

CAVE audio uses speakers and a MIDI-controlled distribution matrix (figure 2). Independent localization of up to 4 sources is provided by MIDI-controlled attenuation at each speaker. Distance cues are provided by MIDI-controlled reverberation and delay; the mix of wet and dry signal may be controlled independently for each source at each speaker. The reflectivity of the screens tends to confound directional cues at excessive loudness levels. Headphones provide better imaging but are encumbering and present problems when multiple users are in the CAVE. Infrared wireless headsets can interfere with two other infrared signal systems in the CAVE (projector control and stereo glasses synchronization). Computing a unique audio focal point for each user is also problematic. Listening positions can be computed only from the position of the tracker. If other CAVE users receive this signal they will be listening "inside" of the active user's head.

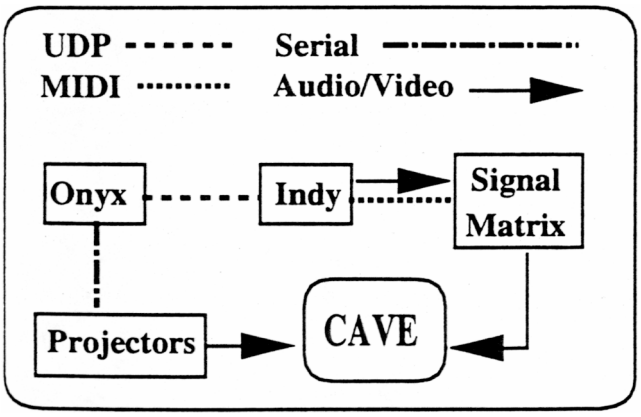

Figure 2. CAVE display systems.

3. Sound Composition Protocols

There is a question whether sound can carry "extra-musical" information without forfeiting its status as a composition. In text-based music, in music drama, in composed quotation, imitation, and sampling we find formulations that address but do not resolve this question. VE expects sound to play an informative role, to carry what has been understood so far as extra-musical information. Acoustic information space is taken as the embedding space for music composition. A composer's task in this case can be to reverse the roles, and make a composition as an embedding space for "extra-musical" sounds.

We designated sound protocols to help construct a relation between informative sound and composition. The functionality and architecture of the audio pipeline was developed according to these protocols.

3.1 Data Driven Sounds

Data driven sounds respond to numerical patterns that are generated in a computational model. We begin by asking "how can this model be interpreted as if it produces sound?" In some cases the data patterns are continuous and the values may be used as sound samples directly. In other cases data values are mapped onto synthesis parameters. Each of the following categories is to some degree data-driven if we consider the measurement of user actions a data stream.

3.2 Field Sounds

Field sounds are dynamical background sounds, constructed ambience generators that have internal behavior and also respond to data from the VE. In this sound protocol we apply the concept of ecosystem as a living system in which many suborganizations are present and interact in order to contribute to the globally evolving changes. Global changes have their own internal clock in automated fashion. In addition to this automation, field sounds are locally responsive to an observer's activities by generating changes that correspond to the actions that observer takes in CAVE, such as physical movements or wand operations over time. These local changes are not isolated signals synchronized with the user, they are changes woven into the fabric of the sound field.

3.3 Flying Acoustic Information Space

In this protocol we use the observer's location and point of view to provide the definition of the scope of information retrieval. We view for example the globe as an information source and storage system from which retrieval of the information will be operated in interactive mode. We store concrete sound samples at points on the earth's surface and these sounds will be retrieved and 'sampled' based upon where the user's point of view is projected. The degree of clarity and complexity in the mixture will vary in correspondence to the location an observer. The flight path through information space can be thought of as operating a dynamical audio mixer. The map of the earth's surface acts literally as a map of information storage locations.

3.4 Sound Cues and Complementary Sounds

The function of the sound cue is to inform or alert an observer to information which her scope of observation (field of view) does not reach. It can be designed to reveal hidden layers of information due to any visual limitations, also to emphasize selected layers of information due to overflowed complexity in data representation. Complementary sounds

can be understood as a subset of the Data Driven Sounds protocol. The emphasis is on the feature that sounds will present complementary information to the visual information. The purpose is to simplify data representation without reducing information by presenting it in a multi-modal fashion.

4. Software Architecture

By including complex systems models in our concept of audio pipeline we make it necessary to import these models into our software environment. There are a formidable number of numerical models that may be valuable for sound synthesis or composition purposes. Since these models are already implemented in software by experts in various fields it does not make sense to duplicate their efforts while trying to generalize their code. We have adopted a client-server architecture so that existing software models can be used as control programs for synthesis software, with a minimum of re-programming.

The server can process messages sent to high-level composition routines or to low-level primitives (figure 3). Our class hierarchy includes 3 subsystems: at the low end, the scheduler and sample buffers along with basic synthesis algorithms, a middle layer that defines note events and synthesis instrument configurations, and a level for describing complex musical events.

4.1 HTM

HTM [Freed, 1992] is a system for real-time interactive sound creation. It is based on the client-server model. In this model, the application program which needs sound is called the client. The client sends requests to the server, which is another program, usually running on a different computer. The server then fulfills the client's requests to the best of its ability. The HTM server is a program that accepts commands from a client application program and schedules message processing and sound sample generation.

On top of this are implemented a number of synthesis algorithms that HTM uses to generate its samples, such as FM, additive synthesis, sample playback, and MIDI. We call this the Vanilla Sound Server (VSS).

4.2 VSS

VSS is based on the concept of a note event, which is a continuing auditory event that has a unique identity. When the client starts a note playing on the server, a note handle is returned. This is a floating point number that can be used to refer to this note in the future, so that the client can, for instance, change the pitch of the note or turn the note off.

4.3 Group Functions

Functions that control groups of parameter changes are implemented above VSS to provide higher level control of the existing functionality of VSS. Groups are composed of dynamic objects that hang around just above the level of VSS. The client program can communicate with these objects to control VSS. In this way, the client can take advantage of the object's built-in rules and knowledge, making the interaction much simpler and higher-level.

The objects that make up the complex models provide access to all the functionality in VSS, and preserve the concept of the note handle. In addition, each object also has a unique handle, so that the client can send multiple message to the same object. As with notes, the object handle is a floating point number returned to the client when an it is created or retrieved.

For each VSS synthesis algorithm, there is a corresponding object that basically functions as an interface wrapper for this algorithm. Many instances of each object can be created, and can either act independently or in tandem.

For every command applicable to a VSS algorithm, there is a corresponding message you can send to its higher-level object, so you don't lose access to lower-level functionality by using these objects.

The messages that objects send to each other are in the same form that the client uses to send messages to the server. The result of this is that an object does not know or care whether a message comes from a client or from another object. This is useful in building up a network, as the client can test different subsets of the network independently.

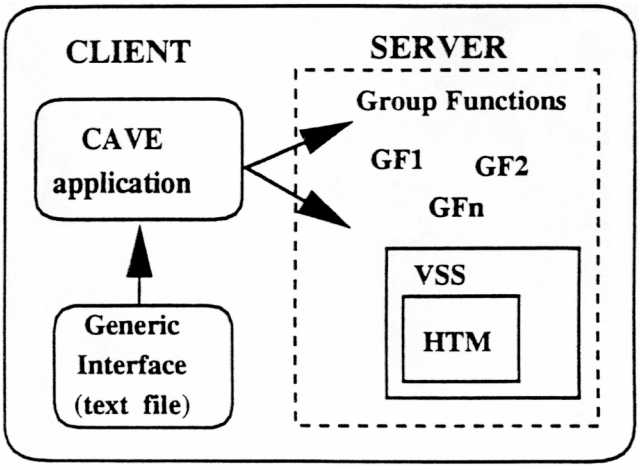

Figure 3. CAVE Audio Software architecture.

4.4 The Generic Interface

The generic interface is intended to simplify the task of adding and modifying sound in an application. It is designed so that although the flow of control and structure is defined in the application code, the types of sounds that are actually played are defined externally, in an input file. This allows an application's sound to be modified without changing or recompiling the application itself. To use the interface, the client must tell the server which objects it wants to use and how it wants those objects configured. Then, the client will send data to the objects configured. Then, the client will send data to the objects, either at regular intervals or whenever a state in the application changes.

5. Computational Models

Models for controlling the server are usually based upon computational systems developed in the sciences. To be

useful in the audio pipeline these models require some adaptations to their software and some interpretive consideration by the composer. When models are incorporated they have a numerical systems component and an interpretive component. The interpretive component is often realized as an interface that selects salient features of the model for interactive control. An interface can help the composer navigate complex parameter spaces and differentiate patterns and features from noisy or redundant regions in the model.

5.1 Chua's Circuit

Many aspects of our current system were prototyped during the study of a chaotic electric circuit. Chua's circuit produces many types of signals, from sine-like periodic patterns to intermittent and unpredictable noise-like patterns [Rodet, 1993]. Using a digital simulation of the physical circuit implemented as a set of ordinary differential equations, we apply navigation and control techniques to chaos for generating musical signals. Sound is created by converting the numerical representation of the voltage directly into sound samples. We designed the manifold interface to facilitate the navigation of the control space of the circuit (figure 4). The manifold describes a function for the continuous transformation of parameters mapped to the axes of the cube. Paths may be traced on a manifold in real-time, and retraced automatically to provide a reproduction of sound sequences. A unique aspect of this interface is the simultaneous presentation of control space and multi phase space of the circuit.

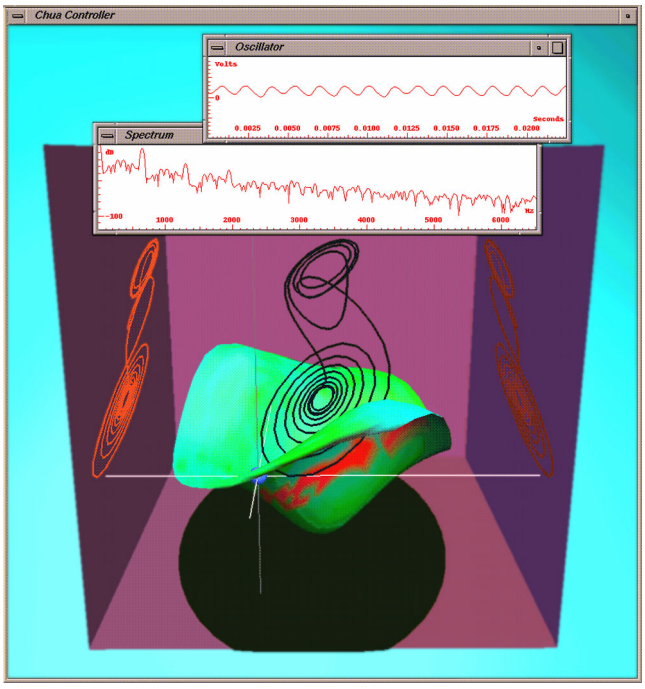

Figure 4. The Manifold interface.

5.2 Alpha Shapes

We have been exploring the use of sound to represent higher-dimensional topological structures, with an interest in the reciprocal use of topology for controlling sound. A finite set of points in 3-dimensional space and a real parameter alpha uniquely define a simplicial complex, consisting of vertices, edges, triangles, and tetrahedral embedded in space. We call this the alpha-complex of the points. The alpha-shape is the geometric object defined as the union of the elements in the complex [Edelsbrunner, 1994].

Alpha shapes can be viewed as generalizations of the convex hull of the point set. It formalizes the intuitive notion of shape, and for varying parameter alpha, it ranges from crude to fine shapes. The most crude shape is the convex hull itself, which is obtained for very large values of alpha. As alpha decreases, the shape shrinks and develops cavities that may join to form tunnels and voids.

An audio experience of the complex is based on beginning at an arbitrary point and advancing through the shape. We map different features of this wave and its history to different sound parameters. The goal is to explore the complex with meaningful auditory and visual cues. For example, the development of the wave is mapped to the sequencing of sound envelopes and thus provides audible expression of topological connectivity information. The smoothness of the wave is reflected by the shape of the sound spectrum, and the combinatorial size is mapped to frequency. The local dimensionality of the complex controls the complexity of the sound through the recursive generation of waves of lower dimensions.

6. Conclusions

Both the Chua's circuit and the Alpha Shapes project have resulted in new representations of science-based models. These models may yield new methods for efficient control of sound synthesis algorithms. The rapid progress on each of these projects during the same one-year period can be attributed to the presence of a real-time interactive sound synthesis pipeline in a graphics-oriented workstation environment.

7. Acknowledgements

We wish to thank Kelly Fitz and Ulrike Axen of the NCSA Audio Development Group for their contributions and suggestions. We are grateful for the support and interaction provided by Leon Chua and Herbert Edelsbrunner.

8. References

[DeFanti, 1992] T.DeFanti, C.Cruz-Neira, D.Sandin, R.Kenyon, and J.Hart. The CAVE: Audio Visual Experience Automatic Virtual Environment. *Comm. ACM 35*, 6, 64-72. June 1992.

[Freed, 1992] A. Freed. Tools for Rapid Prototyping of Music Sound Synthesis Algorithms and Control Strategies. *Proc. Intl. Computer Music Conf.*, San Jose, 1992.

[Rodet, 1993] X. Rodet. Models of Musical Instruments from Chua's Circuit with Time Delay. *IEEE Trans. Circuits and Systems-II 40*, 10, 696-700. September 1993.

[Edelsbrunner, 1994] H.Edelsbrunner and E.P.Mucke. Three-Dimensional Alpha Shapes. *ACM Trans. Graphics 13*, 1, 43-72. January 1994.