BROWeb: An Interactive Collaborative Auditory Environment on the World Wide Web

Eric Metois, *MIT Media Laboratory*
Maribeth Back, *Xerox PARC*

Abstract: We describe an infrastructure for a real-time shared auditory environment on the World Wide Web (WWW). The system consists of the BROWeb "star-shaped" Web server and the StarClient Java class and interface. The BROWeb server is designed to facilitate implementation of Java applets wherein users see and hear each other's activity. Developed as an interactive music system for public performance, BROWeb is robust enough to support hundreds of users in a demanding application.

We also discuss design issues regarding the actual experiences that a Java applet overlying this infrastructure can deliver. We created, for instance, a privileged user--a "director"--who can make decisions affecting the overall system's performance and also the data streams from other users. As for the sound source of the actual experience, we present two alternatives that we have tested: local sounds that download when the experience begins and streamed audio (RealAudio, Xing, or other network broadcasting tool).

Description of the Infrastructure

For this project, we envisioned a set of collaborative instruments that provide sonic experience on the WWW. Our goal was to allow a rich variety of local musical play for the individual at any Web station, while allowing the resulting data to be gathered and redirected by a privileged user (the "director") from a central site.

In this structure, the director's data redistribution can affect the BROWeb user sites as well as control external sound or visual events at the central site.

The system was designed originally to support an interactive music system on the Web in conjunction with the Brain Opera, an electronic interactive opera by Tod Machover (1996). Individual users/players interact with applets designed as instruments or games and connected to an instance of our BROWeb server. Also one of the server's clients, The Brain Opera's performance system stands for the "director" of our virtual community.

Constraints

The major problems with Web-based interactive music are lag and scheduling problems because the limited bandwidth restricts packet-based data delivery and sonic quality. A Java applet allows use of only a small amount of sonic material, and that little is poor quality: Java currently supports only 8-bit mulaw audio files and very limited handling.

These constraints affected our project in the following ways:

Limited download time

> To avoid lag, we vowed to constrain our download time within 100K. Because downloading samples means establishing a new connection for each sound file (which often costs more time than the actual downloading), we limited both the number and file size of samples.

Limited real-time bandwidth

> Responsiveness often is the key to an interactive installation's success. To achieve this, we kept the data that flows back and forth at very low bandwidth (less than one kByte/second).

Unreliable UI (user interface) timing of events prohibited literal mapping.

Underpinning Features

Server Architecture

A central "star-shaped" server is responsible for handling all the interactive connections between web clients and the primary performance system, which is another web client. This program monitors connected clients and dispatches any message that one client sends to the rest of the appropriate community (or group). Groups or user communities can vary in size, and size should be set on the server-side. A group size might be limited to ten, including the primary performance director and the individual local user.

Sound Source

The bandwidth constraints of responsive interactions require that the exchanged messages be very small and low bandwidth (point-and-click level). This requirement prohibits sending actual sound samples in real-time. We thus studied two alternatives. One alternative was to force all sound files to reside locally on the client machine, which means that all sonic material should be downloaded by the Java applet at the beginning of the experience. The other alternative was to use an audio web-broadcasting tool to stream the sonic result of the collaborative experience back to the users. A combination of the two is possible, depending on the actual design and implementation of a specific platform's audio driver. To support as many platforms as possible, however, we considered these two alternatives exclusively.

Timing and scheduling

Using local sound files as our sonic material required that each client schedule the local sound output. This timing is crucial to music and sounds in general. To overcome the jerky timing of network communication, we used a system design that exchanges information on the net as "time-free." Once the information is exchanged, the system reconstructs the actual timing on each client via some simple but effective scheduling. We have built a simple scheduler in Java that performs this task for BROWeb applications that use local files. Using audio streaming, moreover, increases the need for time-free interaction; audio broadcasting utilities can introduce dramatic delays ranging from 5 to 10 seconds.

High-Level Features

Director Experience

For a BROWeb design to create an experience similar to a "director" or a "conductor," its instruments must offer the director levels of control different from the individual users. In one design, users place sound events using their local Web instruments, while the director governs the particular set of sounds, modifies tempos, and chooses types of pattern formation (Yu, 1996).

Musical Community Experience

For the users, a sense of musical community develops. Not only can a user hear the output of other users, but graphics and sounds are designed to humanize the output, to indicate that the humans are the ultimate source of the music. Limiting user groups to manageable proportions (ten or fewer) fosters this sense of personal contact; a player develops some sense of other players' styles.

Technical Description

Server Side (Written in C)

A user group in the system is linked to a specific experience. For instance, each interactive web game/instrument (Java applet) has its own group that is distinct from the "director's" group. This notion of "user group" is associated with the actual port on which this communication occurs. The group structure is scaleable: a single instance of the server deals with a single group. Many instances of that same program running on the same machine, each listening to a different port (and therefore dealing with a different group or community), can occur. The server, therefore, must handle any type of message.

Client ID

Each web client receives a unique ID number. This ID is assigned automatically on the server side. The ID #0 is reserved for the server so that it can occasionally send personalized messages or general announcements. When a new client connects, the first message it receives contains his or her assigned ID.

Messages

From the server's point of view, a message is a string of ASCII characters terminated by the '\n' character. Incoming messages from the clients are multiplexed and redistributed to all the clients, which also means that each client's own messages are echoed to itself. From the client point of view, the first token of that string stands for the ID of the sender: `35 this is a message\n` (i.e., client #35 sent "this is a message") As another example, the first message that a new client will receive after connecting to the server will resemble `0 WELCOME 78\n` (i.e., the new client was assigned ID #78).

Java Side (Java Class and Interface)

To connect a Java applet to one of our servers and to enable communication between the applet and the server's clients, we wrote a very general Java class with an appropriate interface (interface in the Java sense, not the user sense). This class addresses the low-level communication layer, including fetching its client ID from the server, and runs on its own thread on the client side.

Implementation Summary

BROWeb's infrastructure was developed and tested at the Media Lab's Hyperinstruments group in spring of 1996. The first test consisted of a MUD-type simplistic 3-D environment where Web clients, represented as stick figures, can move around and see each other (available at *http://w.media.mit.edu/~metois/MyJava/*). Once stable, this infrastructure was released internally to the Brain Opera team and led to various game and instrument applications (some are available from the Brain Opera's Web site at *http://brainop.media.mit.edu/* (Machover, 1996).

The Brain Opera itself premiered in July 1996 in New York City at the Lincoln Center Festival. As planned, its performance system had a Web outlet which allowed remote clients to interact in real-time with some sections of the piece. Although the original design used local sound files, the Brain Opera ultimately employed audio streaming (using the Xing audio netcasting software from Streamworks) We decided that audio streaming had potential to provide a richer sound stream than a limited collection of local sound files. With the same web implementation, the Brain Opera performed in Linz, Austria at Ars Electronica in September 1996.

Design Choices for the Musical Engine

The choice between local sound files or audio streaming obviously will influence the design of an appropriate musical engine. In the case of audio streaming, the musical engine resides on a single and centralized platform and can mobilize a large amount of CPU and sonic material. Audio streaming has its drawbacks, however: delay

(and hence a lack of responsiveness) and the dedication of a centralized system for the sound engine. Using local sound files, on the other hand, leads to better responsiveness and employs less centralized resources. The drawback of local sound files is that they require duplication of the musical engine on each client machine in Java, which limits the complexity of the engine to the lowest common denominator of client platforms.

After assessing these approaches, we conclude that both are worthy of BROWeb. The choice of an audio source and a design for the music engine, therefore, is determined by the context of the desired experience. As the system already has much CPU dedicated to it locally, audio streaming better serves an interaction with a composed performance such as the Brain Opera. A more "democratic" (in the sense of "less pre-composed") musical experience, could employ local sound files effectively.

Local Sound Source

To develop an original design that could lead towards a multi-user "jamming" experience, we chose "spinning disks" as an analogy for the suggested musical engine. At any time, each disk is associated with a small set of sounds (e.g. five) and an appropriate representation of a score. The spinning represents time; a full revolution is essentially a loop of the associated scores. Each disk, with its own tempo and memory, schedules its score. The memory of the disk determines the life span of any of its sonic events.

The users can place objects on any of these disks, adding sonic events (one of the sounds in the current set) into the appropriate score. Every user receives the outcome of the other users' interactions so that everybody within that user group will hear and shape the same sonic experience.

Although in the example that we show the actual interface on the player side reflects literally the mechanism of the musical engine (spinning disks), its nature could be completely different. Graphics can be any shape that adequately communicates the system's parameters to the user.

Audio Streaming: The Brain Opera

The Brain Opera's performance system is a very complicated structure comprising five computers and a battery of sound modules. Within this system, the Brain Opera team integrated a musical engine that is based on work by John Yu (1996). This engine turns 16 parameters into an appropriate MIDI stream. Ranging from "scales" to "activity" and "rhythm styles," these parameters can be constrained locally to conform to the rest of the performance. The infrastructure described in this paper was used to collect parameter settings from remote users. The Java applet that is downloaded on the client side, therefore, is a directly shared interface to this musical engine, where various clients can compete to tune the desired musical outcome (Palette instrument, 1996). Representing remote clients as small circles that bump into each other, each client's interface illustrates the musical competition (Figure 1).

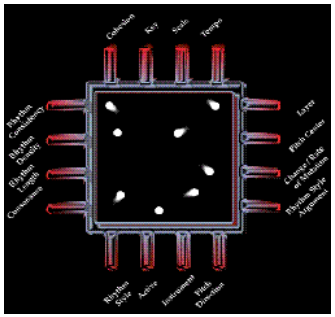

Figure 1: The Brain Opera's Web instrument. The 16 parameters on the sides of the square are mapped to appropriate controls over the music engine. The circles in the middle and their behavior reflect the behavior of the remote community.

Future Work

In future work, we plan to investigate client-side synthesis controllers such as Java MIDI for better sound and more versatile parameters. As the Brain Opera tour continues into 1998, we will adjust the system's design to accommodate new Java audio capabilities. We anticipate the use of the BROWeb system in various applications, including auditory display and graphical MOOs. BROWeb also will aid further investigations of the usage and systemic implications of net broadcasting and multicasting both audio and video within Java applets.

References

Machover, Tod. (1996). "The Brain Opera and Active Music." In *Ars Electronica 96, Memesis, The Future of Evolution.* (p. 300). New York: SpringerWien. Available at *http://brainop.media.mit.edu/*

Metois, Eric. (1996). WebStar Server and StarClient Java class and interface. Group report. Cambridge, MA: MIT Media Lab.

Palette instrument. (1996). Available at *http://brainop.media.mit.edu/online/net-music/net-instrument/net-instrument.html*

Yu, Chong (John). (1996). Computer generated music composition. Master's thesis. Cambridge, MA: MIT, EECS. Available at *http://www.geocities.com/Hollywood/Hills/1197/thesis.html*

Author Information

Eric Metois
MIT Media Laboratory
20 Ames St.
Cambridge, MA 02139
(617) 253-9488 Fax: (617) 258-7168
http://physics.www.media.mit.edu/~metois/
E-mail: metois@media.mit.edu

Maribeth Back
Xerox PARC, 3333 E15-488
Coyote Hill Rd.
Palo Alto, CA 94304
(415) 812-4409 Fax: (415) 812-4471
http://brainop.media.mit.edu/~mbb/
E-mail: mbb@media.mit.edu