Using the Web for Live Interactive Music

John P. Young

Peabody Conservatory of Music
Johns Hopkins University
1 East Mt. Vernon Place
Baltimore, MD 21202 USA
jpyoung@peabody.jhu.edu

Abstract

This paper describes an exploration of utilizing the World Wide Web for interactive music. The origin of this investigation was the intermedia work Telemusic #1, *by Randall Packer, which combined live performers with live public participation via the Web. During the event, visitors to the site navigated through a virtual interface, and while manipulating elements, projected their actions in the form of triggered sounds into the physical space. Simultaneously, the live audio performance was streamed back out to the Internet participants. Thus, anyone could take part in the collective realization of the work and hear the musical results in real time. The underlying technology is, to my knowledge, the first standards-based implementation linking the Web with Cycling '74's MAX. Using only ECMAScript/JavaScript, Java, and the OTUDP external from UC Berkeley CNMAT, virtually any conceivable interaction with a Web page can send data to a MAX patch for processing. The code can also be readily adapted to work with Pd, jMAX, and other network-enabled applications.*

1. Introduction

Telemusic #1 originated from a long-held artistic vision of Randall Packer. Through the process of close collaboration and harmonization of compositional goals with technical feasibility, it became realized as an interactive musical work incorporating live performers, signal processing, and real-time participation via a public Web site. Making this concept a reality required the extension and integration of existing technologies, in ways that had not previously been documented. One primary objective for the software development was to adhere to open, established standards as much as possible. This mandate led to the use of Java and JavaScript (standardized as ECMAScript) for the core components, and the OpenSoundControl (OSC) objects from the UC Berkeley Center for New Music and Audio Technologies (CNMAT) for interfacing with Cycling '74 MAX. The resulting system enables anyone interacting with a Web site to send data over the Internet to a computer running MAX, which can then process the input as it would from any other source. In a sense, one can transform the Web into a musical instrument performed by an unseen ensemble of limitless proportions.

2. Background

Previous attempts to interface standard Web technologies with interactive music have involved substantial limitations. *Variations for WWW* (Yamagishi and Setoh 1998), was a breakthrough for its time. However, it only worked with form-submitted input, substantially limiting the flexibility and reactivity of the user interface, and was based on David Zicarelli's W protocol and W server, which are no longer supported. JSyn (Burk 1998) is an excellent tool for producing interactive audio entirely within a local browser environment. This software was extended with TransJam (Burk 2000), a server-based module enabling communication between multiple JSyn users. However, using the system requires installation of the JSyn plug-in on each client, the TransJam server component is not available to the public, both are closed-source projects, and they do not include any facility for communicating with other common software applications such as MAX. 'Piano Master Classes via the Internet' (Young and Fujinaga 1999) introduced musically sophisticated error-correction for near-real-time collaboration over the Internet. However, the musical elements of the system were entirely based in MAX, and could not be integrated with a browser-based interface.

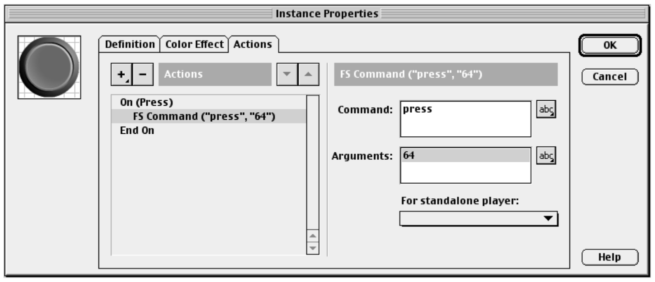
The research supporting *Telemusic #1* aimed to pick up where these predecessors left off, enabling a Web-based interactive interface to control the musical capabilities of MAX/MSP in either a client/server or peer-to-peer fashion, using sophisticated error correction for transmitting live control data over the Internet. In addition, full access to the relevant source code will be made available under the GNU General Public License (GPL), allowing freedom to manipulate the system so long as any modifications or enhancements are submitted back to the community under the GPL, for all to share (Free Software Foundation 2001).

3. Inception

Telemusic #1, a collaborative intermedia work by Randall Packer with Steve Bradley and myself, was the driving force behind development of a distributed input environment functioning in parallel with a live performance. The physical aspects of the piece took place on Friday, November 3rd, 2000 at the Sonic Circuits VIII International Festival of Electronic Music and Art in St. Paul, MN, USA. A Web site, www.telemusic.org, was constructed for the event, presenting an interface for visitors worldwide to participate in real time. The primary controls were implemented in Macromedia Flash. Visitors to the site could define a 'telematic identity' and navigate through a virtual space. By manipulating various elements, they projected their actions into the physical space in the form of triggered samples, filter parameters, and other live sonic results, according to Packer's compositional design. In addition, an active audio feed was streamed back to Web participants using RealAudio, providing a stereo rendering of the performance in progress. Thus it was possible to interact with the Web site and hear concrete audible feedback of one's actions in a matter of seconds. The Walker Art Center, one of the sponsors of *Telemusic #1*, also cooperated by invisibly re-coding their home page, mapping various links so that visitors to their Web site became unwitting participants in the piece. The resulting work was a dynamic balance of precomposed elements and improvisation, as the input received from the Internet was inherently unpredictable. By most accounts, it was a successful premiere, both technically and artistically.

4. Architecture

The original software implementation was narrowly focused on the task of enabling a Flash animation within any Web browser on the Internet to send data to a specific computer running MAX, with minimal delay. Addressing this initial requirement began with the Flash FSCommand(), a built-in method for addressing JavaScript that allows custom functions to be executed based on seven possible object behaviors in a Flash movie: Press, Release, Rollover, Rollout, DragOver, DragOut, and KeyPress. Here is a Flash inspector window attaching an FSCommand() to an object:



FS Command ("press", "64")

```
On (Press)
  FS Command ("press", "64")
End On
```

Command: press

Arguments: 64

FSCommand() specifies two pieces of data—a command, and arguments. In this case the command is "press," signifying the action performed, and the argument is "64", the number that will eventually find its way to MAX.

When an FSCommand() is triggered in a Flash movie, the command and arguments are passed via the Document Object Model (DOM) to the host browser to be processed by an appropriate JavaScript function. Here is a simplified JavaScript function to handle the FSCommand() above:

```
<SCRIPT LANGUAGE=JavaScript>
<!--
function TeleButton_DoFSCommand(command, args) {
   if (command == "press") {
   document.TeleApplet.press(args);
   }
```

This JavaScript simply acts as a middleman to catch data coming from the Flash movie and pass it on to a Java Applet—`TeleApplet`—embedded in the same page. Known as LiveConnect, this communication between plug-ins, JavaScript, and Java Applets originated with Netscape Navigator 3.x, and is now supported in most browsers. The `TeleApplet` also performs as an intermediary, except instead of making a local connection, it transmits long-distance, over the network. Here is the simplified Java code:

```
public class TeleApplet extends Applet {
   public void press(String args) {
      sendData(args);
   }
```

Details of the `sendData` method are not shown because it is simply a standard implementation of TCP/IP socket communication. At the other end of this link is a Java Application—`routeOSC`—listening on the host Web server. `routeOSC` receives the data from the remote client, reformats it into the OpenSoundControl protocol, and finally transmits it to MAX. Here is the basic idea, in pseudocode:

```
public class routeOSC {
  try {
    ss = new ServerSocket(port); // to receive TCP
    buf = new StringBuffer();
    String args = "";
    ia = new InetAddress(host);  // destination IP
    ds = new DatagramSocket(ia); // to send UDP

    socket = ss.accept();        // listening...
    buf = new StringBuffer(socket.read());
    args = String.valueOf(buf);  // parse data

    toOSC(ds, args); // reformat and transmit args
  }
```

Again, details of `toOSC` are not shown because it mostly involves string handling and a standard implementation of UDP datagram communication. Of course, once the data arrives in MAX, it can be freely interpreted as control information no different from that arriving via MIDI, internal processes, the local GUI, or elsewhere.

Hopefully this description has made it clear that the whole process from beginning to end is rather straightforward. Every attempt has been made to maintain simplicity and transparency from concept to implementation, so that others will be able to readily understand the existing code and adapt it to whatever new purposes they can imagine.

The data path may seem rather convoluted, but is necessitated by limitations of third-party software and the intentional containment of browser capability for security reasons. Flash doesn't include any means of external communication other than passing an FSCommand() to JavaScript. JavaScript has no networking functions of its own, so must call methods from a Java Applet in order to connect with anything outside its window. Java Applets are constrained by the well-known 'sandbox', which prohibits establishing a network connection with any machine but the one from which the Applet was loaded—the host Web server. Once the data is received by the Java 'listener' application running on the Web server, it is finally free of restrictions and can be reformatted for any supported protocol and sent to any machine on the Internet. The only way to circumvent all these steps would be to create a browser plug-in specifically designed to violate the well-established browser security model in return for a modest increase in expediency.

After successful initial testing, more features were soon added to offer an increased range of interactive possibilities. Independent of Flash actions, data could also be sent in response to common browser event triggers including page loading (onLoad), clicking links (onClick), element rollovers (onMouseOver), etc. Also, in order to uniquely identify Web visitors, a means of writing cookies, and including that ID in each bundle sent to MAX, was developed. In theory, any interface that can be coded in JavaScript, as a Java Applet, or as a plug-in that can address the DOM, could successfully integrate into this system. Once the compositional challenges of inherent network latency and decoupled feedback to the user are accepted, the potential for realizing distributed interaction is boundless.

5. Limitations

One caveat with the software is that it is currently limited to using the OSC protocol developed at UC Berkeley CNMAT to communicate with MAX. Despite its moniker, OpenSoundControl is not entirely open. Much of the general specification has been published (Wright 1998), but source code to the specific implementation used by the MAX external objects is not available, and substantial portions of the software fall under restrictive copyright of the UC Regents. However, OSC, and OTUDP, its companion object for UDP-based network communication, are the only networking externals currently supported for the MAX environment, and they function quite well in this capacity. OSC/OTUDP will continue to be the primary target, but interoperability will be extended to the open-source projects Pd by Miller Puckette and jMax from IRCAM as well, for those working in these environments where complete end-to-end control of software is desired.

This system was designed for maximum cross-platform compatibility. It has been successfully tested using Netscape Navigator under Windows, MacOS, and Linux, and Microsoft Internet Explorer (IE) under Windows. IE under MacOS does not work because current versions of IE:Mac unfortunately do not properly support LiveConnect, thus communication between JavaScript and Java is broken. The server application has been successfully tested under Windows, Linux, and MacOS X. Because most of the software is written in Java, it is quite portable, and every effort will be made to ensure ongoing compatibility with common operating systems and browsers.

6. Current Development

My collaboration with Randall Packer has continued in the form of his next intermedia work, *Telemusic #2*. This piece is building technologically on its predecessor by interpreting network traffic statistics gathered in real time to generate an organically textured musical atmosphere as a canvas for interaction. A prototype installation, with additional MAX programming by Margaret Schedel, was demonstrated at the "Music Without Walls? Music Without Instruments?" conference at De Montfort University, Leicester, UK from June 21-23, 2001.

The software required was inspired in part by two open-source projects: the Multi Router Traffic Grapher (Oetiker 2000), a tool for aggregating Simple Network Management Protocol (SNMP) queries, and Peep (Gilfix and Couch 2000), a framework for auralizing dynamic network conditions. While there was hope of sampling aggregate Internet traffic from a nearby Internet Service Provider (ISP), the procedural obstacles proved insurmountable. As a result, the prototype monitored network usage at NOVA Research Company, in Bethesda, MD, USA, a small business where I have full authority over the infrastructure, and substantial network traffic is generated during the workday. Raw statistical data was continuously collected using TCPSTAT (Herman 2001) on a GNU/Linux platform. The bursty nature of network activity was smoothed out by sampling over a two-second window, outputting metrics such as total number of bytes, total number of packets, and breakdowns by protocol such as ARP, ICMP, TCP, and UDP. A Java parser application digested the data, formatted it into OSC, and sent it across the Atlantic Ocean to MAX. More extensions to the `TeleApplet` were also written, to enable transmission of a Web client's IP address and a time stamp in addition to the Flash and browser triggers used previously.

This prototype of *Telemusic #2* was received very positively, with the ebb and flow of network utilization providing an intuitive audible representation of the passage of time at the source, particularly in the contrast between day and night. Looking beyond the horizon, Packer sees *Telemusic #3* adding musically responsive visualizations to the environment to create a more immersive intermedia experience.

7. Future Directions

By the time you read this, the initial beta release should have already occurred. The primary milestones for the first public version are to continue adapting the code from its orientation of fulfilling specific requirements towards supporting more general-purpose functionality, and to provide adequate documentation.

Beyond providing a stable and comprehensible code base with which others can experiment, there are numerous other features that have been envisioned:

- Data flow is presently one-way, from browser to MAX only. It would not be difficult to send a response back from MAX to the client Java Applet, allowing bidirectional communication and greater interactive potential.
- It would be prudent to add some form of handshaking and/or error-correction between the `TeleApplet` and `routeOSC`, for the sake of both data integrity and security.
- `routeOSC` is currently configured using command-line options. A graphical user interface for setting parameters and enabling adjustment on-the-fly would be helpful.
- It is possible to re-broadcast incoming data in MAX, but it would also be useful to enable `routeOSC` to send to multiple destinations.

8. Acknowledgements

This research owes a great debt of gratitude to my friend and collaborator Randall Packer, whose wonderfully impractical ideas pushed me beyond the limits I would have otherwise set for myself.

Portions of this paper were first presented at the “Music Without Walls? Music Without Instruments?” conference at De Montfort University, Leicester, UK from June 21-23, 2001, and have been accepted for publication in *Organised Sound* issue 6(2) published by Cambridge University Press.

9. Conclusion

The system described herein has been designed for flexibility and extensibility, offering a framework which can support many potential applications in addition to those already noted. I sincerely hope that development and sharing of this software can inspire further creative use of the Web as a means of participating in live interactive works. Feedback and feature requests are always welcome. The latest version and documentation can be found at http://www.netmuse.org.

References

Burk, P. 1998. “JSyn – a real-time synthesis API for Java.” *Proceedings of the 1998 International Computer Music Conference.* Ann Arbor, MI: International Computer Music Association, pp. 252-255.

———. 2000. “Jammin’ on the web – a new client/server architecture for multi-user musical performance.” *Proceedings of the 2000 International Computer Music Conference*. Berlin, Germany: International Computer Music Association, pp. 117-120.

Free Software Foundation. “GNU General Public License.” <http://www.gnu.org/copyleft/gpl.html>. 15 July 2001.

Gilfix, M., and A. Couch. 2000. “Peep (the network auralizer): monitoring your network with sound.” *Proceedings of the 14th Systems Administration Conference.* New Orleans, LA: USENIX.

Herman, Paul. “TCPSTAT.” <http://www.frenchfries.net/paul/tcpstat/index.html>. 1 May 2001.

Oetiker, T. 2000. “MRTG—the multi router traffic grapher.” <http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/paper/>. 1 May 2001.

Wright, M. 1998. “Implementation and performance issues with OpenSound Control.” *Proceedings of the 1998 International Computer Music Conference.* Ann Arbor, MI: International Computer Music Association, pp. 224-227.

Yamagishi, S., and K. Setoh. 1998. “*Variations for WWW*: network music by MAX and the WWW.” *Proceedings of the 1998 International Computer Music Conference*. Ann Arbor, MI: International Computer Music Association, pp. 510-513.

Young, J., and I. Fujinaga. 1999. “Piano master classes via the Internet.” *Proceedings of the 1999 International Computer Music Conference*. Beijing, China: International Computer Music Association, pp. 135-137.