Eos Pods: Wireless Devices for Interactive Musical Performance

David Bianciardi
Audio, Video & Controls, Inc.
415 Lafayette St. 2nd floor
New York, NY 10003
+1 212 353 9087
david@av-controls.com

Tom Igoe
Interactive Telecommunications Program
Tisch School of the Arts, NYU
721 Broadway, 4th floor
New York, NY 10003
+1 212 998 1896
tom.igoe@nyu.edu

Eric Singer
LEMUR
Brooklyn, NY
[edited out at Eric's request]

ABSTRACT

In this paper, we describe a hardware and software system for an orchestral performance in which the audience members are the performers, led by the orchestra's conductor. The system is designed such that audience members are not required to have previous musical knowledge. Audience members are seated at banquet tables throughout the performance space. By tapping on a lighted dome at the center of the table when prompted by the conductor, the audience members trigger musical sequences in the composition. Coordination is managed through a central software controller. Like all good centerpieces, the domes are not tethered in any way, easily moved, and unobtrusive when the performance is not in progress.

Keywords

New Interfaces for Musical Expression, ubiquitous computing, interaction design, embedded networking, wireless sensor networks.

INTRODUCTION

New York's Eos Orchestra works to test the limits of orchestral performance, through a regular repertoire of experimental works, and through the commissioning of new works for orchestra. For the orchestra's 2003 benefit banquet, Artistic Director Jonathan Sheffer sought to conduct a performance in which the audience at the banquet performed the piece. Terry Riley's minimalist work "In C" was selected for this experiment, due to its significance as a groundbreaking work in non-traditional composition and its algorithmic applicability [1]. The specifics of the performance were kept at a minimum: the audience was to "play" the work, the result should be a musically acceptable performance of Riley's composition, and the audience members should have an engaging experience in playing the piece. All other details were left to the implementation team's discretion.

The design team members have extensive interest in and experience with performance augmented and mediated by digital technology. Eric and David are both central members of LEMUR (league of electronic musical urban robots), a group of musicians, robotics experts, artists and designers dedicated to creating musical robots. Both have a background in musical composition and entertainment system design, and an interest in electronically mediated musical performance. Tom's background is in theatre lighting and design, and is currently the area head for physical computing at the Interactive Telecommunications Program at NYU. His research focuses on the design and use of networked embedded systems.

The parameters that attracted us to the project were as follows: first, it involved a large number of participants, and a need for tight coordination and performance-level response times, drawing from and expanding on team members' prior experience in multi-participant interactive musical environments [2]. Second, as in any musical performance, the physical operation of the instruments must not take the performer's whole focus, so that the performers can focus on the music, not the operation of the instrument; a derivative of one of several of Perry Cook's observations and principles relating to musical controllers [3]. Any user feedback from the device would have to be in the periphery of the audience/performer's attention, not at the center of it. Third, though a significant amount of technology was involved, it would have to be transparent. The interfaces would have to appear aesthetically to be part of the decor, not an alien piece of control technology. In this sense, it let us put into practice many ideas from Weiser & Brown's manifesto of ubiquitous computing [4], specifically their notions of the synergistic effects of networks of microcontrollers and of technology at the periphery of an experience rather than the center. Though heavily reliant on computing and networking power, this project would center on shared musical environment rather than a computing environment.

SYSTEM DESCRIPTION

The performance was held in a banquet hall at which Eos Orchestra would be performing for invited supporters. The tables in the hall were arranged according to an instrument layout for an orchestra, with each table representing one instrument. Each of the thirty tables at the banquet featured an interactive centerpiece consisting of a 12” diameter translucent acrylic dome approximately 8” tall. Electronics housed within the dome allowed the guests to tap the dome and trigger a phrase of music, as long as they had received a cue from the conductor. In a manner similar to MIT Media Lab’s “Tribble”s [5], LED lights within the dome were used to signal status to the guest players. Three states were communicated: Inactive, Enabled (cued by the conductor), and Playing. Domes were “served” to the tables by the technical staff to replace the banquet centerpieces after dinner was over, so no wires could be permitted for power or communications. User input sensors in the domes communicated via a microcontroller back to a master computer, sending data on how hard the dome was tapped. The master computer used this data to trigger a musical phrase played by that table’s instrument with a MIDI velocity relative to the force applied to the dome.

Two Akai samplers with 16 audio outputs each were fed the MIDI generated by a master computer running a Max patch and output the 30 player parts and a click track. Audio of the triggered chunk would be fed to the main mix and routed wirelessly to the table that triggered it and fed to a local, powered monitor. A click track would also be output from the master computer and mixed to the main audio feed.

In the final implementation, the wireless audio return path was not implemented. Instead, clusters of local speakers were positioned overhead around the room. Three tables were served by each cluster of speakers. Audio for each of the tables served by a local cluster was routed to the cluster, so that the sound appeared to be coming from the table itself.

SOFTWARE

Interaction, playback and communication software was implemented on Macintosh computers in Cycling 74’s Max software. A master Max patch was developed for this application with the following features:

- graphical user interface for operator control of performance parameters
- bidirectional UDP communication with pods
- interactive player subpatch for each pod
- algorithms for MIDI playback, interaction arbitration and

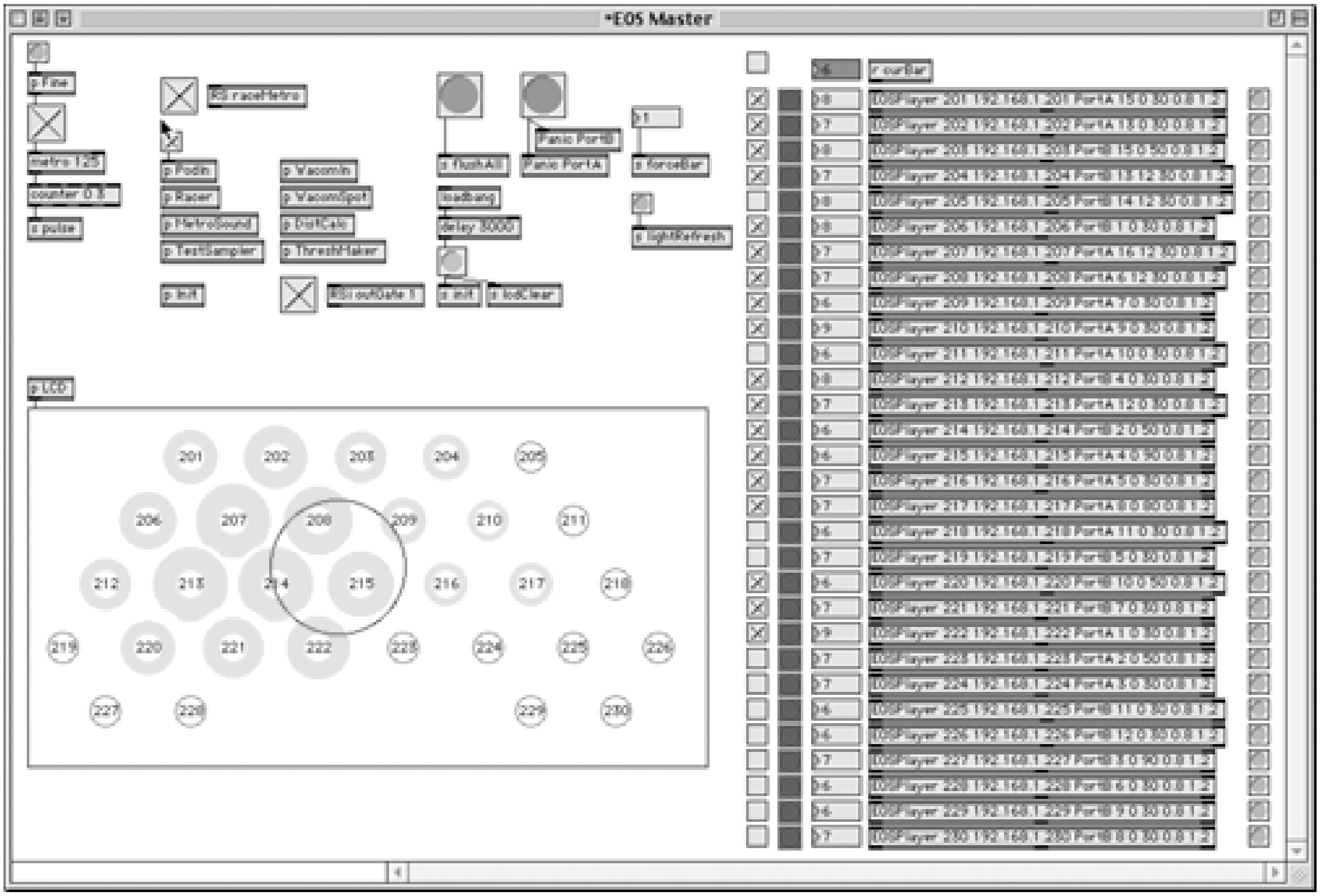
*EOS Master
p Fine
metro 125
counter 0 8
s pulse
RSI raceMetro
p Podin
p Racer
p MetroSound
p TestSampler
p init
p WacomIn
p WacomSpot
p DistCalc
p ThreshMaker
RSI outGate 1
s flushAll
Panic PortB
Panic PortA
loadbang
delay 3000
s init
s lodClear
1
s forceBar
s lightRefresh
p LCD
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
16
r curBar
UDPlayer 201 192.168.1.201 PortA 15 0 50 0.8 1 2
UDPlayer 202 192.168.1.202 PortA 13 0 50 0.8 1 2
UDPlayer 203 192.168.1.203 PortB 15 0 50 0.8 1 2
UDPlayer 204 192.168.1.204 PortB 13 12 50 0.8 1 2
UDPlayer 205 192.168.1.205 PortB 14 12 50 0.8 1 2
UDPlayer 206 192.168.1.206 PortB 1 0 50 0.8 1 2
UDPlayer 207 192.168.1.207 PortA 16 12 50 0.8 1 2
UDPlayer 208 192.168.1.208 PortA 6 12 50 0.8 1 2
UDPlayer 209 192.168.1.209 PortA 7 0 50 0.8 1 2
UDPlayer 210 192.168.1.210 PortA 9 0 50 0.8 1 2
UDPlayer 211 192.168.1.211 PortA 10 0 50 0.8 1 2
UDPlayer 212 192.168.1.212 PortB 4 0 50 0.8 1 2
UDPlayer 213 192.168.1.213 PortA 12 0 50 0.8 1 2
UDPlayer 214 192.168.1.214 PortB 2 0 50 0.8 1 2
UDPlayer 215 192.168.1.215 PortA 4 0 50 0.8 1 2
UDPlayer 216 192.168.1.216 PortA 5 0 50 0.8 1 2
UDPlayer 217 192.168.1.217 PortA 8 0 50 0.8 1 2
UDPlayer 218 192.168.1.218 PortA 11 0 50 0.8 1 2
UDPlayer 219 192.168.1.219 PortB 5 0 50 0.8 1 2
UDPlayer 220 192.168.1.220 PortB 10 0 50 0.8 1 2
UDPlayer 221 192.168.1.221 PortB 7 0 50 0.8 1 2
UDPlayer 222 192.168.1.222 PortA 1 0 50 0.8 1 2
UDPlayer 223 192.168.1.223 PortA 2 0 50 0.8 1 2
UDPlayer 224 192.168.1.224 PortA 3 0 50 0.8 1 2
UDPlayer 225 192.168.1.225 PortB 11 0 50 0.8 1 2
UDPlayer 226 192.168.1.226 PortB 12 0 50 0.8 1 2
UDPlayer 227 192.168.1.227 PortB 3 0 90 0.8 1 2
UDPlayer 228 192.168.1.228 PortB 6 0 50 0.8 1 2
UDPlayer 229 192.168.1.229 PortB 9 0 50 0.8 1 2
UDPlayer 230 192.168.1.230 PortB 8 0 50 0.8 1 2

Figure 1. Operator interface in Max

progression through the piece

- control of pod LED's for user feedback

The piece "In C" is composed of 53 individual melodic patterns of varying length to be played in sequence. In the performance instructions, the composer states that each performer must progress through the patterns in order but may make entrances and repeat each pattern numerous times at his or her choosing.

Our patch was designed to simulate these instructions, with entrances determined by the players and progression through the patterns controlled by the patch. A "horse race" algorithm was implemented with a counter progressing through pattern numbers 1 through 53 and each pod racing towards or past this master number but remaining within +/- 4 of its current value. This was done by periodically incrementing each pod's pattern number, with the odds of incrementing being greater if further behind and less if further ahead of the master number.

Players were able to trigger playback of a sequence by tapping on the pod dome. Tap signals were sent via UDP to the host with a message indicating the velocity of the tap and the pod number. If the pod was enabled, a tap triggered playback of the pod's current pattern, with playback volume scaled by both the tap velocity and the pod's level as set in the operator's interface.

Playback of all parts was synchronized to a master eighth-note clock running at a fixed tempo. Entrances occurred on the next eighth note following receipt of a trigger, playing the triggering pod's current pattern synchronized to the master clock.

Players for each pod were implemented as a parameterized subpatch. Parameters to the subpatch allowed setting of IP number, MIDI port and channel for output voice, trigger threshold and voice scaling. The subpatch handled pod communication, triggering arbitration and synchronization, MIDI playback and scaling, LED feedback and UI feedback for each pod.

To give users feedback about playback, the host controlled RGB LED's inside each pod. A red color indicated that the pod was currently disabled, and green indicated enabled. Tapping the pod caused a white flash (all RGB colors on). During pattern playback, blue LED's were flashed on every eighth note beat and every note in the pattern; this was done to ensure ample player feedback, even during rests.

An operator's interface was implemented using the *lcd* object. The interface presented an analog of the layout of the banquet room, with each table represented as a circle with its corresponding pod/table number inside. A thin red circle indicated a disabled pod; a green circle indicated an enabled pod, with the circle's line thickness representing relative volume level. When a pod was triggered, a blue circle within the green flashed on and off to indicate active playback.

The interface enabled the Max operator to follow the conductor's movement among the tables and assist the conductor in controlling the performance. Using a pen and graphics tablet, the operator could move a circular marquee around the interface, with pen pressure controlling the diameter of the marquee and thus the area of influence. The operator could enable, disable and scale volume of each of the pods using keystrokes, with the keystrokes' effects applying only to the pods enclosed by or touched by the marquee.

Bidirectional network communication was accomplished using the *otudp* object [6] to send UDP messages wirelessly between the Max host computer and the pods. Outgoing messages from the host were addressed to each pod on a local IP address and port, with the last byte of the IP number corresponding to the pod's table number. Messages from the pods to the host were sent on a broadcast IP address, enabling both the host and a backup host computer to receive them.

HARDWARE

Hardware for the pods was implemented on a PIC microcontroller, using NetMedia's Siteplayer web coprocessor to manage TCP and UDP connectivity. The entire module was connected wirelessly using an Ethernet-to-WiFi bridge from D-Link. The pods were powered by 12-volt rechargeable motorcycle batteries.

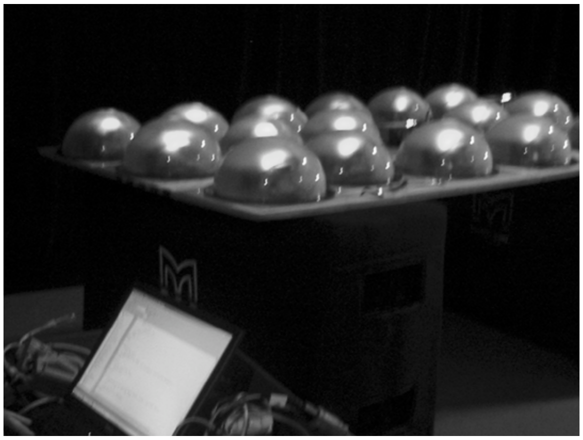

Figure 3. The pods in testing

Sensing was accomplished using a set of four force-sensitive resistors under the rim of the dome. Velocity for the dome was taken as the highest sensor reading on any of the four sensors, provided the value was above a given threshold of sensitivity. Each dome's sensitivity threshold could be set either by sending it via UDP, or by using an HTML forms-based interface on the Siteplayer.

RGB LED output was controlled by the PIC based on UDP messages received from the master control software. The white LED flash on each tap was generated locally by the PIC. All other LED control was received from the master control software. The Siteplayer HTML interface could also be used to set LED intensity levels, and was used during installation for diagnostic purposes, when the master control software was not online.

Message protocol between the pod units and the master control software was kept as light as possible to minimize traffic. Two bytes were sent from each pod to the master control software on each tap: the last octet of the pod's address, and the velocity value. Three bytes were sent from the master control software to each pod: red, green, and blue intensity values for the LED's. Optional fourth and fifth bytes could be sent to adjust the sensitivity of the pod's FSRs, but this was used only during installation.

Because the pods were to be laid out in a specific order in the performance space, fixed IP addresses were used for each pod, to simplify contact with them during installation. Likewise, the wireless bridges were also given fixed addresses in advance. Although it was not strictly necessary to associate IP addresses with instrument numbers, it was convenient, given the time scale of the project.

IMPLEMENTATION FOR UBICOMP 2003

For UbiComp 2003, our implementation of the Eos Pods will be somewhat less extensive than for the initial performance. Between 6 and 12 pods will be used, and only one sampler or synthesizer will be used. Audio will not be routed to local clusters, but will be designed around a central PA system feeding the entire space of the demonstration.

CONCLUSION

The initial performance of the Eos pods at the orchestra's annual supporter's banquet was a successful test run for the system. All technical components performed as specified, with remarkably few technical problems. Communication between conductor, support staff, and audience was somewhat confused as instructions were rushed. For example, the pods were designed to trigger once a player's hand lifted from the pod; not all audience members understood this, and some kept their hand on the pod constantly, wondering why it didn't play. In future performances, we will look for a better solution, allowing for a wider range of playing styles. Overall, however, the audience had an enjoyable experience, and all parties were pleased with the result. Eos has hopes for future and more ambitious performances with the pod system, and we look forward to further refinement of the system as the design team and the orchestra gain more experience with it.

ACKNOWLEDGMENTS

We thank Jonathan Sheffer and the Eos Orchestra for their support and for their openness to experimentation, without which this project would not have been possible. We also thank the staff and students of the Interactive Telecommunications Program at ITP, many of whom were drafted into service in the fabrication of the system; and the staff of Audio, Video, & Controls, whose expertise and enthusiasm made for a more pleasant working experience throughout the process.

REFERENCES

[1] T. Riley, "In C", musical score and performance instructions (1964)

[2] R. Ulyate and D. Bianciardi, "The Interactive Dance Club: Avoiding Chaos in a Multi-Participant Environment" Computer Music Journal , Volume 26, Number 3, MIT Press (2002)

[3] P. Cook, "Principles for Designing Computer Music Controllers," ACM CHI Workshop in New Interfaces for Musical Expression (NIME), Seattle, April, 2001.

[4] M. Weiser and J.S. Brown. "The Coming Age of Calm Technology",http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm (October 1996).

[5] J.A. Paradiso, "Dual-Use Technologies for Electronic Music Controllers: A Personal Perspective," Proceedings of the 2003 Conference on New Instruments for Musical Expression (NIME-03), Montreal, Canada

[6] M. Wright, "otudp 2.4", CNMAT, UC Berkley. http://cnmat.cnmat.berkeley.edu/OpenSoundControl/clients/max-objs.html