Building Collaborative Graphical interFaces in the Audicle

Ge Wang
Department of Computer Science,
Princeton University
35 Olden St.
Princeton NJ 08540 USA
gewang@cs.princeton.edu

Ananya Misra
Department of Computer Science,
Princeton University
35 Olden St.
Princeton NJ 08540 USA
amisra@cs.princeton.edu

Perry R. Cook
Department of Computer Science
(also Music),
Princeton University
35 Olden St.
Princeton NJ 08540 USA
prc@cs.princeton.edu

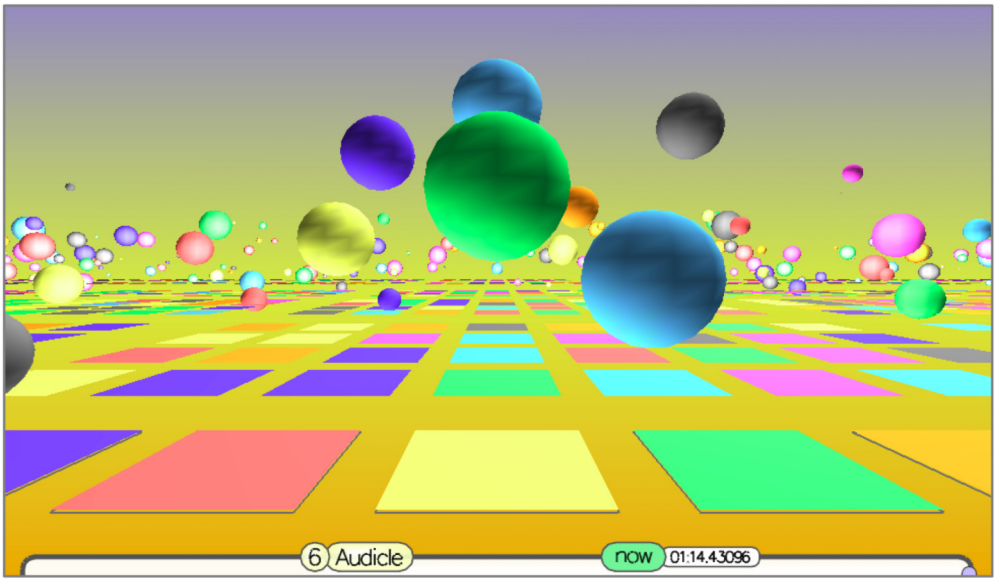

Figure 0. Multiple Bouncing Spheres interfaces visualized from a centralized viewpoint. Each human player manipulates spheres over a portion of the squares. The ensemble is synchronized by computer.

ABSTRACT

Emergence is the formation of complex patterns from simpler rules or systems. This paper motivates and describes new graphical interfaces for controlling sound designed for strongly-timed, collaborative computer music ensembles. While the interfaces are themselves minimal and often limiting, the overall collaboration can produce results novel beyond the simple sum of the components – leveraging the very uniqueness of an ensemble: its strength in numbers. The interfaces are human-controlled and machine-synchronized across a dozen or more computers. Group control, as well as sound synthesis mapping at each endpoint, can be programmed quickly and even on-the-fly, providing a second channel of real-time control. We show examples of these interfaces as interchangeable plug-ins for the Audicle environment, and also document how they are used in a laptop ensemble.

Keywords

Graphical interfaces, collaborative performance, networking, computer music ensemble, emergence, visualization, education.

1. INTRODUCTION

Emergence is the formation of complex patterns from simpler rules or systems. In this paper, we explore minimal, easy-to-learn graphical interfaces that can, as a group, form sound and music that is more interesting and complex than that made by any single component, in a tightly coupled and collaborative environment.

This was motivated by the desire to provide new interfaces for new computer music performance ensembles such as PLOrk: Princeton Laptop Orchestra, currently being developed and instructed by Dan Trueman, Perry Cook, Scott Smallwood, and Ge Wang. In addition to more self-contained, sophisticated instruments, we wished to provide the ensemble with interfaces that require minimal setup and learning time, and with which the participants can immediately and directly influence the overall sound as part of the group. Furthermore, we wanted the option of tightly synchronizing all participating interfaces / machines.

Given this motivation, the research goals are defined as follows.

The interfaces should be:

- simple enough to pick up and use, yet complex enough to generate interesting music/sound as a group
- amenable to collaboration in a tightly-timed setting; for example, a server should be able to synchronize all interfaces with desired musical timing; collaboration is the unifying aspect of all the interfaces we present.
- as direct and as immediate as possible
- as precise as possible, even at the cost of resolution
- easily programmable (i.e. mapped to sound/graphics)

To implement the interfaces, we used the Audicle programming environment [6,7] as the platform, leveraging its existing framework for blending high-performance graphics with precise real-time audio. Extending the Audicle API (in C++), we were able to add and experiment with new graphical interfaces as new faces of the Audicle. We also added a mechanism for accessing and controlling the interfaces directly using ChucK [5], the language for which Audicle was built. In this way, we can write ChucK code to control sound synthesis using data from the graphical interfaces, decoupling the interface from the sound synthesis. Furthermore, ChucK makes it possible to change sound synthesis and interface mapping on-the-fly.

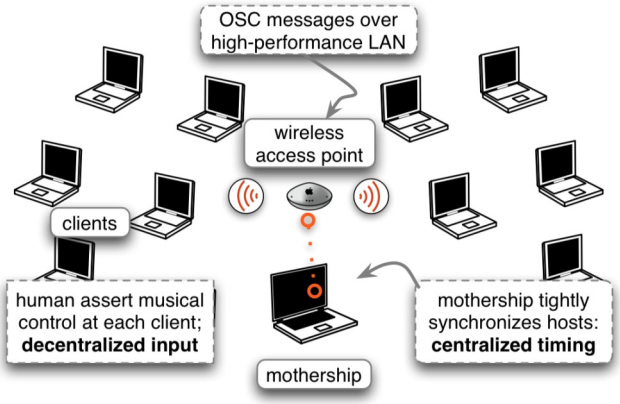

Figure 1. A collaborative interface network model.

The interfaces are synchronized over a wireless local-area network (Figure 1), using Open Sound Control [9]. One or more *mothership* host runs any application that broadcasts messages and synchronization messages to each of the end points. Our current mothership programs are written in ChucK.

2. RELATED WORK

The various graphical interfaces developed in this ongoing investigation derived mainly from three areas: audio/visual interfaces (Figure 2), such as the Painterly Interfaces created by Golan Levin [3] and musical video games created by Toshio Iwai [2,10]; GUI-based frameworks such as the MIDIGrid [1] and ixi software [4]; mainstream puzzle games, such as Chu-chu rocket, Lemmings, and Domino Rally [11,12,13].

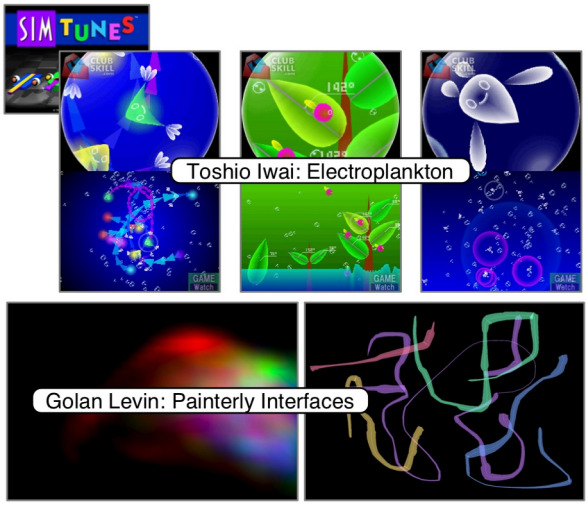

Figure 2. Audiovisual interfaces.

All of these works have significantly influenced and inspired interfaces in this paper – directly (such as in the case of the MIDIGrid) or aesthestically. The contribution of our work is placing interfaces similar to these within a tightly-timed synchronization framework, and finding paradigms to leverage this new collaborative aspect.

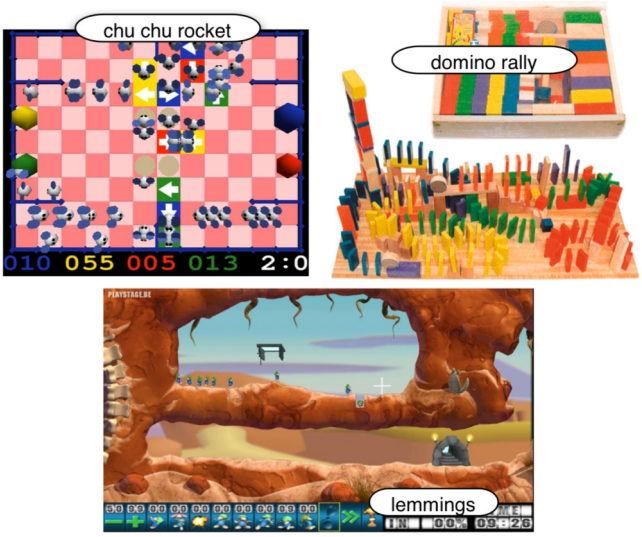

Figure 3. Puzzle games.

3. INTERFACES

Some example interfaces designed for single-server multiple-client setups are described below. As shown in Figure 1, humans assert musical control at each client, while the server centrally synchronizes all the clients.

3.1 Color Squares

In this interface (Figure 4), each client has an N x M grid and a finite color palette. Every color can be mapped to a separate event or to silence, with a potentially different mapping for each client. The user at every client machine selects a color using the mouse or the keys 1-9 and a-z (0 and es are reserved for the silent color), and applies it to any number of squares on the grid.

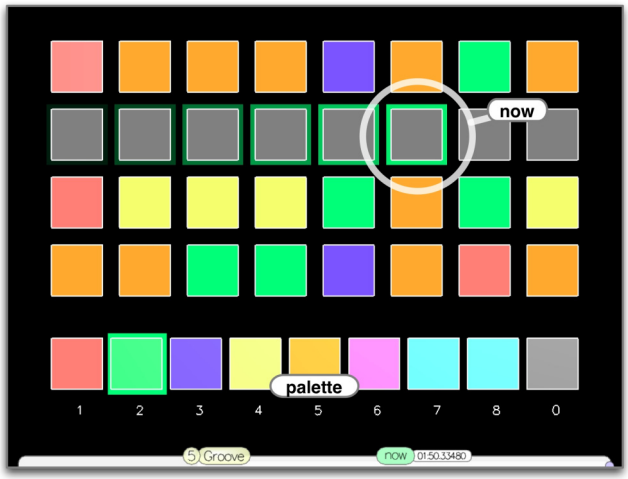

Figure 4. Color Squares

The server program moves through the grid in any desired manner, including sequential, random, or in any other fashion, in any timing pattern as specified in code in the server ChucK program. The client is aware of when a particular grid element is activated and what color was last placed on it, and can respond to this information in any programmatic manner, such as by playing a sound associated with the color of the currently activated grid element. Sample ChucK code for the server and client sides are presented here.

3.2 Spheres

Spheres (Figure 0) is a three-dimensional interface extending the color squares metaphor. It consists of spheres that bounce from a height, with or without energy loss, and colored square covering the "ground". Each time a sphere hits the ground, it can trigger an event based on the color and mapping of the square it touches. Multiple views of the space allow the user to control where a sphere bounces as well as its starting height. The bouncing location (or square) controls which event the sphere triggers, while the height translates to how often it bounces and thus how often the triggered event is repeated.

3.3 Mouse Mania

The Mouse Mania interface draws from the Chu Chu Rocket game described earlier. Each client or host has a virtual entity or "mouse" and a grid that acts as a map for the mouse to follow. Each grid element can have a user specified color and shape. As in the Color Squares interface, the color of a grid element can be mapped to an event that is triggered when a mouse reaches it. In addition, the grid element's shape can control the subsequent movement of the mouse, including its direction, speed, or special dance moves possibly associated with repetition of the related musical event. A mouse need not be confined to a single host; another option is for the server to own many mice that run from host to host, changing the spatial dynamics of the piece.

3.4 Dominotes

This interface uses the visual metaphor of dominoes to control events being triggered in rapid succession. Each user constructs domino rallies and connects sub-rallies to a central launching station when they are ready to be played. The launching station, at the discretion of the server, pushes all adjacent or connected dominoes, triggering a chain reaction. Each domino toppling, as well as special items such as rockets in the dominoes' path, can be mapped to any parameters or events at the discretion of the client. Toppled dominoes can be made upright automatically or manually by the users' selecting any subset of their dominoes. Forks in a domino rally allow each client's musical sequence to follow multiple paths in parallel.

3.5 SaladTossers

This interface is based on the idea of musical "recipes" and consists of salad ingredients, dressing, and a mixing bowl for each client. Ingredients can map to musical events as specified by the client. The user creates a salad by inserting desired quantities of each ingredient into the mixing bowl and tossing it. The tossing causes events to be triggered repeatedly; events associated with ingredients that make up a larger portion of the salad are triggered more often and thus have greater density in the resulting sound. As more ingredients are added to the salad, events are triggered more often. Further, a variety of dressings are available for adding to the mix, each dressing being associated with a different filter or sound processing effect. Finally, there is a "consume" option which gradually empties out the contents of the bowl and thus reduces the overall event density until there is silence. This interface is expected to be especially useful for creating textures where one may prefer to closely control the density of events rather than specifying the exact times at which events are triggered.

3.6 More

The above are some examples of simple interfaces that can produce complex music over a network of collaborators. It is possible to program more such graphical interfaces using the open-source Audicle framework. In addition, the mapping suggestions and time-based behavior described above are optional for each graphical interface and can be easily modified by changing the ChucK code on the client and server sides. Thus, these interfaces are flexible on the visual and auditory levels as well as in the interactions between the two.

4. CASE STUDIES

The Color Squares interface was used in the debut concert of PLOrk: Princeton Laptop Orchestra in a piece called “Non-Specific Gamelan Taiko Fusion Band”. The setup involved one conductor, one mothership laptop, one inkjet printer (from which scores were printed during the performance), and 15 laptop stations, each equipped with powered hemispherical speakers and running Color Squares. The stations were divided into four groups, each with a different sound synthesis mapping.

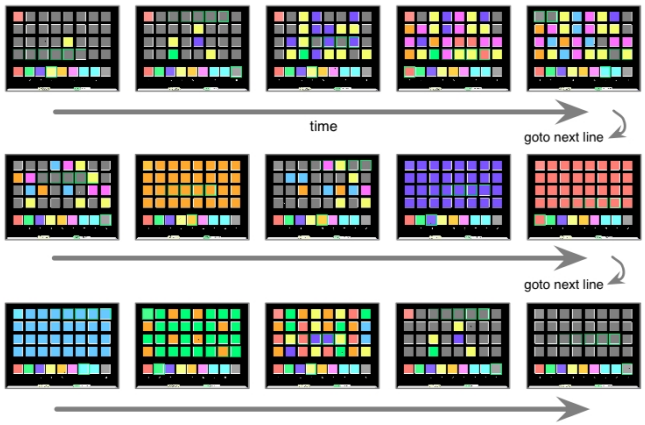

Figure 5. Example score over time.

During the performance (Figure 6), the conductor held up colored print-outs specifying differing densities and timbres (colors) to different groups over time. The score (Figure 5) shows some typical textures and transitions. The study successful demonstrated that the approach fulfilled its major goals. The students learned the interface in seconds and were able to immediately affect change in controllable ways. The resulting performance can be heard at:

http://plork.cs.princeton.edu/listen/debut/nsgamelan.mp3

Figure 6. PLOrk in action.

5. CONCLUSION AND FUTURE WORK

We have demonstrated a variety of graphical interfaces for creating music collaboratively. The simplicity of these interfaces allows new users to grasp the rules quickly and start making music right away. At the same time, the strong central synchronization facilitates collaboration, giving rise to more complex pieces than would be expected from the basic rules for the clients or server. Thus, these interfaces produce a form of *emergent music*. As our investigating continues, we hope to expand on this beginning exploration into collaborative graphical interfaces.

6. ACKNOWLEDGMENTS

We would like to thank Dan Trueman, Scott Smallwood, all the member of PLOrk, Philip Davidson, and Joshua Podolak for their invaluable help and ideas.

7. REFERENCES

[1] Hunt, A. and R. Kirk. "MidiGrid: Past, Present, and Future" *In Proceedings of the International Conferences on New Interfaces for Musical Expression*. May 2003.

[2] Iwai, T. "Images, Music, and Interactivity - the Trace of Media Art" *Keynote Speech. International Computer Music Conference*. June 2004.

[3] Levin, G. *Painterly Interfaces for Audiovisual Performance.* M.S. Thesis, MIT Media Laboratory, August 2000.

[4] Magnusson, T. "ixi software: The Interface as Instrument" *In Proceedings of International Conference on New Interfaces for Musical Expression*. June 2005.

[5] Misra, A., G. Wang, and P. R. Cook. "SndTools: Real-time Audio DSP and 3D Visualization" *In Proceedings of the 2005 International Computer Music Conference*. September 2005.

[6] Wang, G. and P. R. Cook. "The Audicle: A Context-Sensitive, On-the-fly Audio Programming Environ/mentality" *In Proceedings of the International Computer Music Conference*. Novermber 2004.

[7] Wang, G., A. Misra, P. Davidson, and P. R. Cook. "Co-Audicle: A Collaborative Audio Programming Space" *In Proceedings of the International Computer Music Conference*. September 2005.

[8] Wang, G. and P. R. Cook. "ChucK: A Concurrent and On-the-fly Audio Programming Language" *In Proceedings of the International Computer Music Conference*. October 2003.

[9] Wright, M. and A. Freed. "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers" *In Proceedings of the International Computer Music Conference*. September 1997.

[10] http://electroplankton.nintendods.com/

[11] http://www.pressmantoy.com/

[12] http://www.vintage-sierra.com/puzzle/tim.html

[13] http://en.wikipedia.org/wiki/ChuChu_Rocket