ALSKDJALSKDJALSKDJ:

documenting the process involved in writing a timing-accurate networked piece for a laptop ensemble

Konrad Kaczmarek

Princeton University
Graduate Student, Department of
Music

ABSTRACT

The piece, *alskdjalskdjalskdj*, was composed during a three-year period of experimentation using a networked laptop ensemble to create flexible conducting systems and timing-accurate software instruments. During this time the piece existed in several distinct states, each of which reflected the compositional, programming, and creative concerns at that particular time. By examining the evolution of two aspects of the piece, the shared drawing environment and the pulse-based software instrument, this paper charts the process of the piece through these various stages, highlighting the different approaches to programming and the resulting changes in compositional design.

1. BACKGROUND

My first experiments involved generating real-time graphical scores for a group of improvising acoustic performers as part of a graduate seminar on improvisation and alternative performance spaces. In the final performance, musicians were spread out over various indoor and outdoor spaces, and ran an application on their laptops that gave them instructions on how to play in real time. These instructions came in the form of various shapes and colors drawn on their screens that were controlled by a conductor over the wireless network.

The next stage of the piece incorporated a time-accurate software instrument for generating the sounds, and was created for an undergraduate course in computer and electronic music through programming, performance, and composition at Princeton University [2]. The conducted, graphical element of this version of the piece changed to reflect the compositional issues that emerged with the addition of the new instrument. This version of the piece was also shaped by experimentation and user feedback that resulted from weekly workshop sessions with the group throughout the semester [4].

After having performed the piece several times with a professional group, Sideband, I developed a version that brings the graphical elements back into the piece in a meaningful and intuitive way. In this version, various shapes that represent sonic events move around the screen, causing notes to sound as they pass each performer's center-screen mark. The vertical position of the shape determines the pitch, and users can knock down barriers between adjacent screens, allowing their sounds to spatially migrate throughout the ensemble. While the software instrument is conceptually based on the timing system of the previous version of the piece, it uses a completely different approach to programming in order to realize it.

2. THE SHARED DRAWING ENVIRONMENT

The first version of the piece established the framework for the multi-user shared drawing environment that would ultimately be used in all subsequent versions. This environment worked by creating duplicate OpenGL drawing scenes on all of the connected computers, but only rendered a player-specific portion of the global space in each client instance. In the first version of the piece, the conductor used a host application to create and manipulate various objects in the global space, which had clearly marked zones designated for each performer (Fig. 1).

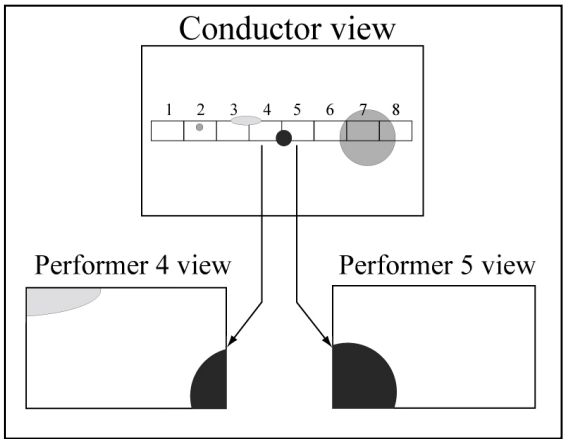

Figure 1. Example of the shared drawing environment.

The different objects represented pre-determined musical instructions, and functioned as individual elements of a larger graphical score. The performers ran a client application that revealed only their designated area of the global space. As the conductor manipulated and moved various objects in and out of the performers' individual

views, he could effectively pass musical gesture around the ensemble.

As the particular state of an object, such as its shape type, size, color, rotational speed, and screen position provided the graphical cues to the performers, it was important that the conductor be able to easily control several of the variables at the same time in order to create more complicated gestures. To achieve this, the conductor used a Wacom tablet as a multi-dimensional real-time input device and premade automation data. By holding down one of the number keys to select a specific object, the conductor could control the object's size, position, and color using the tablet's pen pressure, x/y position, and pen tilt values. The gestures that were generated could be recorded and played back at any point in the performance, or saved to use in subsequent performances.

As the number of parameters increased to reflect the greater degree of control over the desired conducted gesture, I began experimenting with incorporating a simple physics model that could control the objects in a more autonomous way. In this new environment, the conductor had control over parameters such as an object's velocity, coefficient of drag, and mass, and could set the overall system's gravity and control of collisions. The drawing aspect of the program was thus ported from Max/MSP/Jitter to Java, as that platform's text-based programming environment and its direct communication with OpenGL was better suited for these types of calculations. The physics model ran on the conductor's server application, which sent out the relevant drawing commands to the rest of the ensemble. At any point in the performance, the conductor was still able to manually control any individual object, overriding whatever physics model was currently active.

The addition of the networked software instrument in the next stage of the piece greatly altered the role of the shared drawing environment. The musical parameters that the drawn objects communicated to the group of improvisers were either no longer applicable to the new instrument, or could be controlled directly by the conductor over the network. As a result, the drawing environment was rewritten to communicate performance instructions more explicitly in the form of text that was drawn on the performer's screens. The graphical aspect of the drawing environment was limited to the ability to change the color and brightness of individual performer's screens. While the framework of the shared drawing environment was kept in place to render the text-based instructions and differentiate between individual performers, it was no longer used to draw individual objects and therefore no longer functioned as a graphical score. Instead, as the piece was ultimately performed in very low light, the glow that reflected back onto the performer's faces as their screens changed color provided a dramatic visual component to the performance (Fig. 2).

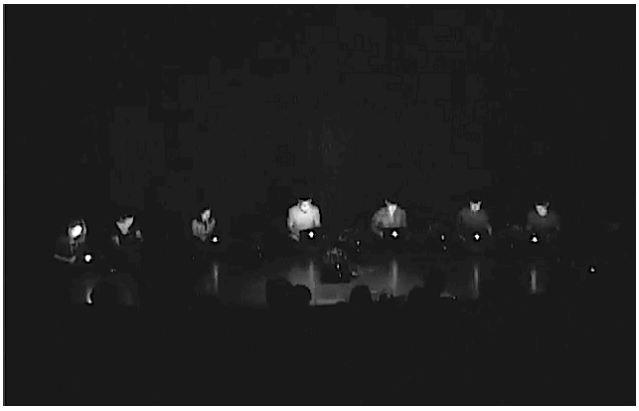

Figure 2. A performance of *alskdjalskdj* in low light.

The most recent version of the piece brought together the underlying shared drawing environment, the physics-based object models, and the software instrument in a meaningful and musically expressive way. The underlying java code was once again rewritten so that drawn objects would now trigger actual sonic events, and could be created and manipulated not only by the conductor, but by the performer themselves. Performers also had control over the boundary functions, allowing the objects that they created to move beyond their own screens and generate sounds on other performer's computers.

The first two versions of the piece used a centralized server/client architecture to control the shared drawing context. This method was well suited for the unidirectional mode of communication that existed between the conductor and performers, as the conductor generated all of the control data, the client application simply received this information over the network and translated it into the appropriate drawing commands. The initial physics model also functioned in this capacity, with the conductor machine running the model and then broadcasting the resulting data to the rest of the ensemble over the network. The final version of the piece, however, used a more distributed control of the drawing context, which effectively turned each performer into a conductor. When a performer created an object on their laptop, the object was also created in the global scene that was running on all of the laptops connected in the network. As a result, each performer application calculated the physics model as well as generating the resulting drawn graphics. The conductor and performers thus ran the exact same underlying code, but used different interfaces to determine what parameters they had control over.

3. THE INSTRUMENT

I wanted to establish a rhythmic language for this piece that could quickly alternate between precise ensemble playing and a more diffused or indeterminate hocketed sound. To achieve this, I created a software instrument that generates looping rhythmic patterns using sampled instruments triggered by performers keystrokes. When a key is pressed, the instrument initiates a repeating pulse of notes that correspond to that particular keystroke. The performer can create complex ostinato patterns by initiating multiple pulses of varying frequency and scale degree. The performer sets the rate of each pulse relative to a conductor-determined base tempo, and then decides when to trigger the pulse within the overall looping texture, effectively determining the phase. Individual pulses can then be manually resynced or turned off, either by the performers themselves or by the conductor. By sending networked controller data routed directly to the software instrument, the conductor also controls the overall tempo, the type of pitch mapping, the key, and the level of the audio effects.

Due to the wide range of musical backgrounds in the group of undergraduates and the improvisatory nature of the piece, I decided not to describe the ostinato patterns using traditional musical notation. Instead, performers received instructions on how to construct their patterns from the conductor in real-time. These instructions came in the form of a "number of voices" parameter that was displayed on their screens and dictated how many note pulses should be sounding at a given time, and other text-based performance instructions such as "listen for a gap in the overall ensemble sound, and try to place a new pulse there". The changing color of their screens indicated what sample bank they should be playing. The sample banks included prepared piano, acoustic guitar, hammer dulcimer, vibraphone, glockenspiel, and an electronic percussion set.

In this stage of the piece, controlling the pulses of notes remotely effectively meant simulating a performer's keystroke. The conductor could turn on or off any individual pulse by sending the corresponding note on, off, or sync keystrokes to a player in the ensemble. This facilitated a form of improvisation in which the conductor was able to set the entire ensemble to a uniform rhythmic pattern, for example by syncing all of their pulses at the same time, and then give them instructions on how to deviate from it either by re-syncing their pulses or by adding new ones. This achieved the desired range of rhythmic language in this piece, and also allowed individual performers to shape the sound of the ensemble in an intuitive way, regardless of their musical background.

After receiving feedback from the ensemble about their desire to be able to shape the sound of the group as a whole in other ways, I added in the ability to save, recall, and retrigger entire ostinato patterns, as well as the ability to share them with each other over the network. With a single keystroke, a performer was now able to retrigger the exact sequence of pulses they had entered, or bring back a multi-pulse sequence they had played earlier in the performance. A graphical interface indicated when a player had shared a pattern, and performers could then choose to adopt that pattern, playing it back unchanged or altering it in various ways. Experimentation with the ensemble also led to implementing a sub-grouping function, which enabled the conductor to send messages or control data to any subset of the ensemble.

The most recent version of the piece brought together the physics-based drawing model and the software instrument, establishing a meaningful correlation between the visual and sonic components. In this version, the repeating pulses of notes were represented graphically by moving objects bounded within a certain area (Fig. 3). The individual object's velocity, vertical height on the screen, and radius determined that pulse's rate, scale degree, and octave. As the objects crossed the center of the screen, they generated note-on messages, which Max routed to the software sampler. By giving the objects different y velocities, the performers were now able to create simple melodies with their pulses in a way that was not possible with the pervious system. Similarly, as the drawing context was exactly mirrored in each instance, the performers could remove the boundaries between adjacent screens and allow their pulses to migrate around the ensemble. This gave them another way to interact with the ensemble as a whole, and to sculpt their sound spatially in a way that was not possible with the previous version.

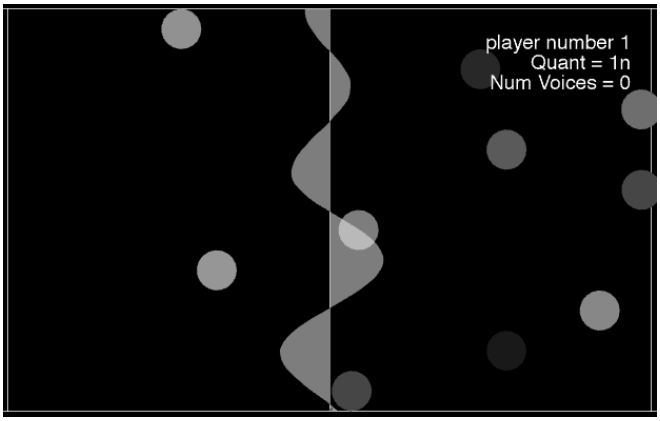

Figure 3. A snapshot of the graphical representation of an ostinato pattern.

Whereas the previous version of the software instrument used a timing system driven by Max/MSP's own scheduler, the note-on messages of the physics-based model were generated within Java using a separate scheduler thread, ensuring that the timing of the note pulses were not affected by the rendering frame-rate.

4. CONCLUSIONS

As with most pieces written for the laptop ensemble, the process of creating *alskjdlaksjdlkajsd* involved aspects of instrument design as well as more conventional compositional concerns [2, 3]. With the addition of network connectivity, this process also incorporated establishing novel means of communication, both on a technological and a musical level. These factors interacted with each other in a complex system of mutual influence and feedback, as a particular compositional idea was inevitably altered by the technology used to realize it. Feedback from performers also played an important role in the development process, providing valuable insight into the effectiveness of the various types of communication.

The changing role of the graphics during the piece's development represented one clear manifestation of the mutual influence and feedback that existed between compositional approach, instrument design, and the various modes of communication. The shared drawing environment was initially conceived of as a way for a conductor to orchestrate an improvising ensemble in real time by functioning as an ensemble-wide graphical score. New modes of communication within the ensemble that were established with the addition of the networked instrument in turn shifted the role of the graphics to a more practical means of delivering performance instructions. Finally, by attaching the physics-based drawing model to the pulse-based software instrument, the role of the graphics changed once again towards a more representative and performative type of functionality.

One important theme that emerged during the development of this piece was the changing role of the conductor, which was manifest in a gradual shift in control away from the conductor towards the individual performer. Initially, the conductor used the technology to control the drawn environment, building on the conceptual model of gesture-based scored improvisation that was developed by musicians like John Zorn, Butch Morris, and Frank Zappa in the later part of the twentieth century [1]. As the programming evolved to incorporate autonomous control of the graphical elements, various aspects of control were effectively taken away from the conductor, freeing him or her to focus on other aspects of the performance. Similarly, the shifting ways in which the performers were able to communicate with each other directly over the network resulted in changes in the conductor/performer dynamic. These changes subsequently lead to more interesting ways for the group to perform, establishing a type of meta-instrument that encompassed the entire ensemble. Finally, by giving the performers control of their own set of drawn objects, the line between conductor and performer that was established in the first version of the piece became increasingly blurred [3].

Each step in the piece's evolution forced a fundamental shift both in the way the underlying code that was implemented and in the ways in which the conductor and the performers functioned within the piece. A flexible approach to programming and compositional design was therefore vital to creating a successful piece within this medium, as it allowed the piece to transcend any singular technological aspect or innovation. For this reason, *aslkdjalskjdalksjd* will undoubtedly continue to evolve and redefine itself in the future.

5. REFERENCES

[1] Brackett, John. "Some Notes on John Zorn's Cobra", in *American Music*, Vol. 28, no. 1 (2010): 44-75

[2] Lansky, Paul. "A View From the Bus: When Machines Make Music", in *Perspectives of New Music* vol. 28/2, (Summer 1990)

[3] Smallwood, Scott; Trueman, Dan; Wang, Ge; Cook, Perry. "Composing for Laptop Orchestra," *Computer Music Journal* Spring 2008, Vol. 32, No. 1: 9–25.

[4] Trueman, D. "Clapping Machine Music Variations: a composition for acoustic/laptop ensemble" in *Proceedings of the International Computer Music Conference (ICMC),* New York City, June 1–5, 2010.

[5] Trueman, D. "Why a Laptop Orchestra?" in *Organised Sound* 12:2, August 2007.