Different Strokes: a Prototype Software System for Laptop Performance and Improvisation

Mark Zadel and Gary Scavone
Music Technology Area
Schulich School of Music, McGill University
Montreal, Quebec, Canada
{zadel,gary}@music.mcgill.ca

ABSTRACT

This paper presents progress in the design of a new software interface for laptop performance and improvisation. These performances can lack a sense of active creation, as well as a visual connection between the performer's actions and the audio output. This stems partly from certain patterns in laptop performance in which musicians resort to heavy automation to cope with performing complex compositions. The software presented here attempts to address this by requiring that the user create all of the control sequences on-stage. The user defines graphical control patterns that are mapped to sample playback. The current prototype resembles a freehand drawing interface where the strokes create looping and cascading animations that generate corresponding audio, ultimately creating music. This style of interface minimizes the use of prepared material and takes advantage of the computer's unique capabilities.

Keywords

Software control of computer music, laptop performance, graphical interfaces, freehand input, dynamic simulation

1. INTRODUCTION

Laptop performance—musical performance on a standard computer system without novel controllers, usually by a solo artist—is an increasingly common mode of live computer music. The novelty of laptop performance is starting to fade, however, and it is sometimes criticized by audiences as being uninteresting. Laptop performances often foster little sense of the effort, difficulty, or activity audiences typically expect. A disconnect exists between the ostensible producer of the music and the music itself: there is no visible causal link apparent between the performer's gestures and the resulting audio[14]. Cascone notes that the same issues facing live acousmatic music exist in contemporary laptop performance[2]; these are essentially transferred from live tape music.

Though the live acrobatics of laptop performers can be intricate and complex, it is more often the case that musicians exercise limited control over a piece. In these instances, performance is reduced to triggering events, playing prepared control sequences, calling up presets, and perturbing scalar parameters. The performance of complex, layered pieces is difficult for a single performer, and the overwhelming number of variables are cognitively impossible to manipulate at once. Performers often have to resort to preparing automated control sequences and structures before going on-stage to cope with the complexity of a piece. This allows performance via a manageable fraction of the system parameters, but comes at the cost of driving fixed processes with a small set of controls[3]. Further, performance software interfaces are typically organized as dense on-screen control panels, featuring independent scalar widgets that are manipulated individually via the mouse[8]. This arrangement constricts the control flow between the musician and the computer system, additionally compromising live control.

"Preparation" as used here has a different connotation than it might in acoustic music. Jazz musicians, for example, practice various patterns and scales intensively for use in improvisation. A control sequence programmed into a computer can be played back exactly, without effort. Well-prepared human performance is interesting due to its inherent difficulty and natural variability, while prepared computer playback is not because of its ease and mechanical consistency.

This research attempts to address some of these issues in laptop performance by offering an alternative to typical performance software designs. The software prohibits the use of predefined control sequences, and aims to allow expressive performance and improvisatory use. It features an interface paradigm that focuses on a visual, spatiotemporal representation. This representation helps establish a link between performer action and musical output, hoping to address laptop performance's lack of visible, causal gesture. Freehand input via a graphic tablet or mouse is used to drive the software. Ultimately, the software aims to allow for an interesting live experience by returning a sense of active creation to laptop performance.

This paper presents this software system in comparison with various other applications for computer music performance. The design of the system is presented and discussed.

2. RELATED WORK

The most widely used examples of laptop performance software can encourage largely automated live use. These typically feature control-panel interface designs, which can constrict live control. The most popular example of contemporary performance software is *Ableton Live*[1], which allows a musician to layer and process audio clips in real-

NIME 06, June 4-8, 2006, Paris, France

time. Music is performed by starting and stopping groups of loops, triggering clips, and modifying effects parameters using the on-screen controls. Though some clip rearrangement can be done in real-time, it is expected that the clips and effects setups are arranged before the performance. *Pure Data* and *Max/MSP* are the canonical examples of dataflow music software[12], and are also very popular for performance. Again, the vast majority of the work is usually done before the performance in preparing the patch. The patch is typically controlled live by using the on-screen widgets.

Less mainstream applications exist that de-emphasize the use of prepared material and that make use of dynamic graphics, visual feedback and creative interface design to enhance interaction and live use. Examples include the works of *ixi software*[9] and the *FMOL*[7] application.

Interfaces for performance which use freehand input also exist. These make use of a more spatial interface metaphor. Golan Levin's work on painterly interfaces[8] features inspiring examples of instruments made possible by digital technology that encourage improvisatory use. The two examples most relevant here are Levin's *Yellowtail* and *Loom* applications, as they make use of freehand drawing gestures in a manner similar to our research. These applications allow the user to draw strokes on the screen using a pen or a mouse. The drawing gesture is recorded and replayed repeatedly to animate the on-screen strokes, and the graphical output is sonified to generate audio output. Amit Pitaru's *Sonic Wire Sculptor*[11] creates rotating, three-dimensional "sculptures" from freehand input trajectories. The shapes are mapped to sound, allowing a performer to interactively create looping, multi-voiced music.

Other musical interfaces also employ a spatial metaphor. Toshio Iwai's *Electroplankton* game for NintendoDS features autonomous agents that create music[6]. These are controlled by the user, and interact with each other and their environment. Each has individual properties and sounds. This example is similar to this research in that the user sets active, spatial structures in motion to generate musical patterns.

Live coding[4][15] is an innovative laptop performance technique where artists play pieces through live computer programming. Our software prototype shares two characteristics with live coding techniques: the desire to build pieces from minimal starting material in performance, and the creation of music via concurrent, generative patterns. Textual programming allows great flexibility and fine granularity, but takes a substantial amount of effort and technical proficiency. Our research also aims to allow the live creation and modification of control patterns, but at a much higher level using graphical tools. The hope is that our approach will be a good compromise between control and efficiency.

3. PROTOTYPE SYSTEM

A prototype system was designed and implemented which attempts to address some of the above issues and encourage active, engaging laptop performance. Instead of preparing note sequences and control mechanisms before going on-stage, the system requires that the musician create these mechanisms *as* performance. The goal was to foster a more interesting experience for the user, as well as the audience, and to allow greater performance flexibility.

The system was intended to permit independent control structures to be assembled quickly and efficiently. The system targets dance and minimal "glitch" styles, which are typically structured as layers of looping audio patterns. A musician is able to generate the same effect by creating multiple simultaneous control patterns in performance.

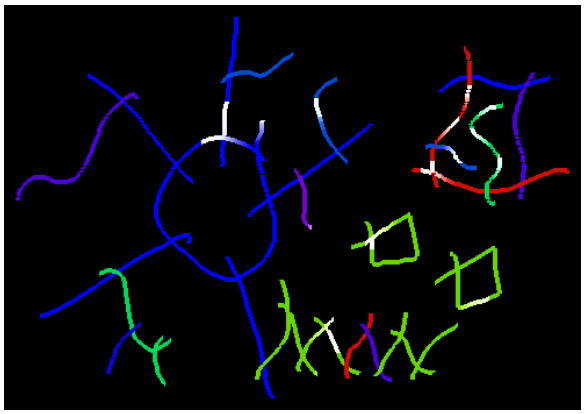

Figure 1: The interface in action

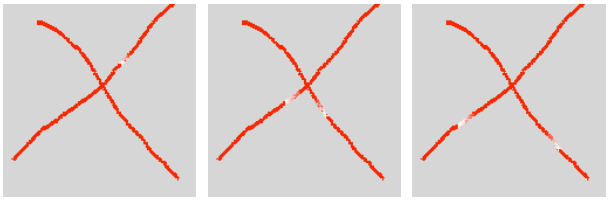

Figure 2: Particle behaviour at points of intersection. The colours have been altered to improve print reproduction.

The prototype system resembles a freehand drawing application, depicted in Figure 1. The program window represents a two-dimensional space in which the user can draw strokes. The strokes define paths along which small, white "particles" may travel. These particles are the active elements of the system; their movements and positions drive the sound playback.

Particles have a special behaviour at points where two strokes intersect. When a moving particle arrives at an intersection point, it is copied to the other stroke, and two particles travel outward from the intersection. This is illustrated in Figure 2.

These simple ideas—particles travelling along strokes that replicate at the intersections—give rise to the system's behaviour. Sets of overlapping strokes thus create figures through which the particles travel and cycle. In turn, these patterns of motion drive audio synthesis, ultimately generating music.

Each drawing gesture made by the user is recorded by the system. Travelling particles mimic these motions as they propagate along the strokes, preserving the gestures' speed and variances. This means that particles always travel along a stroke in the same direction as its original drawing gesture.

Any number of particles may travel along a given stroke at the same time. A particle always follows the user's cursor when drawing, causing new particles to be spawned immediately as intersections are first created.

The motions of these particles govern the sound synthesis via a simple mapping. A stroke can be associated with

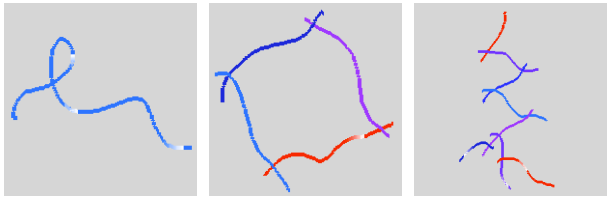

Figure 3: Some typical figures used in performance

a wavetable via keyboard shortcuts. The sound sample is imagined to be "stretched" along the length of the stroke, and a given particle moving along the stroke drives the motion of a corresponding playhead that reads sample data from the wavetable. Thus, faster drawing motions result in higher pitches, and slower ones result in lower pitches. Each stroke is allocated one synthesis voice, and the particle most recently added to the stroke is the one used to drive the mapping. Since there is no limit to the number of strokes, there is no limit to the number of simultaneous synthesis voices.

As mentioned above, wavetables are associated with strokes via keyboard shortcuts. The wavetable is selected from a small, predefined sample set, and subsequently drawn strokes will be associated with that sample until another one is chosen. Strokes are coloured according to their associated wavetable. Silent strokes, with no associated wavetable, can also be selected for control purposes.

Strokes remain stationary after being drawn. The only editing operation currently supported is deletion. A voice may be silenced either by deleting an upstream part of the figure to stop the flow of particles onto the sounding stroke, or by deleting the stroke itself. A stroke's location on the canvas has no effect on the animation or the sound. A figure will behave exactly the same regardless of where it is drawn.

The system can be used with either a graphic tablet or a mouse. One of the key advantages to this system is that it uses freehand drawing gestures, which are known to be very natural[16] and which exhibit interesting, human variability.

There are a number of typical figures that result from this system. The simplest is a stroke with a single loop. A particle orbits around the loop, and a new particle is emitted on each revolution. Loops can also be made using multiple strokes. Not all figures need be loops, and cascading sequences of strokes may also be drawn. These are illustrated in Figure 3.

The system is a self-contained, real-time, graphical application implemented in C++. It uses GLUT[5] for graphics, STK[13] for synthesis, and RtAudio[13] for managing the audio hardware. The code is object-oriented, features design patterns, and uses the C++ standard template library. Further implementation details can be found in [17].

4. DISCUSSION

The prototype system is a first step toward an alternative performance interface that attempts to bring an increased sense of active creation to laptop performance. It aims to address some of the challenges we detail above.

The system avoids a control-panel-style layout in favour of a graphical, spatio-temporal representation. This representation could help visualize and manage the large amount of activity present in complex, multi-layered works. Further, it helps build a connection between the performer's actions, the state of the software system, and the audio output. If projected, the screen display could help to demystify the performance mechanics for the audience.

With this system, we are returning to the acoustic musician's notion of preparation: practicing physical action on an instrument for performance without automatic aids. This is reflected in the fact that there is no mechanism for saving the screen layout or storing predefined control structures. This forces musicians to approach software performance as they might approach acoustic performance: through practice, but not programming. This philosophy also reinforces the interface's usefulness in improvisation.

The use of freehand gestures is very significant to the interface design. Freehand input with a graphic tablet gives the performer a natural, high-bandwidth way to interact with the system that infuses the control data with human variability. This variability can give the music an organic quality that is often absent from computer music. A given piece will sound slightly different each time it is performed, and it is possible to make mistakes. The use of freehand input helps to recapture some of these natural, interesting characteristics of human performance.

The system focuses on the live construction of generative structures for creating musical patterns. This is similar to the live coding approach. This aspect of our research targets styles of computer music where pieces are composed as layers of looping audio patterns. The system hopes to allow these structures to be created quickly and efficiently through graphical means, serving as an intuitive, geometrical way of specifying musical material.

At root, our project aims to be a "spatial language" for quickly specifying audio patterns. The intention was to devise a graphical system with atoms, rules and grammar that could be used for creating music. These visual components would be assembled on-stage by performers to achieve specific effects and patterns. The system's grammar is currently limited, but it will be further developed.

An important point is that the system is intended primarily as an interface for audio performance. The visual component is not necessarily meant to be expressive. In projecting the graphics, we make visible the performer's actions in the same way that one can watch a guitarist's hands as he or she plays. The system is more visually self-explanatory than typical live computer music software, which could help elucidate the performance for the audience. If the graphics are not projected, however, the system's advantages for the performer are still valuable.

There are some outstanding issues present in this early prototype. The current stroke behaviours make it difficult to synchronize between two running patterns. Synchronization could be a useful musical tool to have available. The editing capabilities of the system are currently quite limited, and only support stroke deletion. Some users have expressed that it would be useful to include more subtle editing operations. There is also a problem with the current behaviours for certain stroke topologies. A topology with one loop works well, but topologies with two or more loops constitute a feedback condition. This situation can be encountered unexpectedly in the course of a performance. These challenges will be addressed in future versions of the software.

This project bears similarity to Golan Levin's work in its use of freehand drawing gestures and animated stroke tra-

jectories. While his research provided inspiration for our work, the overall goals of the two projects differ. Levin's work focused on audiovisual performance, the simultaneous creation of abstract visuals and sound. As mentioned above, our project is designed to be a "spatial language" for *audio* performance, allowing one to quickly specify temporal patterns. Levin's focus was on the aesthetic of the interaction, while this research focuses on the interface design issues and the effectiveness of the tool for computer music performance.

Another work to which this research bears some resemblance is Iannis Xenakis's UPIC system[10]. In UPIC, a composer uses a graphic tablet to specify two-dimensional curves that drive various sequencing and synthesis functionality. Our work differs in that we focus on real-time use, whereas UPIC is strictly an offline system. We also have a much narrower scope than UPIC, allowing a particular kind of performance within particular constraints. Xenakis's system is a broadly applicable, general-purpose environment for computer music composition.

An final point to make is that the interface design decisions are idiosyncratic and reflect the opinions of the designers. They were deemed interesting for this iteration of the project, but will be developed further. The current version of the software articulates the flavour of this style of interface and illustrates some of the possibilities offered by our approach. Other stroke behaviours and mappings to audio will be experimented with within the simulation framework, and the system will certainly evolve in the future.

5. CONCLUSION AND FUTURE WORK

This paper has presented research into a software system for computer music performance. Laptop performance is becoming increasingly popular, but audiences can be unsatisfied with the apparent lack of activity and lack of visual cues it sometimes offers. The prototype software system takes the first steps toward a new performance interface design that aims to address these issues. It is a direct-manipulation, graphical interface resembling a freehand drawing program that forces control mechanisms to be assembled live as performance. The system eliminates the use of predefined control sequences and allows improvisation. The variability inherent in the freehand input helps to recapture some of the "humanness" of acoustic performance. This scheme highlights one way in which musicians can exploit the unique affordances of software systems in a performance context.

The prototype system will be further developed. Usability details will be addressed: the system's editing functionality will be extended, menus and other interface items will be added, and the overall graphical feedback will be improved. MIDI or OSC support could also be added for controlling external synthesis software instead of using the application's internal synthesizer.

Most important, the overall system design will be developed. While preserving the core principles and feel of the interface, extended stroke behaviour designs will be found that enhance the system's performance capabilities. The ultimate goal is to have a conceptually minimal system that allows rich, robust and subtle performance.

6. ACKNOWLEDGMENTS

Thanks to Elliot Sinyor for suggesting "Different Strokes" as an entertaining working title for this project.

7. REFERENCES

[1] Ableton homepage. `http://www.ableton.com/`.

[2] K. Cascone. Grain, sequence, system: Three levels of reception in the performance of laptop music. *Contemporary Music Review*, 22(4):101–104, 2003.

[3] N. Collins. Generative music and laptop performance. *Contemporary Music Review*, 22(4):67–79, 2003.

[4] N. Collins, A. McLean, J. Rohrhuber, and A. Ward. Live coding in laptop performance. *Organised Sound*, 8(3):321–330, 2003.

[5] GLUT. `http://www.opengl.org/resources/libraries/glut/`, 2006.

[6] T. Iwai. Electroplankton. `http://electroplankton.nintendods.com/`, 2005.

[7] S. Jordà. FMOL: Toward user-friendly, sophisticated new musical instruments. *Computer Music Journal*, 26(3):23–39, 2002.

[8] G. Levin. Painterly interfaces for audiovisual performance. Master's thesis, Massachusetts Institute of Technology, 2000.

[9] T. Magnusson. ixi software: The interface as instrument. In *Proceedings of the Conference on New Interfaces for Musical Expression*, pp. 212–215, 2005.

[10] G. Marino, M.-H. Serra, and J.-M. Raczinski. The UPIC system: Origins and innovations. *Perspectives of New Music*, 31(1):258–269, 1993.

[11] A. Pitaru. Sonic wire sculptor. `http://www.pitaru.com/sws/`, 2003.

[12] M. Puckette. Max at seventeen. *Computer Music Journal*, 26(4):31–43, 2002.

[13] G. Scavone and P. Cook. RtMidi, RtAudio, and a synthesis toolkit (STK) update. *Proceedings of the International Computer Music Conference*, pp. 327–330, 2005.

[14] A. Tanaka. Music performance practice on sensor-based instruments. In M. Wanderley and M. Battier, editors, *Trends in Gestural Control of Music*, pp. 389–405. IRCAM – Centre Pompidou, 2000.

[15] G. Wang and P. R. Cook. On-the-fly programming: Using code as an expressive musical instrument. In *Proceedings of the Conference on New Interfaces for Musical Expression*, pp. 138–143, 2004.

[16] M. Wright, D. Wessel, and A. Freed. New musical control structures from standard gestural controllers. In *Proceedings of the International Computer Music Conference*, pp. 387–389, 1997.

[17] M. Zadel. A software system for laptop performance and improvisation. Master's thesis, McGill University, 2006.