Laptop Performance: Techniques, Tools, and a New Interface Design

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

Abstract

While personal computers have been used for over twenty years in live music contexts, the proliferation of powerful and affordable portable systems (laptops) has spurred the emergence of new music creation styles and venues outside of academia. Laptop performance is the practice of live computer music using software tools exclusively. This paper surveys the current state of laptop performance practice, discussing the styles of music that are typically performed in this way, and the techniques and tools used. This discussion serves as a context for a new software system for laptop performance. The system's design is motivated by some of the problems encountered in this style of live use. The system supports the real-time definition and modification of generative musical patterns via a novel freehand drawing interface, allowing a solo performer to create multi-layered works on-stage.

1 Introduction

In recent years, there has been an astonishing increase in the computational capacity of computer systems. It has become possible to do real-time DSP on consumer-grade systems, and the emphasis in music-making tools has gone from hardware to software. It is now feasible for the public to make music on their home computers, and these software tools are exceedingly prominent in contemporary music production. This increase in capacity has also made portable computers very common in live performance.

While computer music has been performed in academic research and composition communities for many years, the availability of accessible software music tools has given rise to a computer music culture outside of those circles. Many exciting kinds of music are being made by non-academic artists and producers in home studios all over the world (Hermes 2002).

From this scene, a *laptop performance* subculture has emerged. This term refers less to the fact that laptops are being used in live computer music, and more to *how* they are used. Laptop performance can be defined as live computer music using general-purpose computers, very often without the use of specialized hardware controllers. Its defining characteristic is an emphasis on the live use of software tools. Laptop performers generally sit on-stage in front of a computer, controlling software to play computer music.

The first part of this paper provides a brief survey of the development of computer music performance techniques, followed by a deeper exploration of laptop performance practice. Common software tools are presented, along with how they are typically used in a live setting. Some of the issues present in this style of computer music performance are also discussed.

As laptop performance has become more commonplace, audiences have started to become dissatisfied by its lack of perceivable, causal gestures. The mechanisms by which the music is made are often all but invisible to the audience. The second part of this paper describes the design of a new software interface for laptop performance and improvisation that aims to address this issue. The interface allows the live, graphical creation and modification of generative patterns that drive sound synthesis. It de-emphasizes the use of pre-programmed control sequences and moves toward creating all of the music live, on-stage.

2 Live Computer Performance

This section surveys the evolution of computer music performance techniques, and goes on to explore the contemporary laptop performance phenomenon in depth. Its tools, their use, and some of the issues encountered in laptop performance are discussed.

2.1 History

Computer music was first explored in academic and research settings in the 1950s. In that era, it was not possible to render audio in real-time with the available hardware. In concert, these compositions were necessarily performed

as tape pieces. Audience support eventually started to subside because of the lack of any visible, human component to these performances. Composers began to create pieces for tape and live instrumental musicians in order to make these performances more engaging (Chadabe 1997, p. 68). This strategy proved more satisfying for audiences, and it is still common in today's computer music.

With the emergence of MIDI, it became possible to create interactive computer music that could be controlled during runtime. The use of prerecorded tape in performance poses certain problems for human accompanists since its timing is fixed and rigid. With MIDI, the computer could be made to synchronize with the musicians, making the pieces more flexible and potentially easier to play alongside. Bruce Pennycook's *Praescio* series serves as a good example of this practice (Pennycook 1991). It eventually became possible to create works that could process live audio input from instrumentalists in real-time (Lippe 1993), allowing a new level of flexibility and interactivity in computer music.

Computer music has now been embraced outside of academia due to the availability of powerful and affordable general-purpose computers. The practice of laptop performance has evolved from this trend.

2.2 Laptop Performance

The on-stage use of laptops in non-academic computer music became common over the last decade. While there is now a wide breadth of music being made with these tools, the styles were originally firmly rooted in electronic dance music. Over time, however, many producers drifted away from these influences, developing a new "post-digital" aesthetic. This music showcases the sounds of uniquely digital processes, focusing on errors and failure (Cascone 2000). Traces of dance styles are still evident, however. "[T]he post-digital aesthetic finds itself situated between the popular forms of electronic dance music and the modernist forms of the classical academy" (Turner 2003, p. 81).

A variety of software applications are used for laptop performance, but they generally share several traits. Performance software almost always features a modular paradigm for handling DSP, where sounds are created and treated by chains of individual signal processing modules. The designs differ, however, in the way they handle sequencing and control data. The software can be roughly divided into linear, timeline-oriented solutions and procedurally-oriented solutions, corresponding to the data-flow/control-flow distinction proposed by Duignan et al. (2005).

Figure 1: *Ableton Live*'s session view

2.3 Timeline-oriented Performance Control

Timeline-oriented performance software designs focus on linear pieces of audio and control data, such as sound clips or MIDI sequences. These are positioned in time, processed and overlaid to create entire pieces of music, as in standard sequencing software. The difference is that in a performance context, certain elements of the music are left to be controlled at runtime. Such parameters include, for example, selection and ordering of clips and sequences to be played, as well as signal processing and effects controls. There is an underlying focus on triggering, where the performer explicitly signals when particular parts of the songs should start or stop, and which parts those should be.

The canonical example of this style of performance software is *Ableton Live* (2006). *Live*'s "session view," seen in Figure 1, is the mode used for performance. The basic unit in *Live* is the audio clip, and the session view lays out a collection of clips as a grid. Each column corresponds to a mixer track (i.e., DSP chain), and each row serves as a grouping of clips. Cells, rows and columns feature start and stop buttons for triggering the audio. When a clip is triggered, it can start immediately or on the beat, depending on its configuration.

The typical way this software is used in performance is to prepare clips, their groupings, and DSP chains before ever going on-stage. *Live*'s use in performance can amount to simply triggering groups of clips and presets. The computer handles most of the details automatically, and it is hard for a performer to sound bad or make an obvious mistake.

Another popular timeline-oriented performance tool is *Reason* (Propellerhead Software 2006). *Reason*'s visual design mimics studio hardware units for synthesizing and processing sound, as seen in Figure 2. As such, its use focuses on

Figure 2: A typical interface configuration in *Reason*

patching processing units together to create particular sounds and effects. It supports performance sequencing through its analogue-style step sequencer and drum machine units.

The sequencing modules can hold banks of bar-long sequences. These sequences are programmed into each of the units using their on-screen interfaces. In playback, a performer can trigger and loop each of the different sequences, and they come in on the beat. Performance often consists of running these preset sequences and perturbing effects parameters.

Reason features a container device called the "Combinator" that holds prepared sets of processing units and sequencers. It is meant to be used live to hold songs and patches for performance execution. This makes it easy to switch between or mix different songs and setups on-stage.

2.4 Procedural Performance Control

The other class of popular software performance instruments is dominated by procedural designs. That is, their interfaces concentrate on allowing the user to define procedures that generate and shape the output sequences and sound. These can take the form of visual dataflow interfaces, as in *Pure Data* and *Max/MSP* (Puckette 2002), or that of textual languages, as in *SuperCollider* (McCartney 1998) and *ChucK* (Wang and Cook 2004). These applications tend to be used by more academically-inclined performers since some amount of technical proficiency and understanding is required to program them.

In the case of visual dataflow languages, a patch is prepared before the performance. The patch contains combined sound processing and control machinery. Live performance consists of running the patch, sending control signals and changing its parameters via the on-screen interface as it runs. The contents of the patch can obviously be assembled arbitrarily, leading to a great variety of possible mechanisms. The patch itself is seldom programmed on the spot in performance since it takes a significant amount of effort to assemble a given configuration. However, this is sometimes attempted by more adventurous performers.

Textual audio languages are becoming more popular for live use. In particular, the TOPLAP group (2006) advocates the practice of *live coding*, where audio programs are written and modified on-stage as performance (Collins et al. 2003). This is typically done in a high-level audio-oriented programming language such as *SuperCollider* or *ChucK*, though it is sometimes done using custom setups in generic languages such as *Perl* (McLean 2004). A very interesting notion in live coding is the view that the creation of the procedure itself *is* performance. The algorithms run and generate music while they are being modified, instead of being run after they have been completed.

2.5 Performance Issues

The typical image associated with laptop performance is that of a solo performer staring into the glow of a laptop's LCD screen, executing barely perceivable motions. There is often no visible, causal connection between the performer's actions and the resulting music. The absence of physical gesture and the hidden mechanisms are challenging for audiences, and can hinder their enjoyment of laptop performance.

Another factor that impedes perceivable causality is the fact that laptop performers typically have to resort to preprogramming much of their control sequences and patch setups before ever going on-stage. Their music is often created from many layers of audio, which can prove very difficult to manage live. Automating much of the music production can make multi-layered pieces possible for a solo performer. The drawback is that it detracts from the live experience by hiding much of the performance mechanics.

Performance software interface designs also tend to impede the flow of control between the musician and the computer. They are typically organized as dense control panels, featuring on-screen widgets that are used to individually manipulate system parameters (Levin 2000, p. 38). A performer can only access one or two values at a time using a mouse, compromising live control.

2.6 Recent Trends in Laptop Performance

With the maturing of laptop performance practice, laptops can now be seen as taking a less prominent role on-stage. Instead of being the sole focus, they are being used in conjunction with live musicians and other elements in a bid to

engender a more compelling live experience. The live instrumentalists play over otherwise computer-based audio, which is itself typically controlled by a given performer. For example, artists such as Caribou, Vitaminsforyou and the Books perform in this way. Laptop musicians also often use video projections to further augment the visual component of their performances.

Sometimes a performer will process live audio input to create sound textures. The aforementioned software tools can be used for this, as well as dedicated software packages such as *Guitar Rig* (Native Instruments 2006). We see in these developments a move toward performance techniques that have been popular in the research community for some time.

2.7 Summary

In summary, the laptop performance subculture has become prominent over the last ten years in the wake of powerful software instruments. It is distinguished by a focus on software itself as the primary performance tool, and does not typically employ novel hardware controllers.

Laptop performance software can be divided into two categories, depending on its dominant control paradigm. In timeline-oriented interfaces, linear clips of audio and control sequences are looped, triggered and mixed on-stage to create music. In procedurally-oriented interfaces, music is created from generative control processes that are manipulated live. These programs feature modular DSP architectures, and usually control panel interface designs.

Laptop performance often lacks a sense of active creation since the mechanisms to make the music are largely imperceptible. This has led to the inclusion of instrumental musicians and other elements in an attempt to make laptop performance more compelling.

3 Performance Software Design

A new software system for laptop performance and improvisation was implemented that responds to the performance issues outlined above. It has been presented elsewhere (Zadel 2006; Zadel and Scavone 2006), so here we focus on the ideas underlying its design and how they relate to the above discussion.

3.1 Motivation

The project was motivated by the issues in laptop performance described in Section 2.5. In particular, the design aimed to address the tendency in laptop performance toward simply "piloting" prepared software patches, and to move toward a more active kind of performance that emphasized on-stage action and decision making. This kind of live use could lead to a more engaging experience for the performer, and for audiences as well if the action could be suitably conveyed.

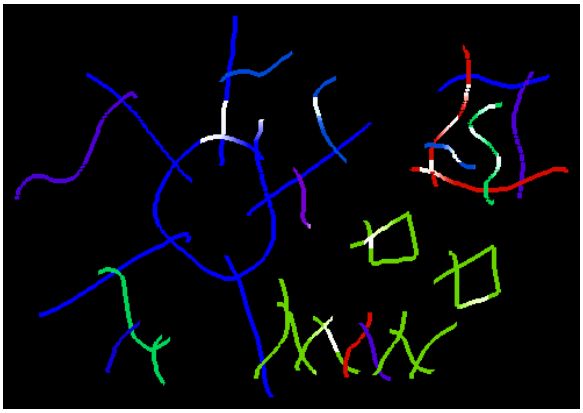

Figure 3: A screenshot of the interface

The system was intended to fit in with the laptop performer's typical toolset. Thus, the interface was not to necessarily require non-standard hardware components in order to be useful. Note that this means that the system does not necessarily attempt to get away from having the performer on-stage sitting at a computer; the aim was to provide interesting performance possibilities *within* the constraints of screen-based software interfaces.

3.2 System Description

The system resembles a freehand drawing interface. On-screen strokes are assembled to create animated figures whose motion drives sample playback. Small, white "particles" travel along the user's strokes, flowing between them at stroke intersections. Samples are mapped to strokes when they are drawn; as a particle moves over a stroke, its associated sample is played back according to the particle's movement. The system is pictured in Figure 3.

The main idea behind the system design was to create a novel, efficient way of defining generative control patterns in a performance setting. The goal was to allow a musician to generate the structures on the fly, on-stage. The system does not allow a performer to create pieces *entirely* from scratch, however, as a small set of samples must be chosen beforehand.

3.3 Design Concepts

This section presents some of design ideas behind the interface and demonstrates how they respond to the above issues in laptop performance practice.

Designing a graphical interface for the computer screen meant that a two-dimensional spatial metaphor could be used. The core idea of the system was to design it as a set of active objects that interacted locally to create temporal behaviours. The user would place and manipulate these objects to create temporal patterns that could be used to drive audio. This had the advantage of being potentially visually self-evident, as well as allowing the computer to manage some of the sound production.

The tool was meant to reduce the reliance on prepared control material, but still allow the performance of multilayered pieces. That meant that the control patterns had to be able to be specified quickly and efficiently, and that the computer would have to be responsible for playing back some of the material once it had been defined. This corresponds to the common form of laptop pieces as multiple tracks of looping audio. Figures can run independently in different parts of the canvas, allowing the artist to layer temporal patterns in multiple voices.

The spatio-temporal representation used by the interface is intended to provide the performer with an intuitive, "physical" way to interact with the system. A nice side effect of this is that it also provides an intuitive way for the audience to understand the system as well. If the screen display is projected, it can serve as a visual manifestation of the performance mechanics that the audience can appreciate, acting as an amplification of the user's gestures.

Some design heuristics were employed to help influence the flavour of the system design. These were not strictly necessary to respond to the above problems in laptop performance, but rather served to colour the interface's look and feel.

First, the design was to be conceptually minimal, but still allow rich sets of behaviours. Instead of offering functionality that had obvious intended uses, the strategy was to provide a small set of atoms that could be combined in oblique, creative ways. This is in contrast to other laptop performance tools, which typically offer a large number of specific controls and features.

Second, the system was to be based on active objects that interacted only as a function of spatial proximity. This mimics objects in real space, which can usually only interact through physical contact. This is also inspired by systems with complex emergent behaviour, like John Conway's *Game of Life* (Dewdney 1988, p. 136), which only allow local interactions. The reacTable* (Jordà et al. 2005) and Small Fish (Fujihata et al. 2000) projects are other examples of interfaces where active objects interact through physical proximity or contact.

Third, the system should try to establish a perceptual connection between the musician's gestures, the system graphics and the output audio. This helps reduce the user's cognitive dissonance and should make for a more usable system (Levin 2000, p. 100). This also aids in visualization as the sound is closely correlated with the on-screen activity.

The system's basis in freehand drawing came later in the design process, and was seen as an elegant way to realize the above design ideas. It also suggested the possibility of using pointer gestures as an extra source of user control and information, though this has not yet been formally investigated. The use of freehand gestures can also help the audience understand what the performer is doing: people are generally very familiar with such gestures, and there is a direct, obvious correspondence between the artist's physical action and the on-screen display.

4 Conclusions

This paper presented a survey of computer music performance techniques, focusing on laptop performance. In the canonical style of laptop performance, where a solo performer uses only software to make music, two control paradigms were identified: timeline-oriented control, where the interface is designed around linear audio clips and control sequences; and procedural control, where the software is designed around the procedural generation of music.

The use of software instruments as the main focus in performance seems to have started to wane in popularity as it can be uninteresting for audiences. Laptop performance does not necessarily provide a visible connection between the performer's actions and the resultant music. The way laptops are used in performance has changed in light of this, and they are now often integrated with other instruments to foster a more engaging live experience.

The design of a new software system for laptop performance and improvisation was described. The system aims to bring a sense of active creation to laptop performance. The system attempts to make laptop performance more engaging by giving the performer a more active on-stage role, and by making her actions potentially visible and meaningful to the audience. The system allows the creation of pieces on the fly from a small set of samples, and its visual representation helps establish a connection between the performer's actions and the resultant music.

5 Acknowledgements

Thanks to Elliot Sinyor for a discussion of contemporary laptop performance practice that helped structure the above survey.

References

Ableton AG (2006). Homepage. <http://www.ableton.com/>, 18 February 2006.

Cascone, K. (2000). The aesthetics of failure: "post-digital" tendencies in contemporary computer music. *Computer Music Journal 24*(4), 12–18.

Chadabe, J. (1997). *Electric Sound: The Past and Promise of Electronic Music*. Upper Saddle River, New Jersey: Prentice Hall.

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

Dewdney, A. K. (1988). *The Armchair Universe: an Exploration of Computer Worlds*. New York: W. H. Freeman.

Duignan, M., J. Noble, and R. Biddle (2005). A taxonomy of sequencer user-interfaces. In *Proceedings of the International Computer Music Conference*, Barcelona, Spain, pp. 725–728.

Fujihata, M., K. Furukawa, and W. Münch (2000). Notes on small fish. In *Ars Electronica 2000 Festival Catalog*, pp. 306–309. Vienna, Austria: Springer.

Hermes, W. (2002). Laptop music. *All Things Considered*. National Public Radio. <http://www.npr.org/programs/atc/books_music/2002/feb/>, 10 July 2006.

Jordà, S., M. Kaltenbrunner, G. Geiger, and R. Bencina (2005). The reacTable*. In *Proceedings of the International Computer Music Conference*, Barcelona, Spain.

Levin, G. (2000). Painterly interfaces for audiovisual performance. Master's thesis, Massachusetts Institute of Technology.

Lippe, C. (1993). A composition for clarinet and real-time signal processing: Using Max on the IRCAM signal processing workstation. In *Proceedings of the 10th Italian Colloquium on Computer Music*, Milan, Italy, pp. 428–432.

McCartney, J. (1998). Continued evolution of the SuperCollider real time synthesis environment. In *Proceedings of the International Computer Music Conference*, Ann Arbor, MI, pp. 133–136.

McLean, A. (2004). Hacking Perl in nightclubs. <http://www.perl.com/pub/a/2004/08/31/livecode.html>, 11 March 2006.

Native Instruments (2006). Guitar Rig 2 product page. <http://www.native-instruments.com/index.php?id=guitarrig2_us>, 11 March 2006.

Pennycook, B. (1991). Machine songs II: The PRAESCIO series—composition-driven interactive software. *Computer Music Journal 15*(3), 16–26.

Propellerhead Software (2006). Reason product page. <http://www.propellerheads.se/reason/>, 11 March 2006.

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

TOPLAP (2006). Homepage. <http://www.toplap.org/>, 11 March 2006.

Turner, T. (2003). The resonance of the cubicle: Laptop performance in post-digital musics. *Contemporary Music Review 22*(4), 81–92.

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

Zadel, M. (2006). A software system for laptop performance and improvisation. Master's thesis, McGill University.

Zadel, M. and G. Scavone (2006). Different strokes: a prototype software system for laptop performance and improvisation. In *Proceedings of the Conference on New Interfaces for Musical Expression*, Paris, France, pp. 168–171.