LATTICE: STRATEGIES FOR AND AGAINST CONTROL IN AN IMPROVISATION INSTRUMENT

Christopher Burns
University of Wisconsin-Milwaukee
Department of Music
cburns@uwm.edu

ABSTRACT

Lattice is a laptop improvisation instrument designed to balance user control with unpredictable behavior. The operator triggers synthesis events, specifying their timbre, and upper and lower boundaries on their frequency and duration. The specific frequency envelope and duration of the event are then algorithmically determined, as are all other synthesis parameters relevant to the chosen timbral type. The resulting tension between performer control and algorithmic specification leads to unfamiliar and interesting improvisational situations. The implementation of *Lattice* also responds to design goals including portability, learnability, and expressive sound synthesis. This paper describes the instrument and its implementation.

1. INTRODUCTION

As many authors have noted, the increasing power of computer hardware and computer music software have encouraged composers and performers to make use of laptops for live electroacoustic performance [4]. My personal interest in laptop performance comes from two directions: first, an increasing awareness of and connection to the San Francisco Bay Area improvisation scene; and second, my experiences realizing and performing live electroacoustic music of other composers. As I heard others perform, I became increasingly excited about the musical possibilities of improvisation; the more I performed with electronics, I increasingly understood the ways that electronic instruments could expose the intellectual challenges of performance, without making the technical demands of a traditional instrumental education [3, 6].

The instrument under discussion, *Lattice*, is one of an ongoing series of composerly responses to the challenges of laptop performance and improvisation, and the first which requires me to perform as a full-fledged improviser, without the scaffold of another composer's structure or aesthetic. *Lattice* makes little distinction between composition and instrument; the software is designed to shape and constrain improvisation, both as a spur to the novice performer and as an embodiment of particular compositional choices.

2. DESIGN GOALS

The most important design goal for the *Lattice* instrument was to balance flexible and expressive performer control with unpredictable behavior. This balance reflects the instrument's intended use for improvisation, and in particular a desire to disrupt habit and cliché. Models for this approach to improvisation include John Cage, who said, "I want to find ways of discovering something you don't know at the time that you improvise.... The first way is to play an instrument over which you have no control, or less control than usual", David Tudor, whose creative work centered on inherently unstable feedback systems, and Luigi Nono, whose *La Lontananza Nostalgica Utopica Futura* constrains the electronics operator's improvisation inside the boundaries of a prerecorded multichannel tape and a violinist's performance [1, 9].

A second objective was to develop an expressive sonic vocabulary using only synthesized timbres. While the absence of samples increased the challenge involved in producing interesting sounds, it enhanced the timbral flexibility of the instrument, as the synthesis parameters are continuously varied. Recording plays no role in the instrument, but real sound was influential in the design; the software was created during an artist's residency in the Santa Cruz mountains of California, and a number of the synthesis engines take some aspect of the soundscape of my daily hikes as their model or jumping-off point.

The third major goal for the instrument was learnability. This objective was approached in two ways: first, the instrument was to enable fluid performance without requiring the technical sophistication needed for traditional acoustic instruments. The interface was to facilitate the intellectual challenges and responsibilities of improvisation, while minimizing any technical learning curve. Second, the instrument was to use similar or identical interface concepts across different synthesis techniques. This reuse of interface concepts allows the timbral vocabulary of *Lattice* to be expanded or altered without significant changes to the interface, and reduces the learning time for the instrument.

Additional design objectives included portability and modularity. *Lattice* is hosted on a laptop, and no additional equipment is required. It is easy to set up and travel with, and can even be performed unplugged (albeit quietly) through the laptop's built-in speakers. This choice required a commitment to use the input devices native to laptops. Software modularity facilitates both development and maintenance of the code.

3. IMPLEMENTATION

Lattice is written in Miller Puckette's Pd language [7]. Pd facilitates rapid application development, is cross-platform, and is free and open-source software.

3.1. Interface Design

Lattice uses the QWERTY keyboard of its host laptop as the main device for user input. Two interface frames are present; single keypresses initiate one set of functions, while keypresses combined with the shift key

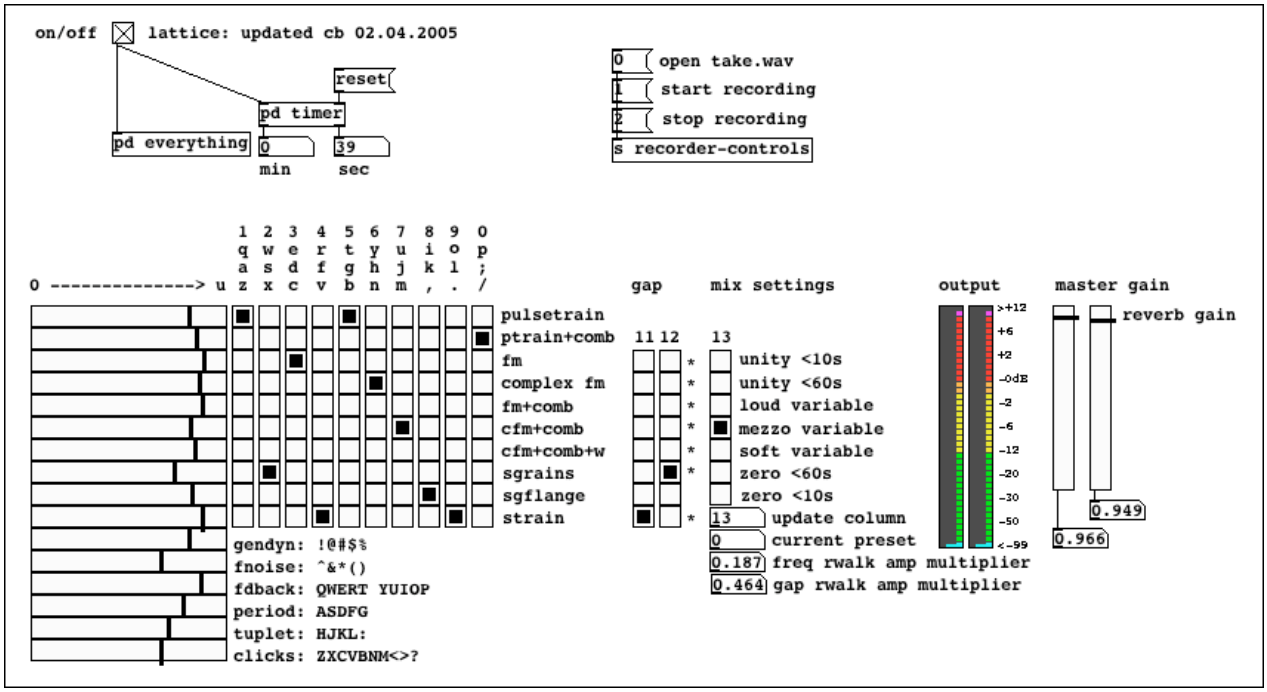

Figure 1. Screenshot of the *Lattice* software.

initiate a second set of functions. While these input gestures may not be the most obviously musical, at least they are familiar to anyone with typing experience.

The first interface frame (single keypresses without the shift key or caps lock) organizes the keyboard as a two-dimensional grid, with pitch register mapped from top to bottom, and duration range from left to right. The grid is organized in ten columns of four keys each, with a variable timbre assigned to each column. The four keys in each column trigger synthesis events of the column's timbral type. They also specify the register of the event; the topmost key in the column will produce higher frequencies, the bottommost lower. The precise registers assigned to the four keys vary somewhat from timbre to timbre; typical values for the lowest register are 50-250 Hz, and for the highest 950-1950 Hz.

The columns of keys are organized according to their upper and lower boundary on duration ("duration range"); the leftmost column produces synthesis events spanning the shortest durations, while the rightmost column produces the longest. As with register, the duration ranges assigned to a particular column of keys can vary from timbre to timbre; typical values are 0.2-0.3 seconds for the shortest range, and 30-60 seconds for the longest.

Ten rows of radio buttons onscreen provide visual information about the current assignments of timbres to columns of keys (see Figure 1). Timbre assignments can be changed using the trackpad to click different radio buttons. Keys outside the 10 X 4 grid can also be used to alter timbre assignments. The left and right arrow keys navigate from column to column, while the up and down arrow keys change the timbre assignment for the current column. The minus and equals keys step back and forth through a series of twenty-one presets, which update the timbre assignments for all ten columns. Finally, the delete key randomizes every assignment. As the performer of *Lattice* I tend to favor the randomization function, because it serves as part of the balance between control (the performer requests a change to the current assignments) and unpredictability (a stochastic algorithm determines the actual new assignments).

The second interface frame (invoked by holding down the shift key) lays out a similar two-dimensional grid, now organized horizontally into rows. This frame has fixed timbre assignments, with four timbres each occupying half (five keys) of the top two ten-key rows. Duration range is still mapped left-to-right, but the ranges repeat with each half-row, so that e.g. the T and P keys will produce similar durations with different timbres. The synthesis techniques assigned to these keys are not particularly amenable to pitch control, and so registral information is not captured.

The third row from the top in this frame is similarly divided in two; the fixed assignment here is not to a synthesis technique but rather to percussion phrase generators. (These phrase generators are discussed in greater detail in section 3.2). The duration range implied by the left-to-right layout of keys here applies to phrase length, rather than synthesis event duration; a single phrase will consist of many discrete synthesis events. Seven unique percussive timbres occupy the seven leftmost keys on the bottom row; the only user input to these synthesis types is triggering, since neither registral nor durational control is applicable. Three of these timbral types are repeated on the <, >, and ? keys, so as to fill out the complete 4 X 10 grid of the second frame.

A few additional controls (not particularly associated with either frame) are available. The arrow keys can be used to navigate not only the timbre assignments for the key columns in the first interface frame, but also the assignment of two post-processing techniques to specific synthesis types. (The post-processing cannot be switched off, only reassigned to different timbres). The arrow keys also enable choices about the overall dynamics of the instrument: the current state can be set to "loud," "mezzo," "soft", "unity <10sec," "unity <60sec," "zero

<10sec," and "zero <60sec". Independent time-varying amplitude scaling for each synthesis type is derived from these global states; the last four settings force all the amplitude scalers to maximum or minimum within the specified timeframe. Trackpad-only controls provide "set-and-forget" options including a master on/off switch, master volume, master reverb volume, start and reset controls for a performance timer, and controls which enable recording the instrument output to disk.

The interface also displays additional information about the state of the instrument. Each synthesis type is associated with a horizontal fader which displays the current amplitude scaler for events of that type. Output meters display the current output RMS volume. The most recently invoked timbre preset (for interface frame 1), the current navigation column for the arrow keys, and the current value of random walk parameters internal to the instrument are also presented onscreen.

3.2. Control Algorithms

The perfomer triggers synthesis events, and in most cases specifies duration range and register for those events. All other parameters are determined algorithmically, without user input. This is *Lattice*'s primary approach to the balance between performer control and unpredictability; the user initiates events, and the software determines the details of their realization. The closest analogy is to percussion performance, where a musician controls the attack of an event, and the state of the drum determines its decay. In *Lattice* the situation is more extreme: as if the size, material, and construction of a drum were all fluctuating continuously during performance!

As noted above, the actual duration of synthesis events is not specified by the user, only the duration range (expressed as an upper and lower boundary). Upon triggering of an event, the software randomly generates an actual duration from inside these boundaries. This value is then propagated to additional parameter-generating functions.

Most other synthesis parameters are governed by random walks (a technique inspired by Xenakis' GENDYN synthesis algorithm) [5, 8]. Random walks determine both x and y values of parameter envelope breakpoints connected by linear ramps. The upper and lower boundaries for both dimensions of these random walks are individual to each synthesis parameter, except in the case of frequencies, where the upper and lower boundaries are generally specified by the performer. Walk durations are randomly chosen below a maximum specified as a percentage of the event duration (sixty percent is a typical upper bound). A final segment exhausting the event duration is initiated in any case where the duration remaining to the total event after the latest walk segment falls below a fixed threshold. The final parameter value may be fixed (as in amplitude envelopes, which ramp to zero at their conclusion) or randomly determined (as in frequency envelopes, which may end anywhere in their available registral space).

The use of line-segment random walks means that most parameter envelopes have complex shapes; single synthesis events have independently evolving frequency, amplitude, and spectral envelopes. The consistent deployment of walks to generate envelopes across synthesis types also means that this aspect of *Lattice*'s code can be extremely modular; reusable panning, amplitude, and frequency walk modules each appear dozens of times in the *Lattice* software.

Random walks are also deployed to generate parameters which vary with each new event, rather than continuously in time. These parameters include filter Q (for the filtered noise synthesis), grain amplitude (for the granular synthesis types), flanging parameters (for the flanged noise synthesis), and most parameters for the various types of click and percussion synthesis. As with the continuously-varying random walk envelopes, these random walks are generated using a single module of code provided with parameters appropriate to the desired range of output. (This module is also used inside each of the random walk envelope modules).

A few additional synthesis parameters are governed by simple random values rather than walks. Wavetables for the pulse-train generators are loaded with five randomly distributed fixed-amplitude pulses. Random values also govern parameters of the highly simplified GENDYN-style noise synthesis.

The final set of control algorithms pertain to the percussive phrase generators. There are two different types, "period" and "tuplet." The period module generates a number of random values when triggered: tempo (which ranges from 40-240 bpm), number of beats per loop (11-31), number of variations per loop (2-6), and the initial distribution of sounding beats. The initial state of the loop is then sonified at the specified tempo using the form of click/percussion synthesis associated with the duration range chosen by the performer. With each repeat of the loop (which continues until the overall duration is exhausted) its contents are randomly varied. If sounding beats are selected for variation they are switched to silence; silent beats are toggled to sounding. The result is a shifting pattern of pulses within a consistent tempo; note that the performer has no control over the sparsity or density of sounding beats, nor over their evolution through time. As mentioned in section 3.1, there are five period modules, each associated with a particular timbre and duration range. All five can be activated simultaneously to create complex and polytemporal rhythmic textures.

Similarly, there are five tuplet modules, each associated with a duration range. These function similarly to the period generators, but now every active beat is given a periodic subdivision (from one to twelve events per beat; the maximum allowable tempo is correspondingly slowed to 120 bpm). As with the beats, not all subdivisions are sounded; a threshold value is set randomly for each new sounding beat, and each subdivision is assigned a random value which must exceed the threshold in order to trigger a synthesis event. Like the period modules, all five tuplets can be active simultaneously. Unlike the period modules, they can switch between different percussive synthesis types. A randomly chosen synthesis type is assigned to each module at random durations between 3-13 seconds; the same synthesis type may be chosen repeatedly, and for more than one module simultaneously.

3.3. Synthesis techniques

Each of the synthesis types controlled by the first interface frame have three polyphonic voices available. Once all three voices are active, additional performer requests for events of a given type are blocked and discarded until one of the sounding events is completed. A variety of synthesis methods are available in this interface frame, including pulse-train synthesis, comb-filtered pulse-trains, frequency modulation synthesis (in versions with one and three modulators, with and without comb filtering), comb-filtered three modulator FM with an additional quartic waveshaping stage at the output, two flavors of granular synthesis with sinusoidal waveforms (one texturally oriented, with rapid streams of grains, the other gesturally oriented, with small isolated sets of flanged grains), and comb-filtered, ring-modulated noise. Two types of post-processing can be applied to any of these synthesis methods. Both types temporarily ramp the output volume to zero; the second type also applies ring modulation during the transitions. Reverberation is also applied to all synthesis outputs after the final mixing stage.

The top two rows of the second interface frame host four types of synthesis: a limited form of GENDYN synthesis which uses random walks to define the output waveform directly; a module which injects extremely short burts of white noise into a flanger; and a pair of waveguide-like feedback networks, with waveshaping stages to compensate for deliberate gain mismatches and idiosyncratic topologies [2]. One of these networks is tuned to emphasize noisy textures, the other to produce pitched timbres (with swooping glissandi as the delay lines constituting the network continuously change in length). The feedback networks are perhaps the paradigmatic synthesis technique for the instrument, since their behavior is particularly unpredictable; when triggered they may produce anything from silence to maximum volume across a noise-pitch continuum.

The phrase generators and the single percussive events triggered by the bottom rows of the second interface frame provide a catalog of different click synthesis types, including noise-modulated FM, ring-modulated noise, flanged noise, various configurations of filtered noise, and low-frequency sinusoidal chirps. While these techniques are all extremely simple, the parameter alterations corresponding with each new event subtly differentiate and enliven the timbres.

4. CONCLUSIONS

Experience performing with *Lattice* in solo, duo, and ensemble contexts suggests that the design goals were largely met in the implementation. There is a clear balance and tension between performer control and algorithmic behavior, with the software occasionally disrupting the performer's intentions. The performer has some ability to affect the amount of entropy (depending on the use of randomized timbre assignments and feedback networks). While the synthesis methods are simple in technique, they are carefully tuned and continuously variable, produce complex, grainy, and expressive sound, and function well in combination with acoustic instruments.

Lattice has proven easy to learn; in performance I think about making music, rather than navigating the interface. The consistent interface concepts have made it easy to integrate new synthesis techniques, and the built-in hard disk recorder facilitates rehearsal. The modular design has also paid off: the "period" generator found a second life in one of the granular synthesis types, debugging has improved the random walk code across the instrument, and the addition of randomization of the timbre assignments was trivial thanks to the architecture. Finally, the instrument is undeniably portable!

The current state of *Lattice* is only a step in a continuing engagement with laptop improvisation; the instrument can be extended with additional synthesis techniques (and chording possibilities will provide appropriate "space" in the interface). *Lattice* will also serve as a springboard for future projects exploring other relationships between control and unpredictability.

5. ACKNOWLEDGEMENTS

Thanks to the Djerassi Resident Artists Program for the opportunity to create *Lattice*, and to Chris Jones, Matt Ingalls, Dan Chudnov, and Steve Nelson-Raney.

6. REFERENCES

[1] Adams, J. D. S. “Giant oscillations: the birth of *Toneburst*.” *Musicworks* 69, pp. 14-17, 1997.

[2] Burns, C. "Emergent Behavior from Idiosyncratic Feedback Networks", *Proceedings of the International Computer Music Conference*, Singapore, 2003.

[3] Chadabe, Joel. "Remarks on Computer Music Culture." *Computer Music Journal* 24/4, pp. 9-11, 2000.

[4] Collins, Nick. "Generative Music and Laptop Performance." *Contemporary Music Review* 22/4, pp. 67-79, 2003.

[5] Hoffmann, Peter. "The New GENDYN Program." *Computer Music Journal* 24/2, pp. 31-38, 2000.

[6] Jordá, S. "Digital Instruments and Players: Part II-Diversity, Freedom, and Control," *Proceedings of the International Computer Music Conference*, Miami, 2004.

[7] Puckette, M. "Pure Data: another integrated computer music environment," *Proceedings of the International Computer Music Conference*, Hong Kong, 1996.

[8] Xenakis, I. *Formalized Music*, rev. ed. Pendragon Press, Hillsdale, New York, 1992.

[9] Zaparinuk, Peter. "David Tudor's Performance Composition." *Musicworks* 71, pp. 47-51, 1998.