Complexity and Control in a Software Improvisation Environment

Christopher Burns, University of Wisconsin-Milwaukee (cburns@uwm.edu)

Abstract: *Kepler's Monsters* is a software improvisation environment comprised of synthesis and signal processing modules that can be activated, interconnected, and arranged in various feedback loops by the performer. The performer assigns rhythmic profiles to active modules; these profiles then trigger the algorithmic generation of the sound-defining parameters. The potential for rhythmic counterpoint exists not only between materials (four independent processing chains are available for layering), but also among different timbral aspects of a single sound event. This paper describes the design of the software, and the tools it provides to manage complexity and unpredictability.

1 Introduction: Design Goals

Kepler's Monsters is a software improvisation environment, designed for musical expression by a single performer or in an ensemble setting. *Kepler's* is one of a series of works responding to the challenges of laptop performance, open form, and improvisation; other software projects in the group include *Lattice* (2004) and *Periodic Table* (2006) [Burns 05].

A number of goals guided the design of the *Kepler's* software. One of the most important was unpredictability - surprising sonic and musical behavior that makes the software as much "performer" as "instrument." Unexpected materials and developmental processes work to foster a feeling of improvisational dialogue between the performer and the instrument, discourage the facility of improvisational clichés and rehearsed gambits, and encourage formal exploration in performance. David Tudor's "performance compositions," especially *Toneburst*, were an important inspiration for this relinquishing of control. Ron Kuivila describes Tudor's live electronic music as "a musical situation in which advance planning is only partially useful, perfect compliance is impossible, and the concepts of contingency and action are essential" - all desirable outcomes for *Kepler's Monsters*, and ways in which laptop performance more generally might rejuvenate and participate in the tradition of open-form music [Kuivila 04].

A second key objective for *Kepler's Monsters* was to develop the entire sound-generating apparatus as an integrated, highly idiosyncratic feedback network. After several years of experimentation with feedback networks as one sonic element within a larger set of possibilities, it seemed possible to treat the entire set of sound generating and processing modules as nodes of an all-encompassing network [Burns 03]. Feedback structures naturally encourage the kind of unpredictability sought for the *Kepler's* system; they also produce flexible, variable, and articulate pitch and timbral contours. The concept of a feedback loop also informed the model of interaction between the performer and the software; the *Kepler's* electronics are designed to follow the performer's musical intentions, but also to channel, divert, and resist them, depending upon the overall state of the feedback network. Models for this push-pull interactive relationship include Tudor's *Toneburst*, George Lewis' *Voyager*, and Luigi Nono's *La Lontananza Nostalgica Utopica Futura* [Lewis 00].

A third design goal for the software was multiplicity: the software should be able to produce several simultaneous textural layers in counterpoint, with complex melodic, rhythmic, and timbral expressions taking place in each individual layer. This polyphonic conception drove a number of decisions about the synthesis engine and the user interface; in particular, the proliferation of sound-defining parameters across four semi-independent textural layers necessitated some of the algorithmic parameter generation and higher-level user interface strategies.

Two other criteria deserve mention. The *Kepler's* software generates sound without the use of samples; prerecorded material seemed antithetical to an "anything-can-happen" spirit of improvisation. And on a more practical level, the software is designed for performance using only the resources of a laptop. While a QWERTY keyboard is not the most obviously musical interface, it has the advantages of familiarity and accessibility (what Sergi Jorda terms "efficiency"), and the advantages of a highly portable, self-contained environment are too great to resist [Jorda 04a].

2 Synthesis Engine

Kepler's Monsters is implemented in the Pd computer music environment [Puckette 96]. Pd facilitates rapid prototyping, is cross-platform, and is free and open-source software.

Sound generation in *Kepler's* begins from four discrete synthesis sources. There are ten synthesis types available for each of the four sources: three sinusoidal oscillator types, two pulse-train oscillator types, two noise models, and three percussive models. The oscillators include a sine oscillator with waveshaping for additional harmonics, a basic frequency modulation carrier/modulator pair, and an FM carrier/modulator pair incorporating glissando. The first pulse-train oscillator drives a resonant bandpass filter, while the second drives a comb filter. The noise models include bandpass filtered noise which is then ring modulated three times in series by a single oscillator, and a simplified "flute-like" waveguide model. The percussive models include Karplus-Strong synthesis, an oscillator frequency-modulated by noise, and an FM bell model corresponding to John Chowning's recipe [Chowning 73].

Each of the four synthesis sources feeds into its own fixed sequence of nine signal-processing modules. In order, each chain of modules consists of:

1. a discretely-variable frequency modulation operator (applied to arbitrary synthesis input by means of a interpolating variable-length delay line),
2. a continuously-variable FM operator (tending to produce glissandi),
3. a continuously-variable bandpass filter,
4. a discretely-variable bandpass filter,
5. a continuously-variable waveguide-like feedback/delay structure,
6. a gate,
7. a continuously-variable amplitude envelope,
8. a delay line with regenerative echo, plus the idiosyncratic feature that it adds percussive clicks to its output when sweeping to a new delay length, and
9. a delay line with regenerative echo.

Each module has two states - active and bypassed - and each module can be switched on and off independently.

These four signal chains (each comprised of the sequence of nine signal-processing modules enumerated above) can then be interconnected to form the unified feedback network. The output of each signal-processing chain can be routed to the input of the two "adjacent" chains (the output of chain one can be fed into the inputs of chain four and chain two, the output of chain two can be feed into the inputs of chain one and chain three, etc.) The output of each chain can also be routed to the inputs of any or all of its own modules. In both cases, a time-varying delay is inserted into the feedback loop to preserve causality, and a compressor/limiter is applied to the signal chain output to prevent runaway amplitudes. As a result of this arrangement, all of the signal processing modules can be connected to one another, either directly or indirectly. Only the four synthesis sources exist outside the global feedback network (necessarily - without some initial injection of signal, an all-digital feedback network will only output silence).

Kepler's Monsters offers two options for the spatialization of its output. For stereo diffusion, the four signal chain outputs are panned to equally-spaced fixed positions within the stereo field. For quadraphonic diffusion, each signal chain output is routed to its own loudspeaker. While there is no explicit motion within the system, the propagation of sound around the feedback network often results in the sensation of spatial animation.

3 Algorithmic Parameter Generation

The audio engine for *Kepler's Monsters* includes a large number of sound-defining parameters. With the exception of the simplest modules (gate and amplitude envelope), there are several parameters for each signal-processing module, and even more for the synthesis sources. Multiplied by the four signal chains, there are too many parameters for one performer to reasonably control. Instead, *Kepler's* generates these parameters algorithmically, and provides the performer with higher-level control over the temporal behavior of the algorithms.

Kepler's includes nine rhythmic generators (hereafter referred to as "timebases"). Each timebase takes two parameters: a rhythmic behavior, and a value from one to ten expressing a range of durations (with one having the shortest minimum and maximum duration, and ten having the longest). There are ten rhythmic behaviors:

1. "periodic" - strictly periodic events (with the period chosen randomly from within the performer-specified duration range),
2. "tuplet" - a shifting periodicity, encompassing various integer subdivisions of a slower basic tempo chosen randomly from within the duration range,
3. "random" - random durations generated inside the range,

4. "randomwalk" - randomly walking between durations inside the range,

5. "lfo" - durations following a sinusoidal curve inside the limits of the range,

6. "growing" - continuously increasing durations (reset to a small value when they cross the range maximum),

7. "shrinking" - continuously decreasing durations (reset to a large value when they cross the range minimum),

8. "phrase" - state-machine logic defining transitions between "short", "medium", and "long" value types (with the actual values randomly generated based upon the range),

9. "meta" - random walk transitions between the previous eight timebase types (in the order presented),

10. "reservoir" – randomly selecting one of a small set of duration values, and occasionally overwriting values in the set with new randomly determined durations (from inside the range).

The actual minimum and maximum values set by the duration range parameter depend upon the rhythmic behavior; musically meaningful minima and maxima depend upon the particular types of rhythms created. In general the ranges are intuitively defined to correspond with a rough scale from *presto* to *largo*.

	timebase type	range	duration
1	shrinking	8	248.18
2	periodic	4	305
3	reservoir	8	5766
4	growing	2	6883.95
5	tuplet	2	58.3846
6	meta	10	13891
7	randomwalk	7	996.698
8	tuplet	4	132.1
9	lfo	2	1111.79

Figure 1: Display of timebase information. "Range" is a value from 1 to 10; "duration" is the current value in msec.

Once activated with a rhythmic behavior and a duration range, the timebases broadcast messages (one per rhythmic event) specifying their next duration in milliseconds. The performer then turns on the various synthesis sources and signal processing modules by assigning them to "listen" to a specific timebase.

Modules within the same signal chain can each be assigned to different timebases. This produces a situation where different timbral features of a composite sonic event have independent rhythmic identities – a kind of “timbral polyrhythm.” On the other hand, multiple sources and modules can be assigned to receive messages from a single timebase. Since there are only nine timebases, complex configurations of modules inevitably involve some shared rhythmic aspects. Polyrhythmic independence, counterpoint, and synchrony are all possible.

As each source or signal processing module receives duration messages, it algorithmically generates all its necessary parameters. With continuously-varying parameters, the duration value is used as a ramp time to create linear or curved transitions from one parameter value to the next. The actual algorithm used varies from module to module and parameter to parameter. Many parameters are generated through first- or second-order random walk processes, but other kinds of algorithmic logic also apply. For instance, the frequency parameters for most of the synthesis sources use a pitch equivalent of the "reservoir" timebase, with the next frequency chosen from a small, incrementally varied set of possible values.

4 User Interface

The performer's main role is to control the parameters (rhythmic behavior and duration range) for each of the nine timebases, and to activate and deactivate synthesis sources and signal processing modules by syncing them to the timebases. In addition, the performer can control which synthesis types are used by the four sources, set the strength of feedback connections across the network, and generate random values for specific groups of parameters. All these actions are triggered via the QWERTY keyboard, with related functions mapped along rows of keys to facilitate the performer’s learning curve.

Most actions are specified via three keystrokes. The first keystroke, from the top row of the keyboard, specifies the parameter group in use. The "`" key selects the timebases, the "1" through "4" keys select each of the four synthesis sources, and the "5" through "8" keys select the four signal processing chains associated with those sources. For timebases, the second keystroke (from the second row of keys) specifies the particular timebase to be configured, with "q" representing timebase one and "o" representing timebase nine; the third keystroke then sets the duration range (mapping the row of keys "a" through "l" to ranges one through ten) or the rhythmic behavior (with the row of keys "z" through "/" selecting one of the ten behaviors). The pattern is similar for synthesis sources, with the row of keys from "q" to "p" specifying which of the ten synthesis types will be used, and the "a" through "l" row identifying the timebase to respond to. (In all

cases, the performer doesn't need to repeat the first or even second keystroke for additional actions, as long as the parameter group or parameter remains the same).

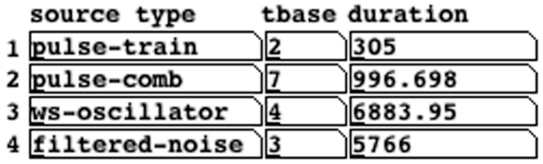

Figure 2: Display of synthesis source information. "Tbase" is the active timebase for each source; "duration" is the current duration for that timebase in milliseconds.

For the signal processing chains, the second keystroke (from the "q" through "p" row) selects a particular module. The third keystroke then syncs that module to a timebase (using the "a" through "l" row to specify timebases one through nine), or sets the amplitude coefficient for the feedback input to that module (with "z" turning feedback off, and "x" through "/" specifying geometrically increasing values from 0.0125 to 3.2). The feedback coefficients for input from adjacent signal chains are accessed with the "[" and "]" keys, with "z" through "/" again specifying the coefficient values.

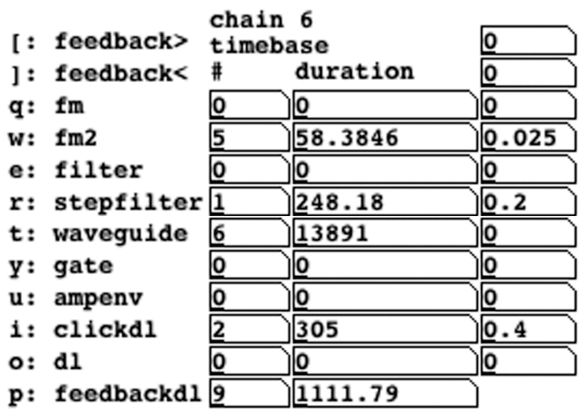

Figure 3: Display of signal processing chain information. The rightmost column displays the feedback coefficients at the input of each module.

The randomization functions can be accessed through a single keystroke (in combination with the shift key). All from the top row of the keyboard, "~" randomizes all the timebase parameters, while "!" through "$" randomizes the parameters for each of the four synthesis sources. The "%" through "*" keys apply a masked randomization to each of the four signal-processing chains, with some timebase values and feedback coefficients altered, and some left as-is; "(" through "+" randomize all the parameters for their particular signal chains. In both types of randomization, there is a bias towards deactivating signal processing elements - if too many elements are activated, the computing resources, the performer, or both can be overwhelmed. Finally, the "delete" key randomizes all the user interface parameters - a useful way to create a sharp sectional contrast, or to create a unexpected situation.

Visual feedback about the state of the environment is provided through a number of tabular displays (examples are shown in Figures 1-3). A performance timer and VU meters for each of the four signal chains are additional conveniences.

5 Assessment

The control scheme for *Kepler's Monsters* is necessarily quite high-level; there are simply too many parameters to expose them all to the performer. While pitch can only be controlled indirectly, the timebase design provides the performer with substantial control over the rhythmic and textural behavior of the system. Relative to its sister project *Lattice*, the keyboard interface is more challenging to navigate; where *Lattice* actions are triggered with one keystroke, most *Kepler's* parameter changes require three. However, the learning curve is still quite reasonable. *Kepler's Monsters* is musically and intellectually demanding to perform, but with rehearsal, the interface itself becomes transparent.

The feedback network produces rich, articulate synthesis, facilitates timbral diversity, and offers the kind of surprising behavior desired for the system. (The role of feedback in the system can be made even more prominent if the synthesis sources are turned off after the network is seeded with some audio). Since the four signal processing chains can be held independent or interconnected, the performer can create complex polyphonic textures, or relate materials into a larger composite.

The feedback network is unpredictable without being out of control. In his discussion of desirable qualities for digital musical instruments, Sergi Jorda writes, "non-linearity should not inhibit the performer from being able to predict the outputs related with small control changes, which seems necessary for the development of a finely tuned skill and an expressive control.... A balance between randomness and determinism, between linear and non-linear behaviors, needs therefore to be found" [Jorda 04b]. *Kepler's Monsters* satisfies

this criterion, with surprising but also manageable behavior. If the performer desires additional unpredictability, the parameter randomization functions are available.

Kepler's Monsters occupies a middle position between "composition" and "instrument" - hence its description as an "improvisation environment." The work is considerably less specific than we usually understand the term "composition": the software is useful both in solo performance and ensemble improvisation, and two performances can potentially sound quite different from one another. At the same time, the software doesn't offer the stylistic breadth (Jorda's term is "diversity") we expect from an acoustic instrument. For this composer, the middle ground is the most interesting space to occupy - why not design the instrument and its music at the same time, and allow each to inform the other?

Acknowledgements

Special thanks to Steve Nelson-Raney.

References

[Burns 03] Burns, C. "Emergent Behavior from Idiosyncratic Feedback Networks", *Proceedings of the International Computer Music Conference*, 2003.

[Burns 05] Burns, C. "*Lattice*: Strategies for and against control in an improvisation instrument," *Proceedings of the International Computer Music Conference*, 2005.

[Chowning 73] Chowning, J. "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation," *Journal of the Audio Engineering Society*, vol. 21, no. 7, pp. 526-534, 1973.

[Jorda 04a] Jorda, S. "Digital Instruments and Players: Part I – Efficiency and Apprenticeship," *Proceedings of the New Instruments for Musical Expression Conference*, 2004.

[Jorda 04b] Jorda, S. "Digital Instruments and Players: Part II – Diversity, Freedom, and Control," *Proceedings of the International Computer Music Conference*, 2004.

[Kuivila 04] Kuivila, R. "Open Sources: Words, Circuits, and the Notation-Realization Relation in the Music of David Tudor," *Leonardo Music Journal*, vol. 14, pp. 17-23, 2004.

[Lewis 00] Lewis, G. "Too Many Notes: Computers, Complexity, and Culture in *Voyager*," *Leonardo Music Journal*, vol. 10, pp. 33-39, 2000.

[Puckette 96] Puckette, M. "Pure Data: another integrated computer music environment," *Proceedings of the International Computer Music Conference*, 1996.