Brock University

Department of Computer Science

User-Guided Evolution of Granular Synthesis

Corrado Coia and Brian J. Ross
Technical Report # CS-09-09
July 2009

Brock University
Department of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
www.cosc.brocku.ca

User-Guided Evolution of Granular Synthesis

Corrado Coia
Brock University
Department of Computer Science
500 Glenridge Ave.
St. Catharines, Ontario
Canada L2S 3A1
corrado.coia@gmail.com

Brian J. Ross
Brock University
Department of Computer Science
500 Glenridge Ave.
St. Catharines, Ontario
Canada L2S 3A1
bross@brocku.com

ABSTRACT

Granular synthesis is a form of audio synthesis that consists of breaking audio into tiny millisecond chucks. This paper describes the EGDE system. EGDE (Evolutionary Granular Delay Environment) is a software plug-in that permits its granular delay effects to be time synchronized with the host application. Its interface features an interactive genetic algorithm, to be used for user exploration of granular synthesis parameters. Interactive evolution is ideal in this application, as it permits a user to interactively explore a wide variety of effects that arise with different combinations of the granular delay parameters. A goal of EGDE is to provide a granular synthesis interface with an intuitive and efficient means of auditioning and rating effect parameters, while minimizing user exhaustion. This is particularly important with a granular synthesis, since many parameter settings will be undesirable to most listeners. Typically, each parameter set in the population can be auditioned in a matter of a few seconds or less. A minimalist 3-value evaluation scheme lets the user either clone, use, or delete candidate parameter sets. Mutation and crossover rates can also be fine tuned as desired. The interface lets the user directly tweak parameters, thus permitting direct user editing of chromosomes. The evolutionary interface could be easily adapted to other musical applications in the future, for example, generalized synthesis engines.

General Terms

Human Factors

Keywords

Granular Synthesis, Computer Music, Interactive Genetic Algorithm

1. INTRODUCTION

Recent years have seen a technological revolution in music production. Digital audio technology has made it possible for anyone with a desktop or laptop computer to run inexpensive music production systems with audio quality that rivals that of professional recording studios of decades past. Furthermore, contemporary music software is capable of audio and musical processing that simply could not be conceived of in the past. Digital audio systems have become essential tools for all aspects of music production – composition, editing, recording, mixing, and final production.

One esoteric area of electronic music is granular synthesis [17]. Granular synthesis treats audio as a collection of tiny asynchronous audio clips or *grains*. Each grain is between 5 to 100 milliseconds in duration, and might be extracted from a larger audio signal, for example, from an audio file or live microphone input. There are a large number of parameters which define the nature of these grains and their playback. Different combinations of parameters can generate drastically different results, ranging from controlled pitch and time transpositions of the original sound signal, to grotesquely mangled mutations that bear little resemblance to the source. The flexibility of granular synthesis has made it a favourite tool of experimental musicians. However, this flexibility also means that it is complex to use. It can be difficult to find parameter combinations that "sound right", especially when many combinations are musically unpleasant.

This paper describes EGDE (Evolutionary Granular Delay Environment), which is a granular synthesis system that features an interactive genetic algorithm interface. The high number of parameter settings required with granular synthesis suggests that interactive evolution is worth consideration. Genetic algorithms and related technologies have a track record for finding acceptable solutions in problems with high dimensional parameter spaces. Furthermore, interactive evolution is of particular importance in applications in which there is no known computable aesthetic evaluation function, and so the user remains the best judge of fitness [2]. With the system in this paper, the user can explore different parameter sets interactively, in an attempt to discover new and interesting effects from the wide variety possible.

The kind of granular synthesis performed by EGDE is *granular delay*. Delay is a well-known effect that simulates an echo with possible feedback. Granular delay performs a delay at a granular level, in which sampled grains of an input signal are fed back into the output stream at some

temporal delay. Some of the parameters controlled by the system include duration of grains, frequency of creation, delay time, play-back pitch, forward or reverse direction, and synchronization intervals. The interactive genetic algorithm (IGA) interface permits all or some of these parameters to be set and explored. The user is presented with a population of 16 parameter sets. The IGA interface is constructed to expedite the fast audition and evaluation of individuals. A unique tri-state evaluation interface is used, in which parameter sets are evaluated as one of: delete, use, or hold. This prevents the need for subjective quantitative scales for evaluation, and also permits settings to be quickly deleted, which is often the most desirable action. Because the system's sound engine continuously pipes a source signal through the granular delay, individuals can be auditioned and rated very quickly, and often in a matter of seconds. This makes audio effect evolution nearly as efficient as IGA interfaces for graphical applications, in which candidates can be evaluated at a glance. The effect is written as a VST plug-in[1], which permits its use in a large number of commercial applications [20].

Section 2 gives some background on granular synthesis, and section 3 reviews related work in interactive genetic algorithms and music. The implementation of the granular delay engine is described in Section 4. A walk-through of the EGDE system in use is given in Section 6. Conclusions and directions for future work are discussed in Section 7.

2. GRANULAR SYNTHESIS

Granular synthesis is an esoteric audio synthesis technique in which audio is comprised of minuscule grains [16] [17]. A single grain of sound is generally between 5 to 100 milliseconds in duration, and when combined together, form complex and dynamic soundscapes. The power of granular synthesis stems from its ability to combine and manipulate grains in countless creative ways. For example, the user can specify the manner in which individual grains are created (location from source audio, duration, pitch, direction of playback, amplitude envelopes,...) and multiple grains are generated (mixing, frequency of generation, order of generation with respect to different grains, repetition rate,...). The control of all these parameters is often specified stochastically or algorithmically, and controlled via software. For example, the CSound language's *granule* operator has 22 parameters that specify the characteristics of granular synthesis generation [21].

The composer Iannis Xenakis created and documented the first granular synthesis principals and techniques in the 1950's [15][16]. His analog approach involved slicing the audio tape into small pieces and then taping them back together in a new order. The first computer-based granular synthesis systems where introduced by Curtis Roads in 1991 and by Barry Truax in 1988 [16]. The digital approach allows for accurate control over all aspects of creating, manipulation, and playback of grains. Hence, more complex compositions are possible.

There are many variations of granular synthesis, for example, granular wave synthesis, granular audio playback, and real-time granulation or granular delay, which is the form of granular synthesis used in this paper. Granular delay is accomplished by having the granular synthesis engine accept a constant audio steam. The engine then creates grains from the audio currently present within the a finite-duration buffer, which presents a moving "window" of audio from which to generate grains. Due to the fact that the audio is streaming into the granular synthesis engine in real-time, the engine is limited to only the audio currently present in the finite-sized buffer and not the entire audio sample. Hence, this method is considered a delay-effect, as the audio input is stored and then processed and played in unison with the live signal.

Many free and commercial software systems support granular synthesis, for example, CSound [21] [4], Native Instruments Reaktor [8], Crusher XStudio [19], and Ableton Live [12]. Ableton Live's granular delay effect is in fact one of the inspirations for this paper. Live's granular delay implements the aforementioned granular delay effect in real-time. A number of parameter controls are available, such as granular frequency, pitch, and feedback level, as well as stochastic ranges for parameters. Furthermore, since Ableton is designed towards rhythmic music and real-time playback, the granular delay can be synchronized to generate grains within timed windows of the source audio.

3. EVOLUTIONARY COMPUTATION AND MUSIC

The application of genetic algorithms in music composition is widely known [5]. Most of this work involves evolving music at the note level, for the purposes of generating compositions.

Of more relevance to this paper is the use of evolutionary computation towards the processing or synthesizing of audio signals at the timbre level. One direction for this research is to use evolution to produce a synthesizer or sound source that generates output matching characteristics of an example sound sample [7] [11] [13]. These techniques use automatic fitness evaluation, to determine the closeness of match between candidate and target sound data.

An alternative approach is to use interactive genetic algorithms (IGA's), in which the user uses the genetic algorithm as a guide in which to interactively explore variations of sounds. Here, the genetic algorithm acts as a tool for sound discovery. Johnson uses a mutation-based IGA to explore different granular synthesis settings [9]. It uses a minimalist interface, in which evolutionary controls and parameter settings are kept hidden from the user. Mandelis describes an IGA for the exploration of synthesis parameters [10]. The system makes use of various crossover operators, and a real-time interface controlled by a data glove. Dahlstedt also uses an IGA to explore synthesis parameters [6].

Other biologically-inspired techniques have been applied to sound generation, and granular synthesis in particular. For example, Blackwell uses particle swarms to generate granular audio [3]. Miranda uses a cellular automata to define granular synthesis frequency and duration values [14].

4. GRANULAR DELAY ENGINE

Audio delay is a well-known audio effect, which could also be called an "echo". To create a delay, an audio signal is mixed with a delayed version of itself. The original analog delay units performed this by using a loop of recording tape, in which the record and playback head worked on the same audio tape in a repetitive, looping fashion. It is easily im-

[1]http://www.cosc.brocku.ca/∼bross/EGDE/

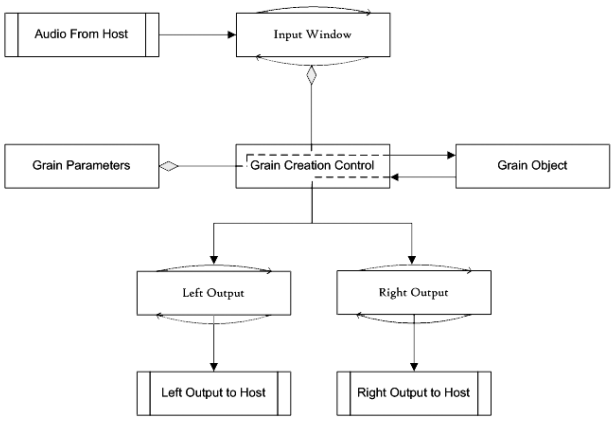

Figure 1: Granular Delay Engine

plemented in digital audio, by mixing a digital signal with a temporally delayed version of itself, saved in a buffer. There can be a variable amount of feedback gain for the delayed signal, which controls the strength of the old signal mixed into the current one. The delay time of the old signal (ie. how far back in the buffer to read the old signal) is also a user-controllable parameter.

EGDE implements a granular delay effect, which is inspired by the granular delay included with the Ableton Live music production system [12]. Rather than performing a uniform delay on a uniform signal as done with a normal delay, granular delay mixes in grains (segments) of the old signal into the current audio. Then the feedback loop takes this audio output, and remixes it back as a delay signal. Hence the overall effect arises due to granular playback and delay feedback, both of which are separately controllable. Almost all of the parameters for controlling granular synthesis are available when generating grains, such as grain duration, grain start time, volume, direction of grain, playback pitch, and others. The difference between it and full granular audio synthesis is that granular delay has a limited window of audio available from which to generate grains, while granular audio synthesis can select grains from any point in a source audio file.

Figure 1 shows the architecture of the granular delay engine. An audio stream from the host is fed into the engine, into a circular buffer (finite queue). This buffer represents the maximum delay time available for the delay effect, and is currently set to 5 seconds. The grain creation mechanism will use the current parameters to extract a grain from this buffer, and generate a grain. A grain is essentially a small clip of audio, with an envelope convolution applied to the start and end to create smooth transitions to zero. This is always performed on grains, or else noise will arise when a grain makes a sudden transition on or off. The generated grain is then mixed with the current signal, and the result is again mixed into the delayed signal buffer, which results in the granular delay effect.

In following the approach of Ableton's granular delay, a constraint of our granular engine is that grains are generated linearly at some fixed frequency rate. Grain duration will be inversely proportional to the frequency. This disallows generated grains to layer directly over each other, which in turn reduces peaking or overflow of audio signals that will arise when grains are directly mixed together. Multi-grain mixing will occur during delay mixing, which will be easily controlled via a feedback level parameter.

The granular delay parameters are as follows:

1. **Grain density and offset**: Density controls how frequently grains are created, and therefore the length of duration of each grain. Grain density offset gives a range of values from which to select a density. Hence when a grain is to be generated, its frequency will be the grain density value plus/minus the offset.

2. **Pitch and offset**: A zero pitch value means the grain has the same pitch characteristics of the original audio. Positive or negative pitch values will either increase or decrease the pitch of the grain with respect to the original audio. Offset is the degree to which pitch changes can vary between generated grains.

3. **Delay and offset**: The delay is the amount of time it takes for a grain to be played back. By mixing a new grain with the input signal into the output buffer at the specified delay time (plus/minus offset), the delay effect is produced.

4. **Feedback**: This controls the amplitude of grains that mix into the output buffer.

5. **Granular spray**: If set to zero, then grains are created from the current input signal. When increased, grains may be created from further back in time. A high spray will create a more "scrambled" delay effect.

6. **Reversal**: This is the probability that a generated grain is played in reverse direction.

7. **Dry/Wet**: This controls the mix balance of the original signal with the granular delay effect, as written to the output.

A new contribution of EGDE is its ability to synchronize grain parameter changes with the tempo from the host, which results in rhythmic changes of granular effects. This provides a rich number of musically interesting effects. Parameter changes can occur at three levels of synchronization, controlled by the following parameters:

1. **Cloud** (cloud length, offset): Clouds define time-determined durations for multiple parameter changes. A cloud is a duration of time in which all the parameters are locked together. If cloud duration is activated (ie. non-zero), then all parameter changes will occur together within the cloud duration. It is akin to a window in which all parameters are set and used. They will only change in unison at the beginning of the next cloud.

2. **Beat synchronization** (grains density, pitch, delay length): If the parameter is set to synchronize on beats, it will do so with respect to k quarter-notes (k between 1 and 16). Different parameters can synchronize at different note rates, which can result in complex rhythmic effects.

3. **Unsynchronized**: Here, a parameter change can occur asynchronously, irrespective of the host clock.

Activated cloud synchronization pre-empts beat synchronization and unsynchronized changes. Likewise, if clouds are not used, then beat synchronization pre-empts unsynchronized changes.

EGDE is implemented as a VST plug-in [20]. VST is a popular software specification developed by Steinberg Music, which permits the implementation of music software plug-ins hostable by various commercial music systems. Because the system is a plug-in, it relies on a host application from which to read and write audio signals, clock tempo information, and other information. Similarly, the plug-in will write the granular delay effect back to the host for playback. The VST environment also provides a GUI environment.

5. INTERACTIVE GENETIC ALGORITHM INTERFACE

Figure 2 shows EGDE's user interface. The population is on the left, the granular delay parameters on the top-right, and evolution controls on the bottom-right.

The population consists of 16 individuals. Each individual contains rating controls, an age indicator, and a "sample" audition button. We forgo a numeric rating system common in many interactive evolution applications, for the following simple tri-state rating scheme (from highest to lowest fitness preference):

1. **Hold** (blue): Clone this parameter set into the next generation, and also use the set for reproduction (a parent). This is a form of elitist reproduction. ;
2. **Use** (green): Use this set for reproduction, but do not clone it.
3. **Delete** (red): Do not use this set for reproduction, and do not clone it.

Note that if the user would rather use a smaller population, say 9 individuals, then individuals 10 through 16 can be set to *delete*, and ignored during subsequent generations. Clicking a sample button will activate that individual. Only one individual is active at one time, as indicated by its activated sample button. When sample is activated, that parameter set is loaded into the host, as well as the top-right parameter control section. It will then immediately process the audio, which will be heard by the user.

Evolution controls are given on the bottom of the interface. A chromosome is a list of floating-point values, one value per granular delay parameter. The *crossover/mutation balance* control determines the probability that crossover is selected for reproduction, verses mutation. If the user would rather use mutation exclusively, then this control should be set to 100. Either one-point crossover or N-point crossover may be chosen by the user. One-point crossover tends to preserve recognizable groupings of parameters, while N-point results in more varied permutations of parameter settings. The *Mutate % of the effects* control determines the probability that each parameter will undergo a mutation, should mutation have been chosen to perform reproduction. Mutation occurs on a gene by perturbing its current value, by the range specified by the *% mutation variance* control. Setting the variance low will result in a small tweak of a value, while a high value may result in a drastic alteration. Finally, *Next generation* will create a new population of parameter sets, using the current population ratings.

The parameter control section shows the parameter values of the currently active population member. These parameters are described in Section 4. They permit direct user editing of parameter values within the active chromosome, and result in immediate audio feedback of edited changes. User editing of chromosomes could be considered a form of interactive Lamarckian evolution [18]. Parameter editing can be undone with the *undo* button, which replaces the parameters with the values that existing before editing commenced. In addition, parameters can be *frozen* (red fields). Frozen parameters will no longer be subject to mutation. However, these parameters can still have crossover applied to them. Freezing parameters lets the user stop the introduction of new values into the population.

6. USING THE SYSTEM

The primary motivation behind the design of the EGDE interface described in Section 5 is to provide the user with a means for quickly and effectively auditioning and evaluating granular effects, as formulated by the genetic algorithm. Because many granular delay effects will create drastic distortions of the audio, many will likely be undesirable. It is therefore useful to let the user quickly evaluate the population, and move through the evaluation process as efficiently as possible. The recommended way to use the interface is to have the host play a looped audio clip, which is immediately processed through the granular delay. The currently activated population member (parameter set) will be the one that processes the audio signal. Therefore, the user must merely click each sample button to immediately hear the effect of that parameter set. Depending on the duration of the loop being processed, an effect may be auditioned within seconds (or less, if the effect is clearly undesirable), and hence the entire population in perhaps 20 seconds.

Evaluating individuals is performed quickly. The user decides if an individual should be held (cloned and used for reproduction), used (removed but used for reproduction), or deleted. We feel that this tri-state evaluation is a natural and intuitive fitness scale for interactive applications such as this, in which the user must make a series of reactive aesthetic judgements. The alternative, more conventional approach of evaluating each individual on a scale of (say) 1 through 10, is arguably of less utility, since numeric evaluations have less intuitive meaning to the user, and even less use for the genetic algorithm that is working with a very limited population of individuals. To promote efficiency, individuals set to *hold* are kept on a hold state for subsequent generations. Otherwise, the default rating for non-held individuals is *delete*. Therefore, users will often only hit the *sample* button to hear an effect, and move on to the next individual. When an effect is deemed worthy of further exploration, then its rating may be changed to *use* or *hold*.

The interface is flexible with regards to the flavours of evolution to be used. Conventional strategies for using interactive genetic algorithms are applicable. During initial use of the system, the user would be recommended to use a roughly equal combination of crossover and mutation. The mutation perturbation rate should be set high as well. As the user samples different individuals, many initial results will be deemed unacceptable, and will be pruned from the population. Interesting effects will be retained. As generations progress, a particular effect may be especially pleasing to the user. This parameter set may be set on *hold*; the

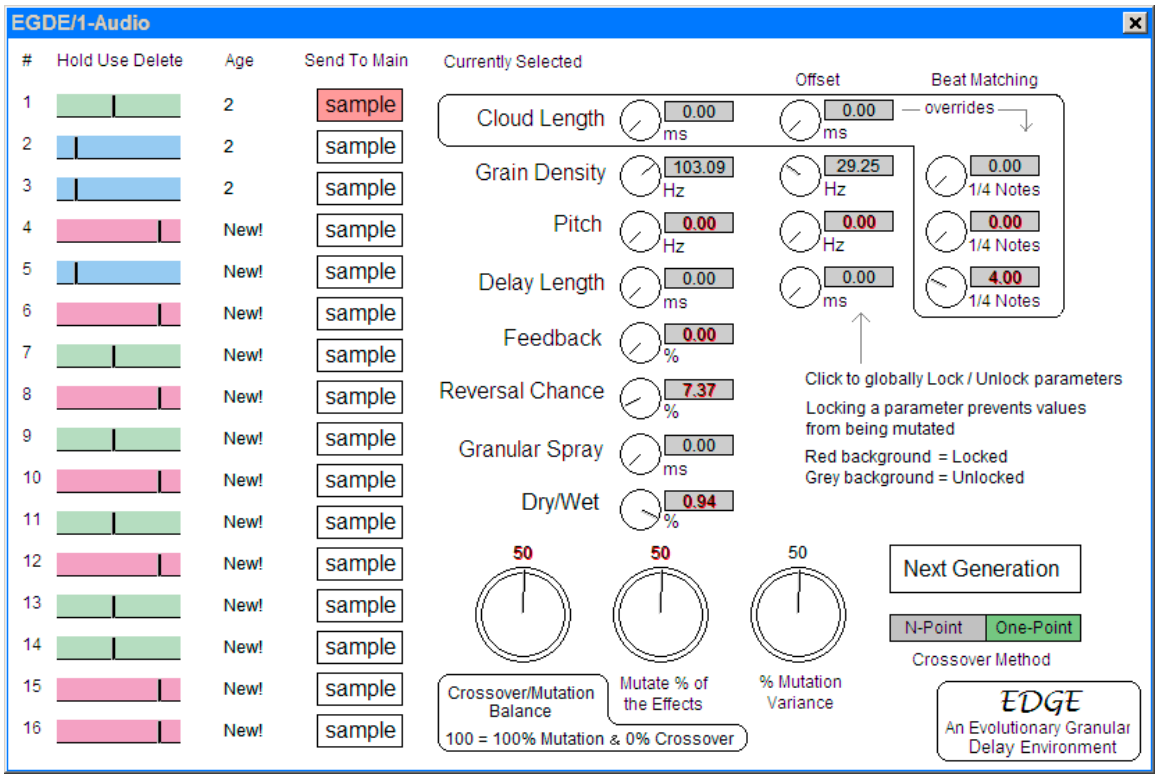

Figure 2: Granular Delay Interface

crossover rate can be reduced, as can the mutation rate. Then evolution will be used to tweak the effect, until a final desired version is found. At any time, the user is free to directly edit parameters. This will directly alter chromosomes, and user changes will be available for use during subsequent evolution.

Alternatively, exclusively mutation-directed evolution is common in interactive genetic algorithms. This may be done by turning the crossover rate to zero. As usual, the mutation perturbation rate can be fine tuned as generations progress. Mutation-based evolution can be used on multiple individuals, or just one individual set to *hold*. Of course, in the case of multiple individuals being mutated, the lack of crossover prevent shared characteristics from appearing in offspring.

The VST interface permits the user to save a selected parameter set as a parameter file. This can be loaded and used by the granular delay engine at any time. It can also be read and used outside the interactive genetic algorithm interface, since the granular delay engine can be used independently of the genetic algorithm interface. The user will therefore save favourite effects in a library, for future reference.

7. CONCLUSIONS

Our granular delay engine is inspired by the granular delay effect bundled with the Ableton Live music system [1]. Along with many of the parameters used by Ableton's effect, our engine additionally permits complex beat-synchronized parameter changes. However, like other granular synthesis systems and digital synthesis systems in general, there are a vast number of parameter settings possible. Parameters interact in complex, nonlinear ways. This makes using granular effects difficult, especially when many parameters create extreme results.

We implemented a interactive genetic algorithm which the user can use to explore the vast space of granular possibilities, and iteratively refine and tweak until a desirable effect settings are found. User evaluation of individuals is usually a bottleneck in interactive evolution, due to user fatigue when confronted by the task of manually evaluating many populations of possibly unimpressive individuals. To help overcome this problem, we designed an interface that promotes fast and effective auditioning of candidate effects. This is particularly important in music and audio, because the user must *invest time* to hear an audio signal to completion. This contrasts to evolutionary art applications, in which the time invested is the significantly shorter duration it takes to glance at and evaluate a visual effect. Knowing that many granular delay effects will not be useful to a user during early generations, our interface lets users quickly hear an effect, and just as quickly remove it from further consideration. When effects are of more interest, rather than assign a score (which takes a certain amount of effort to determine, and is probably an arbitrary value anyway), the user merely indicates that that parameter set is to be flagged for use for reproduction. If the effect is of particular interest, it is flagged for use and cloning. And finally, the best results can be saved, and used in the main host application as desired.

Our use of an interactive genetic algorithm to explore granular synthesis parameterizations differs from the approach of Johnson [9]. His system uses mutation exclusively, utilizes numeric evaluation of individuals, and keeps

the granular synthesis parameters hidden from user inspection and editing. The use of IGA's with general synthesis systems in [6] and [10] is comparable to our approach, although our IGA interface design is motivated for fast sample evaluation. Our interface should be readily adaptable to other synthesis and effects technologies. Although work using other techniques towards granular synthesis, such as swarms [3] and cellular automata [14], is quite different than ours in approach, all share the goal of helping the user navigate the complex parameter space of granular synthesis.

Future revisions of EGDE are being considered. The granular synthesis engine could be enhanced with additional parameters, for example, parameter acceleration, inter-grain spacing, envelope control, panning, amplitude, and others. We are also considering the idea of having some sort of visualization for IGA population members, to give the user a visual representation of a parameter set. This would allow the user to see how generated parameters relate to their parents, which might be of use during auditioning and evaluation. Finally, more population editing controls for sharing parameters amongst the population members, and perhaps removing parameters from evolution entirely, are being considered.

8. REFERENCES

[1] Ableton. Ableton live 7. http://www.ableton.com. Last accessed Sept 26, 2008.

[2] P.J. Bentley and D.W. Corne. An Introduction to Creative Evolutionary Systems. In P. Bentley and D. Corne, editors, *Creative Evolutionary Systems*, pages 1–75. Morgan Kaufmann, 2002.

[3] T. Blackwell. Swarm Granulation. In J. Romero and P. Machado, editors, *The Art of Artificial Evolution*, pages 103–122. Springer, 2008.

[4] R. Boulanger. *The CSound Book*. MIT Press, 2000.

[5] A. Burton and T. Vladimirova. Generation of musical sequences with genetic techniques. *Computer Music Journal*, 23(4):59–73, Winter 1999.

[6] P. Dahlstedt. Creating and Exploring the Huge Space Called Sound: Interactive Evolution as a Composition Tool. In *ICMC 2001*, pages 235–242, 2001.

[7] A. Horner, J. Beauchamp, and L. Haken. Machine tongues XVI: Genetic algorithms and their application to FM matching synthesis. *Computer Music Journal*, 17(4):17–29, 1993.

[8] N. Instruments. Reaktor 5. http://www.native-instruments.com. Last accessed Sept 26, 2008.

[9] C. Johnson. Exploring Sound-Space with Interactive Genetic Algorithms. *Leonardo*, 36(1):51–54, 2003.

[10] J. Mandelis. Genophone: An Evolutionary Approach to Sound Synthesis and Performance. In *EvoWorkShops*, pages 535–546, 2003.

[11] J. Manzolli, A. M. Jr., J. Fomari, and F. Damiani. The Evolutionary Sound Synthesis Method. In *MM'01*, pages 585–587, 2001.

[12] J. Margulies. *Ableton Live 7 Power!* Course Technology PTR, 2008.

[13] J. McDermott, N. Griffith, and M. O'Neill. Evolutionary Computation Applied to Sound Synthesis. In J. Romero and P. Machado, editors, *The Art of Artificial Evolution*, pages 81–101. Springer, 2008.

[14] E. Miranda. On the Origins and Evolution of Music in Virtual Worlds. In P. Bentley and D. Corne, editors, *Creative Evolutionary Systems*, pages 189–203. Morgan Kaufmann, 2002.

[15] T. Opie. Granular synthesis resource site. http://www.granularsynthesis.com/. Last accessed Sept 26, 2008.

[16] C. Roads. *The Computer Music Tutorial*. MIT Press, 1996.

[17] C. Roads. *Microsound*. MIT Press, 2001.

[18] B. Ross. A Lamarckian Evolution Strategy for Genetic Algorithms. In L. Chambers, editor, *The Practical Handbook of Genetic Algorithms*, volume 3, pages 1–16. CRC Press, 1997.

[19] J. Stelkens. Crusherx-studio. http://www.crusher-x.de. Last accessed Sept 26, 2008.

[20] S. M. Technologies. Cubase. http://www.steinberg.net. Last accessed Oct 3, 2008.

[21] B. Vercoe. *The Public Csound Reference Manual*. Media Lab MIT, 1992.