

# The Evolving Drum Machine

Matthew John Yee-King<sup>1</sup>

Creative Systems Lab, Department of Informatics, University of Sussex, Brighton, UK

**Abstract.** The expectation of the listener from house and techno music seems to be that percussion sounds will maintain the same timbre for the duration of a piece of music. For the composers of such musics the synthesizing of drum sounds of a quality equal to those available from commercial drum machines or samples is difficult and seems unnecessary. A system is presented here which provides a unique method for the composition of rhythmic patterns with dynamic timbres. A genetic algorithm using a heterogeneous island population model is applied to the problem of percussion sound synthesizer design. Multiple percussion sounds are evolved simultaneously towards different targets where the targets are audio files specified by the user. The fitness function driving the evolution compares the evolving sounds to the target sounds in the frequency domain, awarding higher scores for closer matches. The system was tested using a simple step sequencer interface, as found in classic drum machines and a MIDI controlled version has also been implemented. The system provides the user (and listener) with a tangible sense of timbral transformation as the performance proceeds, where the timbres move ever closer to the target sounds. This represents an effective application of an artificial life technique to real time, algorithmically enhanced music composition.

## 1 Introduction

Percussion sound timbres are often static in music. Acoustic percussion instruments can generally produce a limited set of timbres with varying loudness but apart from retuning, the instrument's timbral output cannot easily be stretched beyond this palette during a performance. Even the snare drum, the cornerstone of jazz drumming, with its huge variety of timbres cannot change its shape or material during a performance. In 'popular' forms of electronic music such as house and techno music, the situation is more extreme. The vast majority of drum tracks for this music are constructed either using classic analogue drum machines such as the Roland TR909 or using samples of such drum machines (or other drum samples). This seems at odds with the electroacoustic and experimental strains of electronic music, where timbral richness and dynamics play a key role. Modern, live performance orientated drum machines such as the Electribe series from Korg provide means for editing the drum sounds 'live' but generally this is limited to 'tweaking' the filter cut off or similar transient effects, it is not feasible to reconfigure the underlying synthesizer live.

The system presented in this paper attempts to redress the balance by providing its user with a means for composing rhythmic patterns which are sonified using evolving percussion synthesizers. A genetic algorithm is used to explore the space of possible parameter settings for a general purpose, subtractive-style percussion synthesizer modelled after the systems found in classic analogue drum machines. The composer can work with the sounds whilst they are evolving as the current fittest sound can be triggered by MIDI or using the step sequencer interface shown in Figure 1.

User testing with the system using the step sequencer interface suggests that this can be rather an inspiring and enjoyable way to construct rhythm tracks and also to design percussion sounds that may be used later as samples. The development and analysis of the system has shed some light on the particular properties of the GA which lend themselves to rapid evolution of 'good enough' solutions, such as population size, population models and genomic manipulations.

## 1.1 Related Work

Genetic algorithms have been applied to sound synthesis problems many times. This system employs a similar fitness function to [1] (later developed in [2]) where GAs were used to find settings for FM synthesizers. In [3], GAs are used to control complex granular synthesis and in [4] GAs are used to match timbral classes. See [5] for an overview of the motivation and techniques. Applications of GAs to percussion sounds are less common - see [6] for an example of an application of an Interactive GA to the design of percussion sound synthesizers.

The integration of algorithmic processes to live music composition and performance is inspired after [7] [8] [9], where different methods are used to generate machine improvisers and machine enhanced composition components. In [10] an interactive genetic algorithm which allows a group of users to collaboratively rank and thereby evolve sounds is described, work which has in part inspired the exploration of different population models here. In [11] Whitley discusses the island population model as a method for avoiding premature convergence on local maxima. For a discussion of fixed architecture networks vs changeable size networks, the basic for the exploration of growable genomes mentioned in this paper, see [12]. The user testing carried out for this work was designed after the Contextual Inquiry model proposed in [13] wherein the test is carried out in the context of the user's work, which in this case involves an alternation between composition and discussion. The sound synthesis set up used in this work was inspired particularly after that found in the Waldorf Attack [14] but also based on the sounds found in classic analogue drum machines. The drum machine samples used as evolutionary targets in the experiments were obtained from the Hollow Sun website [15]. The system was developed using the Super-collider sound synthesis language and server [16] and the Java language. where the separate parts of the program communicate using OpenSoundControl [17]

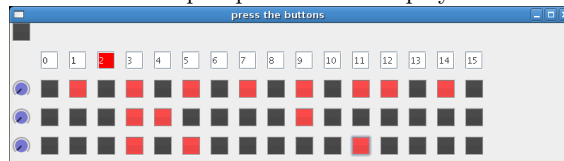
## 1.2 Overview

This paper consists of a technical description of the system, a results section providing analysis of the performance of the GA along with user experiences and finally a conclusion.

## 2 Implementation

The system consists of two programs. The first program carries out the search for percussion sounds using a genetic algorithm and provides a graphical interface which allows the user to specify population size, target sound and other GA parameters. The second program allows the user to play back the sounds. For the user tests reported here, this was a simple drum machine-like sequencer which plays back the sounds in a pattern defined by the user, shown in Figure 1. The second program has also taken the form of a MIDI controllable system where the sounds are triggered using an Akai MPD16 MIDI drum pad. In essence, the search program evolves a separate population of sounds for each track on the sequencer or pad on the MPD16 and notifies the playback program of the fittest settings for the synthesizer at each iteration of the genetic algorithm. In this way, the play back program is always using the fittest sounds found so far by the search program. The system is intended for live performance, where the changing sounds are part of the aesthetic of the performance.

Fig. 1. The 3 track step sequencer GUI employed for user testing



### 2.1 Search

The search program employs an unsupervised genetic algorithm which searches a 20 dimensional space which represents settings for a synthesis algorithm. This synthesis algorithm is described in detail in the section 'Synthesis'.

**Genetic Encoding** The genome contains one or more sets of 20 floating point values in the range 0-5. The values are scaled by the synthesis engine into appropriate ranges. Table 1 lists the parameters along with example values before and after scaling. When a growth operation occurs (see section on genome manipulation), an additional set of values will be added in which case the sound will be made using 2 complete synth graphs. A single synth graph is shown in Figure 2.

**Fitness Function** At the beginning of the evolution the user specifies a target sound which should be an uncompressed audio file. A Fast Fourier Transform (FFT) is performed on the target audio file where up to 50 consecutive, non-overlapping snapshots are taken of the spectrum with a window size of 512 samples. The snapshots are evenly spread throughout the sound so the interval between snapshots is greater for longer sounds. The resulting real and imaginary values are interpreted to power readings for each frequency bin. The fitness function then translates each of the genomes in the population into an uncompressed audio file and performs an FFT. with the same parameter settings. The sum of squared error between the two sets of power readings is calculated. This value is then multiplied by the ratio of the target sound's length to the candidate sound's length as it was previously observed that the evolving populations would exploit a lack of length punishment and converge on very short sounds otherwise.

**Table 1.** The parameters encoded in the genome with example values for the generation 100 sound shown in Figure 3. Example values rounded from 16 decimal places.

	Parameter	Range	Genome value	Scaled value
1	Oscillator base frequency (BF)	0-500Hz	1.649	164.852
2	Envelope 1 attack	0-0.005s	0.2688	0.0003
3	Envelope 2 attack	0-0.05s	0.803	0.008
4	Envelope 1 decay	0-2.5s	1.563	0.781
5	Envelope 2 decay	0-2.5s	0.677	0.338
6	Envelope 2 peak	0-base frequency Hz	4.021	109.205
7	Square oscillator level	0-1	4.3124	0.8625
8	Sine oscillator level	0-1	1.0579	0.2116
9	Noise oscillator level	0-0.5	4.977	0.498
10	Clean mix level	0-1	4.94	0.988
11	Low pass filtered mix level	0-1	1.201	0.24
12	High pass filtered mix level	0-1	1.499	0.3
13	Low pass filter cut off	$0-2*BF + BF$ Hz	0.583	196.52
14	High pass filter cut off	$0-2*BF + BF$ Hz	2.402	295.293
15	Low pass filter resonance	0-0.05 reciprocal of Q	3.625	0.036
16	High pass filter resonance	0-0.05 reciprocal of Q	4.731	0.047
17	Envelope 2 - low pass filter cut off	0-1	3.208	0.642
18	Envelope 2 - high pass filter cut off	0-1	3.323	0.665
19	Envelope 2 - square osc base frequency	0-1	0.984	0.197
20	Ring modulation	0-1	1.752	0.35

**Breeding strategies** The system provides different breeding strategies. The most basic uses a single population of individuals with a uniform mutation rate where the next generation is made by crossing over and mutating the two fittest

genomes from the previous generation. The fittest genome is also added to the next generation. A more complex breeding strategy splits the population into several islands. The genomes from each island are assessed for fitness independently and the islands can have a varying mutation rate. Migration can occur from one island to another, where a random individual from one island is moved to another island. This means the islands can change size.

**Genome Manipulation** New genomes are created by crossing over the two fittest genomes then mutating them. Mutation can occur in different ways:

1. *All loci* mutation adds or subtracts 0.05 from all values in the genome for a 1% mutation rate.
2. *Jump* mutation assigns a new random value between 0 and 5 to 1% of the loci for a mutation rate of 1%.
3. *Creep* mutation adds or subtracts a random value between 0 and 0.5 (10% of the maximum locus value, 5) from 1% of the loci for a mutation rate of 1%.

The genome can also increase in size by adding a complete set of synthesis parameters to the end of the genome. These parameters can be a copy of the existing set in the genome or a set of random values. The effect of these different mutation types on the GA performance is discussed in the results section.

## 2.2 Synthesis

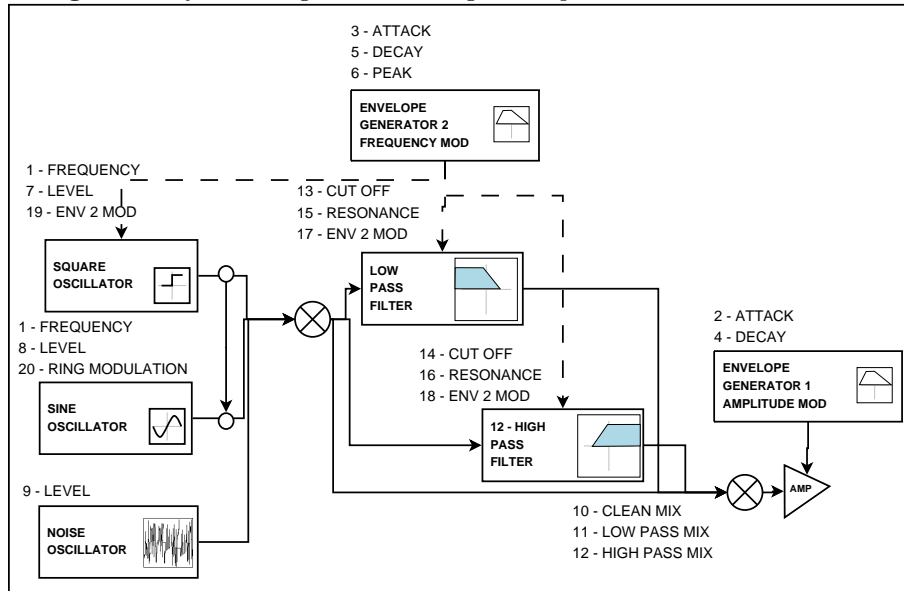
The synthesis algorithm follows a subtractive synthesis model, inspired after the models found in classic analogue drum machines. (see [14] for a description of some classic drum machines' synthesis algorithms). The justification for using such a synthesis model as opposed to something more advanced or esoteric is that to explore a well known drum sound aesthetic which is prevalent in much electronic music and held in high regard will immediately provoke contrasting reactions in electronic musicians. Also this places demands on the system to not just provide 'interesting' or 'esoteric' sounds, but to provide quality, useable sounds. The synthesis graph is shown in Figure 2. and is configured via 20 parameters which are listed in Table 1. The genome is split into one or more genes, each of which encodes settings for a complete set of parameters. If there is more than one gene (as will be the case if a growth operation has been carried out), there will be more than one instance of the synthesis graph used to produce the sound.

## 3 Results

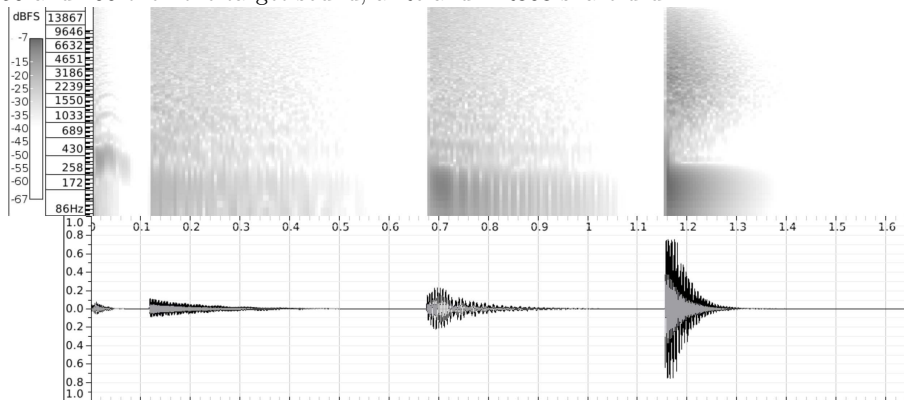
### 3.1 GA performance evaluation

**Assessing genomes** On a dual core Athlon X2 2.2 GHz system evolving 3 populations simultaneously, the assessment time per genome was approximately

**Fig. 2.** The synthesis algorithm showing the 20 parameters listed in Table 2.1



**Fig. 3.** Example sounds from an evolutionary run, time domain plot at the bottom and frequency domain plot at the top. The sounds shown are the fittest from generation 1, 50 and 100 then the target sound, a Roland TR808 snare drum.



150ms, which varied depending on the length of the sound. With 3 populations of 50 genomes to assess, that meant the sounds heard from the drum machine changed once every 22.5s. On the 1.8 GHz single core Intel Dothan system employed for user testing, the time increased to around a minute per change, so the population size was decreased to 20 for the user tests. The effect of waiting for the sounds to change was lessened since the 3 populations were evolving asynchronously.

**Algorithm settings and their effect on musical output** Whilst part of the focus of this work was the artistic applications of genetic algorithms and the musical results of this, some analysis of the system's search performance was essential to indicate future technical directions for the work. Decisions made at the technical level clearly have an impact on the sound and behaviour of the system. To further understand the possible behaviour of the system, several test evolutionary runs were carried out with different algorithm settings. A run lasted for 1000 generations and the convergence time and fitness were measured as the time to reach 90% of the maximum fitness and the highest fitness in the population at this time, respectively. All of the runs were evolving towards one of three target sounds: a TR808 kick drum, a TR808 snare or a TR808 closed hi-hat. It was found that the populations all reached a similar fitness for each sound, 0.03, 0.01 and 0.06 respectively.

In [11], it is argued that the island population model provides increased performance in the case where the different objectives of a multi-objective optimisation problem are linearly separable and can thus be solved individually in parallel. Migration then allows the combination of the individual solutions to provide what is likely to be a global maximum. The problem of synthesizer design could be described as multi-objective since the requirement is to generate a series of spectral feature vectors in the correct order and different parts of the synthesizer can generate different feature vectors at different times. The question of linear separability is beyond the scope of this paper, but the results of several evolutionary runs did not indicate any performance advantage was gained from the island model which may indicate that this particular problem is not linearly separable. Simple epistatic effects between loci reduce separability, for example the base frequency parameter affects envelope 2 and both of the filters. The parameters for envelope 2 have more complex epistatic effects as it modulates several parts of the synthesizer.

To assess the effect of the different mutation types, several runs were carried out with population size 50 and different mutation types. Jump mutation provided consistently poor performance. All loci mutation provided excellent performance for the hi-hat but poor performance for the snare drum. 10 more runs were carried out to check this observation and it was observed that all loci mutation provides consistently faster convergence than creep mutation for the hi-hat sound but that the convergence fitness was consistently lower than the creep mutation runs - premature convergence. The length punishment would be more severe for a very short sound such as the hi-hat and given that all loci are

mutated, there is a good chance of getting a shorter and thus much fitter sound with each iteration. This suggests that varying the mutation type across islands as well as mutation rate might increase the performance.

So what is the musical result of a genome with or without epistatic effects? What is the effect of different types and rates of mutation? For the purposes of this discussion the operation of the evolving drum machine in a live performance context is considered, where the performer is triggering the sounds using a MIDI drum pad. The first, rather obvious observation is that the evolving drum sounds move gradually from nondescript start sounds towards the target sounds. But why do they not make larger jumps? Possibly since the epistasis in the genome prohibits the cleanly modular evolution possible with a linearly separable problem. The population size has an impact here as well - a larger population produces the same rate of increase of fitness but in fewer iterations, so small populations are preferable to provide musically interesting, frequent and small jumps.

In [18], time domain audio data is evolved directly towards a target sound and the author explicitly states that the evolutionary optimisation process is being used as part of the style of the piece. The target sound is changed mid evolution and different types of mutation are applied and it is observed that this adds to the musicality of the piece, varying its dynamics. It was observed with the evolving drum machine that the initial phase of evolution was more rapid and that this provided an interesting movement to the musical output. Changing the target sound automatically and periodically before the fitness increase levelled off kept the system in a perpetual state of rapid convergence. Changing targets is also a challenge to the performer, who must decide which of the available sounds is appropriate for which part of the rhythm. A low mutation rate can slow the evolution down, offering a more gradual exploration of the space of possible drum sounds. A high mutation rate provides a series of much more random sounds, but requires that the system be configured not to keep the fittest sounds from each population.

### 3.2 User experiences

Besides the author's experience of using the system discussed in the previous section, initial user testing was carried after the Conceptual Inquiry model which aims to answer the question:

'How can we get detailed information about how people work when they cannot articulate it on their own?' [13]

The inquiry is carried out by means of an interactive session with a user, an interviewer and the system in question. The user attempts to carry out the task for which the system was designed, a task with which they have familiarity, and the interviewer attempts to gain information about the way the user does this. It is observed that the user finds it easier and more pertinent to express opinions in this context than in a non-contextual interview. Another approach would

be a questionnaire, but this forfeits the opportunity for impromptu, contextual discussion. This has proved to be a useful first approach to assessing the evolving drum machine as the users were able to immediately start composing with the system, then to learn about and respond to its peculiarities. So far, two musicians have been invited to compose rhythm patterns with the system whilst asking and answering questions. This is not sufficient to constitute a full evaluation of the system but it does provide some insight into the usefulness of artificial life derived music tools for musicians.

They were first presented with the sequencer interface with which they interacted whilst the sounds evolved. Some initial questions were asked and then the underlying evolutionary process was explained. They were then given access to the search algorithm interface and allowed to continue composing, but this time they could change settings on the GA such as the target sound file. The musicians interviewed had different backgrounds: musician 1 uses Roland analogue drum machines and synthesizers in an improvisational context to compose somewhat noisy electronic music and could be seen as a drum machine virtuoso; musician 2 is a skilled guitar player who composes melodically sophisticated music using industry standard tools such as Cubase and GigaSampler.

Musician 1 noted that the sounds reminded him of the sounds from early drum machines such as the Roland CR78. He also observed that the sounds improved in quality as the evolution progressed and (once he was informed as to the underlying process) that they became recognisably similar to the specified target sounds (which were in this case samples of a Roland TR808). He would be interested in using these drum sounds in his compositions, wherein the drum sounds are normally sourced from analogue drum machines, synthesizers and effects units in various configurations. He made the observation that sometimes the sound that was supposed to be the snare was actually of more use as a bass drum. This ties in with the GA performance observation that there is a larger variation in the convergence fitness for the snare drum than for the other sounds - perhaps the snare drum is more difficult to match due to having the tonal component of the bass drum along with the noise component of the hi-hat. He stated that the relinquishing of control to the machine and the deliberate exploration of the accidental, a process embodied in such a system, was a valid aesthetic in itself but that in this case the human does need to maintain editorial control. He suggested various improvements that could be made to the interface, such as displaying more information about the target sound.

Musician 2, with their leaning towards a more traditional musical aesthetic found the evolutionary method of exploration to be a refreshing approach to making drum sounds, which he normally constructs from samples or simple virtual synthesizer patches. He was happy to relinquish control of the sound design process to the machine, accepting this as a valid technique by which unique sounds could be made. Before being told much of the underlying system, he said he was using pitch and volume dynamics to decide which sound should be used for which part of the drum kit. If the sound that was evolving towards a snare sample was pitched lower than the sound evolving towards a bass drum

sample, he would use the snare seeker as a bass drum. As the evolution progressed, he tended to map the sounds to their appropriate parts. He suggested that the system would be more useful if the sounds could be triggered from a MIDI sequencer and that volume controls (later added) would be of use.

## 4 Conclusion and Future Plans

A genetic algorithm controlled drum synthesizer has been successfully implemented and this represents a novel application of artificial life inspired technology to a well established musical tool. The system can run in real time in the context of a live percussion performance where the populations of drum sounds are heard to converge as the performance proceeds. The construction and testing of the evolving drum machine has generated encouraging results from the musicians to whom it was presented, both in terms of its usefulness as a creative tool and its performance as a classic drum machine sound re-synthesizer. The evolutionary process as an aesthetic for composition may as yet be something that falls beyond the bounds of 'popular' electronic music but could provide an intriguing compositional framework, where varying the properties of the search algorithm generates a variety of musical behaviours.

The future plans for the work centre around development of more sophisticated fitness functions and synthesis algorithms. The first step will be to implement some perceptual weighting in the fitness function so that the importance of different parts of the spectrum is weighted according to a human listener's response to the spectrum. In [19] different methods are discussed for recognising the variety of timbres that can be made with a snare drum. These methods revolve around eliciting archetypal representations of the snare drum timbres. Evolving towards archetypes of different drum sounds as opposed to recordings of said sounds will allow the fitness function to operate in a lower dimensional space. More sophisticated synthesis techniques such as physical modelling could be implemented to allow the sounds to escape from the classic analogue drum machine model. The author is excited about the implementation of these ideas and the observation of their effects on composers.

## 5 Acknowledgements

The author would like to thank Nick Collins for his guidance and enthusiasm and the reviewers for their constructive criticism.

## References

1. Horner, A., Beauchamp, J., Haken, L.: Genetic Algorithms and Their Application to FM, Matching Synthesis. *Computer Music Journal* **17**(4) (1993) 17–29
2. Mitchell, T., Charles, J., Sullivan, W.: Frequency Modulation Tone Matching Using a Fuzzy Clustering Evolution Strategy. In: AES 118th Convention, Barcelona, Spain. (2005)

3. Takala, T., Hahn, J., Gritz, L.: Using Physically-Based Models and Genetic Algorithms for Functional Composition of Sound Signals, Synchronized to Animated Motion. In: International Computer Music Conference (ICMC). (1993)
4. Gounaropoulos, A., Johnson, C.: Timbre interfaces using adjectives and adverbs. In: Proceedings of the 2006 International Conference on New Interfaces for Musical Expression. (2006) 101–102
5. Miranda, E.: t the Crossroads of Evolutionary Computation and Music: Self-Programming Synthesizers, Swarm Orchestras and the Origins of Melody. *Evolutionary Computation* **12**(1) (2004) 137–158
6. Collins, N.: Experiments with a new customisable interactive evolution framework. *Org. Sound* **7**(3) (2002) 267–273
7. Bown, O., Lexer, S.: Continuous-time recurrent neural networks for generative and interactive musical performance. In: *EvoWorkshops*. (2006) 652–663
8. Collins, N.: Towards Autonomous Agents for Live Computer Music: Realtime Machine Listening and Interactive Music Systems. PhD thesis, Centre for Science and Music, Faculty of Music, University of Cambridge (2006)
9. Blackwell, T., Young, M.: Swarm granulator. In: *EvoWorkshops*. (2004) 399–408
10. Woolf, S., Yee-King, M.: Virtual and Physical Interfaces for Collaborative Evolution of Sound. *Contemporary Music Review* **22**(3) (2003) 31–41
11. Whitley, D., Rana, S.B., Heckendorn, R.B.: Island model genetic algorithms and linearly separable problems. In: *Evolutionary Computing, AISB Workshop*. (1997) 109–125
12. Gruau, F., Whitley, D., Pyeatt, L.: A comparison between cellular encoding and direct encoding for genetic neural networks. In Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L., eds.: *Genetic Programming 1996: Proceedings of the First Annual Conference, Stanford University, CA, USA, MIT Press (28–31 1996)* 81–89
13. Holtzblatt, K., Beyer, H.: Making customer-centered design work for teams. *Commun. ACM* **36**(10) (1993) 92–103
14. Waldorf: Waldorf Attack Percussion Synthesizer, <http://www.waldorfmusic.de/en/products/attack>. (2007)
15. Hollow Sun: Vintage Keys/ Beatboxes. Website (2007)
16. McCartney, J.: SuperCollider, a real time audio synthesis programming language. Website (2007)
17. Wright, M.: Open sound control: an enabling technology for musical networking. *Org. Sound* **10**(3) (2005) 193–200
18. Magnus, C.: Evolving electroacoustic music: the application of genetic algorithms to time-domain waveforms. In: *Proceedings of the 2004 International Computer Music Conference*. (2004) 173–176
19. Tindale, A., Kapur, A., Fujinaga, I.: Towards Timbre Recognition of Percussive Sounds,. In: *Proceedings of the International Computer Music Conference (ICMC)*, Miami, Florida. (2004)