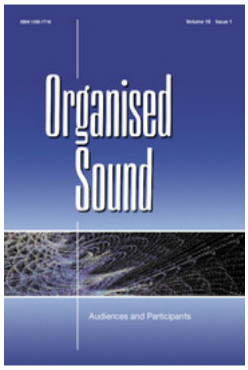
Organised Sound

http://journals.cambridge.org/OSO



Experiments with a new customisable interactive evolution framework

Nick Collins

Organised Sound / Volume 7 / Issue 03 / December 2002, pp 267 - 273
DOI: 10.1017/S1355771802003060, Published online: 26 June 2003

Link to this article: http://journals.cambridge.org/abstract_S1355771802003060

How to cite this article:
Nick Collins (2002). Experiments with a new customisable interactive evolution framework. Organised Sound, 7, pp 267-273 doi:10.1017/S1355771802003060

Experiments with a new customisable interactive evolution framework

NICK COLLINS

London, UK
E-mail: nick@sicklincoln.org
URL: www.sicklincoln.org

This article collates results from a number of applications of interactive evolution as a sound designer's tool for exploring the parameter spaces of synthesis algorithms. Experiments consider reverberation algorithms, wavetable synthesis, synthesis of percussive sounds and an analytical solution of the stiff string. These projects share the property of being difficult to probe by trial and error sampling of the parameter space. Interactive evolution formed the guidance principle for what quickly proved a more effective search through the multitude of parameter settings.

The research was supported by building an interactive genetic algorithm library in the audio programming language SuperCollider. This library provided reusable code for the user interfaces and the underlying genetic algorithm itself, whilst preserving enough generality to support the framework of each individual investigation.

Whilst there is nothing new in the use of genetic algorithms in sound synthesis tasks, the experiments conducted here investigate new applications such as reverb design and an analytical stiff string model not previously encountered in the literature. Further, the focus of this work is now shifting more into algorithmic composition research, where the generative algorithms are less clear-cut than those of these experiments. Lessons learned from the deployment of interactive evolution in sound design problems are very useful as a reference for the extension of the problem set.

1. INTRODUCTION

This work was principally inspired by two sources. The first was the MutaSynth software of Palle Dahlstedt (Dahlstedt 2001) which is readily adaptable to different creative situations. The generality of the system is accomplished by sending MIDI messages, the mapping being the choice of the designer of a specific synthesis patch. Secondly, this author recently witnessed an interactive fractal evolver called ArtE-Fract (Chapuis and Lutton 2001) which allowed effective user exploration of the parameter space. Whilst static visual images will always allow a much quicker assessment than time-consuming sound, the utility of the UI and the readiness of freezing or adjusting particular parameters on the fly suggested good dividends in audio parameter exploration.

This paper works with interactive genetic algorithms without any attempt to reduce the fitness bottleneck of the human decision-maker, using neural networks or similar tools (Biles, Anderson and Loggi 1996, Todd and Werner 1999). These sound synthesis experiments deal with situations where the object to be generated is not of great musical complexity compared to, say, the evolution of composing systems in Jacob (1995).

Other previous efforts in the use of genetic algorithms in sound synthesis include Fujinaga and Vantomme (1994), where large sets of granular synthesis parameters are evolved, and Yuen and Horner (1997) show a use of genetic algorithms for optimisation of parameters within a general synthesis method related to wavetables. Colin Johnson's paper (Johnson 1999) sees him beginning to establish a basic user interface for interactive evolution of sound. That work was surpassed by Dahlstedt's user configurable MutaSynth. A further attempt at a generally applicable user interface system for interactive evolution is introduced in this paper. The applications described demonstrate the range of investigation supported by the library.

2. A FRAMEWORK FOR INTERACTIVE EVOLUTION EXPERIMENTS

No attempt will be made to provide an introduction to genetic algorithms, since this has been covered many times before in the literature. It should be warned that the literature is not consistent in the use of genetic algorithm terminology. The terms chromosome/allele, genome/gene and parameter set/parameter are used interchangeably. It is probably most helpful to think in terms of a parameter space of defined extent (defined by whatever synthesis algorithm we are investigating), and of individual parameter sets that are fixed points in the multi-dimensional space.

The basic type in this research (for an indivisible allele) is a single float parameter. The use of genes at the bit level is avoided since the arbitrary flipping of bits within a float makes little sense as 'musically meaningful mutation' (Biles 1996). Instead, these atomic float parameters have an assigned range, and degree of allowed variation under mutation. This is a specification of the bounds of the parameter space, and the way in which variation may occur to points in that space under genetic algorithm operations. In this paper, tables will be provided listing the range of all parameters in a space. There may be multiple copies of a particular parameter, corresponding to many instances over a number of simultaneous voices or other cloned components of the synthesis method.

In conducting each experiment, the user was provided with a graphical user interface supporting the auditioning and rating of eight candidate parameter sets per generation. At any point, the facility was available to examine and tweak manually any given parameter value, to reset a candidate to a random sample from the parameter space, to change the chances of mutation, alongside options to produce a new generation via best two parents reproduction, asexual reproduction, weighted parenting and interpolation between the top two rated candidates. Save and load facilities were integral to the implementation, allowing favourite sets to be used over multiple sessions.

The exact user interface and implementation is described in section 3, since the general reader may not be familiar with SuperCollider. For now, we proceed directly to consider the experiments in turn.

2.1. Reverb

Genetic algorithms have been used in filter design problems by audio engineers. Here we explore some of the possible (monophonic) reverbs allowed by particular reverberation algorithms. Working as sound designers, with no need to match a real room response, we have the freedom to explore the 'space' of rooms within the purview of a formula.

As a warm up, bare multitaps were evolved. The parameter sets were simply arrays of the delay time and amplitude of impulses, with amplitude in decibels relative to one in a sixteen bit dynamic range. There was a fixed limit on the maximum number of taps, but within this limit, the number of active taps was a parameter. No statistical information was imposed to promote natural reverberation shapes for the first generation, but interactive evolution allowed one to home in on both conventional and more left-field impulse responses.

N stage delay networks were created with feedforward and feedback lines between any two distinct stages. *N* was fixed for a specific run of such chromosomes. In practice, values of *N* up to eight were practical for realtime auditioning. The data template consisted of delay times and amplitudes for any permissible connection, alongside parameters holding the number of current active connections. For this experiment, up to ten feedforward and five feedback lines could connect any two stages. The synthesis code was implemented with the resolution of the sum to each buffer, one buffer per stage. The combinatorics of indexing connections for an *N* stage network were not trivial, but were comfortably programmed within the paradigm.

Figure 1 shows a diagram of a three-stage network designed through a genetic algorithm. Figure 2 shows the impulse response of this network normalised over a half-second range (the initial impulse is missing). In practice, good results were achieved for five-stage networks and above. A seven-stage network plan would run at about forty per cent CPU cost. It would not be advisable to run the exact delay networks once discovered, rather, the calculated impulse responses can be more cheaply modelled with a multitap or FFT convolution.

For a third reverb algorithm, the Schroeder design (Roads 1996: 483) of parallel comb filters, series allpass units and separate multitap early reflections was adopted as a model for the genome. Coprime delay times for the comb filters were not made compulsory. The data template for the genome is given in table 1, giving a sixty-three dimensional parameter space. With this machinery in place, it was a simple matter to find pleasing reverberations. A number of general controls independent of evolution were given for the Schroeder reverb in the form of stretching parameters to try out different time scales, and final absorption of a lowpass filter. They were not added to the data template itself (though they easily could be) because they do not determine the impulse response's form.

The interactive evolution of reverbs was an ideal way to get interesting custom impulse responses. The benefit of the genetic algorithm was in cross-breeding favourite responses, in nosing around the space without aimless randomisation. Whilst a great deal of refinement would be needed to engineer realistic quality reverbs, the more perverse potential of the reverb algorithms were a delight to discover. Future work could explore stereo reverbs, then possibly interactive evolution for spatialisation algorithms, generating particular spatial paths or room models.

2.2. A non-standard crossfade between breakpoint set wavetables

In this experiment, a global synthesis method instrument was evolved with parameters for the three wavetables required. The instrument was auditioned as a MIDI instrument, so it could be played and tried out by the sound designer in context rather than just through a single fixed bleep. The synthesis method described here is relatively simple, but to the author's knowledge has not previously appeared in the literature. The non-standard crossfade refers to the following adaptation of a crossfade (linear combination) signal.

me *A*, *B* are periodic of period *p*, and that *C* is periodic of period *q*, taken for this experiment as $q = p*2^k$ for some integer *k*. *k* corresponds to some number of octaves separation in frequency, with normal value $k = 0$ to preserve an exact pointwise relationship as defined below. Define the output signal as:

$$\text{OUTPUT} = A * (1 - S) + B * S$$

where

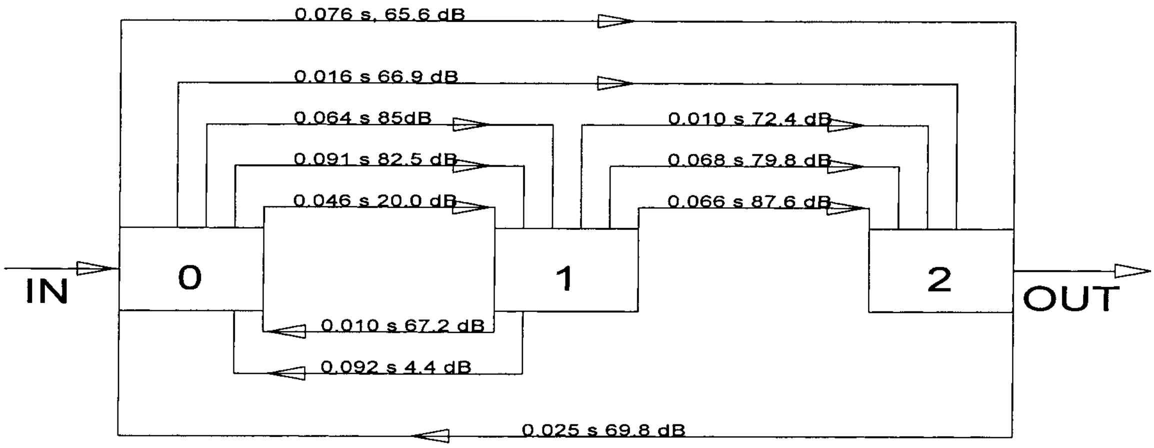


Figure 1. An example three-stage delay network.

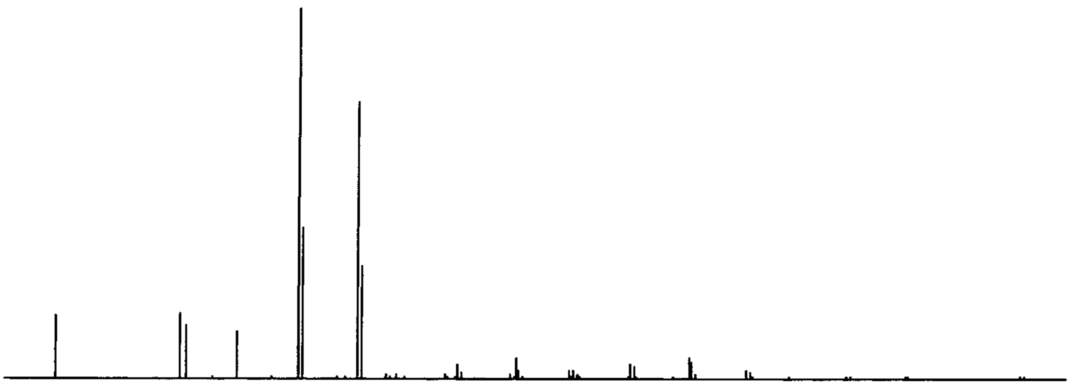


Figure 2. Impulse response for the delay network in figure 1.

Table 1. Schroeder reverb parameter space.

Parameter name	Valid range	Number
Comb delay time	0.05–0.2	10
Comb decay time	0.1–1.0	10
Early delay time	0.05–0.1	10
Early amplitude	0.001–0.5 (exponential)	10
Allpass delay time	0.001–0.1	10
Allpass decay time	0.01–0.1	10
Number of combs	1–10	1
Number of early reflections	1–10	1
Number of allpasses	1–10	1

$$S(t) = \begin{cases} 0 & t < 0 \\ t/C(t) & 0 < t < C(t) \\ 1 & t \geqslant C(t). \end{cases}$$

C can be thought of as the set of interpolation times, specified pointwise from A to B. A target point in B is reached in a proportion of the standard crossfade time, the proportion held by C. This is a time domain trick which distorts the intermediate waveforms in a crossfade from a pure linear combination.

The wavetables gave the oscillators A, B and C. They were specified by ten ordinate values equally spaced in x, in the range 1.0 to –1.0 for A and B and in the range 0.01 to 1.0 for the interpolation times C. Zeroes were always grafted on at the start and end of the A and B tables to avoid a click. The non-standard crossfade was mapped to MIDI controllers by pitch bend controlling p, modulation controlling k in determining q, and a further slider controlling time t manually, with a MIDI note on triggering a particular fundamental frequency.

The fun of this experiment was in sound design on the fly. MIDI patches let you try out generated sounds in context, even in the practice room or studio. One could really see the potential of IGA parameter exploration as a generic method of sound design. The drawback of the MIDI instrument is that auditioning it is more time consuming, exacerbating the human bottleneck in the genetic algorithm search. The results of the experiment were encouraging in terms of the fun of auditioning and finding some useful sounds, though single wavetables themselves (especially with even x positions and fixed numbers of breakpoints) did not give quite enough variety of output. Future experiments with multiple arbitrary breakpoint set wavetables should allow more scope.

As an alternative approach, one imagines a second stage where a settled synthesis patch has its control mappings evolved. The integrated approach presented, however, was quite effective in showing up weak sounds in context, and the author felt that he got to know the limits of the synthesis method through such exploration far more effectively than in auditioning a single output tone. That live controllers were available to manipulate the sound, rather than fixed envelopes evolved alongside the other parameters, meant that from a point in the parameter space the controller dimensions were available for immediate auditioning. The IGA covered the variations amongst many dimensions for the specification of the breakpoint sets, something a human being would have difficulty tracking, whilst the four controller dimensions were few enough to explore manually. This made for a good combined search through the sound space, and one might claim the extra auditioning time was warranted.

2.3. Percussion sounds using Klank for body resonances

The language SuperCollider in which these experiments were conducted (see section 3) has a powerful unit generator called Klank, which takes many arguments. Klank provides a set of resonant filters with user-specified centre frequencies, amplitudes and ring times. These parameters are perfect for GA exploration. In this experiment, a percussion model based around Klank is the subject of the interactive evolution. The percussion template was given additional parameters for ten noise generator units of varying type and envelope, and AM and FM synthesis. The template for the chromosome is revealed in table 2, showing a total of 221 float parameters in the genome.

Table 2. Percussion parameter space.

Parameter name	Valid range	Number
Number of noises on	1–10	1
Noise type	0–9	10
Noise frequency	20–10,000 (exponential)	10
Envelope peak time	0.0001–0.1 (exponential)	10
Envelope decay time	0.01–0.5 (exponential)	10
Envelope curve	−16 to 16	10
Vibrato frequency	0.01–10,000 (exponential)	10
Vibrato amplitude	0–90 dB	10
Tremolo frequency	0.01–10,000 (exponential)	10
Tremolo amplitude	0–90 dB	10
Resonance frequency	20–20,000 (exponential)	40
Resonance amplitude	70–90 dB	40
Resonance ring time	0.001–1.0 (exponential)	40

Auxiliary variables (which were not evolved) were provided in the user interface to turn on or off AM, FM and resonance components of the synthesis to aid in isolating sound characteristics. The noise type parameter referred to ten different noise generators provided by SuperCollider, being WhiteNoise, BrownNoise, PinkNoise, PinkerNoise, ClipNoise, LFNoise0, LFNoise1, LFNoise2, LFClipNoise and Dust. FM was only possible for the last five of these. The percussive sound generator gave a good space of sounds to explore, with much of the power coming from the Klank module. Different resonant recipes gave different body sounds to the percussion, varying from bells and metallophones to snares and industrial noises.

No sane human being would search through all the 221 dimensions of the model blindly hoping to find some optimal sound! The use of some guidance principle like a genetic algorithm is therefore critical to the development of finished synthesis patches for this parameter space. Again, cross-breeding of percussion sounds had a lot to offer, with the capacity to find children sharing useful attributes of both parents, and home in on the most interesting characteristics in candidate sounds.

2.4. Analytical solution of the plucked stiff string

Rossing and Fletcher (1995: 61) give an analytical solution of the plucked stiff string including energy loss due to air, bridge and internal damping (Rossing and Fletcher 1995: 50). The stiff string no longer has a strictly harmonic overtone recipe, with intervals becoming increasingly stretched between higher modes. To investigate the model via synthesis, the parameter set necessary to drive the various equations was templated. Important single parameters included those given in table 3. The AM, FM and resonance model from the previous experiment was later added on top of this basic set to try to give some extra timbral activity to the solutions. The permissible ranges used for the percussion were found to be very excessive in this context and were made far subtler.

Table 3. Stiff string parameter space.

Parameter name	Valid range
Young's modulus	2e4–2e11 (exponential)
Density	100–7,800 (exponential)
Tension	1e3–1e5 (exponential)
Length	0.1–0.2 (exponential)
Radius (assuming circular cross section)	0.001–0.1 (exponential)
Pluck point (assuming linear sharp pluck shape)	0.01–0.99

The model still remains sinusoidal, and of fixed pluck envelope, so further work would concentrate on adding some noise sources and slight individual enveloping for all partials. The value of this experiment was in giving a good sense of the possibilities of the acoustic equations. Plain sounds of conventional strings and metallophones were readily obtainable. The bass string sounds in particular were rich and engaging. Without some systematic method of sampling the parameter space, the author would never have seen the true scope of the equations in sound synthesis, but would perhaps have dispensed with them too quickly to reveal their potential. The exploration of the argument space of analytical solutions provides a fertile ground for the mathematically inclined sound designer.

3. THE GAPARAMS LIBRARY

For an environment for varied interactive evolution experiments, SuperCollider 2 (McCartney 1998) was a natural choice, not least because of the combination of real-time synthesis and UI facilities. After initial attempts at interactive evolution it was found propitious to factor out reusable code into a class library. The payoff was the development of a general framework for interactive evolution experimentation, broad enough to support all the work in this paper and further projects besides.

The most basic unit of genetic data in the library is a floating point valued parameter, and template and instantiation classes handle the schema and the actual instance of such primitives. Singular parameters as well as parameter arrays are supported, with linear or exponential parameter ranges. User interface and chromosome base classes use this 'GAParams' paradigm. With the library code in place, each project derives a specific parameter set class from the base chromosome class. The UI master object can be simply invoked with the type of this derived class and the interactive evolution system runs immediately.

Figure 3 shows the user interface facilities provided by the library code. Save/load and creation of a finished patch allow a genome bank to be created over multiple sessions, and immediately run with a finalised parameter set. Many types of 'new generation' are available, based on best two parents, weighted parent choice algorithms, asexual reproduction and the technique of interpolation, recommended by Dahlstedt despite its non-GA basis. The emphasis is on fast access to assessment, hence individual play buttons for the eight candidates in the current generation. The library actually supports an arbitrary number of candidates per generation, though eight was found to be a useful default. There are class-specific windows for derived chromosome classes and for global parameters affecting all playback; reverb time stretching was mentioned above as one such case. There is a mutation chance window covering all parameter groups in the parameter set template, allowing one to freeze the mutation of a given parameter or increase the likelihood of its being nudged in a new generation. Edit windows allow modification to the level of any individual parameter. In the figure, you can also see SuperCollider's dualScope Synth method, used to give access to spectral and time domain views of the output.

The parameter space for a new experiment is lightly described in a single class written by the experimenter for a specific evolutionary role. The extent of the parameter space is mapped out, and the nature of any variation due to mutation allocated. The author also writes a synthesis function so as to hear back a current candidate parameter set for the algorithm being investigated.

The library's paradigm was often challenged but the design succeeded in supporting the varied experiments described above. The greatest trick to this generality was in setting a maximum initial size to the parameter spaces, that is, the library supports static allocated arrays but not dynamic any-size parameter spaces. Since infinitely expandable parameter spaces are somewhat intractable to search, this is no great loss. It was particularly gratifying to create the N stage delay networks comfortably within the library's paradigm.

4. CONCLUSIONS

The experiments have been encouraging. The fitness bottleneck feared by some authors was not found to be a great problem in the restricted musical applications attempted here, though perhaps with some reservations for trying out evolved MIDI instruments. Few problems were found with the use of a standard genetic algorithm for these limited situations. Coevolution (used in musical applications in Todd and Werner 1999, Dahlstedt and Nordahl 2001) is not currently supported in the GAParams Library, but since previous research has demonstrated its importance for the evolution of more complex entities, there will be a limit to the capabilities of this system without it.

Within the restrictions of float parameter genes this interactive evolution process gave exciting results. Convergence of chromosomes tended to happen relatively fast. It was great for trying out loose parameter spaces where specific engineering knowledge was absent. Parameter spaces have associated psychoacoustic sensation spaces, which means that there is a danger that some of the mutation changes are not perceivable to the ear. But where it is a long task to engineer useful parameter values, interactive genetic algorithms come into their own. Making and modifying experimental set-ups became a fast process using the GAParams library, and so sensation space redundancy was not too much trouble.

One facet of this paper is that random generation of

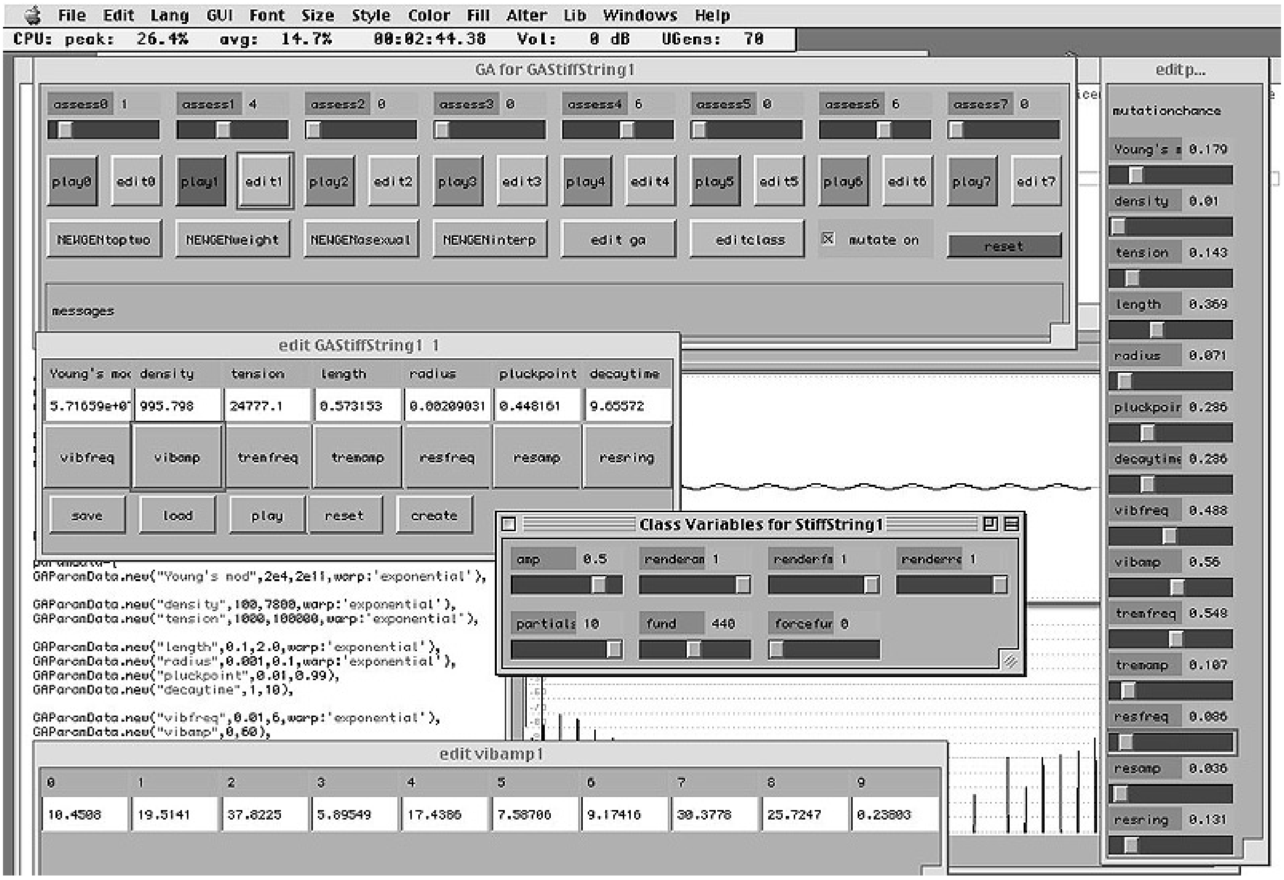


Figure 3. User interface for the GAParams library.

sound without acoustical knowledge built in is often very disappointing, but with exploration guided by a genetic algorithm, it may have a little more scope. The benefits gained from synthesis algorithms were heightened by having an interactive genetic algorithm to explore their potential.

The library of common code for the project was invaluable. With issues raised by the experiments, the GAParams Library has grown in power and has now been released to the SuperCollider community. The library is available from the author's website given above, and includes all of the experimental set-ups from this paper. A further paper using the GAParams code (Collins 2002) in combination with another library for audio cutting develops the work in this paper into the domain of algorithmic composition. Whilst this article might describe a success story of dividends from IGA sound design, the picture there is qualified by the greater problems of evolving arguments to generative algorithms. There are surely many pitfalls waiting as we attempt to effectively explore more and more complex parameter spaces. Yet not to end on a discouraging note, a great deal of future work could be carried out in the library, and the reader is encouraged to write a short SuperCollider class, and get exploring!

REFERENCES

Biles, J. A., Anderson, P. G., and Loggi, L.W. 1996. Neural network fitness functions for a musical IGA. In *Proc. of the Int. ICSC Symp. on Intelligent Industrial Automation (IIA'96) and Soft Computing (SOCO'96)*, pp. B39–44, March 26–28. Reading, UK: ICSC Academic Press.

Chapuis, J., and Lutton, E. 2001. ArtiE-Fract: interactive evolution of fractals. In *Proc. of Generative Art*. Milan, 2001.

Collins, N. 2002. Interactive evolution of breakbeat cut sequences. In *Proc. of Cybersonica*, the ICA, London. Also forthcoming in the *Digital Creativity Journal*. Available as a PDF from www.axp.mdx.ac.uk/~nicholas15/publications.htm

Dahlstedt, P. 2001. Creating and exploring huge parameter spaces: interactive evolution as a tool for sound generation. In *Proc. of the Int. Computer Music Conf.*, Habana, Cuba.

Dahlstedt, P., and Nordahl, M. G. 2001. Living melodies: coevolution of sonic communication. *Leonardo* **34**(3).

Fujinaga, I., and Vantomme, J. 1994. Genetic algorithms as a method for granular synthesis regulation. In *Proc. of the Int. Computer Music Conf.*, DIEM, Denmark.

Jacob, B. L. 1995. Composing with genetic algorithms. In *Proc. of the Int. Computer Music Conf.*, Banff, Alberta.

Johnson, C. G. 1999. Exploring the sound-space of synthesis algorithms using interactive genetic algorithms. In A. Patrizio, G. A. Wiggins and H. Pain (eds.) *Proc. of the AISB'99 Symp. on Musical Creativity*, pp 20–7. Brighton, April.

McCartney, J. 1998. Continued evolution of the SuperCollider real time synthesis environment. In *Proc. of the Int. Computer Music Conf.*, Ann Arbor, Michigan.

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

Rossing, T. D., and Fletcher, N. H. 1995. *Vibration and Sound*. Springer-Verlag.

Todd, P. M., and Werner, G. M. 1999. Frankensteinian methods for evolutionary music composition. In N. Griffith and P. M. Todd (eds.) *Musical Networks*. MIT Press.

Yuen, J., and Horner, A. 1997. Hybrid sampling-wavetable synthesis with genetic algorithms. *Journal of the Audio Engineering Society* **45**(5): 316–30.