

# Genophone: An Evolutionary Approach to Sound Synthesis and Performance

James Mandelis

[jamesm@cogs.susx.ac.uk](mailto:jamesm@cogs.susx.ac.uk)

*School of Cognitive and Computing Sciences, University of Sussex, Brighton, United Kingdom*

Genophone is a hyperinstrument implementing an Artificial Life paradigm for the creation of new sounds and the way they change during performance. The project addresses the problem of required intimate knowledge of sound synthesis techniques in order to design sounds that are novel and of musical interest, and suggests the "selective breeding" paradigm as a solution. This approach can be used successfully for various different synthesis techniques. The level of abstraction of the parametric space used also facilitates the utilisation of different external synthesisers via System Exclusive MIDI messages.

## 1. Introduction

This paper describes the Genophone, a hyperinstrument developed for sound synthesis and sound performance using the evolutionary paradigm as the driving process. Sound design on most current commercial systems relies heavily on the intimate knowledge of the sound synthesis technique employed by the sound generator (hardware or software based). This intimate knowledge can only be achieved by investing long periods of time playing around with sounds and experimenting with how parameters change the nature of the sounds produced. Often such experience can be gained after years of interaction with one particular synthesis technique. The program presented here attempts to aid the user in designing sounds without the necessity of deep knowledge of the synthesis techniques involved. This method of design is inspired by evolutionary techniques, where the user expresses how much particular sounds are liked and then uses these sounds through mutation and genetic recombination (Goldberg 1989) to create new ones. The aim of the program is to encourage creation of novel sounds rather than designing sounds that satisfy specific a priori criteria.

Through the use of "locked" parameter sets (where their values are unchangeable), variable control is exercised on the non-deterministic effect of the evolutionary processes. This feature, on the one hand, exercises some control on the shape evolution takes and on the other, allows a gradual familiarisation with the synthesis technique involved (if desired).

Genophone is a "Hyperinstrument" or "Virtual Musical Instrument" (Machover & Chung 1989; Pressing 1990; Mulder, Fels & Mase 1997; Mulder 1994), comprising a dataglove, synthesiser and a PC that runs the evolutionary software. Real-time information from the glove is used to manipulate parameters that affect the sounds produced. The mapping of the finger flex and the sound changes is one-to-many, that is a single finger flex (one of five) can control multiple parameters. This problem of mapping lower dimensionality performance controllers to higher dimensionality parameters (Pressing 1990) is also tackled within the same evolutionary framework. The resulting mappings are mainly used during performance by changing sound characteristics in real-time.

The selective breeding process generates System Exclusive MIDI messages (SysX) for sound definitions, which are then sent to the synthesiser to be rendered. This level of abstraction facilitates the use of different external synthesisers with minimal effort. It also taps into the ability of commercial synthesisers to produce musical sounds by their design, and the existing wealth of sounds available for them. In previous attempts of using AL for sound synthesis a lower level definition was used, therefore limiting the usage of the system to one particular piece of hardware (or software emulation) that employed only a single synthesis technique. Also musical sounds are much harder to evolve when lower level definitions are used (Yee-King 2000; Woolf 1999).

This project has also shown that evolutionary methods can be used successfully on several sound-synthesis-techniques, demonstrating the feasibility of a generic approach in sound design for all different sound-synthesis-techniques. This approach is fast; often just a few generations are needed for evolving sounds that are interesting and of good quality. It is also very easy and fun to use, as well as being easy to learn.

## 2. Setup Description

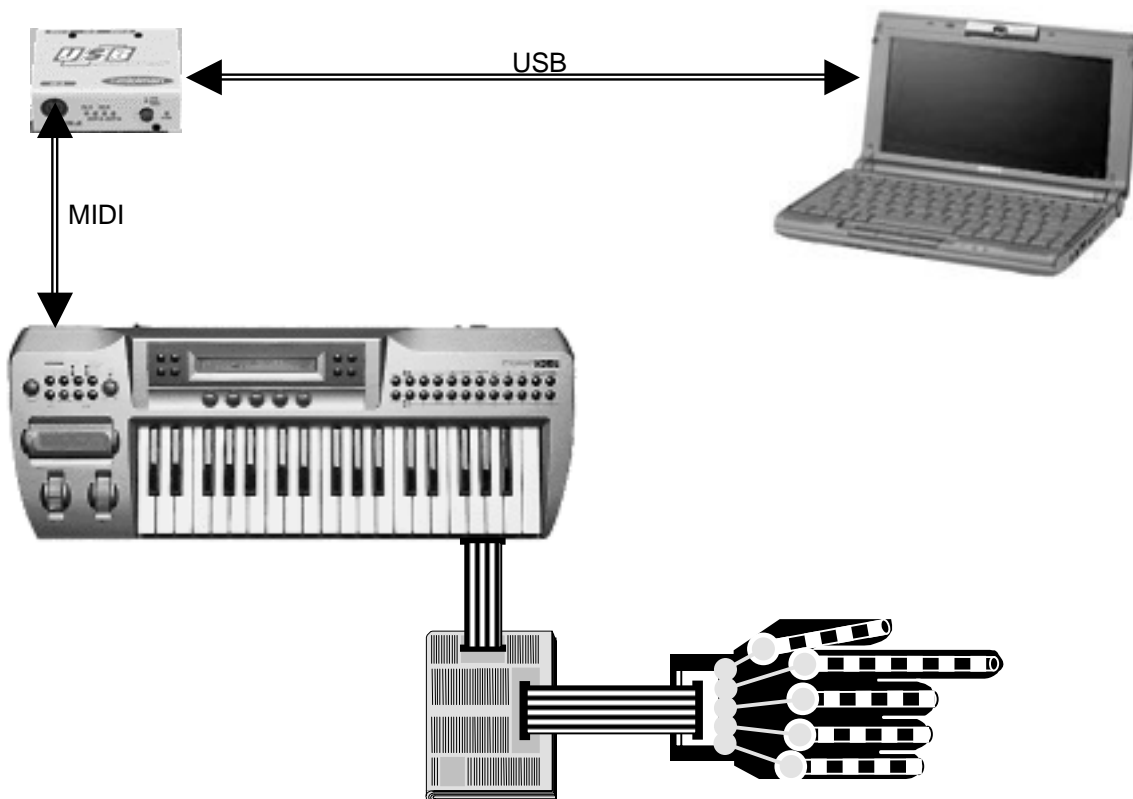
This section describes the physical setup and hardware connectivity of the system.

### 2.1. Parts

- A *KORG Prophecy* solo-synthesiser, which produces the actual sound.
- A PC with MIDI interface. Used to run the evolutionary software.
- A glove with five flex-sensors for each finger. It is mainly used as a performance orientated device, but also for navigation phenotypic space.
- An interface board for signal conditioning. It translates the finger flex of the glove (expressed as variable resistance) to five voltage signals 0-5V appropriate for internal use on this specific synthesiser.

### 2.2. Connectivity

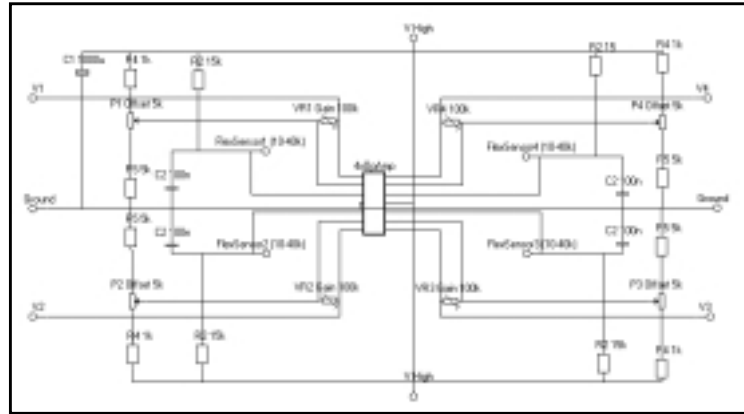
- The PC is connected to the Synthesizer via MIDI. If the PC is equipped with a MIDI interface then it can be connected directly to the synthesizer. In this case a USB to MIDI interfaced has been used with the specific notebook. The MIDI connection is only necessary when new sound definitions need to be exchanged between these two units. After that, the Synthesizer can be used for performance without the use of the PC connection.
- The glove is connected to the interface via a 6-wire cable. One +5V and five returns.
- The interface board is connected directly to the synthesizer's five input knobs (performance knobs) outputs. Allowing finger flex positions to take the place of knob rotational positions.
- The synthesizer is connected to an external amplifier or headphones.



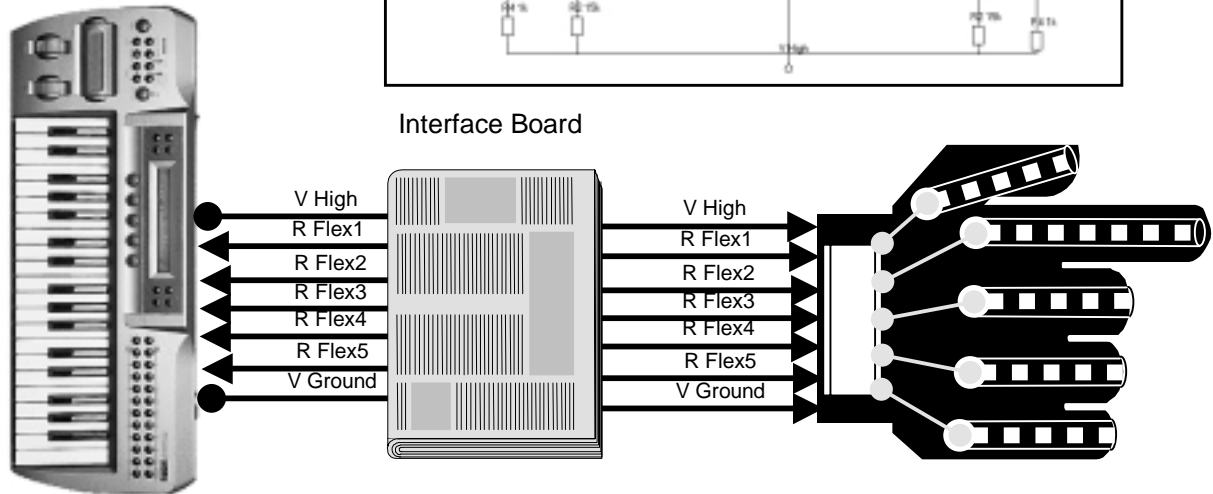
### 3. Hardware Description

#### 3.1. Interface Board

The interface board is constructed on a strip-board, it has eight operational amplifiers (only five are used), in the form of two National Semiconductor Rail-to-Rail 4 x op-amp ICs (LMC6484). There are also capacitors for noise filtering and several resistors. The following schematic describes half the board (only 4 op-amps, one IC), the rest of the board is duplicated. Terminals; to V5, V High and Ground are connected to the synthesiser's internal PCB board. Terminals; Sensor 1 to 5, V High and Ground are connected to the glove's flex sensors.

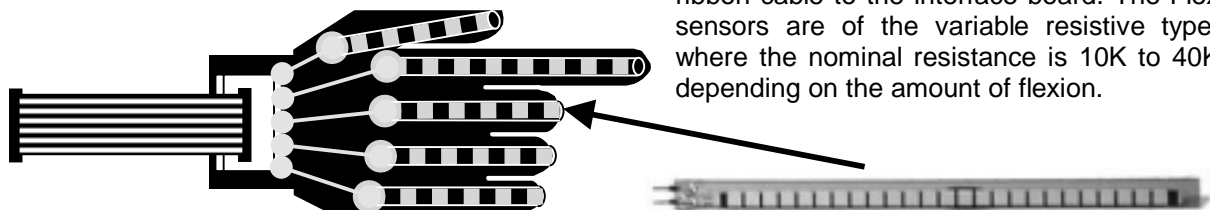


V1  
Flex



#### 3.2. Glove

The glove has flex-sensors attached to each finger. The sensors are enclosed in flexible tubing to allow longitudinal movement of the sensor when the finger is flexed, but restricting movement in any other direction. One end of each sensor is fixed on the tube, but the other end can slide freely within the tube. The fixed end is plugged on to a micro-socket at the end of a wire that connects it to a small distributor attached on the wrist of the glove. On the distributor there is a socket for connecting a ribbon cable to the interface board. The Flex sensors are of the variable resistive type, where the nominal resistance is 10K to 40K depending on the amount of flexion.



#### 3.3. Synthesiser

The synthesiser used at this stage of the project is a *KORG Prophecy* solo-synthesiser. This particular synthesiser has been chosen because it supports seven different sound synthesis techniques. The available multitude of synthesis techniques is used to show the suitability of this type of interface to a variety of sound synthesis paradigms. The sounds are modelled by 200+ parameters and since the synthesiser is geared towards performance, its internal architecture allows for real-time manipulation of those parameters, via five "performance knobs". The knobs can be assigned to control up to four parameters each. Also the way those parameters are changed by the knob's rotation can be specified i.e. the lowest and highest value, the change function (linear, exponential, logarithmic).

# Genophone: An Evolutionary Approach to Sound Synthesis and Performance

## Main Features of the Prophecy

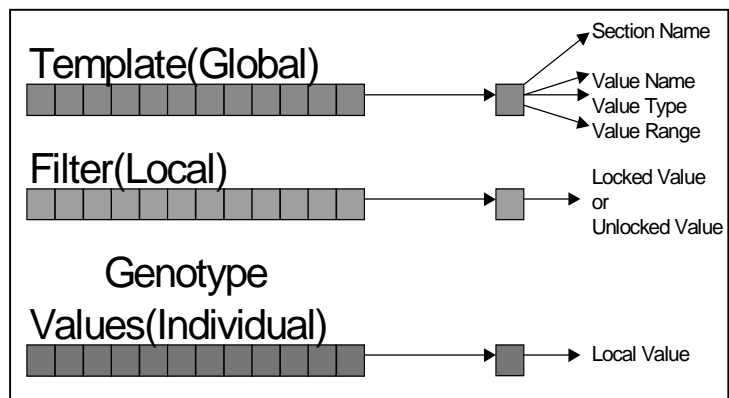
- Seven different synthesis techniques are supported, ranging from analog synthesiser oscillators to physical models such as sax or bass guitar.
- The Oscillator block provides seven types of oscillators, such as Analog, VPM, and Physical Modelling, and also contains a Sub Oscillator and Noise Generator. The Wave Shape block can be set to either Clip or Resonant wave shaping, and determines how the waveform is shaped and the balance at which it is mixed with the original waveform. The Mixer Block determines the levels at which the two systems of Oscillator, Sub Oscillator, Noise Generator, and Feedback are sent to the Filter block. The Filter block provides two multi-mode filters (switchable between LPF/HPF/BPF/BRF), and can be placed in either series or parallel to control the output. The Amp block lets you independently control the level of each output signal. The Effect block provides seven types of effect; Distortion, Wah, Chorus/Flanger+Delay, Reverb, and Dual Parametric EQ. (You may select either Chorus/Flanger+Delay or Reverb.)
- The Performance Editor function lets you assign parameters to each of the five knobs for real-time control. Four Performance Editor sets are provided. For each set, any of the more than 200 program parameters can be assigned to a knob, meaning that up to 4 parameters can be assigned for control by one knob. Performance Editor settings are also stored independently for each program.

## 4. Software Description

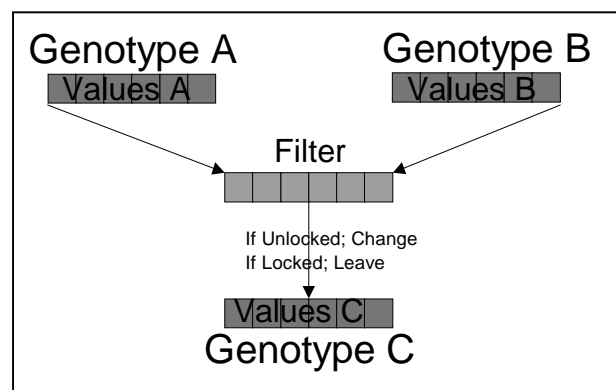
### 4.1. Overview

There are three main structures used in this program;

- The Instrument Template; describes the parameters used in making up a patch in a particular instrument (in this case the Prophecy). It defines things like SysX headers, parameter names, type of values they take, and range of values. It also defines sections of parameters (and sections of sections) that reflect the logical design of the instrument. The state of parameters “Locked” or “Unlocked” is also stored here temporarily as well as the current parameter values (not persistent).
- The Values (Genotype); is a particular set of values for the parameters defined above. I.e. of the required type in the required range. Each Genotype translates to a SysX message that defines a sound.
- The Filter; is a list of “Unlocked” parameters. When a filter is applied to the current patch it has the effect of “Unlocking” any “Locked” parameters.

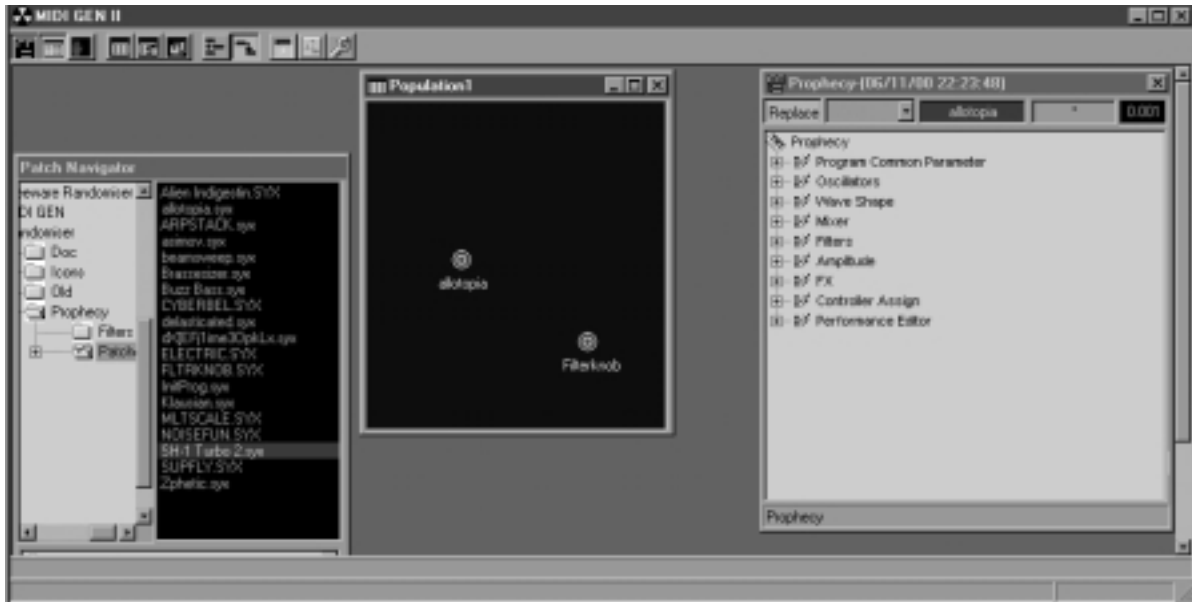


- The state of parameters (“Locked” or “Unlocked”) is used for limiting the effects of, loading new patches, mutation, and recombination. Filters are used to fill the gap between total knowledge of the synthesis technique (knowing how each parameter changes the sound) and total ignorance of the principles underlying the sound production. By enabling or disabling sets of parameters is possible to influence just parts of the instrument’s logical structure i.e. the effects section, the mixer section or the oscillator section etc.



# Genophone: An Evolutionary Approach to Sound Synthesis and Performance

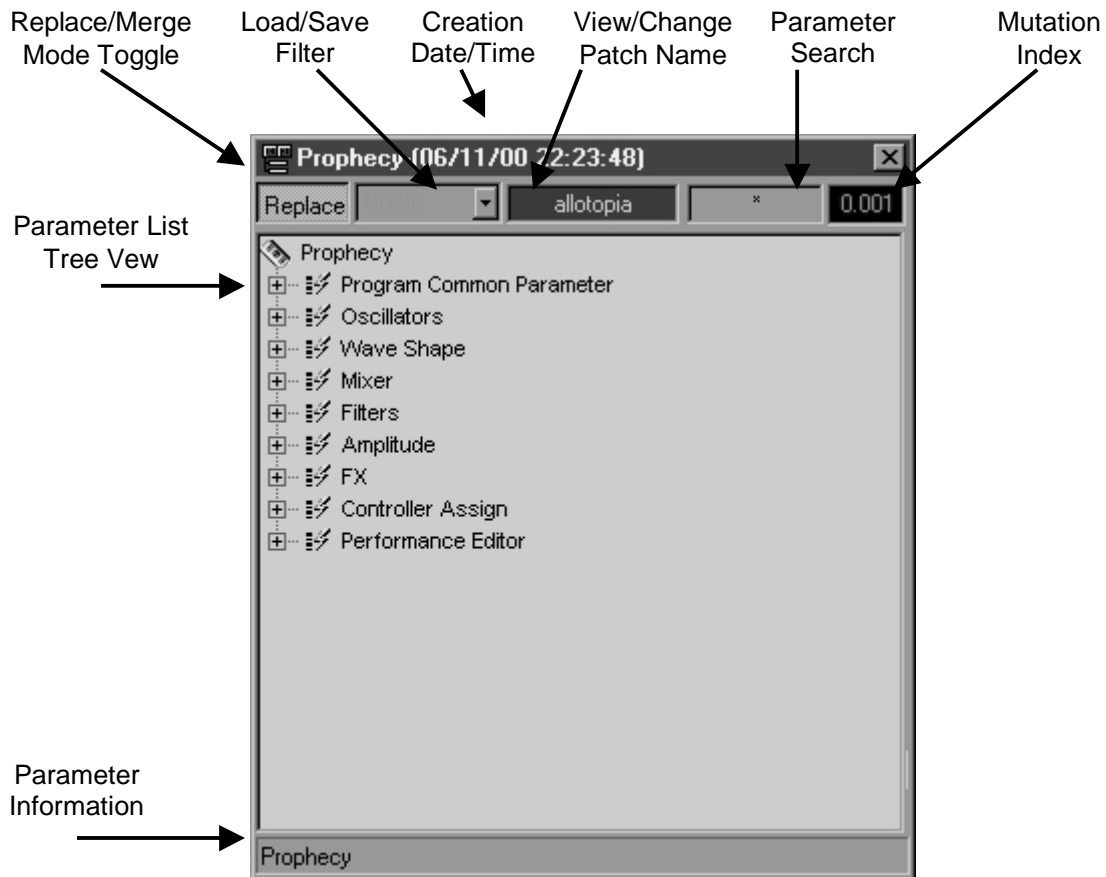
The software written for this project is a multiple-document interface (MDI) application. It contains several tools/components for manipulating and viewing sound parameters. The application uses extensively drag-and-drop functionality for most operations.



A description of the components follows;

## 4.2. List View

The most important component is List View; its main function is to display the current patch's parameters in a tree view form, lock/unlock parameters and load filters.

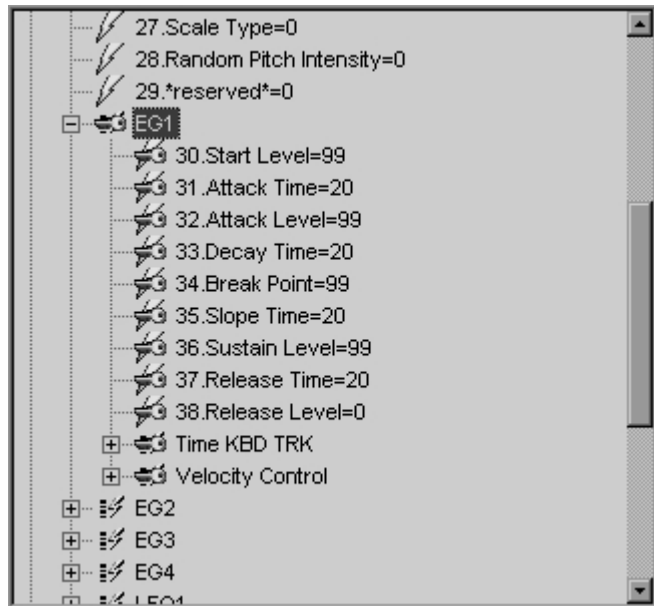


# Genophone: An Evolutionary Approach to Sound Synthesis and Performance

The patch files can be dragged-and-dropped into this component, which will display the new parameters, upload the patch to the synthesiser (optional) and play a note (optional) in order to preview it. The parameters are aggregated into sections (and sections of sections) that reflect the logical structure of the synthesiser.

Parameters and sections can be in two states, "Locked" and "Unlocked", these states can be applied recursively on sections. When a parameter is "Unlocked" its value can be changed freely, when "Locked" its value is frozen. Their icon in the List View indicates the state of parameters, and sections.

A collection of "Unlocked" parameters (only their state not their value) is called a "Filter". Filters allow parts of the patch to be frozen and parts to change freely, filters can be saved and loaded. Each time a Lock or Unlock operation takes place a new filter is created, which can be saved for later use.



The "Replace" option button, when pressed it changes to "Merge" and back. When a new patch is loaded the default behaviour ("Replace" mode) is to load the new patch afresh ignoring any current loaded filter and overwriting any existing values. In "Merge" mode, the values of the existing patch are overlaid with the values of the new patch, but only unlocked parameters defined by the current filter, the rest remain as they were.



Also in List View it is possible to use a context menu (right click) for extra functions;

**Lock** – Locks/Unlocks (recursively) parameters and sections.

**SysX** →

**Load** – Load a new patch from file.

**Save** – Save current patch in a file.

**Download** – Download patch from synthesiser and make it current.

**Upload** – Upload current patch to synthesiser.

**MIDISetup** – Enter MIDI set-up dialog.

**Undo** – Undo previous change, send or patch upload.

**Randomise** – Change the parameter(s) value to a random one between the Minimum & Maximum values.

**Send** – Upload this parameter's value.

**Edit** – Edit a parameter's value.

**Auto Send** – (Option) Send individual parameter changes as they happen.

**Auto Upload** – (Option) Upload new patches (whole patch), after change or load.

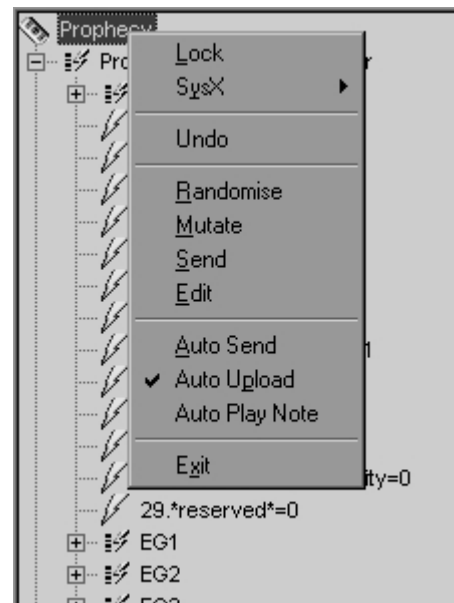
**Auto Play Note** – (Option) Play a note after the uploading a new patch.

**Mutate** – Change the parameter(s) value according to the following formula;

$$X = \text{Mutation Index} * ((\text{Max} - \text{Min} + 1) * \text{Rnd1} + \text{Min})$$

$$\text{New Value} = \text{CurrentValue} + \text{Int}((X + X + 1) * \text{Rnd2} - X)$$

where Rnds are random numbers between 0 & 1

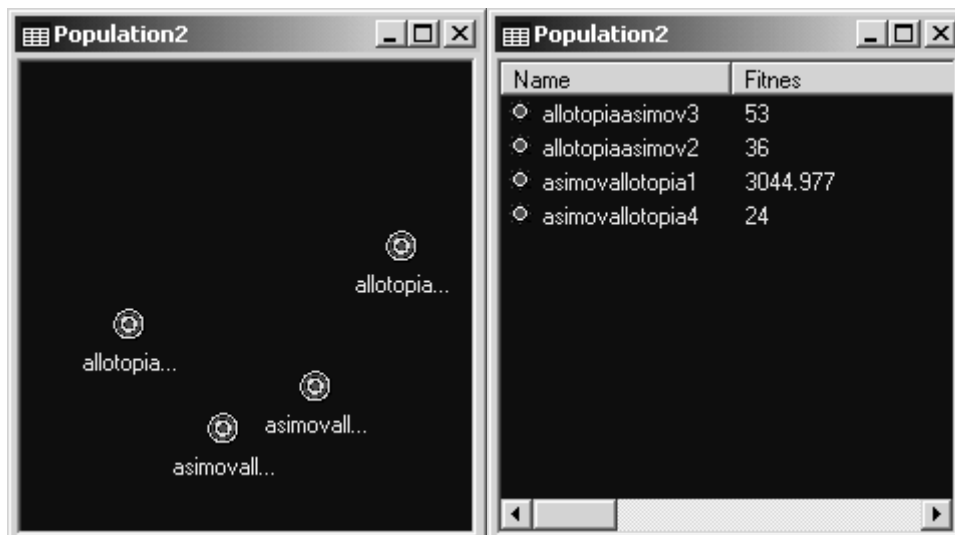


# Genophone: An Evolutionary Approach to Sound Synthesis and Performance

## 4.3. Population Window

Population windows are containers for groups of patches they support the following functionality.

- Files can be dropped into them from the Patch Navigator, List View and other Population windows.
- Pressing Delete, will remove individuals from the population.
- Double-clicking (or pressing Enter) on a selected individual loads it to the List View in “Replace” mode (which is then loaded to the synthesiser if Auto Upload is on).
- Dragging and dropping an individual from a Population to List View will take account the filter mode, if in “Merge” mode, the current filter will be used to load the new values, so only “Unlocked” parameters will be changed.
- Multiple individuals can be selected either by Shift-Click or using the Lasso, as a group they can be deleted, dragged & dropped, or used as parents for reproduction.
- For each individual in the population, its fitness value is derived from its relative height within the window (vertical axis position). This way the user can show relative preference of patches by positioning the more liked ones closer to the top.
- Population windows can have two views. Double-Clicking on the background cycles through the views;



## 4.4. Mutation

The mutation button creates a new population of five mutants based on a previously selected individual;



# Genophone: An Evolutionary Approach to Sound Synthesis and Performance

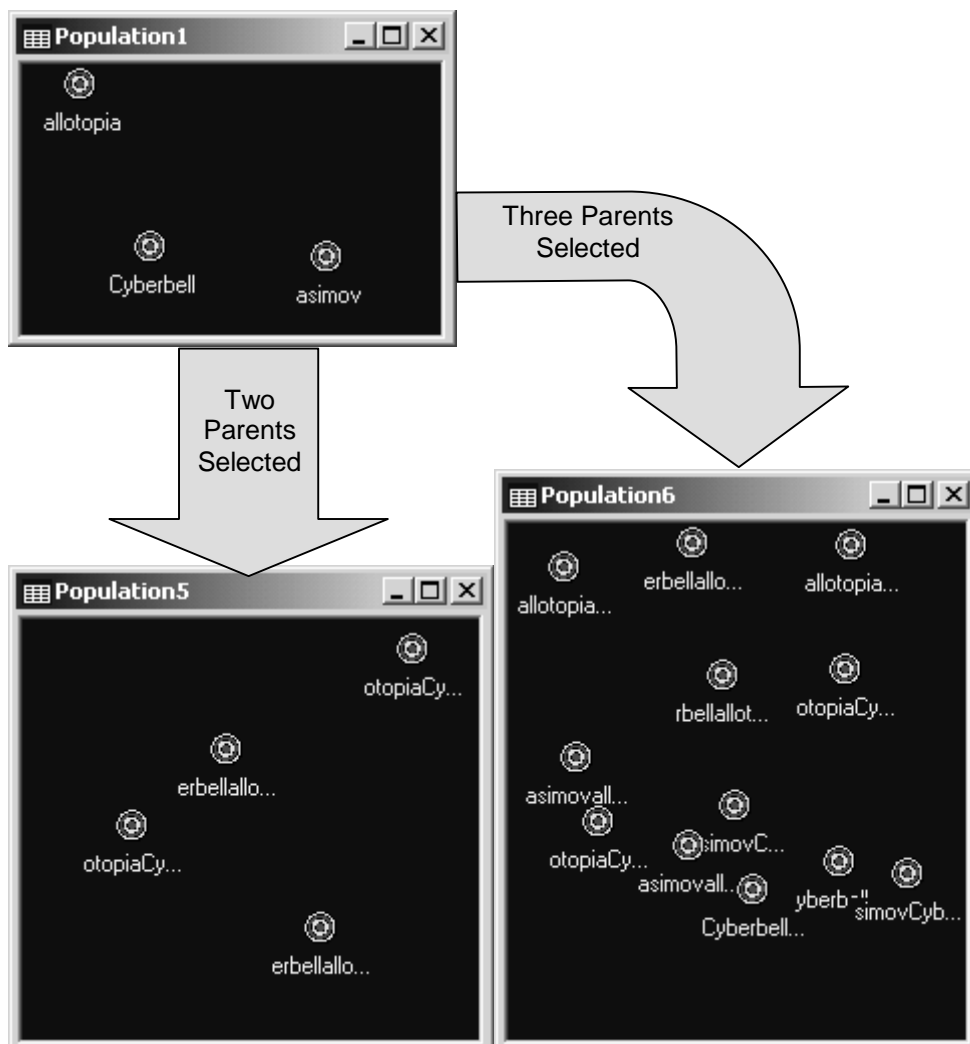
The user previews, and then repositions the mutants according to how pleasing they are.

Only the “Unlocked” parameters are mutated, “Locked” parameters just get copied across from the original patch. Each mutant parameter value is produced by the following formula;  
$$X = \text{Mutation Index} * ((\text{Max} - \text{Min} + 1) * \text{Rnd1} + \text{Min})$$
$$\text{New Value} = \text{CurrentValue} + \text{Int}((X + X + 1) * \text{Rnd2} - X)$$
where Rnds are random numbers between 0 & 1



## 4.5. Reproduction

When a number of individuals are selected, the “Reproduce” button creates a new population made up of two offspring from each possible combination of parent couples.



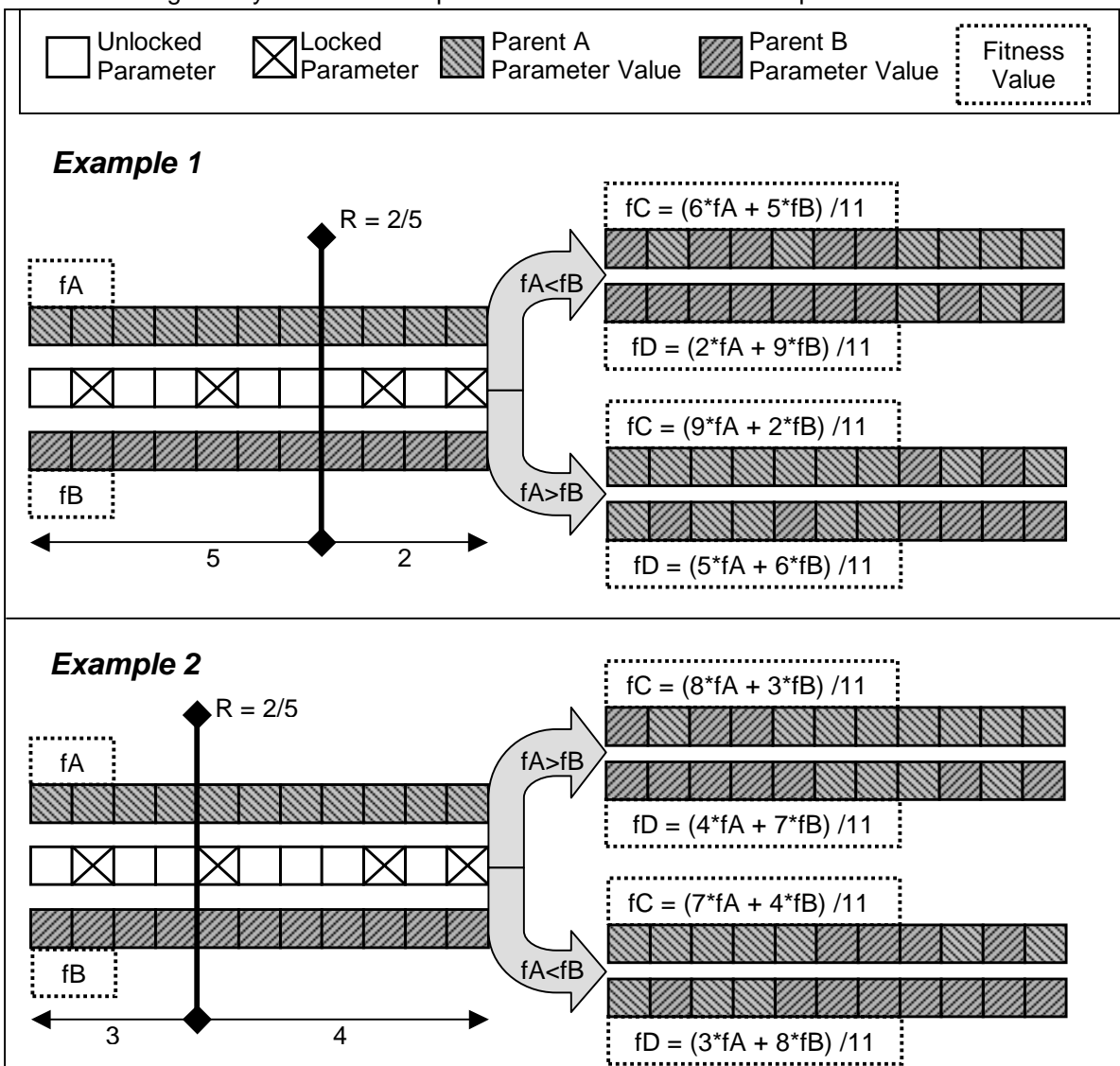
### 4.5.1. Reproduction Types

There are three reproduction types available implementing different recombination techniques (description follows). In the course of experiments certain idiosyncrasies became apparent.

- Type B seemed more likely to disrupt clusters of related parameters with the resulting sounds been more likely to be unfit (no sound produced at all, or was too alien from the parents to be usable), sometimes this could be considered an advantage since the resulting sounds though alien where quite attractive. One could say that results from this recombination type are more radical.
- Type C also seemed likely to produce unsuitable sounds for different reasons, by using “in between” values quite often the sounds were indistinct, quiet or just silent. One could say that this recombination type is too conservative or uncommitted in its results.
- Type A is the one used most, it preserves clusters of parameters and their values, and its results are a bit more predictable.

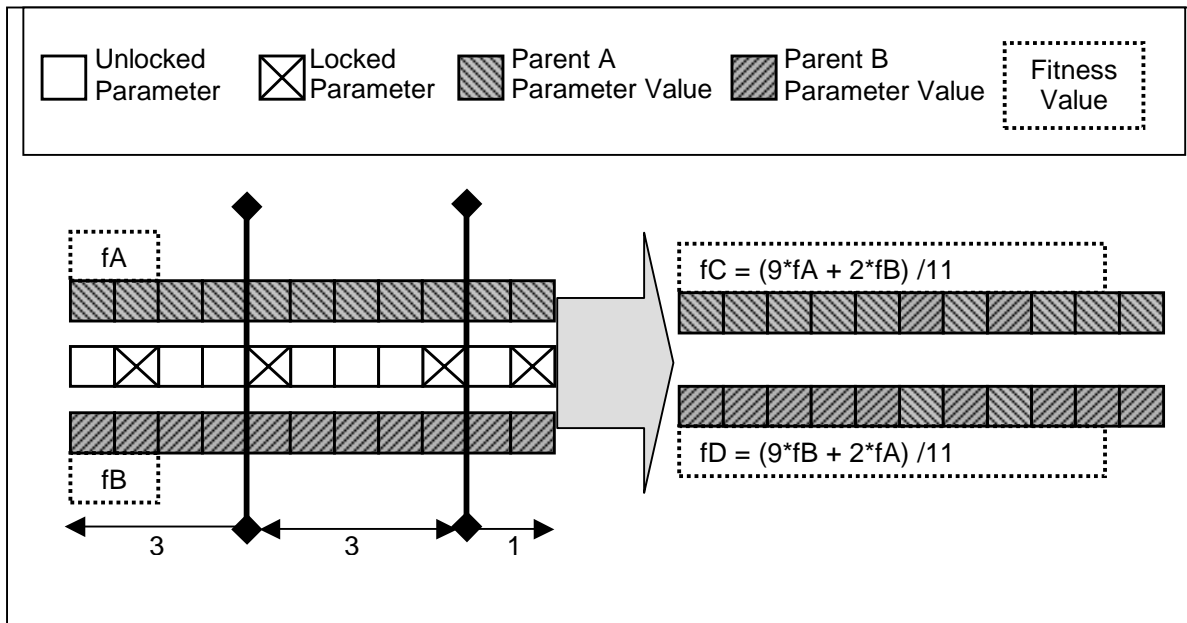
### 4.5.2. Reproduction Type A

For each couple, a random crossover point is selected in the range of 1 to the number of “Unlocked variables”. Originally two offspring are produced as parent clones (one of each parent), then their unlocked parameters are overwritten by the other parent’s, but only those defined within the range of 1 to the Cross-Over-Point for one, and from the Cross-Over-Point to the number of “Unlocked variables”, for the other. Which parent is used first, in this parameter copying process, depends on whether the Cross-Over-Point is before or after the middle of the unlocked variables range and by which parent has the larger fitness. Also each offspring is assigned a fitness derived from its parent’s fitness’s weighted by the number of parameters inherited from each parent.



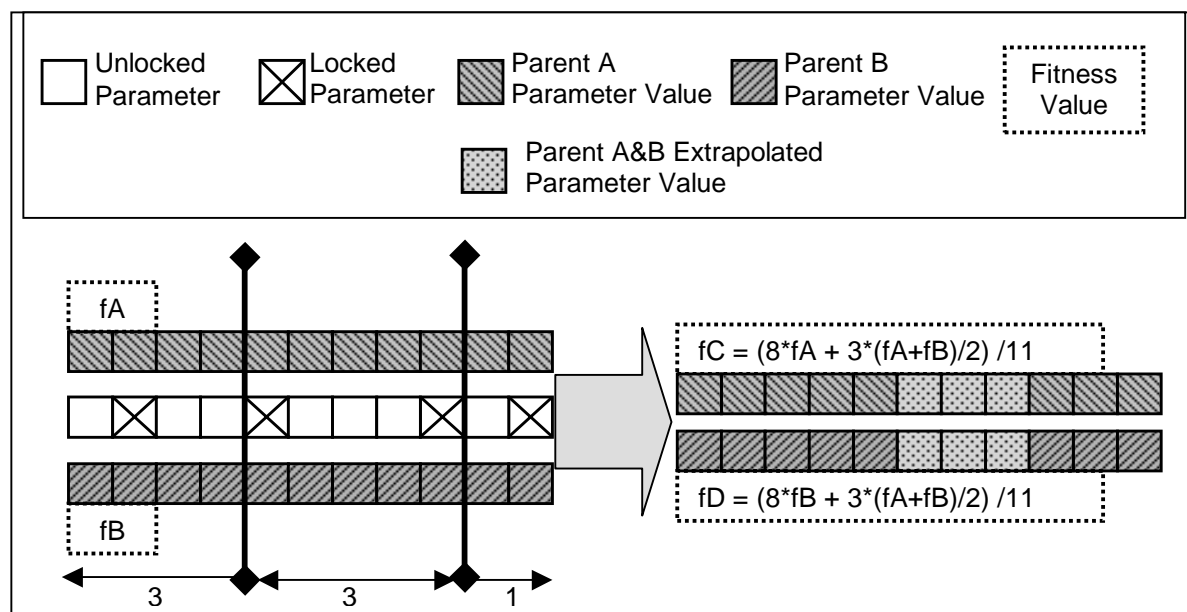
### 4.5.3. Reproduction Type B

For each couple, two random crossover points are selected in the range of 1 to the number of “Unlocked variables”. Originally two offspring are produced as parent clones (one of each parent), then their unlocked parameters are overwritten, but only those defined within the range between the Cross-Over-Points. The parameter value used for overwriting is selected probabilistically from the parents, where the probability is proportional to the fitness value of the parents. Also each offspring is assigned a fitness derived from its parent’s fitness’s weighted by the number of parameters inherited from each parent.



### 4.5.4. Reproduction Type C

For each couple, two random crossover points are selected in the range of 1 to the number of “Unlocked variables”. Originally two offspring are produced as parent clones (one of each parent), then their unlocked parameters are overwritten, but only those defined within the range between the Cross-Over-Points. The parameter value used for overwriting is derived between the parents’ values weighted by their fitness. Consequently, in the two offspring, the region between the Cross-Over-Points is identical, the rest of the genotype is identical to one parent for one offspring and identical to the other parent for the other. Also each offspring is assigned a fitness derived from its parent’s fitness’s weighted by the number of parameters inherited from each parent.



## 5. Experiments

This section describes a set of experiments performed on the above system in order to ascertain its efficacy in designing novel sounds. A small number of users were involved and their reactions and comments, as well as their derived sounds were noted. Some users had some experience in sound synthesis and some none; also some users had extensive experience with music as musicians, DJs etc, and most had a rather eclectic (and demanding) sonic taste. The experiments were performed in two flavors, in the first, a single note was used to preview the generated sound, on the second, loops of short musical phrases (arpeggio patterns), were used.

### 5.1. Experiment 1

In this experiment the users were simply asked to use the dataglove to modify the evolved sounds. A loop of notes was played and the users were able to transpose the sequence on the keyboard (arpeggios) and change the sound characteristics via the glove as they performed. The general impression and comments were noted during the sessions.

The aims of this experiment were to test;

- The responsiveness of the setup.
- The expressiveness of the setup.
- How useful and/or interesting was perceived to be.
- If it was easy to “play”.

The users commented on the following;

The response was instantaneous, that is there was no perceived delay between motion and change in sound.

It usually took a couple of minutes before they were able to play a dynamic changing sound. Often it helped if there were no visual cues that is if they looked away from their moving hand and Prophecy's display (where value bars are shown). Somehow concentrating on the sound and the kinesthetic feeling of the finger positions facilitated the internalization of the motion-to-sound mapping and allow them to “play” the glove, creating dynamic sounds with changing rhythms. Another interesting observation was that explaining before hand how the glove changed the sound did not help using it. It was more productive if users were left to discover it themselves. There is definitely an element of direct experience that is hard to communicate verbally but is so blatantly obvious when one had the chance to explore how the changes in finger position changed the sound characteristics.

The users must have enjoyed playing with the Genophone because they either wanted to play more or they wanted one for themselves.

### 5.2. Experiment 2

In this experiment a factory-preset patch from the Prophecy was used as a seed. Successive generations were created by mutation only (no recombination), until the user arrived at a sound that considered aesthetically pleasing enough to use in a musical composition or as a sound effect. The mutation factor was set for each generation.

The aims of this experiment were to test;

- The usability of the interface.
- The effectiveness of the mutation operator in creating new sounds.
- Effect of the mutation factor and observe its general behavior.

After several sessions with different users the following observations were made;

The results were in general pleasing to the users and well received.

Mutation as a concept was easily understood and grasped by the users.

The value of the mutation factor was somewhat difficult to predict beforehand. Often it was too high (producing unfit sounds) or too low (producing uninteresting sounds). One would have to go back to the parent sound and spawn a new generation with a more appropriate mutation factor value. Also the mutation factor's effect varied for different seed sounds, though this did not come as a surprise considering the different synthesis techniques used and their inherent non-linearity.

## 5.3. Experiment 3

In this experiment a small number of factory-preset patches from the Prophecy were used for breeding through reproduction (recombination). Successive generations were created by reproduction only (no mutation), until the user arrived at a sound that considered aesthetically pleasing enough to use in a musical composition or as a sound effect. The relative fitness of each offspring was indicated by the user via the vertical placement of the sounds within the “population container”. All three types of recombination in reproduction were used.

The aims of this experiment were to test;

- The usability of the interface.
- The effectiveness of the reproduction operators in creating new sounds.
- The effect of each type of recombination and observe its general behavior.

After several sessions with different users the following observations were made;

Reproduction as a concept was easily understood and grasped by the users, some commented that it reminded them selective breeding of plants or domestic animals.

The results were in general pleasing to the users and well received. Some commented on the ability of the operator to produce offspring that were quite different from the parents, yet retaining characteristics from both. Also sounds sounded more “interesting” compared to the sounds produced by the previous experiment (through mutation). The three different types of recombination were evaluated and exhibited the following characteristics. Although Type A was the one most used the others proved useful also, especially for injecting novelty into the strain.

- Type B seemed more likely to disrupt clusters of related parameters with the resulting sounds being more likely to be unfit (no sound produced at all, or were too alien from the parents to be usable). Sometimes this could be considered an advantage since the resulting sounds though alien were quite attractive. One could say that results from this recombination type are more radical.
- Type C also seemed likely to produce unsuitable sounds for different reasons, by using “in between” values. Quite often the sounds were indistinct, quiet or just silent. One could say that this recombination type is too conservative or uncommitted in its results.
- Type A was the one used most, as it preserves clusters of parameters and their values, and its results are a bit more predictable.

## 5.4. Experiment 4

This experiment was a synthesis of the previous two experiments. Users were given the same set of four factory-preset patches. They were free to use those patches for mutation and recombination in any way they deemed fit. Successive generations were created by reproduction and mutations, until the user arrived at a sound that considered aesthetically pleasing enough to use in a musical composition or as a sound effect. All three types of recombination in reproduction were used and various mutation factor values.

The aims of this experiment were to test;

- The usability of the interface.
- The effectiveness of combining different operators in creating new sounds.
- To show how individual aesthetic criteria derived diverse sounds.

The users commented on the following;

The “selective breeding” paradigm was quite intuitive and fun to use.

The interface was quite straightforward by extensive use of drag-and-drop.

The complexity and the subjective quality of the sounds were surprising, considering the non-stochastic process that produced the sounds.

The whole process was faster than anticipated. Often good usable sounds were produced within a few generations.

The previewing of sounds could be perceived as tedious especially if there was minimal variation.

Occasionally sounds were produced that were good but very quiet. Since that can be the result of many interrelated parameters the only way to remedy this was to “dive” into the individual parameters and reprogram the patch by hand. This was both time consuming and irrelevant to the general idea of non-stochastic processes producing the sounds explored in this project.

## 6. Conclusions

The preliminary results from this project are encouraging and will be followed by system enhancements that will allow more complex experiment to be performed and move into the next phase. Most of the initial aims for this initial phase have been satisfied and can be summarized as;

- The Evolutionary Paradigm can be successfully applied for the creation of novel sounds often with surprising complexity. It seems that viable (fit) parameter sections are preserved through the genetic recombination, as it is also the case with Genetic Algorithm optimisation (Goldberg 1989). In other words if the starting sounds are professionally designed ones then the offspring are likely to be of comparable quality. This also shown by the observation that genetic recombination produces higher quality results than if mutation is used alone. In implementations (Yee-King 2000) where no genetic recombination is used, and mutation or a type of “genetic space crawling” is used instead, it is much harder to produce sounds that are complex and of high (subjective) quality.
- Different Sound Synthesis Techniques can be used without the use of Specific Domain Knowledge. It was an initial requirement that no specific domain knowledge should be used in the system, that is the parameters are treated as going into a black box, no knowledge of their function is kept in the system. As a result a new synthesis technique can be added by just specifying the System Exclusive Implementation Chart of the new synthesiser. As a down side, when sounds are produced where they are fit but very quiet then there is no way in the current framework to address the problem. The only solution is to either selectively breed part of the genotype that may be suspected of being responsible, or manually tweak individual values until the desired result is achieved.
- The level of abstraction (SysX MIDI) was the right choice. Synthesisers are designed for musical sounds and aspects like keyboard mapping are already implemented in them. If an approach was taken where the parameters were encoding low-level sound production (*Woolf* 1999), it would have been much harder to produce musically acceptable sounds.
- The use of the glove is responsive and expressive. Although there is no structured a priori mapping between hand movements and sounds changes (since they are also evolved), it is usually easily internalised by the player (Krefeld 1990). That is after few finger flexions the brain seems to be able to assimilate the correspondence of movements and sound changes. This actually came as a surprise, it was thought originally that some rigid, direct mapping would have to be used in order for the brain to learn the movement-to-sound mapping. One example of such mapping is “sound sculpturing” (Mulder, Fels & Mase 1997) in which the sound is represented by a three-dimensional object where changes in its shape are translated to changes in sound. It was also thought that some kind of structured language would have to be used for describing those mappings (Keane & Gross 1989; Machover & Chung 1989; Pressing 1990; Mulder, Fels & Mase 1997; Mulder 1994; Rován, Wanderley, Dubnov & Depalle 1997) if no direct one-to-one mapping was used. The use of such formalisation hasn't been necessary yet partly because the Prophecy implements its own mapping formalisation through SysX parameters, and partly because the relatively low dimensionality of the input device (dataglove) which has only five degrees of freedom. It would also be interesting to see if the ease of internalising mappings is retained when input devices of more channels are used i.e. more than five. In the future when different synthesisers and input devices (with more degrees of freedom) are used, the issue of a mapping formalisation will have to be addressed. Also the two processes for sound evolution, and motion-to-sound mapping evolution will have to be separated.
- The ease of use of the interface was also a surprising outcome. The selective breeding paradigm is an accessible one and users were able to breed complex sounds after only a brief introduction. The sounds produced were of such quality that would take someone with quite a bit of experience in the synthesis technique if they were to be programmed directly by hand. The fact that the overall process is not directional, that is not designed to satisfy a priori sound specifications i.e. “I would like to produce a bell sound”, does not preclude the possibility of doing so in indirect ways. For instance, if bell or bell-like sounds are used as a starting point, then it is conceivable that a satisfactory bell sound will be produced within a few generations of selective breeding.

# Genophone: An Evolutionary Approach to Sound Synthesis and Performance

## References

- Goldberg, D.E. (1989). *Genetic Algorithms in search, optimisation and machine learning*. Reading, MA: Addison-Wesley.
- Keane, D. & Gross, P. (1989). The MIDI baton. *Proceedings International Computer Music Conference*, Columbus, Ohio, USA. San Francisco CA, USA: International Computer Music Association.
- Krefeld, V. (1990). The Hand in the Web: An interview with Michel Waisvisz. *Computer Music Journal*, 14 (2), 28-33.
- Machover, T. & Chung, J. (1989). Hyperinstruments: Musically intelligent and interactive performance and creativity systems. *Proceedings International Computer Music Conference*, Columbus, Ohio, USA. San Francisco CA, USA: International Computer Music Association.
- Mulder, A.G.E. (1994). Build a better powerglove. PCVR, 16, pp 10-14. Stoughton, WI, USA: PCVR Magazine.
- Mulder, A.G.E. (1994). Virtual Musical Instruments: Accessing the Sound Synthesis Universe as a Performer. *Proceedings of the First Brazilian Symposium on Computer Music*, pp. 243-250
- Mulder, A.G.E. Fels, S.S. & Mase, K. (1997). Empty-handed Gesture Analysis in Max/FTS. *Proceedings of the AIMI international workshop on Kansei - the technology of emotion*, Antonio Camurri (ed.), pp 87-90.
- Mulder, A.G.E. Fels, S.S. & Mase, K. (1997). Mapping virtual object manipulation to sound variation. *IPSJ SIG notes Vol. 97, No. 122, 97-MUS-23 (USA/Japan intercollege computer music festival)*, pp. 63-68.
- Pressing, J. (1990). Cybernetic issues in interactive performance systems. *Computer Music Journal*, 14 (1), 12-25.
- Rovani, J.B. Wanderley, M.M. Dubnov, S. & Depalle, P. (1997). Instrumental Gestural Mapping Strategies as Expressivity Determinants in Computer Music Performance .Presented at "Kansei - The Technology of Emotion" Workshop.
- Woolf, S. (1999). Sound Gallery: An Interactive Artificial Life Artwork. *MSc Thesis, School of Cognitive and Computing Sciences, University of Sussex*.
- Yee-King, M. (2000). AudioServe - an online system to evolve modular audio synthesis circuits. *MSc Thesis, School of Cognitive and Computing Sciences, University of Sussex*.