Durham University

Durham E-Theses

Algorithmic Compositional Methods and their Role in Genesis: A Multi-Functional Real-Time Computer Music System

JULIAN WILLIAM LYWOOD-MULCOCK

How to cite:

LYWOOD-MULCOCK, JULIAN WILLIAM (2015) Algorithmic Compositional Methods and their Role in Genesis: A Multi-Functional Real-Time Computer Music System. Doctoral thesis, Durham University.

Use policy

The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a https://etheses.durham.ac.uk/id/eprint/11033/ is made to the metadata record in Durham E-Theses
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

Please consult the full Durham E-Theses policy for further details.

Academic Support Office, The Palatine Centre, Durham University, Stockton Road, Durham, DH1 3LE
e-mail: e-theses.admin@durham.ac.uk Tel: +44 0191 334 6107
https://etheses.durham.ac.uk

Julian Lywood Mulcock

Algorithmic Compositional Methods and their Role in *Genesis*: A Multi-Functional Real-Time Computer Music System

Algorithmic procedures have been applied in computer music systems to generate compositional products using conventional musical formalism, extensions of such musical formalism and extra-musical disciplines such as mathematical models. This research investigates the applicability of such algorithmic methodologies for real-time musical composition, culminating in *Genesis*, a multi-functional real-time computer music system written for Mac OS X in the SuperCollider object-oriented programming language, and contained in the accompanying DVD. Through an extensive graphical user interface, *Genesis* offers musicians the opportunity to explore the application of the sonic features of real-time sound-objects to designated generative processes via different models of interaction such as unsupervised musical composition by *Genesis* and networked control of external *Genesis* instances. As a result of the applied interactive, generative and analytical methods, *Genesis* forms a unique compositional process, with a compositional product that reflects the character of its interactions between the sonic features of real-time sound-objects and its selected algorithmic procedures.

Within this thesis, the technologies involved in algorithmic methodologies used for compositional processes, and the concepts that define their constructs are described, with consequent detailing of their selection and application in *Genesis*, with audio examples of algorithmic compositional methods demonstrated on the accompanying DVD. To demonstrate the real-time compositional abilities of *Genesis*, free explorations with instrumentalists, along with studio recordings of the compositional processes available in *Genesis* are presented in audiovisual examples contained in the accompanying DVD. The evaluation of the *Genesis* system's capability to form a real-time compositional process, thereby maintaining real-time interaction between the sonic features of real-time sound objects and its selected algorithmic compositional methods, focuses on existing evaluation techniques founded in HCI and the qualitative issues such evaluation methods present. In terms of the compositional products generated by *Genesis*, the challenges in quantifying and qualifying its compositional outputs are identified, demonstrating the intricacies of assessing generative methods of compositional processes, and their impact on a resulting compositional product. The thesis concludes by considering further advances and applications of *Genesis*, and inviting further dissemination of the *Genesis* system and promotion of research into evaluative methods of generative techniques, with the hope that this may provide additional insight into the relative success of products generated by real-time algorithmic compositional processes.

Algorithmic Compositional Methods and their Role in *Genesis*: A Multi-Functional Real-Time Computer Music System

Julian William Lywood Mulcock

PhD Thesis

Department of Music

Durham University

2014

Table of Contents

Chapter 4: Interactivity in Digital Music Systems

Chapter 5: The *Genesis* System

Chapter 6: Evaluation of the *Genesis* System

List of Illustrations and Tables

DVD Table of Contents

***Genesis* Folder**

Genesis.rtf

Genesis

***Thesis Recordings* Folder**

1 – BBCut2.aif

2 – Genesis Quantized.aif

3 – BBCut2 Determined.aif

4 – Genesis Click.aif

***Audiovisual Examples* Folder**

1. Quick Start Guide.mov
2. MIDI Implementation.mov
3. Local Static Onsets.mov
4. Local Dynamic Onsets.mov
5. Grain Freeze.mov
6. Slave Sound-Object Enveloping.mov
7. Network Set Up.mov
8. Network Onsets.mov

9. Local GAs.mov
10. Fractals Static.mov
11. Fractals Dynamic.mov
12. Local Spectral Following.mov
13. Pitch Track Both Inputs.mov
14. Pitch Track Slave Pitch Fixed.mov
15. Tempo Following.mov
16. Network Onsets and Pitch.mov
17. Pitch Fix with Original.mov
18. Pitch Fix without Original.mov
19. Dynamic Scoring System.mov
20. Call and Response.mov
21. Live Routine, Live Sample.mov
22. Random Search Processes.mov
23. Markov Chain.mov

Genesis Performances **Folder**

John Sniders Folder

John Snijders Free Exploration.mov

Mark Carroll Folder

Mark Carroll Free Exploration 1.mov

Mark Carroll Free Exploration 2.mov

Shelly Knotts Folder

Shelly Knotts Free Exploration 1.mov

Shelly Knotts Free Exploration 2.mov

Shelly Knotts Free Exploration 3.mov

Shelly Knotts Free Exploration 4.mov

Statement of Copyright

The copyright of this thesis rests with the author. No quotation from it should be published without the author's prior written consent and information derived from it should be acknowledged.

Chapter 1

Introduction

1.1 An Overview of the Research Topic

Since the advent of modern computing, computational algorithmic techniques have been applied to generate and analyse musical compositions. In addition, numerous methods of interaction between a computer and a human user have been suggested for the purpose of modifying, manipulating and arranging musical structures within a composition, such as the pitch, rhythm and timbre of selected synthesized instruments. The efficiency and digital accuracy with which modern computers can calculate such musical structures has enabled composers to explore novel and extra-musical approaches as part of a compositional process, or, to form an entire compositional process itself resulting in a proposed autonomously generated musical composition.

Generative processes offer composers the capability to create parametric values of a compositional system relative to a selected algorithmic structure. The conditional behaviours of such algorithmic processes can greatly influence the product of an algorithmic compositional process. For example, the use of indeterminacy can generate a significant variety of compositional outcomes, which can be bound (or not) to chosen minimum/maximum values, thereby causing a level of unpredictability in a compositional process and resulting in an output that has the potential to provide numerous compositional products. Furthermore, with the onset of technological advancements, real-time execution of generative processes permits composers to generate algorithmic compositional products on-the-fly, as a generative process is running, thereby allowing instantaneous modification, manipulation and arrangement of a musical structure.

Analytical processes, which can be applied algorithmically, are currently based upon either conventional musical analysis such as Schenkerian analysis or note-event assessment through musical values (for example pitch, tempo and onset), or novel

methods of evaluation formed of perception models such as those proposed within the field of psychoacoustics based on auditory phenomena such as loudness, timbre and spatialisation. Despite numerous suggested analysis techniques, due to the unavoidable limitations in the explanation of our own listening experience, no conclusive method of analysis is currently available, with the computational analysis of musical structures reflecting such constraints.

As a result, decisive analysis by computers of musical composition is distinctly relative to the type of composition analysed and the type of analysis applied. For example, many conventional musical analysis methods use strictly formalist and orthodox principles of musical description such as scales, key and tempo, which can be applied to conventional approaches to musical composition. However, for musical compositions that do not pertain to such formalist explanations, such a musical analysis method is void and necessitates perceptual models for interpretation of musical gesture.

Moreover, real-time functionality of analytical techniques must also be extensively reviewed, as instantaneous analytical results are crucial to the maintenance of interaction between generative processes, which require such assessment values to function; the more complex an analytical process, the more time it may take to complete its assigned task, thereby introducing latency between interactive processes and disrupting the unfolding dialog between them, potentially impacting on the fluency of the compositional product.

As noted previously, there are numerous interactive methods that have been applied to algorithmic compositional methods. The method of interaction dictates the relative level of influence a composer may (or may not) wish to have on a resulting compositional outcome. With communication protocols such as Open Sound Control (OSC), interactions can be sent instantaneously from sources such as physical digital hardware to the sonic features of an analog sound signal, extracted through analytical algorithms and represented as symbolic or subsymbolic musical values. In addition, protocols such as OSC allow such representations of sonic features to be broadcast

over computer networks, offering ensembles of computers to communicate and interact through their respective musical values.

So, with the application of computational algorithmic processes, it is possible to create extensive real-time digital music systems that generate musical compositions, relative to their defined interactive, generative and analytical processes. With computer programming languages such as SuperCollider and Pure Data, composers can investigate computational algorithmic methods of compositional processes using open-source classes to form the fundamental architecture of a digital music system, within the prescribed language. Consequently, composers can dictate each algorithmic process's influence on a resulting compositional product through the hierarchy of each algorithmic process's status in a real-time digital music system's fundamental architecture, and therefore its role in the compositional process.

The result of such extensive real-time digital music systems is musical compositions that can be generated instantaneously, applying selected conditional behaviours that can be modified, manipulated and arranged by the interactions of sources, extracted and represented through analytical algorithms. The implications of a composition generated by such a real-time method present distinct challenges in concluding the nature of the compositional process and the assessment of its compositional product.

Therefore, despite the promise of extensive real-time digital music systems for the generation of musical compositions, a significant number of aesthetic issues must still be considered when creating such systems, for the purpose of warranting their validity as a method musical composition; with the acknowledgment of aesthetic considerations such as the purpose of applying a chosen algorithm to a compositional process or a deliberation by the composer of the influence an algorithm may have on a resulting compositional product, it is possible to resolve concerns over the cogency of extensive real-time digital music systems and their role in musical composition.

Considering composition within the context of the research presented within this thesis, the primary focus is relative to the real-time algorithmic method applied in *Genesis*. The *Genesis* system uses the sonic features of real-time sound-objects, such

as timbre, pitch and onset, to modify a number of generative processes mapped to relatable values for a series of granular synthesisers and associated filters. Therefore, the compositional approach is founded upon real-time interaction, thereby applying 'virtual scores', as proposed by Manoury (1990), through which composition with the *Genesis* occurs in musical time, generating the musical score as part of the real-time interaction process (this is discussed in detail in chapter 4.2 *Composition with Real-time Interactive Music Systems*). As a result of such an approach, pre-compositional devices, such as a predefined score are not necessary, but can still be applied should a composer wish to dictate specific compositional material to interact with the system.

Furthermore, due to the granular synthesis method of realisation by the system for generating sound-objects relative to the outputs of the generative processes controlled by the sonic features of real-time sound-objects, an 'acoustical model' [1] is implemented. Through an acoustical model, 'the program that carries out the steps required to produce a sound realizes a given acoustic description of musical sound'[2]. Consequently, through the applied analytical models, the sonic features of the real-time sound-objects are applied to relative parameters within the granular synthesisers and filters, interpreting and then realising the sonic outputs of *Genesis* in real-time. As a result of the implementation of granular synthesis techniques in *Genesis*, a combination of microsound compositional methods and conventional musical formalisms are applied, thereby merging timbral manipulation of sound-objects with defined musical values such as pitch and duration.

In addition, through the application of a real-time compositional methodology, the concept of improvisation is not considered mutually exclusive to the process of composition; due to the input and realisation of compositional material when interacting with *Genesis*, the user and machine generate responses as part of the ongoing compositional procedure, thereby altering their response strategy in real-time relative to creative methodologies of both parties (again, this is discussed further in chapter 4.2 *Composition with Real-time Interactive Music Systems*).

[1] Truax, B. For Otto Laske: A Communicational Approach to Computer Sound Programs. *Journal of Music Theory*. 20 (2): 233
[2] Ibid

The algorithmic implementations, response strategies, analytical processes and modes of interaction with *Genesis* are discussed relative to existing research in chapter 2 *An Introduction to Algorithmic Composition*, chapter 3 *Real-time Computational Algorithmic Systems in Musical Practice* and chapter 4 *Interactivity with Digital Music Systems*, with chapter 5 *The Genesis System* detailing how the research has been applied in *Genesis*. Consequently, chapter 6 *Evaluation of the Genesis System* discusses the success of *Genesis*, relative to the research presented in the chapters listed above.

1.2 Personal Motivation

In consideration of the approach taken to this research, which predominantly assesses the musicality of real-time computational algorithmic processes, it is necessary to contextualize this in relation to my background. In my youth, I participated in all that I could which had a musical focus. Through the flute and violin, I learned the fundamental formalist approaches to musical composition in both solo and ensemble scenarios. However, I would often seek to explore what lay beyond the formalisms of the symphony orchestra, often challenging (with not much success) the reasons for such methods of compositional process.

With technological advancements prominent in the media throughout the mid 1990s, such as the showcasing of 'virtual reality' headsets, 3-D graphics formed of blocky polygons and the 'World Wide Web', an article on the BBC children's television programme Blue Peter demonstrated a system very similar to Piano Tutor[3] (indeed it may have Piano Tutor itself, but I cannot confirm this through relevant searches). I was fascinated by the process through which the system was able to assess and adapt to a performer's interaction with the computer, and, in my naivety considered the computer to be as good, if not better than a human at the assessment and adaptation of a real-time performance, indeed considering it to present an incredible level of artificial intelligence.

[3] Dannenberg et al. 1990. An Expert System for Teaching Piano to Novices. *proceedings of the ICMC'90*: 20-23

Furthermore, prior to the mid 1990s, popular music videos were saturated with images of recording artists stood behind racks of analog and digital synthesizers creating unfamiliar sounds, often accompanied with CRT computer screens flickering illogical numbers and graphs, which were somehow meant to represent the ongoing compositional process in the audio recording. Therefore, at the time, I was convinced that exploration of music with computers would address any questions I had regarding the necessity for formalisms in musical composition.

Upon acquiring Opcode's MusicShop with issue one of Computer Music Magazine and having purchased a Yamaha CS2x digital synthesizer, I believed I could not only replicate the sounds of the popular artists on the radio, but also have the computer assess and adapt my compositional outputs for the purpose of improving my compositions as well as exploring music that was not bound the formalisms of the symphony orchestra. However, much to my disappointment, none of this happened; MusicShop had no facilities to assess my compositions, and although the Yamaha CS2x digital synthesizer could generate sounds similar to those used in professional recordings, recording and editing of the interactions was distinctly limited by the formalist representation of sound through a pitch/duration paradigm in the MusicShop sequencer.

Throughout my undergraduate degree in Music Informatics and my Masters in Electroacoustic Studies, it became evident that I had indeed been highly naïve in my assumption of what computers, and the algorithmic processes they can execute, are actually capable of. However, instead of becoming disgruntled and resentful of the fact the apparent artificial intelligence computers present is in reality highly limited in comparison to our own intellectual prowess, I became encouraged to investigate what could indeed be generated with the limited 'intelligence' a computer has and what impact this may have in a musical composition process. Consequently, I discovered the myriad of computational algorithmic processes that have been applied to form compositional processes from the mid-20th Century, which inherently challenges the perception of artificial intelligence, the role of algorithmic processes and the importance of computers in musical composition.

One such algorithmic process encountered during my undergraduate degree is concatenative synthesis (Schwarz, 2006; Casey, 2004; Lazier and Cook, 2003; Momeni and Mandel, 2005), a method of sound synthesis through which the sonic features of a target sound-object are compared to a database of sound-objects, with the best match to the target within the database used as the synthesizer's audible output. Such a synthesis method applies extensive analysis techniques in order to compare adequately the sound-objects, which is used to assess their suitability to a target, relative to the description of a best match algorithm.

Remembering the Piano Tutor (Dannenberg, 1990) I had seen years before, I immediately began to make comparisons to the process; both systems assessed an input, compared them to a suitable descriptor resulting in an output based on its assessment. However, clear distinctions in the method of representation are present; concatenative synthesis requires analytical algorithms that apply feature extraction to represent sonic features of an acoustic signal where as Piano Tutor uses symbolic MIDI messaging, thereby limiting its application significantly to the formalist structures of MIDI and to the use of MIDI instruments.

Seeing the potential of applying sonic features of a sound-object for the control of other sound-objects, and my introduction to SuperCollider, a music programming language that permits extensive real-time functionality, upon completing my Master's dissertation in *Psychoacoustics and its role within Machine Listening*, I wished to explore extensively the possibilities of real-time compositional processes using the sonic features of sound-objects, extracted through methods such as psychoacoustic models, to generate real-time compositions. This principle led to the beginnings of the Genesis standalone program, which accompanies this thesis.

1.3 Aims of the Research

Through the use of sonic features extracted from an acoustic source, it is possible to apply the extracted values to control or influence a chosen parameter within a digital music system. This thesis, and the accompanying *Genesis* system, investigates and demonstrates methods of interactive, generative and analytical processes which can be used to apply such sonic features for the purpose of musical composition. Therefore,

the aim of this research is to present methodologies, aesthetic considerations and the implementations available to form such a digital music system, in combination with a detailed account of those applied in *Genesis* and why.

Below is a summary of the primary issues raised within the chapters of the thesis:

- The influence computational algorithmic procedures may have on a compositional process
- The acknowledgement of the constraints in feature extraction from acoustic sources, and indeed, the limitations in our understanding of the listening experience
- The effect of interaction methodology on a compositional process and associated models of interaction
- Implementation of creativity with machines
- The importance of efficiency in real-time compositional processes
- The challenges in forming formal evaluation of real-time interactive music systems
- The difficulties in comparing the compositional products of digital music systems considering the absence of conclusive analysis
- The advantages and disadvantages of using music programming languages for the construction of digital music systems
- The necessity of a composer to acknowledge the possible outcomes of a compositional process that applies extensive computational algorithmic processes

In addition, the thesis aims to provide the following original contributions to the research topic:

- A novel method of real-time interaction with a digital music system through the use of real-time sound-objects as a predominant interface device
- A unique approach to the evaluation of the processes and products of real-time interactive music systems based upon extensions in current HCI evaluative techniques

- Critical review of interaction methodologies and their relevance to real-time interactive music systems
- Detailed audiovisual examples of, and performances with, the *Genesis* system, thereby providing researchers with documented evidence of its algorithmic implementations
- Complete and thorough explanation of the generative, analytical and interactive processes in *Genesis*, and their relationship to existing algorithmic methods
- Discussion regarding the implications of real-time compositional techniques for composers and performers
- Proposed consequences of applying random and unpredictable methodologies to provide creative outputs in real-time interactive music systems

1.4 Implementation

For the dissemination of the research, a thesis is provided, detailing the topics described above, along with a DVD which includes the Genesis standalone application which will run on any Mac OS X system 10.6+ and three folders, containing examples of its generative and interactive processes, and audiovisual examples of the functionality of *Genesis*, all of which are referenced within their relevant sections in the thesis.

1.5 Evaluation Criteria

A principle objective in evaluating *Genesis* is to identify a methodology that would systematically evaluate *Genesis* relative to three key areas:

- interaction with *Genesis* via the GUI
- interaction with *Genesis* by instrumentalists and musicians
- the global products of composition with *Genesis*

As a result, in section 6.1 *Evaluation Methodology*, approaches to, and the challenges in the evaluation of real-time interactive music systems are discussed. Consequently, a performer-centred evaluation methodology is applied, accompanied by an approach based upon the evaluation method proposed by Stowell et al (2009) extending existing HCI techniques, which uses questionnaires and a Likert scale. This method involves supervised and unsupervised exploration with the *Genesis* system, with consequent discussion completed afterwards in the form of a questionnaire comprised of critical questions and Likert scale responses in order to generate quantitative and qualitative results relative to the three key areas listed above.

To obtain the evaluative results, three experienced musicians were invited to attend solo sessions with author, through which a range of the interactive and generative properties in *Genesis* are explored, relative to the evaluation method suggested by Stowell et al (2009). Each participant engages with *Genesis*, and is asked to generate a real-time composition/s with the system, which is documented audiovisually in the folder *Genesis Performances* on the accompanying DVD. The questionnaire is designed to provide valuable and balanced perspectives of the success of the *Genesis* system, in relation to the three key areas above, with the audiovisual examples documenting the interaction and products of each performer's solo session.

With the feedback generated from the evaluation, combined with the audiovisual performances, the research presented in chapter 2 *An Introduction to Algorithmic Composition*, chapter 3 *Real-time Computational Algorithmic Systems in Musical Practice* and chapter 4 *Interactivity with Digital Music Systems* is directly and explicitly applied to the responses provided by the performers to form critical review of the process and products of the *Genesis* system, thereby providing insight into the aesthetic value and context of *Genesis* and its associated interactive, analytical and generative implementations.

Chapter 2

An Introduction to Algorithmic Composition

2.1 Algorithms in the Compositional Process

It is proposed that the creative process involves four stages; stage one - preparation, stage two - incubation, stage three - illumination and stage four - verification[4]. If we are to apply these four stages of the creative process to musical composition then: stage one - musical objectives are chosen and researched, stage two - those objectives are considered by our subconscious, stage three - 'eureka', a possible solution is created based on stage one and two, and stage four - it is put into practice, and the result is scored or performed. However, the use of only four distinct stages oversimplifies the creative process as the differences between the individual need to be represented. Guildford (1950) considers such dissimilarities between an individual's creative process by suggesting that 'a sensitivity to problems, a capacity to produce ideas (fluency), an ability to change one's mental set (flexibility), an ability to reorganize, an ability to deal with complexity, and an ability to evaluate'[5] each impact on the capability of one's creative process.

There are many models that exemplify the abilities of the individual in the creative process as well as Guildford's (1950) such as Busse and Mansfield's (1980) which proposes the steps of 'a) selecting a problem to solve among several other problems, b) engaging in efforts to solve the problem, c) setting constraints on the problem solution, d) changing the constraints and restructuring the problem (which if successful leads to an illumination) and e) verifying the proposed solution'[6]. To take into account the importance of the role of the individual and their influence on the musical composition process, it is possible to conclude that their abilities to perform the tasks outside of the four initial stages will have a significant impact on a compositional *product* as well as the composition *process* itself.

[4] Lubart, T. 2000. Models of the Creative Process: Past, Present and Future. *Creativity Research Journal* 13(3/4): 295-308
[5] Ibid: 295
[6] Ibid: 304

The result or *product* of a creative process and the reflection and assessment of the outcome can become part of the creative process itself, demonstrating a potentially recursive nature, implying that the creative process is an infinite loop, in which, an individual creates, assesses, creates, assesses and so on. If this is true, then the decision to render a 'final' result can always be questioned; is there room for improvement? This question highlights again the individual's importance in shaping the creative process, as it is ultimately their decision when the outcome becomes definitive, if indeed it ever does. As a result, it could be considered that the creative process and the products it generates are idiosyncratic, thereby reflecting the behaviours, understanding and conditioning of the individual.

Algorithms, which are a set of formalised rules with the aim of producing a result bound by the instructions used, can thus be applied at each of the four stages of the creative process. As a basic example, if an algorithm is applied to all four stages of the creative process, it could at stage one - create a series of musical phrases, at stage two - employ random generators to model the subconscious, at stage three - generate a result based on stage one and stage two and at stage four - notate the result ready for performance.

As stated, models representing the abilities of the individual can also be applied to each of the stages, and to extend the previous example may at stage one - use prior knowledge based upon musical formalisms to create musical phrases, at stage two - use attractors to collect specified values that are created by a prediction-driven generator, constructing a disposition within the subconscious, at stage three - make further use of musical formalisms to assess the results that are based on stage one and stage two, and at stage four - use scoring rules for notation of the results.

It is evident that many aspects of the creative process can be modelled with a variety of structures, rules and formalisms, and although it is possible to engage all manner of discussion as regards to their appropriateness and significance, their presence cannot be ignored in any consideration of the creative process. Algorithms are a suitable method for writing rules and models of processes, as they are able to follow a series of instructions forming a calculation that outputs a result based upon the instructions

they have followed. For example, algorithms can be used to complete the following functions at different stages of the creative process:

1. Model the *entire* creative process including analysis of the input/output
2. Model *stages* of creative processes including analysis of the input/output
3. Generate results based on a creative process without analysing the input/output
4. Randomly generate results relative to conceptual constraints

The use of algorithms, and therefore the use of rules and formalisms is prevalent in the Western Art Tradition. Melodic rules can be found dating back as far as the 11th Century, 'when Guido d'Arezzo used a scheme that assigned a different pitch to each vowel in a religious text'[7] and, as the Western Art Tradition evolved, harmonic, structural and rhythmic rules did so too. For example, 'the 14th and 15th centuries saw the development of the quasi-algorithmic isorhythmic technique, where rhythmic cycles (*talea*) are repeated, often with melodic cycles (*color*) of the same or differing lengths'[8]. This application of repetition is a divisive compositional procedure and a foundation of the compositional process in the Western Art Tradition, and therefore forms a basis from which its musical rules and formalisms developed 'seen in various guises: the Classical Rondo (with section structures such as ABACA); the Baroque fugue; and the Classical sonata form with its return not just of themes but tonality too'[9].

Considering the progression of the Western Art Tradition in the 20th Century by the Second Viennese School, predominantly led by Schoenberg, the inception of serialism and its use of pitch classes 'can be viewed as no more than a continuation of the tradition of formalising musical composition'[10]; strict organisation rules dictate the relationships and structures between each of a tone row's pitches. As the Serialist movement developed, so did its formalisms, giving rise to the concept of Total serialism through which composers such as Boulez and Pousseur subjected further musical values such as rhythm and dynamics to the organizational principles of

[7] Järveläinen, H. 2000. *Algorithmic Musical Composition*. University of Technology, Helsinki: 1
[8] Edwards, M. 2007. *Algorithmic Composition: Computational Thinking in Music*. University of Edinburgh: 2
[9] Ibid
[10] Ibid

Schoenberg's serialism. Therefore, algorithmic methodologies (the use of rules, structures and formalisms) are undoubtedly synonymous with the compositional processes of the Western Art Tradition.

Furthermore, rules can be found within compositions that seem apparently free of much of the Western Art Tradition such as that of John Cage's *Music of Changes* (1951) for solo piano. This work makes use of chance and indeterminacy based upon a modified *I Ching*, a Chinese text believed to be dating back as far as 1000BC which features 64 hexagrams each denoting a possible 'change' or 'wisdom'. The score was 'decided' by flipping coins and consequently choosing a 'change' based upon the result of a series of coin flips and then referring to a modified *I Ching,* which featured musical events instead of the original's 'wisdoms'. Interestingly, although a meticulous process of selection for each musical event was employed, Cage described the work as having 'a freely moving continuity'[11].

The use of chance and indeterminacy is by no means a contemporary convention; 'the invention of musical dice games by composers like Johann Philipp Kirnberger, Maximilian Stadler and Joseph Haydn enabled amateur musicians to generate numerous variants of dance pieces'[12]. Mozart also embraced such a composition process in his *Musikalisches Würfelspiel* (1787) in which 'eleven different versions of each bar of the minuet have been composed beforehand'[13] allowing the performer to present many different versions of the same composition based on the outcome of their dice rolls.

As demonstrated by Cage's *Music of Changes* (1951), the rules that can be applied to composition may also exist in other disciplines and subjects. For example, Joseph Schillinger (1895-1943) explored the application of mathematical processes to musical composition. His work then 'penetrated modern compositional practice, from Allen Forte's work on pitch-class sets, to Karlheinz Stockhausen's so-called 'Formant-Rhythmik' or Gottfried Michael Koenig's concept of periodicity as it is

[11] Cage, J. 1961. *Silence: Lectures and Writings.* Connecticut: Wesleyan University Press: 1
[12] Essl, K. 2007. 'Algorithmic composition' in *The Cambridge Companion to Electronic Music*, eds N Collins & J d'Escrivan, CUP, Cambridge: 109
[13] Ibid

implemented in his algorithmic composition software *Projekt 1*'[14]. However, the use of methods from subjects outside of music may not always be successful; much in the same way that not all rules found in the Western Art Tradition will not be applicable for all genres of music. Notwithstanding this fallibility, many composers continue to search for new ways to apply algorithmic methods to the composition of music.

Reflecting on the prominence of algorithms in the Western Art Tradition, it would appear that for Western Art Tradition to progress, it must build on its algorithmic foundations. However, there is a common misconception that algorithmic composition is bound to the application of computers for musical composition. As demonstrated, this is clearly not the case - algorithms have evidently been implemented in musical procedures for hundreds of years. Rather, the use of algorithms and computers facilitates the exploration of new, novel and complex rules, formalisms and structures. Therefore, it could be considered that algorithmic composition with computers is the modern development of the Western Art Tradition, building upon its intrinsic algorithmic methodologies through computational processes.

2.2 Generative and Analytical Algorithms

Algorithms, when applied to musical composition, may be generative, analytical or both generative and analytical. Many conventional formal musical rules are relatively simple to write as an algorithm but there are a number of issues that can affect the quality of the result. For example, if a generative algorithm is used to dictate the rhythm of a chosen phrase, it is possible to simply choose a set of rhythms from a pre-selected rhythm table and use the result as the solution. However, a composer may wish to give an algorithm a more developed context in relation to what comes before and what comes after a chosen passage, adding another level of intricacy to the generative process, requiring an analytical algorithm to assess the relationship between what has occurred and the following output by a generative algorithm. The composer may also wish to give the algorithm the ability to use unpredictability or

[14] Essl, K. 2007. 'Algorithmic composition' in *The Cambridge Companion to Electronic Music*, eds N Collins & J d'Escrivan, CUP, Cambridge: 111

randomness as part of the possible solution, requiring careful decision-making on how unpredictable or random they would wish the outcome to be.
The *Genesis* system applies a number of generative processes controlled by the user and/or by data gathered from analytical algorithms, which are mapped in real-time to selected musical values, relative to the conceptual constraints of the composition system (detailed fully in chapter 5 *The Genesis System*). It is therefore necessary to define what generative processes are, and how chance and indeterminacy can affect the outcomes of generative algorithms.

A generative process can be considered to be that which may 'create a new entity or bring about a novel circumstance'[15]. The application of such a process can also provide 'the flexibility to build processes which generate new sequences of events every time it is executed, and processes which respond to environmental and human interference whilst remaining within the boundaries imposed by the programmer'[16] Due to its fundamentally creative properties, it is an attractive method for use with not only musical composition but also among others, visual art and computer programming.

Generative processes, which allow for the possibility of chance and instance, can be seen as an exciting and interesting characteristic to their capabilities. A distinction must be made, however, as the use of chance is not intrinsic to a generative process; composers have the choice of how strictly and to what extent they wish to enforce the use of chance, or whether to apply it intentionally at all. When making this decision, it is important that the composer is aware that, much like the process of choosing algorithms for a composition, the application of chance and its possible outcomes are carefully considered in the context of the intended outcomes and conceptual constraints of a composition. If not, the resulting composition may be far from what the composer originally intended or may not be performable.

Considering the ability of generative algorithms to generate 'novel circumstance'[17], to what extent do generative algorithms 'create a new entity'? Surely, the concept of

[15] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49
[16] Ibid
[17] Ibid

'new' is subjective to the composer and the creative process they have applied. For example, if an algorithm is following a formalism such as cadential progression, the result will perhaps be entirely predictable, meaning the product that is output by a generative algorithm is simply an entity of which all of its properties are known and therefore *may* not be considered 'new'. As a result, such a creative process that does not incorporate unpredictable or random behaviours, yet is still reliant on a generative algorithm to generate its products are 'new' relative to the judgments of the observer.

However, in reference to the concept of 'new' and its association with chance, the use of chance serves to alter the level of which a result appears 'new' to the composer; the use of chance does not guarantee newness as it is the amount of change from a known, recognisable and/or anticipated outcome that renders a result 'new'. Therefore, the incorporation of chance in a generative process could 'bring about novel circumstance'. So, generative algorithms may or may not 'bring about a novel circumstance'[18] dependent on the composer's specification.

As noted, the conceptual constraints of a composition must be considered in relation to the application of generative processes for the generation of musical ideas. So, the mapping between generative process and musical values is paramount to ensure cogency between a generative process and a resulting musical composition. Generative processes can be used for the creation of patterns, sequences or single events but these are, at their most fundamental level, values, which 'mean' nothing musically; a generative process does not create music in itself, it is the application of the result by the composer that is ultimately the most important factor when using generative processes for musical composition. Therefore, it is the composer's use of a generative process's results relative to the conceptual constraints of the compositional process that renders such a process's products applicable to a musical composition.

The results of a generative process and their consequent use in a musical composition are in no way limited; any musical value such as pitch, duration, spatialisation, dynamic or timbre may be controlled by the results. This leaves the composer with the exceptionally difficult and meticulous task of choosing which values represent what

[18] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49

purpose. A composer has the option of relying on an algorithm to allocate a set of results to a creator-selected series of musical variables, which will make the process considerably more efficient but increase an algorithm's influence on a composition. It is therefore important that the size of the results created by a generative process are manageable by the composer themselves or that the composer has taken in to account the possible requirement of an analytical algorithm to aid the application of the results, and the influence this may have on a generative algorithm and its compositional role.

With the necessity for time to analyse and assess a generative process's output taken in to consideration, the situation of the performance must impact on the decision to allow time for analysis as, for example, in a live performance setting, direct analysis by the composer may be unfeasible, thereby necessitating the application of a computational real-time analytical process. *Genesis* is designed for real-time functionality and must therefore use selected real-time analytical algorithms to obtain sonic features from real-time sound-objects. The results of the applied analytical algorithms are then mapped in real-time to their respective values in the selected generative processes used in the *Genesis* system. It is therefore necessary to describe analytical algorithms in the context of musical analysis and how the results can be applied to generative processes.

Many qualities of music can be identified and analysed through the use of algorithmic techniques. For example, analytical algorithms can be executed to define the pitch, loudness and onset of individual note events, the tempo of specified phrases, the timbre of instruments or sound-objects, the genre of a selected work, the key signature of a phrase and any harmonic structures or melodic patterns that may be present in a chosen section. There are three predominant stages required for an analytical algorithm to perform a task successfully:

Stage One - Extract a data set suitable to the analytical process desired
Stage Two - Use a clearly defined rule base from which to make an analysis of the supplied data set
Stage Three – Resynthesize the results created by the analysis

In summary, analytical algorithms can be applied to assess inputs and/or outputs at various stages of the compositional process, for the primary purpose of the automation of analytical tasks. So therefore, any computer system that requires the use of analytical algorithms must:

1. Have the capability to adequately and accurately extract sonic features from a desired source relative to the analytical task
2. Have the relevant rules and perceptual models with which to analyse the supplied sonic features
3. Organise and represent the results of the analysis in relation to the required analytical task

To elaborate, with regards to stage one, a relevant data set must be defined; in order to ensure an analytical algorithm can complete a specific task, it must be supplied with the sonic features relative to the process. For example, for a task such as melody analysis, which can be applied to assess melodies and compare them with a bank of preexisting melodies, thereby finding or 'recognising' melodic familiarities from a specified phrase, the pitch, onset and duration data are necessary (and possibly amplitude depending on the algorithm's structure); if an incorrect or limited data set is used in relation to the defined task, the resulting analysis may not be accurate or, in the worse case, fail in the task it is required to complete.

For instance, if only duration data is supplied for the above melody analysis algorithm, the task will be unable to allocate pitches and the onset of note events, therefore failing to analyse the melodies of a chosen phrase, and rendering any comparison with a bank of melodic phrases redundant.

Once the relevant sonic features have been identified for the analytical task, there are four key methods of representing the sonic features from which an analytical algorithm can extract them. These methods are defined in the following example:

1. A pre-defined data set such as a musical score

2. A live symbolic source such as MIDI IN
3. A recorded acoustic audio source/s
4. A live acoustic audio source/s

This example highlights the capability of computational analytical algorithms to assess different sources and forms of data for both real-time and offline musical applications. Furthermore, a combination of methods can be used together for purposes such as live scoring of unspecified instrumentation; a pre-defined pitch set can define the pitches played by a input live acoustic source (method 1), with a timbre and onset analysis of the input live acoustic source for the consequent use of defining instrumentation and rhythmic content (method 4).

Examples 1, 3 and 4 of the four methods of representing sonic features listed above can be described in symbolic or subsymbolic terms, which in itself, raises challenges in relation to the best method of representing sonic features for a chosen task. For example, if a pre-defined data set is used for melodic pattern recognition, the data set could contain MIDI note events, which are symbolic, or as vectors containing melodic contours, which are subsymbolic.

If we are to assume an analytical algorithm's rule base is the same for both types of data (symbolic/subsymbolic), it would yield different results, and with differing degrees of efficiency. Perhaps the vector-based data set would produce a more efficient and higher quality result due to the nature of a melodic pattern recognition task, which suits the searching of recurring contours; iterating over a collection of single MIDI Note events to find such contours would be much less efficient and less accurate in comparison. Therefore, the use of symbolic or subsymbolic data and its impact on an analytical task must be carefully considered relative to the desired process.

With regards to obtaining sonic features necessary for analytical algorithms, the purpose of feature extraction is to obtain a defined sonic feature from the various musical representations of pre-defined data, live symbolic sources, live audio streams or recorded audio. The effectiveness of the feature extraction process itself is also

affected by the differences between symbolic and subsymbolic data. With symbolic data sets, musical feature extraction is a relatively simple task; a feature can be identified in relation to its corresponding symbol or collection of symbols, such as pitch from a MIDI note number or melody from a series of MIDI note events, but its application is limited to analytical tasks that function well with symbolic data representations such as harmony classification or chord transcription.

In contrast, accurately obtaining sonic features from subsymbolic sources is a considerably more complex process. When exclusively using acoustic sources, either live or recorded, a feature extraction algorithm must use low-level spectral data (a waveform), with no symbolic data from which to begin. From the outset, this implies that the application of such sources only performs well with analysis rules that function with low-level subsymbolic data such as gestural or timbral classification. However, through the use of psychoacoustic and mathematical models, it is possible to define symbolic features from low-level spectral data, which in turn, opens up a considerable number of possibilities for their consequent application to symbolic and subsymbolic analytical algorithms, with the ability to map the results to relative generative processes (detailed further in chapter 3.2 *A Brief Summary of Machine Listening*).

The various methods used in algorithms for musical composition are continually restructured and adapted by composers. The following examples are indicative of the variety of approaches composers have used to suit their particular compositional objectives; stochastic models, Markov chains, cellular automata, flow control and grammars, generate and test (GATs), expert systems, fractals, artificial neural networks, genetic algorithms and creation by refinement (CBR). This list is not fully inclusive of all algorithms used by composers, but it identifies the algorithms that are more commonly used for musical algorithmic processes and demonstrates the broad range of techniques a composer may wish to use.

As algorithms can grow in complexity depending on their context and application, so too does the problem of assessing the quality of the solution; the opportunities for producing solutions of a variable quality are that much greater, thus a more complex

algorithm does not necessarily produce a more satisfactory result. The quality of a result and its usefulness may well be a subjective judgment left to the composers concerned, but they must be realistic in their assessment of the success or quality of an output from an algorithm in terms of achieving the task it has thus been set.

Directly related to the problem of the quality of an output is the composer's application of an algorithm in a context that can be justified musically. Algorithms, much like the formalisms found in Western Art Music can be used for producing melody, harmony, structure and rhythm. This is not to suggest that algorithms must follow the rules found in Western Art Music, but that they can be applied to the same key elements of a musical composition. The mapping of an algorithm to one of these key features, much like the potential complexity of an algorithm requires careful assessment and monitoring by the composer and also helps to ensure an algorithm is fit for purpose.

Furthermore, in terms of *Genesis,* detailed fully in chapter 5 *The Genesis System*, many compositional techniques are applied using methods beyond the conventions of melody and harmony established in the Western Art Tradition. For example, algorithmic compositional methods making use of microsound techniques are free of the restrictions imposed by the concepts of the diatonic scale and its associated harmony by the dissection of a digital waveform into sound events of around 50ms. Each sound event is considered an element of the sum of its parts, permitting the manipulation of sound-objects at their most fundamental level. This creates primary structural modifications of a sound-object's timbre, thereby offering the ability to generate new sound-objects through acute adjustments of a sound-object at its micro structure, outside of the pitch/duration paradigm associated with Western Art Music.

Therefore, algorithms are able to perform many tasks. The following list, although far from exhaustive, exemplifies the scope and nature of the tasks algorithms can perform within a compositional process; automated mappings to any modifiable parameter of a chosen live synthesis technique, selection of melodic phrases, creation of rhythmic phrases, editing of macro structures, psychoacoustic analysis of data for use (or not) with associated mappings, random allocation of silences, comparison of chosen works

for the selection of similar external compositions to those chosen, suggestions of antecedents or consequents, automation of a generative performance, rearrangement of a set of selected melodic phrases, audible granulated responses to a live performer's input, complex mathematical equations mapped to the parameters of additive synthesisers and live graphical scoring. From this summary list alone it becomes clear that there are many possible applications for algorithms in a compositional context, but as already noted, their success or otherwise ultimately depends on the skills and judgment of the composer.

2.3 Computers and Algorithms

The purpose of using a computer in algorithmic composition can be divided into three general categories[19]:

1. Modeling traditional, non-algorithmic compositional procedures
2. Modeling new, original compositional procedures, different from those known before
3. Selecting algorithms from extra-musical disciplines

The reason for an algorithm's application can be considered highly contentious. For example, one may question the reasoning behind 'modeling traditional, non algorithmic procedures'[20]. A warranted question, as superficially, the need for a computer to complete a task we are able to do ourselves does not appear necessary but the context of this form of algorithmic procedure must be taken in to account. For example, later in this chapter, I demonstrate with *Chord Creator* such a case where it would be entirely possible for a composer to achieve the desirable outcome without computational assistance. However, the efficiency and accuracy with which the algorithm is able to complete the task certainly merits its application.

Using algorithms for 'Modeling new, original compositional procedures, different from those known before' and 'Selecting algorithms from extra-musical disciplines'[21] is perhaps less contentious in comparison to modeling traditional compositional

[19] Supper, M. 2001. A Few Remarks on Algorithmic Composition. *Computer Music Journal* 25(1): 48
[20] Ibid
[21] Ibid

procedures. The question of the algorithm's suitability however must always be asked in any critique. As discussed earlier, a composer must ensure that an algorithm is fit for purpose. An example of the importance of this notion is provided by Iannis Xenakis, a composer strongly involved in the use of strict formalist rules in composition and author of *Formalized Music: Thought and Mathematics in Composition* (1971). His works focused on testing an algorithm's suitability to a task as well as experimenting with algorithms based upon mathematical concepts such as game theory and stochastic processes.

The use of a computer is not essential to execute an algorithm. As stated before, it could be argued that *all* compositions use algorithmic functions, including those predating the digital age. The role of the computer is to increase efficiency; it is a tool that can calculate complex problems much faster than a human. It can also control many different variables simultaneously while we only have one pair of hands. It is the complexity of tasks, which necessitates the use of a computer to complete them. This is not to say that the computer's output is *better* than a human's, it is simply faster; the 'quality' of the result is the same (assuming both a human and computer have not made errors), in turn depending on the appropriateness of the criteria used to determine its actions.

Due to the efficiency and power of modern computers, real-time compositional systems can be conceived and applied to form real-time compositional processes such as those found in *Genesis*. Prior to modern computing, 'compositional algorithms were used 'out of time' (Xenakis 1971) for creating musical scores… a symbolic output in the form of a score list had to be translated into musical notation in order to be performed by musicians'[22]. Such a process would interrupt the direct flow of interaction between compositional outcomes and their realisation. In contrast, through the real-time compositional methods that modern computing affords, it is possible to generate and realise instantaneously, providing composers with immediately relatable outcomes to the compositional approaches they have applied. Therefore, real-time interaction between a compositional process and its realisation is achievable, enabling composers to create compositions in real-time.

[22] Essl, K. 2007. 'Algorithmic composition' in *The Cambridge Companion to Electronic Music*, eds N Collins & J d'Escrivan, CUP, Cambridge: 122

The deliberate and extensive application of algorithms and their contribution to a compositional process must thus be carefully deliberated over by a composer right from the outset. An algorithm is a tool with which is it possible to obtain a result as described by Supper (2000), but this is not to imply that by using an algorithm, a successful composition will always be written. Music, unlike the objective behaviour of algorithms that can respond *yes* or *no*, *1* or *0* or *true* or *false*, is everything in between a *yes* and *no*, *1* and *0* or *true* and *false* demonstrating Music's intrinsic subjectivity and stark contrast to objective algorithms. This sentiment must be addressed when composers are making decisions regarding the use of algorithms in their works, otherwise there is a risk that their composition may no longer be considered a musical work, but instead, as an objective process or series of processes that take on an identity of their own.

In addition, despite a composer's application of an algorithm or algorithms to a musical composition, much like the notion that an algorithm does not ensure a successful composition, neither does an algorithm necessarily make a great composer; it is how the composer has contextualised the result of an algorithm or algorithms in a musical composition that will ultimately determine a composer's 'success'. It is also key to note that the complexity of an algorithm does not affect the validity of a work; the composer's application of it, regardless of its complexity, is the most important factor when assessing the effectiveness of an algorithm or algorithms in a composition.

To elaborate, and in relation to Cage's extensive application of chance, which has a strong connection with algorithmic processes, Burt (1996), when discussing 'successful' use of chance in composition, states 'a popular misconception of the use of chance in art is that it should be judged by criteria of winning and losing. For many, winning means 'sounds like something I'm already familiar with', or, 'makes me happy in ways I know''[23]. However, the true value of applying algorithmic approaches to composition would appear to be its generative possibilities, allowing composers to search and explore for unique, unknown musical outcomes. For

[23] Burt, W. 1996. Some parentheses around algorithmic composition. *Organised Sound* 1(3): 167

example, Burt(1996) muses 'I, at any rate, find it much more valuable to use algorithmic methods as a means of finding out what I don't know, rather than making what I do know'[24].

Furthermore, through an algorithmic process, which can present the 'unknown', it is the composer's control over an 'unknown' event that impacts on its perceived success 'since the composer's control over events ceases after he has shaped his prescription, and since the prescription is necessarily very partial in the case of chance music, his suggestions need to be firm and striking if they are to produces distinctive results'[25]. Therefore, the perceived 'success' of a compositional product is accountable to the composer through the environment they have created and the prerequisites, conditions and boundaries contained within.

A composer must consider, along with the suitability of an algorithm for a composition, the *need* for a computer in a compositional method. As an example, the number of physical mappings would be a consideration if electroacoustic techniques were to be used. If the number of mappings were few, for instance, a linear curve assigned to the filter frequency on one channel, it would appear unlikely that any automation (and therefore a computational algorithm) would be needed as the performer has only one variable to control at that one time.

However, if the composer wished to set specific frequencies in 0.015s intervals, a computational algorithm could do this with digital precision, whereas a human performer would struggle to perform accurately the task within such a stringent time scale. It is therefore imperative that the composer understands the capabilities of both performers and computational algorithms; if these considerations are ignored, the artistic quality of the resulting realisation will be seriously degraded. It is also important to mention here that despite digital systems having an implicit reliability and accuracy, errors can sometimes occur and indeed can be deliberately precipitated allowing composers to explore the aesthetics of failure (Cascone, 2000; Vanhanen, 2003).

[24] Burt, W. 1996. Some parentheses around algorithmic composition. *Organised Sound* 1(3): 168
[25] Reynolds, R. 1965. Indeterminacy: Some Considerations. *Perspectives of New Music*. 4(1): 137

With the advent of modern computing, not only has the application of algorithms within the compositional process become more prevalent, but also, and perhaps more importantly, due to the procedural nature of all algorithmic processes, their use with computers allows for a symbiotic relationship to be established between computer and algorithm. This symbiosis between computer and algorithm gives the composer considerable power to capatalise on the possibilities thus opened up in establishing a compositional process. Algorithms can be applied at many different levels of the compositional method. For example, an algorithm such as the following written by the author in SuperCollider, called *Chord Creator*, may be used for the selection of the number of possibilities of *n* note chords, without octaves:

```
nNoteChord = { arg array, n;

var max = array.size - n + 1;
var index = Array.fill(n, 0);
var end = Array.fill(n, max);
var auxIndexValue;
var auxRes, res = [];

while({index != end}, {
                // subset
                auxRes = [];
                index.reverse.do({ arg i, j;
                auxRes = auxRes.add(array[j + i]);
});
res = res.add(auxRes);

// index change
index[0] = index[0] + 1;
(index.size - 1).do({ arg i;
if(index[i] == max, {
          auxIndexValue = index[i + 1] + 1;
          (i + 2).reverseDo({ arg j;
          index[j] = auxIndexValue;
          });
          });
        });
        });
res;
};
```

```
bench { shoAll = nnoteChord.value((shoArray), voices); };
shoOut = shoAll.select{ |x| (x%12).asSet.size == voices};
shoSelector = shoOut[start.asInt..finish.asInt]
```

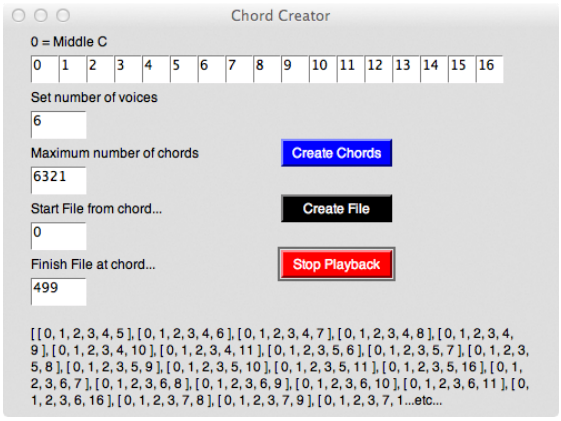


Figure 1. Chord Creator GUI

Figure 1 illustrates the graphical user interface applied to implement the *Chord Creator* algorithm. The composer can thus set the number of voices, the root note of the chord and the notes above the root that they wish to use. From this data, the algorithm calculates every possible chord of six notes without octaves, which, for in this case, there are 6321. The algorithm will then audibly play back the output in sequence. There is also the option of saving the data to a MIDI file available for use in any compatible program such as Sibelius or Logic. *Figure 2* below shows an excerpt of this final output opened in Sibelius ready for editing and playback.

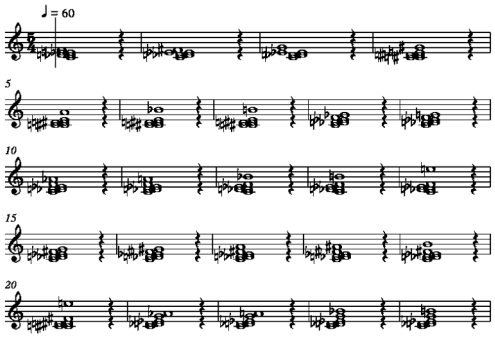


Figure 2. MIDI Output of Chord Creator

The composer determines the level of application to a compositional process by the results generated by Chord Creator. For example, the composer may wish to use the raw data thus generated to determine both the harmonic and rhythmic content of the entire work. This gives the algorithm a high-level of control over a composition's harmony and rhythm. On the other hand, the composer may selectively choose chords and associated rhythmic characteristics from the resulting data and also choose to dictate their rhythms (perhaps using another algorithm such as Cage's coin flip), hence fulfilling the reverse of the previous example, thus giving the algorithm a lower level of control over a composition's harmony and rhythm. These examples represent different approaches to the application of such an algorithm to these two key aspects of a musical work, but by no means embrace the range of possibilities that are thus available.

The following table identifies the five key attributes that influence, and in many instances determine, the use of algorithms within a composition:

1. Human Control – Free of all algorithmic methods (composer's perception)
2. Low Level – Algorithms used but composer decides upon result
3. Mid Level – Composer explicitly uses algorithms but uses own intuition
4. High Level – Algorithms decide upon result. Composer oversees
5. Algorithmic Control – Algorithms control result. (composer's perception)

In relation to the breadth of influence an algorithm may have, it is ultimately determined by the composer's application of it, so its relative influence to the macro or micro level of the composition itself is a matter of choice on the part of the composer. So, for example, if an algorithm was chosen to select pitches for one phrase of a 128-phrase composition, on the macro-level is has a low-level of influence, but on the micro-scale, for that chosen phrase it has a high-level of influence. Of course, the importance of events in both the macro and micro-scale are subjective, so the phrase controlled by the algorithm could be considered influential on the macro-level, but this is merely an arbitrary example of how the influence of algorithms must be considered in context of their place in a composition. The

‘composer’s perception’ marks the subjectivity involved in the use of algorithms, formalisms, rules and structures. As stated earlier, algorithms, formalisms, rules and structures are argued to be intrinsic to the composition of all musical works, but a composer may wish to dispute their use or not; perhaps the ego can be held accountable for such contrary beliefs.

Considering again the *Chord Creator*, the composer may wish to use another algorithm in combination with outputs generated by *Chord Creator*, replacing the composer’s role in the selection process of the chords and therefore giving this algorithm a high-level of influence on the creative process. As suggested, this could be a coin flip or other indeterminate process, but there are more linear and predictable approaches to data selection. For example, the use of feature extraction and search-based algorithms are an effective way of finding solutions with little or no input from the composer other than stating the rules by which the algorithm will make its choices. Such an algorithm could use *if* and *while* statements stating the feature the composer wishes to search for. So, if a particular feature is found, a relative result will be output. If not, the algorithm may do nothing, or propose a new, creative output. The pseudo code example below shows how this could be put into practice:

```
// If the note equals the composer's specification, output the note
if ( note == note,

{
output = note
});


// If the note does not equal the composer’s specification, randomly generate a note value of 3, 4 or 5
if ( note != note,

{
output = rrand(3,5);
});
```

There are many options available to a composer for algorithmically generating values such as mathematical models, Markov chains, grammars, genetic algorithms and

neural networks. Such methods are considered artificial intelligence, which is the 'science of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but does not have to confine itself to methods that are biologically observable'[26]. Through the use of the algorithms proposed, it is possible for a computer to propose creative ideas based upon the rules of the system.

The level of intelligence demonstrated by a system that applies artificial intelligence techniques however, cannot be easily defined as 'we cannot yet characterise in general what kinds of computational procedures we want to call intelligent. We understand some of the mechanisms of intelligence and not others'[27]. It is therefore difficult to categorise definitively which algorithmic processes within computational algorithmic musical composition demonstrate intelligence and at what level. Despite this, it is possible to apply various branches of artificial intelligence research to musical composition for creative proposition tasks with methods such as *search*, which examine large numbers of possibilities, or *learning from experience*, which continually adapt and learn behaviours from their numerous states.

Reflecting on the proposed artificial intelligence an algorithmic *process* may be considered to demonstrate, the composer must be aware that the authorship of a composition may come into question. For example, 'much of the resistance to algorithmic composition that persists to this day stems from a basic misunderstanding that the computers compose the music, not the composer'[28]. This belief centres on the presumption that the composer has no input in the design and construction of the algorithmic procedures that define the resulting composition, but instead, applies responsibility to the computer system, which generates the music. Moreover, such an idea would conclude that both the *process* and *product* are consequences of a computer's own making: that the entire authorship and accountability of a computational algorithmic composition is solely attributed to its digital source.

[26] McCarthy, J. 2007. *What is Artificial Intelligence?* Stanford University: 2
[27] Ibid: 3
[28] Edwards, M. 2007. *Algorithmic Composition: Computational Thinking in Music.* University of Edinburgh: 9

In contrast, 'Curtis Roads points out, it takes a good composer to design algorithms that will result in music that captures the imagination'[29]. Therefore, resolving this notion, authorship of such music implicitly requires a *human* composer who has programmed an algorithmic methodology with a *process* and *product* attributed to them. However, there must be a consideration of the divide between the authorship of the *process* and *product* in relation to the roles between the composer and the machine. For example, is the *product* accountable only to the machine that made a realisation possible, disregarding the composer's programmed *process*? Or is it accountable to the composer who designed a *process* through which a *product* could be realised by the machine?

There are a number of methods that attempt to discover the accountability of an algorithmic *product* such as the Turing Test[30] and the Lovelace Test[31]. Both tests endeavor to determine the extent of a machine's intelligence, with the Lovelace Test focusing on a machine's ability to *create* autonomously. In particular reference to the Lovelace Test, its fundamental principle is to ascertain whether a programmer can account for all actions of an algorithmic *process* that led to a *product*, with those that cannot be accounted for considered creative and accountable to the machine, rendering it a truly autonomous agent. However, as yet, no such system has conclusively passed the Lovelace Test, which would negate any such notion that *any product* of a computational algorithmic methodology is the full responsibility of a machine.

So, it must be proposed that accountability and authorship of *products* generated by algorithmic *processes* must be attributed to a human composer. However, the degree to which this responsibility is recognised *must* be relative to the composer's chosen level of influence of any implemented algorithmic procedures. As a result, the authorship and accountability is variable from composition to composition with the necessity to always acknowledge the use of a human composer as the instigator for a resulting algorithmic *product*. This is discussed further in chapter 4.1 *Interaction with Creative Systems*.

[29] Edwards, M. 2007. *Algorithmic Composition: Computational Thinking in Music*. University of Edinburgh: 9
[30] Turing, A.M. 1950. Computing Machinery and Intelligence. *Mind* 59: 433-460
[31] Bringsjord et al. 2001. Creativity, the Turing test and the (Better) Lovelace Test. *Mind and Machines* 11: 3-27

2.4 Unpredictability and Randomness in the Creative Process

Primarily, it is necessary to highlight the distinction between unpredictability and randomness as the two concepts have differing implications on the creative process. An absolute unpredictable outcome is one that cannot be foreseen. However, it is possible to consider a *relative* unpredictability in which a number of events may be predictable, with an outcome comparative to its probability of occurring. Therefore, specific conditions can be set through which events may or may not transpire, relative to the probabilities of each event's outcome. As a result, it would appear that the implementation of relative unpredictability in a compositional process, and thereby constructing a scenario in which a number of predictable *known* outcomes can occur, is applicable to the proposed models of creative processes based on structures and order, which enable relative predictions to be made.

On the other hand, absolute randomness is the absence of order, and consequently cannot be predicted due to the privation of rules that enable the prediction of its outcomes through probability. However, a *relative* randomness can be proposed, one that is 'the lack of order or structure *relevant to some specific consideration*… identified by reference to something people might have regarded as relevant… the potentially relevant 'something' is usually the creator's own knowledge, the structure of conceptual constraints into which the novel idea may be integrated'[32]. Therefore, the result of such randomness, although unpredictable, can still form an acceptably creative idea, relative to a composer's compositional aims.

Consequently, both unpredictable and random methods of idea generation are proposed to form part of the creative process. However, as Boden (2005) states 'our ignorance of our own creativity is very great. We are not aware of all the structural constraints involved in particular domains, still less of the ways in which they can be creatively transformed'[33]. Thus, it is possible to include unpredictability and randomness in a creative process but their specific roles in its structure are uncertain.

[32] Boden, M. *The Creative Mind: Myths and Mechanisms.* New York: Routledge: 239
[33] Ibid: 246

Boden (2005) proposes psychological (P-creative) and historical (H-creative) components in the creative process. P-creative are 'ideas (whether in science, needlework, music, painting, literature…) that are surprising, or perhaps even fundamentally novel, with respect to *the individual mind* which had the idea'[34] with H-creative being ideas 'that are novel with respect to *the whole of human history*'[35]. However, 'whichever type of creativity is involved, it's historically creative if no one has had thought before'[36]. Therefore, a P-creative idea is one in which the *individual* has not thought of before, unique to them, whereas an H-creative idea is one that has not occurred throughout human history.

With regard to the role of unpredictability and randomness for such creative ideas, 'many P-creative ideas can actually be predicted. For instance, people typically ask certain exploratory questions, and notice certain structural facts… All H-creative ideas are (so far as is known) *unpredicted*, since an H-creative idea is one which (again, so far as is known) no one had ever thought of before'[37]. So, in relation to a P-creative idea, although as Boden (2005) states that such an idea can often be predicted, the presence of *relative* randomness, and therefore the acceptance of a random event considered by a composer to fit the conceptual constraints of a composition, resolves that P-creative ideas can indeed be just as unpredictable as H-creative ideas. As a result, the previously proposed notion that relative random events can be used to propose creative ideas within a creative process can be validated for both the P-creative and the H-creative.

Considering the function of randomness in a truly unpredictable, H-creative idea, it could be argued that absolute randomness is intrinsic to such a creative process: if an absolute randomness were implemented 'there is the total absence of *any order or structure whatever* within the domain concerned'[38], rendering an H-creative idea absolute random due to its inexplicable and unpredictable occurrence. However, as the creative process is believed to be formed of a variable *structure*, relative to the individual, then it may be thought implausible that the result of such absolute

[34] Boden, M. *The Creative Mind: Myths and Mechanisms*. New York: Routledge: 43
[35] Ibid
[36] Ibid
[37] Ibid: 233
[38] Ibid: 239

randomness can be considered creative, or attributed to any subject other than absolute random.

Furthermore, the notion that an H-creative idea is an absolute unpredictable would appear questionable itself; how would one go about testing the series of events that led to an apparent absolute unpredictable? Conceivably a method similar to the evaluation approach of the Lovelace Test could be appropriate, requiring every individual event to be accounted for, necessitating *every* event to be unpredictable and random to then qualify the outcome as an H-creative. Therefore, the concept of an H-creative idea is perhaps flawed, that no idea is truly unpredictable, that creativity is founded in determinism, that all ideas exist, have existed or will exist. Yet, determinism cannot predict *all* creative ideas (if that were true, surely all creative ideas would have occurred or have been predicted), implicating that indeterminism, with its acceptance of random, unpredictability and stochastic processes, can offer insight into the methods behind our creative process and resulting creative ideas, both P-creative and H-creative.

Since both deterministic and indeterministic ideologies can be applied to the creative process, this validates the assertion that the use of both unpredictability and randomness are suitable methods for the generation of creative ideas. However, when deliberately attributing such methods to a creative process, that is forming compositional approaches that embrace unpredictability and/or randomness, the consequences of their outcomes can differ substantially.

Considering the key difference, unpredictable methods such as Markov chains have a *relative* unpredictability whereby the output values are predetermined with each individual output constrained by an implemented probabilistic model. On the contrary, random methods such as white noise generators cannot be predicted, and are instead the result of uncorrelated random variables, reflecting their absence of order. Therefore, it is suggested that the *methodical* application of unpredictability and/or randomness in a compositional process is the choice of the composer, with the need for a full understanding of the implications of the outcomes on the creative process.

The use of random functions is perhaps the simplest method to generate a creative idea. However, due to the lack of order and structure in randomness, such an approach may not be considered the most efficient: if a composer is searching for a *specific* idea, then application of *relative* unpredictability would be more suitable as possible outcomes are known, and are therefore known to occur relative to their probability of happening.

Random affords a significantly more serendipitous method of acquiring creative ideas, in which an outcome can be applied that is not being sought, thereby generating results that are not known prior to the instigation of a compositional process. Yet, the composer must be willing to accept that the outcomes of a random function must be concluded by the composer themselves: each result is related to a composition *through* the composer, thereby indicating the importance of understanding that randomness is able create a relatable idea just as it is unable to create a relatable idea. Therefore, the efficiency of using randomness in a compositional process is comparative to a composer's conceptual constraints of a composition, and to what extent this allows the results of such a random compositional process to influence the resulting work.

Random leads to unpredictable and non-linear outcomes, which may be a composer's intentions. However, as highlighted previously, if an entire composition is made up of random functions, therefore implementing absolute randomness, the attributed subject of a compositional process must be considered, and therefore the question of 'who is the composer?' must inevitably be asked. The significance of such a question will largely depend on the level at which the composer has control over the random functionality of an algorithm and what bounds have been imposed on its application, thus the *relative* randomness within the compositional process. For example if a composer stated 'any value from infinity', the level of randomness is maximal with absolutely no constraints. In contrast, if the statement was 'any value between 0 and 10', such an operational restriction facilitates much more control over the output. Furthermore, it is possible to add further constraints such as only odd or even numbers, only prime numbers, only powers and so on.

A composer must not only carefully choose the bounds of any random function, but also the mapping to which is related. For example, if a composer were to choose 'any value from infinity' to map to a pitch, the resulting data stream might overwhelm a loudspeaker and/or greatly confuse an associated instrumentalist. The importance of mapping effectively between the constraints of a random function (or any type of function for the matter) is paramount. With random however, it is even more important that the composer is aware of all possible outputs, otherwise the result may be unplayable.

Therefore, as highlighted previously, the outcomes of *absolute* randomness are certainly not creative ideas: considering alone the issue of mapping such outputs to instrumentation, this negates the plausibility of *absolute* randomness as a compositional tool. However, the following code demonstrates a method through which *relative* randomness can be applied to a compositional process:

```
//Array of all notes

arrayOfNotes = [note0, note1, note2, note4, note5];

// If the note does not equal the composer's specification, make a 'suggestion'

if( note != arrayOfNotes[i],

{

//What is the prior note?

priorNote = arrayOfNotes[i-1];

//Check for 8ves

if(   (priorNote != arrayOfNotes[i])
&& (priorNote+12 != arrayOfNotes[i])
&& (priorNote+24 != arrayOfNotes[i])
&& (priorNote-12 != arrayOfNotes[i])
&& (priorNote-24 != arrayOfNotes[i]) ,

{
```

```
//If priorNote is not the same as note

if(   priorNote != arrayOfNotes[i],

{

noteOutput = rrand(priorNote, arrayOfNotes[i])

});


//If priorNote is the same as note

if(   priorNote == arrayOfNotes[i],

{

noteOutput = rrand(priorNote+1, arrayOfNotes[i])

});

});

});
```

This example constricts the random search process algorithm considerably, using the prior notes to ensure no repeats, as well basing the random function's bounds between the prior note and the current note, prioritizing the removal of octaves above all other rules, resulting in a serendipitous response to a finite set of results. So, although the selection of the chosen note's output is unpredictable, the choice is relative to the compositional process and the conceptual constraints of the composition.

When considering random data generation techniques and their efficacy and efficiency, an interesting question arises with regards to autonomy. If a system were to use only random functions, does it become truly autonomous? As highlighted previously, the implementation of absolute randomness may result in a composition that cannot be attributed to a subject other than absolute random. The complex tasks, calculations and routines required by algorithms making use of neural networks, expert systems, and various grammars are often implied to have the objective of being

or becoming autonomous. If random is taken in to account, is it not itself acting autonomously, governing itself by its randomness? So, should this mean that all autonomous systems require random functionality to call themselves truly autonomous? It is not within the remit of this thesis to offer a possible solution to consider the philosophical issues thus arising in a wholly definitive manner, but a considered awareness of the resulting implications must be taken into account appropriately.

2.5 Further Considerations of Applying Computational Algorithms within a Compositional Process

The nature of a task an algorithm is required to execute will have a great impact on the number of calculations required from the computer. For example, selecting a random integer from a set of chosen bounds to decide the pitch of a single instrument in an offline method demands very little processing power. On the other hand, if an algorithm is written that relies on a continuous loop of feature extraction of a live instrument with the comparison of this feature to selected variables in order to choose the frequencies of a filter bank, the number of processes will significantly increase. In this case, the algorithm must be especially efficient in terms of extracting data and using the information thus produced to select the variables of the filter bank.

This highlights a fundamental issue that warrants further consideration: at what level can algorithms be used to control and influence composition? In demonstrating the capabilities of the *Chord Creator*, it was shown that algorithms could be applied at different levels of the compositional method. An algorithm can be employed to construct and control a significant proportion of a composition's material, for example, by using cellular automata to determine the pitch, duration and onset of sine waves over a specified period. Cellular automata are 'discrete, abstract computational systems… composed of a finite or denumerable set of homogenous, simple units, the *atoms* or *cells'*[39]. The system's cells evolve over time, with their evolution dictated by a set of transition rules and update functions, such as the *Game of Life*[40]. Therefore,

[39] Berto, F. and Tagliabue, J. (2012). *Cellular Automata*. [online] Plato.stanford.edu. Available at: http://plato.stanford.edu/entries/cellular-automata/#Bib [Accessed Feb. 2015]
[40] Gardner, M. 1970. The Fantastic Combinations of John Conway's new solitaire game "life". *Scientific American* 223: 120-123

once the cellular automata have been initiated, the pitch, duration and onset is determined entirely by the rules of the model, with no external manipulation of the audible outcome other than the choice of sine waves or other waveforms for the instrumentation.

The example of the use of cellular automata displays how an application of a chosen method can have a major impact on a composition, and therefore a high level of influence. It is important to make a distinction however that, by using a specific model as in that example, does not necessarily mean that the model's application is restricted exclusively to the specific task chosen in that instance. In terms of the level of influence and control algorithms may have on a compositional method, a composer must use discretion not only in the methods they wish to use, but also the manner of their application.

To use the cellular automata example again, the composer has many options of how the model may be applied such as selecting only pitches, or the composer may wish to review the output, editing the result to their own requirement, or, to extend the use of algorithms within a compositional method, execute an algorithm that decides an output by reviewing the result of the model's calculation, in effect acting as a fitness-for-purpose function.

There are many methods of obtaining data for review in association with an algorithm and a major part of considering which methods to use is the type of data a composer wishes to submit for analysis. As an example, *Chord Creator* writes a MIDI file, which contains a detailed amount of symbolic data including note duration, pitch, and onset. Depending on the requirement of a chosen algorithm's task, for example, to find notes of a specified duration, pitch and onset, the resulting MIDI file from *Chord Creator*, or any MIDI file for that matter, is a suitable format for a 'note/event' algorithm to analyse.

For the same task of obtaining pitch, duration and onset information, the MIDI format is nevertheless totally unsuitable in a non-symbolic situation such as a live analysis of

the acoustic data generated by an acoustic instrument; although there may be a requirement by the composer to score the live analysis to a MIDI format, this process cannot be completed until the pitch, note duration and onset characteristics have been extracted by an algorithm from the low-level spectral data of a live acoustic instrument's waveform.

To elaborate on the difference between using a MIDI file or any other saved data and live streams for use by an algorithm for analysis, it must be noted that MIDI files or saved data, as far as a computer is concerned 'exist' as data prior to an algorithm being executed, allowing for offline processing operations to be completed at the composer's convenience, prior to the use of the results. On the contrary, the real-time processing of live streams requires either loops that run at specified time intervals or external triggers such as amplitude peaks found by amplitude followers (which themselves require looped routines), gathering data at the time of their respective execution. In such circumstances it may not be possible to generate the results with the required immediacy, giving rise to undesirable latency. A composer must address these differences appropriately distributing tasks between saved data and live streams prior to algorithm selection for a composition, otherwise the outcomes may suffer from data corruption, or at worst, a non-recoverable termination of the program.

To expand on the issue of latency and time in live performance, the requirement to eliminate any audible timing errors in music can be a limiting factor in the methods of both analysis and generative algorithms. Although the ear in most circumstances will not recognise small timing errors of the order of 20 milliseconds or less, this leaves a very small window in computing terms for accommodating variations in processing requirements from sample to sample. Added to this essentially practical problem, and with particular reference to analytical algorithms, there are substantive issues to be considered in terms of the nature of these processing algorithms, and the ways in which we react subjectively to the varying complexities of the results. Thus if we consider our own listening experience, it is the relationship of events over time that allows us to analyse, contextualise and assess the connection between events and as a result, give them musical value.

In an offline situation, it is possible for a computational analysis algorithm to 'listen' to saved data such as a WAV, AIFF or MIDI file and thus extract all the information necessary for the intended computational process. Although the difference between using non symbolic musical data as of that found in WAV or AIFF files and the symbolic musical events within MIDI files and the possible issues this may cause in terms of creative intention has been highlighted previously, the use of algorithms with saved data does not necessitate real-time, or 'on the fly' processing; an entire file can be 'assessed' with the efficiency of its result limited only by the time the algorithm requires to complete the calculation. Therefore, in offline scenarios, it is possible to process the stored source data in advance of a performance, which, in some circumstances will prove an expedient and effective solution. However in a real-time analysis situation, such stored data is unavailable as data and must be constantly streamed, only representing what *has* happened as opposed to both what *has* and *will* happen, as represented in a saved and compiled finite data set.

As a result, algorithmic iterations can occur within musical time (in a live scenario) or outside of musical time (in an offline scenario). For example, the duration of algorithmic iterations may be set to the chosen musical time intervals of a live performance, perhaps following a strict a rhythmic metre of 110bpm. In contrast, the algorithmic iterations may be run out of musical time in an offline scenario where the constraints of such a metre and time are unnecessary, such as a Schenkerian analysis of Mozart's *Haydn Quartets*.

However, it is important to note that live algorithmic iterations can be set to specific durations outside of the rhythmic metre as well as to a stated metre of a live musical time scale. For instance, an algorithmic iteration can follow a time scale of 1/30s to obtain relatively accurate loudness data while another algorithmic iteration can generate filter resonance values relative to a selected rhythmic metre of 90bpm. The loudness could then be means averaged over the interval between each beat (0.67s) with the averaged value from around 22 loudness values used to modify the filter resonance parameter. Therefore, one algorithmic iteration follows the rhythmic metre, with the other occurring outside of this. Thus, algorithmic iterations can follow musical time relative (or not) to a selected metre, offering the composer a wealth of

options to apply strict musical intervals alongside the onset of time outside of such musical intervals and durations.

In reference to the listening experience, although deeply personal and subjective, it is apparent that a reactionary and predictive process could take place; we have a reaction to events, defining their value and their possible progression based only on the events heard, or we may attempt to predict a consequent event based upon events previously heard and our prior knowledge. When considering algorithmic composition, the two approaches of reactionary and predictive processes can be used independently of one another or in combination. The choices made in this context, however, can have a considerable effect on the time required by an algorithm or combination of algorithms to output a result. For example, depending on the window size given for the time to gather event data, the length of time an algorithm is required to perform a task can be divided by the window size, and at the end of each time division, a result can be produced.

So, if for instance, pitch events are chosen and assuming the feature extraction algorithm can accurately identify separate pitch events over a chosen phrase of thirty-two pitch events, the algorithm could use a window of every four events to produce an output, therefore considerably reducing the amount of time needed to calculate a response by a factor of eight, from every 32 events to every 4. This choice of using groupings of four events at a time and/or consolidating the contents of each window to create a larger table for analysis are just some examples of how such an algorithm can be adapted by a composer for use in musical composition.

It is therefore clear that there are many considerations a composer must take into account when making extensive use of algorithms in their composition. Perhaps the most important is the level at which they wish an algorithm to influence the resulting composition and the suitability of an algorithm for a chosen task. Along with these notions, a composer must be pragmatic in considering what they wish an algorithm to do, and this highlights the need for awareness by composers of the capabilities of algorithms prior to their use in a composition; there must be a sense of realism when passing a significant amount of compositional tasks to an algorithm, because

currently, an algorithm can only act within the bounds it has been set and nothing more.

In many respects the above account raises more questions than it answers, not least in terms of the possible relationships that can be established between a composer and the possibilities of computational processing in the production of works that meet the creative expectations of the originator. A fundamental objective of this thesis is to develop a deeper understanding of the ways in which such processes can be enhanced both in terms of their operational characteristics and also the means by which they may be interactively controlled via a suitable interface. It is through this advancement of knowledge and understanding that it is hoped that these fundamental issues of musical creativity can usefully be progressed.

Chapter 3

Real-time Computational Algorithmic Systems in Musical Practice

3.1 An Introduction to Real-time Generative Algorithmic Systems

As highlighted in chapter 2 *An Introduction to Algorithmic Composition*, composers have explored the application of computers to implement a variety of algorithmic compositional methods to generate musical compositions within the three categories proposed by Supper (2000). The real-time functionality of such algorithmic methods allows algorithmic procedures and outputs to occur in musical time thereby offering composers and performers instantaneous results to an unfolding musical dialog, relative to a defined creative process.

The *Genesis* system uses an assortment of real-time generative algorithmic processes to generate its outputs based upon its inputs and interactions with the user for consequent control of a series of granular synthesisers. It is therefore necessary to contextualize the real-time algorithmic procedures used within *Genesis* and the methods through which other systems applying similar algorithmic approaches have offered users interactivity (or not) with the musical composition process.

The use of real-time computational algorithms that apply significant levels of unpredictability and randomness are applied extensively within *Genesis*. As proposed in chapter 2.4 *Unpredictability and Randomness in the Creative Process*, the application of algorithms applying such procedures is a valid method of musical composition and creativity. To elaborate on the use unpredictability and randomness in music, indeterminate processes have defined musical genres by composers such as John Cage ('chance music'); Karlheinz Stockhausen, Pierre Boulez and Luciano Berio ('aleatoric music'); and Iannis Xenakis ('stochastic music')[41].

Stochastic models are one such method for the application of indeterminate processes to music. Considering its exclusive role in indeterminate methodology, it 'is based on

[41] Bokesoy, S and Pape, G. 2003. Stochos: Software for Real-Time Synthesis of Stochastic Music. *Computer Music Journal* 27(3): 33

a process in which the probabilities of proceeding from one state, or set of states, is… defined. The temporal evolution of the process is therefore governed by a kind of weighted randomness, which can be chosen to give anything from an entirely determined outcome, to an entirely unpredictable one'[42]; the composer is able to have substantial control of the level of indeterminacy governing a chosen process, and therefore the amount of relative unpredictability within an algorithmic procedure.

If we now consider in more detail the impact of probability on the application of algorithmic processes, it is important from the outset to recognise that the bounds controlling a relative unpredictability have a significant effect on the scope and nature of the resulting output (as discussed in Chapter 2.4 *Unpredictability and Randomness in the Creative Process*). The definition of each individual stochastic process, set by the composer, gives the option of applying a composer's requirements of predictable, linear results or unpredictable, chaotic results, and many of the variations in between.

Perhaps the most prominent use of stochastic processes in 20th Century musical composition has been 'stochastically distributing sonic sound events in sound space as first realised by Iannis Xenakis, beginning with his work *Achorripsis* (1957)'[43]. As Xenakis continued researching and studying stochastic models and their uses for musical composition, he constructed the 'Dynamic Stochastic Synthesis'[44] concept. This is 'an approach to microsound synthesis that uses probability distributions to manipulate individual digital samples, as if they were indivisible elementary particles'[45].

Dynamic stochastic synthesis is an example of nonstandard synthesis, which can be described as the 'manipulation of individual digital samples. Amplitude and duration values are obtained through musical procedures, they are not based on an acoustical model'[46]. Therefore, it allows for the application of real-time stochastic processes for microsound composition, instead of more conventional harmonic approaches such as those found in the melody generation systems Probabilistic Model to Melodic

[42] Wishart, T. 1994. *Audible Design*. York: Orpheus the Pantomime Ltd
[43] Bokesoy, S and Pape, G. 2003. Stochos: Software for Real-Time Synthesis of Stochastic Music. *Computer Music Journal* 27(3): 33
[44] Ibid
[45] Luque, S. 2011. Stochastic Synthesis: An Overview. *proceedings of the Xenakis International Symposium*: 1
[46] Luque, S. 2006. *Stochastic Synthesis: Origins and Extensions*. Royal Conservatory, The Hague: 7

Segments Generation (PMMSG) by Carbonera and Silva (2005) and Musical Weighted Synchronous Calculus of Communicating Systems (MWSCCS) by Ross (1995).

The aim of the dynamic stochastic synthesis method is to 'unify the macrostructure and the microstructures of compositions, to use synthesis techniques idiomatic to computers and to open an experimental field in sound synthesis'[47]. Xenakis describes the ways in which this is to be implemented through the use of[48]:

- Mixing 'pure' electronic sounds with 'concrete' sounds
- Stochastic processes to efficiently produce sonorities with 'numerous and complicated' transients
- An approach in which sound synthesis is performed only in the time domain

These descriptions are each important in relation to the aesthetics of microsound composition. Microsound composition is the use of grains or 'sound quanta'[49] comprising of very short sound events (under 50ms) from either synthetic or digital waveforms. The relationship between the 'sound quanta' and the control of them are the focus of this compositional technique.

Iannis Xenakis' computer program *GENDYN* (1992) reflects the aesthetics he defined for microsound composition. *GENDYN* (1992) is a stochastic algorithm that uses the 'mathematical concept of random walks to produce both duration structure and timbral fluctuations in computer-generated sound. This means that the probabilistic movement of random walks is used for wave-shaping sound synthesis as well as for controlling aspect of musical form (i.e. composing a 'score')'[50], therefore applying a stochastic model for the description of wave shapes and their durations (Random walks are most commonly applied through the use of Markov chains, which are 'discrete systems, in which the present outcome depends on a number of previous outcomes. In other words, the present outcome is not independent, but the process has

[47] Luque, S. 2011. Stochastic Synthesis: An Overview. *proceedings of the Xenakis International Symposium*: 1
[48] Ibid
[49] Xenakis, I. 1960. *Grundlagen einer stochastischen Musik*. Berlin: Ars-Viva-Verlag
[50] Hoffmann, P. 2000. The New GENDYN Program. *Computer Music Journal* 24(2): 31

'memory' of the past events that affect the future'[51] allowing a user to weight a probability distribution for particular events to occur).

In terms of the compositional results produced by Xenakis' *GENDYN* (1992), it is proposed by Ikeshiro (2011) that its outputs are considered noise music, a genre which itself is not immediately quantifiable; it 'appears to be contradictory by satisfying the conditions of both noise and music'[52]. Such a notion raises many aesthetic questions surrounding the issues of 'what is noise?' and 'what is music?'(see Cascone, 2000; Hegarty, 2007; Kelly, 2009). However, for the purposes of this thesis, it is proposed that noise, that is sound which features no fixed periodicity such as those created by *GENDYN* (1992), can constitute valid sound-objects for use in musical composition.

It is key to point out that the resulting noise music generated by *GENDYN* (1992) is not a circumstance of using stochastic models, rather that it is due to the dynamic stochastic synthesis method. Therefore, it is the realisation approach that results in the proposed noise music that is formed as opposed to the application of probability distributions to dictate the unfolding dialog of the composition's structures; as stated, microsound techniques are inherent to the formation of sound-objects in *GENDYN* (1992) and the assimilation with stochastic models is the causal factor in generating such noise music.

With regards to the method of controlling the random walks in the synthesis method in *GENDYN* (1992), once the system is executed, the values of the probability distributions cannot be changed, resulting in no real-time interactivity between the composer and the system. The consequence is that that 'the spectrum of probabilistic functions allows for one only global property to emerge, an ineluctable rush toward the average, final point or 'mean state value' (i.e., *stochos*, destination, destiny)'[53]. So, the compositional outputs of *GENDYN* (1992), although unpredictable in terms of their microstructures, are entirely predictable in their macrostructures, always

[51] Järveläinen, H. 2000. *Algorithmic Musical Composition*. University of Technology, Helsinki: 3
[52] Ikeshiro, R. 2011. *GENDYN and Merzbow: A Noise Theory Critique of Xenakis' Dynamic Stochastic Synthesis and its Relevance Today*. Goldsmiths College, University of London
[53] Di Scipio, A. Systems of Embers, Dust, and Clouds: Observations after Xenakis and Brün. *Computer Music Journal* 26(1): 25

concluding to a relative knowable outcome.

Composers of stochastic methods in musical composition have gone on to adapt Xenakis' *GENDYN* (1992) algorithm, adding their own techniques and influences in stochastic process. *Stochos*[54] and the 'new *GENDYN* program'[55] are two examples, which build upon the original *GENDYN* (Xenakis, 1992) algorithm. *Stochos* provides 'multiple control sources working in parallel to manipulate the sonic parameters on any event time level…'[56], a 'flexible algorithm, which distributes the events by assigning probability distributions for onset and time event duration…'[57] and a density parameter which is controlled by the probability distributions found in Xenakis' *Achorripsis* (1957).

The 'new *GENDYN* program' is a 'reimplementation of dynamic stochastic synthesis in a graphical, interactive, real-time environment'[58]. Both systems introduce interactivity between the user and the ongoing stochastic processes, thereby presenting an increased variety of unpredictability in terms of the resulting macrostructures of each system's outputs relative to modifications of the probability distributions made by the user. Furthermore, emulations of *GENDYN* (1992) algorithm have been written, such as the SuperCollider class GENDY1[59] which manipulates a grain's amplitude and duration based on the processes within *GENDYN* (1992), reflecting the impact and importance of Xenakis' compositional work.

In contrast to the application of stochastic models to algorithmically generate creative processes that feature extensive unpredictability, self-organising systems offer composers the ability to obtain both determinate and indeterminate outcomes of a desired creative process. A self-organising system is a mathematical model that uses a rule base to define local interactions between its values, with the culmination of these values resulting in global structures. Therefore, the development of the interactions at the local level influences the outcome on a global level. Considering such a system's

[54] Bokesoy, S and Pape, G. 2003. Stochos: Software for Real-Time Synthesis of Stochastic Music. *Computer Music Journal* 27(3): 33-43
[55] Hoffman, P. 2000. The New GENDYN Program. *Computer Music Journal* 24(2): 31-38
[56] Bokesoy, S and Pape, G. 2003. Stochos: Software for Real-Time Synthesis of Stochastic Music. *Computer Music Journal* 27(3): 34
[57] Ibid: 36
[58] Hoffmann, P. 2000. The New GENDYN Program. *Computer Music Journal* 24(2): 31
[59] Collins, N. 2012. *Gendy3 Class Help File.* SuperCollider 3.5.3

use for musical composition 'the development of higher-level musical structure arises from interactions at lower levels'[60] and as a result, self-organising systems are suitable candidates for the development of compositional structures and ideas.

Swarm Music (Blackwell and Young, 2004) is a self-organising system intended to mimic the local behaviours of insect swarms to develop an unfolding global musical structure over time, thereby creating artificial improvisations in real-time. In Swarm Music, through using real-time MIDI data input by external performers, the MIDI event data defines attractors, which draw in particles from an autonomously generated swarm, with the resulting local organisations of the particles around the attractors creating the global structures in the form of improvised melody streams. So, the values of the MIDI input influence the swarm on a local-level causing the system's global output to be relative to the swarm's particles' individual local interactions. As a result, the attractors determine the points to which the particles are more likely to travel towards with the consequent outcomes generated by their local interactions, thereby introducing a relative unpredictability within the artificial improvisation.

With regard to the compositional outputs of Swarm Music (Blackwell and Young, 2004), Young noted 'you were definitely aware of a response, and a performance loop emerging. Extremes of material seemed to work best – soft chords played slowly would soon change the kind of material coming from the swarm, after fast loud single lines for instance'[61]. Therefore, the system convincingly reacted, and subsequently improvised, relative to the real-time inputs of a human performer. Furthermore, although the system's generated swarms and consequent improvisations are autonomous, an operator also has the ability to modify the current state of the attractors through its user interface, thereby manipulating the local structures to increase the influence of the performer at the global level and the system's resulting improvisation.

Considering the structural level at which Swarm Music (Blackwell and Young, 2004) dictates, 'parameters are extracted at the mini- and meso-levels. There is a tantalizing possibility that interpretation could take place at the smallest perceivable level, the

[60] Blackwell, T and Young, M. 2004. Self-Organised Music. *Organised Sound*. 9(2): 123
[61] Ibid: 133

micro-level, and the musical structure at every level upwards could arise through self-organisation'[62]. This limitation is fundamentally due to the MIDI implementation of Swarm Music (Blackwell and Young, 2004), which does not offer parameterization of micro-level structures such as wave shape.

A granular synthesis version of Swarm Music (Blackwell and Young, 2004) was developed titled Swarm Granulator (Blackwell and Young, 2004), which enabled micro-level manipulation of the granular synthesiser's pitch, amplitude, duration and duration between events. The Swarm Granulator (Blackwell and Young, 2004), therefore enabled the artificial improvisations to produce extensive timbral modifications over time, much in the same vein as the microsound approach to composition implemented by Xenakis in *GENDYN* (1992).

Both Swarm Music (Blackwell and Young, 2004) and Swarm Granulator (Blackwell and Young, 2004) are unified in the treatment of each structural level of the musical hierarchy, resulting in changes at a constant frequency throughout an improvisation. Considering that 'since organisation at higher and higher levels would be expected to take place with diminishing frequency, it could be that a hybrid multi-level approach is preferable'[63]. Such an implementation would result in different rates of change between each structural level of the self-organising system, causing its artificial improvisations to better reflect the musical hierarchy of micro-, mini- and meso-levels.

So, despite the capability of self-organising systems to generate global musical structures based upon the lower-level interactions, the hierarchy through which these lower-level interactions form the global structure is currently inconclusive. However, this is not to denigrate the authenticity of the outputs of such systems, and perhaps such developments may indeed degrade the quality of the results through convolution of the desired creative process.

Cellular automata implement self-organisation methods to dictate their resulting global structures. As previously described in Chapter 2.5 *Further Considerations of*

[62] Blackwell, T and Young, M. 2004. Self-Organised Music. *Organised Sound*. 9(2): 136
[63] Ibid

Applying Computational Algorithms within a Compositional Process, cellular automata are 'dynamic systems in which space and time are discrete and they may have a number of dimensions, single linear arrays or two-dimensional arrays of cells being the most common forms. The cellular automata algorithm is a parallel process operating on this array of cells. Each cell can have one of a number of possible states. The simultaneous change of state of each cell is specified by a local transition rule. The local transition rule is applied to a specified neighbourhood around each cell'[64]. Their self-organization methods may then be qualitatively divided in to four different classes[65]:

1. Evolution leads to a homogeneous state
2. Evolution leads to a set of separated simple stable or periodic structures
3. Evolution leads to a chaotic pattern
4. Evolution leads to complex localized structures, sometimes long lived

These classes act to define the extent of change from one state to the next, and therefore the relative unpredictability and randomness present at local level and consequently at the global level. The amount of change between each state of cellular automata is dependent on a cell's current state, the size of the array and the algorithm that is processing the cells. The classes described above are not conclusive however, as other methods of defining the behaviours of cellular automata also attempt to explain the level of difference between states such as those proposed in the following six categories[66]:

1. Spatially homogenous fixed points
2. Spatially inhomogenous fixed points
3. Periodic behaviour
4. Locally chaotic behaviour
5. Chaotic behaviour
6. Complex behaviour

[64] Burraston, D and Edmonds, E. 2005. *Cellular Automata in Generative Electronic Music and Art: Historical and Technical Review*. University of Technology. Sydney: 5
[65] Wolfram, S. 1984. Universality and Complexity in Cellular Automata. *Physica* 10D: 1-35
[66] Li, W, Packard, N, and Langton, C. 1990. Transition Phenomena in Cellular Automata Rule Space. *Physica* 45D: 77-94

Despite these and many other differing explanations of the behaviour of cellular automata, they all represent the ability of cellular automata to offer a variety of behaviours that give a user a wide range of possible applications such as the modeling of natural phenomena and creation of artificial life.

One of the main attractions of cellular automata for generative tasks is that 'Cellular automata are sufficiently simple to allow detailed mathematical analysis, yet sufficiently complex to exhibit a wide variety of complicated phenomena'[67]. Their ability to 'exhibit a wide variety of complicated phenomena'[68] can clearly be seen in the classes that their behaviour can be defined as, as described previously. So, it is therefore possible to generate an extensive and varied set of results from a relatively efficient process, making them a strong candidate for use in real-time generative techniques.

There are many examples of compositions using cellular automata such as work of Beyls (1980, 1989, 1990), Millen (1990), Miranda (1993, 2001) and Kirk and Orton (1991) who applied real-time applications of cellular automata with MIDI using a variety of self-organising models. Furthermore, and in relation to the use of granular synthesis within *Genesis*, Miranda demonstrated extensively the use of cellular automata with granular synthesis and microsound composition methods with Chaosynth (Miranda, 1993). The self-organising model applied in Chaosynth (Miranda, 1993) is based on 'the behaviour of a type of catalytic chemical reaction know as Belousov-Zhabotinskii reactions… the cellular automaton models the way in which most natural sounds produced by an acoustic instrument evolve: they tend to converge from a wide distribution of their partials to form oscillatory patterns'[69].

So, in terms of Wolfram's (1984) classes, perhaps the outputs of Chaosynth (Miranda, 1993) could be proposed to reflect 'complex localized structures, sometimes long lived'[70] if we are to consider the complexities of sound-object formation outside of the

[67] Wolfram, S. 1983. Statistical Mechanics of Cellular Automata. *Review of Modern Physics* 55(3): 601
[68] Ibid
[69] Miranda, E. 2002. *Evolving Cellular Automata Music: From Sound Synthesis to Composition*. Sony Computer Science Laboratory, Paris: 2
[70] Wolfram, S. 1984. Universality and Complexity in Cellular Automata. *Physica* 10D: 1-35

digital domain. The results of the local interactions in Chaosynth (Miranda, 1993) are mapped to the frequency, amplitude and duration of each grain with Miranda commenting that its output 'resembles the morphological evolution of sounds produced by most acoustic instruments'[71], mirroring the supposition that the applied model forms 'complex localized structures, sometimes long lived'[72] that are present in real-world sound-objects.

Another method of applying cellular automata to real-time granular synthesis techniques is demonstrated in 'ca' (Vaidhyanathan, Minai and Helmuth, 1999). The ca system 'investigates the effects of change in the timbre of sound using a cellular automaton in real-time… the cellular automaton generated by the chosen rule controls parameters of a bank of filters. The system uses standard infinite impulse response filters and a general model of three neighbourhood cellular automata. The composer can configure the filter banks by adjusting the bandwidths and center frequencies though the graphical user interface'[73]. As a result, the ca system (Vaidhyanathan, Minai and Helmuth, 1999) allows the user to select a model which can fall into one of the four defined by Wolfram (1984). Consequently, the composer can generate a number of harmonic structures through the manipulation of the filter banks relative to the selected model and therefore transform the timbre, 'creating a new palette of sounds'[74].

Fractals are also examples of self-organising systems but demonstrate self-similarity; when considering the approaches to formalising rules for musical composition, Benoit Mandelbrot (1975), a mathematician who focused his research on fractal geometry, suggested that self-similarity, the concept that low scale and high scale structures bear similarities, could be found within music. He asserted that 'music exhibits fractal behaviour for the reason that much of it is hierarchic and even self-similar in structure. Pieces are broken down into movements, sections, phrases and notes'[75]. Therefore, in a similar principle to the explicit application of self-organising systems as seen with Swarm Music (Blackwell and Young, 2004), the correlation between

[71] Miranda, E. 2002. *Evolving Cellular Automata Music: From Sound Synthesis to Composition*. Sony Computer Science Laboratory, Paris: 3
[72] Wolfram, S. 1984. Universality and Complexity in Cellular Automata. *Physica* 10D: 1-35
[73] Vaidhyanathan, S, Minai, A, and Helmuth, M. 1999. ca: A System for Granular Processing of sound using Cellular Automata. *proceedings of the 2nd COST g-6 Workshop on Digital Audio,* Trondheim, *1999*: 1
[74] Ibid
[75] Järveläinen, H. 2000. *Algorithmic Musical Composition*. University of Technology, Helsinki: 8

interactions at the local level are accountable for those on higher levels, demonstrating a commonality between the use of fractals and other forms of self-organising systems.

The correlation between local and global interactions in self-organising systems such as fractals, and their proposed role in music, has been demonstrated in the work of Voss and Clark (1975). Their research showed 'statistically that most widely acclaimed music has a very similar distribution to fractals that have what is called a 1/*f* or *inverse frequency* distribution'[76]. Their work also applied the 1/*f* principle to music composition: 'they used a 1/*f* noise generator as a pitch selection unit. The random numbers from the noise process were rounded and scaled to produce pitch values in a range of two octaves'[77]. The use of noise processes such as Brownian, White and Pink motions serve to affect the type of movement between increments, and their correlation, giving a definitive characteristic to each method used.

Fractals may also be applied to rhythmic functions as recent research analysing 558 compositions of Western classical music showed that 'the ubiquity of 1/*f* rhythm spectra in compositions spanning nearly four centuries demonstrates that, as with musical pitch, musical rhythms also exhibit a balance of predictability and surprise'[78]. It must be noted however, that the function of fractals does not have be limited to pitch and rhythm; their outputs may be attributed to any mapping a composer desires, making them an exciting tool for the algorithmic control of many discrete parameters.

There have been many implementations of fractals through the use of MIDI that are founded on mapping between the results of the fractal processes to pitch and duration (Diaz-Jerez, 2000. Dunn, 2003. McDowell, 1994. Greenhouse, 1995), thereby each offering similar musical outputs relative to the applied fractal. In addition, attempts have been made to introduce fractals for the manipulation of a granular synthesiser's parameter settings such as the self-similar grain distribution granular synthesiser proposed by Chapman et al (1996). The system attempted to map the values of local interactions of fractals to individual grains, applying the 'audification' methods

[76] Leach, J and Fitch, J. 1995. Nature, Music and Algorithmic Composition. *Computer Music Journal* 19(2): 24
[77] Järveläinen, H. 2000. *Algorithmic Musical Composition*. University of Technology, Helsinki: 8
[78] Levitin, D. 2012. Musical rhythm spectra from Bach to Joplin obey a 1/f power law. *proceedings of the National Academy of Sciences* 109(10): 3716

suggested by Kramer (1996), which advise on audio parameter mappings relative to musical perception. Despite this, Chapman et al's (1996) system 'is restricted in terms of the amount of data described by a single point, making mapping to complex grains overly trivial'[79]. As a result, it would appear that fractal processes applied to extensive microsound control are limited by the level of data in the output, whereas in contrast, for higher-level macro structures such as pitch and duration which require considerably less data over time, their suitability is established.

Considering the role of self-organising systems in the compositional process and the resulting compositions they generate, Supper (2001) remarks that 'simulating natural phenomena raises the question whether composers secretly see algorithmic composition as a way of generating natural forms naturally – forms which are taken to justify themselves by their naturalness alone'[80]. However, such a notion directly implies that composers applying self-organising systems are intentionally exploring the use of natural phenomena within their creative process. In certain implementations, this is certainly true, such as in the applied chemical reaction model for acoustic modelling found within Chaosynth (Miranda, 1993) and the insect swarm mimicry of Swarm Music (Blackwell and Young, 2004).

In contrast, in a system such as ca (Vaidhyanathan, Minai and Helmuth, 1999), the user is given the option to apply a model of their choosing, which therefore suggests that an understanding of the selected model is required. However, what if the composer were to choose a model that they felt sounded 'good', thereby achieving their desired outcome without specific knowledge of the process behind the result? Should this render such a product of a self-organising system inferior?

Burraston (2005) states 'the sonic artist and musician must be prepared to investigate the theoretical background in order to successfully employ this vast behaviour space within their compositional strategy'[81]. Such a proposition would certainly conclude that any implementation of self-organising systems in which the composer does not

[79] Chapman et al. 1996. Self-Similar Grain Distribution: A Fractal Approach to Granular Synthesis. *Proceedings of the ICMC'96*: 213
[80] Supper, M. 2001. A Few Remarks on Algorithmic Composition. *Computer Music Journal* 25(1): 53
[81] Burraston, D and Edmonds, E. 2005. *Cellular Automata in Generative Electronic Music and Art: Historical and Technical Review*. University of Technology. Sydney: 24

understand the underlying local interactions and their consequences to the global output is not valid, thus judging the product of a self-organising system by the composer's awareness of the process that generated its outcome.

So, perhaps it is therefore reasonable to apply both Supper's (2001) and Burraston's (2005) proposals in circumstances where the *explicit* execution of models are used. However, in circumstances where the composer has chosen to implement a model without expressed knowledge of the applied model whilst still achieving the required compositional structure or idea, then surely this must justify such an application. The above discussion highlights a considerable issue in the evaluation of algorithmic music that uses complex processes and procedures; a complex *process* is often used to validate a *product*, but to what extent should this validation be recognised? This is detailed further in chapter 6 *Evaluation of the Genesis System* in which the role of a *process* in the evaluation of a *product* is deliberated over in the context of real-time systems such as *Genesis*.

As well as processes demonstrating self-organising properties, *Genesis* uses search-based algorithmic procedures to generate results relative to the outputs of its implemented machine listening processes and to preset variables. When considering the application of musical formalisms in algorithmic processes (either conventional or novel), it is possible to find and search for solutions from chosen variables using conditional statements made up of rules and structures defined by a composer.

The use of conditional statements for search functions requires a significantly high amount of symbolic data; rules must be defined by symbolic values, and therefore, so must the input. Therefore, low-level structures such as timbre and gesture are not as suited for use in such a symbolic representation required for conditional statements. These factors should be considered by a composer in relation to their effect on a required task; ideally, conditional statements should be used when definitive and concise rules can be formulated.

Expert systems employ heuristics that allow the algorithm to make approximations and use those estimates as its output. The use of heuristics creates an optimal system, which, although may not necessarily find the best result, is notably more efficient than

a complex series of conditional statements that may potentially never find a solution. Expert systems are 'most suitable for generating music whose style can be codified by faces, rules and heuristics, such as the musical style of Bach's keyboard pieces'[82]. Therefore, again, it is important that a composer considers the implicit limitation of using highly structured rules and their required symbolic representations as it could be argued the 'biggest constraint of expert systems is that new musical styles are not that well-defined or have not developed enough to be codified extensively'[83].

In musical practice, search-based systems have demonstrated considerable strength in musical applications for music formed of well-established rules and structures (Ebcioglu, 1998. Tsang and Aitken, 1991, Pachet, 1992). Indeed, Hiller's *Illiac Suite* (1957), the very first piece of music written exclusively using computational algorithms used search-based methods to identify desirable outputs. However, if a search-space is too large or a rule set too extensive, the intricacies of organising a rule set hierarchy or obtaining a manageable and applicable output can seriously affect the relative cost of the outcome; the implementation required to reflect the complexity of the rule set or dataspace may not equate to the perceived value of the result.

Methods such as genetic algorithms attempt to obtain desired outcomes through searching a data-space comparative to a fitness function, resulting in potentially efficient results in terms of the prospective complexity of a fitness function and/or a search space. Fitness functions can be applied within methods such as the previously discussed cellular automata, but they are not compulsory, as they only serve to refine a processes' outputs towards a set goal; cellular automata can still create complex structures without the use of fitness functions. Genetic algorithms however, model natural selection and therefore they must intrinsically apply fitness functions. Therefore, they implement the principle of search-based methods to identify a result based on specified criteria, instead described within a fitness function which is a definition of a preferred result that can have a threshold dictating the relatedness of data found within a search-space to an ideal outcome.

There are two main methods for measuring fitness in a genetic algorithm; a human

[82] Järveläinen, H. 2000. *Algorithmic Musical Composition*. University of Technology, Helsinki: 7
[83] Ibid

critic or an automatic fitness assessment[84]. The use of a human critic requires an interactive system in which the user assesses each generation's population, with the user then measuring each member's fitness. In contrast, an automatic fitness assessment uses prior knowledge defined by the user for the genetic algorithm to assess a candidate's fitness. Both approaches have advantages and disadvantages but the key issue dividing the methods is the relationship between the efficiency and the quality of the results. To demonstrate, a human critic can lead to higher quality results in relation to the fitness they have determined, but it requires time to measure a candidate's fitness, making the process inefficient. On the other hand, automatic fitness assessment can be executed almost instantaneously, but the quality of the results is limited to the criteria applied within the fitness function. With these factors in mind, a composer must consider carefully the context and the requirements of the process they wish the genetic algorithm to complete.

In relation to the creative process, introduced in Chapter 2 *An Introduction to Algorithmic Composition*, there are four stages; stage one - preparation, stage two - incubation, stage three - illumination and stage four – verification. It is possible to use a series of genetic algorithms to model different parts of the creative process. For example, David E. Goldberg, a leading researcher in the application of genetic algorithms has stated that the use of such algorithms could be used to model 'different *facets of human innovation*'[85]. He proposed that the ability of genetic algorithms to use selection, crossover and mutation processes allow for the possibility of '*improvement* and *crossfertilizing* types of innovation'[86], defined more precisely as:

'Selection + Mutation = Continual Improvement

Selection + Recombination (crossover) = Innovation'[87]

The concept of 'continual improvement' can be represented as a '*hillclimbing* mechanism, where mutation creates variants in the neighbourhood or the current

[84] Gartland-Jones, A and Copley P. 2003. The Suitability of Genetic Algorithms for Musical Composition. *Contemporary Music Review* 22(3): 43-56
[85] Goldberg, D E. 1998. *The Design of Innovation: Lessons from Genetic Algorithms, Lessons for the Real World*. University of Illinois: 3
[86] Ibid: 4
[87] Ibid: 3

solution and selection accepts those changes with high probability, thus climbing toward better and better solutions'[88]. The idea of 'innovation' in the context of using selection and crossovers can be explained as 'grasping at a notion of a set of good solution features in one context, and notion in another context and juxtaposing them, thereby speculating that the combination will be better than either notion taken individually'[89]. It is certainly clear that genetic algorithms offer the possibility of creating better solutions to an initial problem; however, this is highly reliant on the fitness function and its method of implementation be it a human critic or an automated fitness assessment. So, despite their apparent capability of modelling creative processes, it must not be assumed that genetic algorithms can as a matter of course completely replace the need for human input.

There are many examples of computational algorithmic methods that make extensive use of genetic algorithms for real-time interaction between users and an ongoing search process. *GenJam* (Biles, 1994) applies a genetic algorithm to model a novice jazz musician learning to improvise. As the system 'plays its solos over the accompaniment of a standard rhythm section, a human mentor gives real-time feedback, which is used to derive fitness values for the individual measures and phrases. *GenJam* (Biles, 1994) then applies various genetic operators to the populations to breed improved generations of ideas'[90]. *GenJam* (Biles, 1994) therefore uses a human critic to evaluate the fitness of its outputs and bases its future generations on those selected to satisfy the composer's requirements.

The *GenJam* system was then adapted further and was developed into *AutoGenJam* (Biles, 2001) removing the necessity for a human critic, being replaced by a predefined population of 'licks', in addition to applying an intelligent crossover operator and mutating repeated events. Furthermore, the system has the option to 'trade fours' with a live instrumentalist via MIDI, allowing for the program to provide responses based upon a performer's most recent phrases.

---

[88] Goldberg, D E. 1998. *The Design of Innovation: Lessons from Genetic Algorithms, Lessons for the Real World*. University of Illinois: 4
[89] Ibid: 3
[90] Biles, J. (1994). *GenJam - Original Paper*. [online] Igm.rit.edu. Available at: http://igm.rit.edu/~jabics/GenJam94/Paper.html [Accessed Mar. 2013]

The intelligent crossover produces crossover points that will produce children most like their parents, thereby increasing the likelihood of desirable results by retaining similarity between evolution points. So, if a parent, founded on the predefined 'licks' database is deemed fit, then so will its children, allowing the removal of a standard fitness function, to be replaced by the intelligent crossover.

The outputs of the two systems *GenJam* (Biles, 1994) and *AutoGenJam* (Biles, 2001) have been considered by listeners to be superior in the case of *AutoGenJam* (Biles, 2001). Despite this, the limited database used for *AutoGenJam* (Biles, 2001) restricts the creative space in comparison to the random populations created at the start an improvisation by *GenJam* (Biles, 1994). Therefore, if *AutoGenJam* (Biles, 2001) is repeatedly used in performance, repetition and familiarity will become apparent in its responses to the user. Indeed, this may be highly desirable, as it will demonstrate the characteristics of its predefined 'licks' and thus the resulting characteristics of the system when used with such data. Nevertheless, if an extensive search space is required to explore many different possibilities, then this will be a significant limitation.

Another aspect to consider is the assessment of the human performer by *AutoGenJam* (Biles, 2001). As the system currently stands, the system always responds to the phrases provided by the performer, regardless of the quality of the input. In contrast, with *GenJam* (Biles, 1994), due to the use of a human critic, a 'bad' performance can be erased from the search space, essentially forcing the system to respond only to desirable inputs.

Biles (2001) concluded when discussing *AutoGenJam* (Biles, 2001) that a fitness function may be required 'to determine if a human's four in a solo is worth keeping and breeding. This is not an issue when trading fours because the occasional bad four is gone as soon as it is played, but a bad lick breeding in a fitness-free environment could ruin a soloist'[91]. Such a proposition highlights the possible consequences in an automated fitness environment, yet *AutoGenJam* (Biles, 2001) was considered a 'better' system in terms of its product.

[91] Biles, J. 2001. *Autonomous GenJam: Eliminating the Fitness Bottleneck by Eliminating Fitness.* Rochester Institute of Technology. New York: 6

When considering the real-time functionality of the systems, and the ability to provide *AutoGenJam* (Biles, 2001) with real-time inputs via MIDI, the use of an automated fitness function is perhaps the most efficient and effective method of breeding the input data; with a human critic fitness function, a musician is required to break away from performance to assess possible future outputs, the ongoing dialog between the performer and the system may be interrupted, disrupting the performance. Therefore, perhaps a combination of both an automated fitness function and a human critic may resolve such an issue; the use of a 'remove' button, which allows previously input data to be removed by the user while continued use of the current automated fitness function could offer a suitable solution.

The *AudioServe* (Yee-King, 2003) system is an 'implementation of a collaborative, interactive genetic algorithm that allows multiple users to evolve and share audio synthesis circuits using a web-based java interface'[92]. The program allows local exploration of synthesis parameters that can be sent via network to a central system that holds a global population. As a result, the users on local machines may evolve their populations, with data sent from the central population consequently bred into the local population. Therefore, the preferences in synthesis parameter settings of those around the user directly influence their outputs forming the proposed collaborative functionality.

Approaches to applying genetic algorithms are not in anyway restricted to 'conventional' methods of interfacing such as user interfaces displaying numerical parameters and wave shapes; *Feeping Creatures* (Berry, 1999) and *Gakki-mon Planet* (Berry et al, 2003) attempt to free a composer from the constraints of search spaces bound to numerical display by creating a 'worlds' in which the user is able to roam freely creating music from the environment that surrounds them. For example, 'the world of *Feeping Creatures* is a flat green grid across which the inhabitants – "feeps" – and the observer move… Each feep has a sequence of musical pitches that form its chromosome. When two feeps mate, portions of each parent's note list are passed on to their offspring to form a new chromosome or pitch series. Some will seek out

[92] Woolf, S and Yee-King, M. 2003. Virtual and Physical Interfaces for Collaborative Evolution of Sound. *Contemporary Music Review*. 22(3): 37

partners that are, on average, musically constant to themselves, while others prefer dissonance'[93]. The aim of such programs is to not restrict the search space whatsoever, potentially allowing complete freedom within a hypothetically infinite exploratory search space.

Other examples of systems using genetic algorithms for musical composition include *MutaSynth* (Dahlstedt, 2000) and *IndagoSonus* (Gartland-Jones, 2003) each with their own adaption of the process of natural selection and mappings to musical values. However, from a preliminary investigation of their characteristics using examples such as those described above, the effect a fitness function can have on the process and the consequent success of the results of a genetic algorithm can be readily confirmed. It is also soon becomes clear from more detailed investigations the extent researchers have gone to in order to improve a fitness function or remove the requirement for a fitness function altogether. The capability of genetic algorithms to perform different tasks at different structural levels in real-time is evident as well as their application to innovative forms of user interfacing. Their proficiency to explore predefined musical phrases, for example in *AutoGenJam* (Biles, 2001), and with collaborative functionality as seen in *AudioServe* (Yee-King, 2003), is certainly impressive and reflects well on their suitability for real-time musical composition.

On the basis of the evidence presented above it is reasonable to assert that genetic algorithms are an exceedingly powerful tool for algorithmic composition. As demonstrated, they can be used for a variety of problem-solving applications as well as for exploratory tasks. Also, the possibility that they can model creative processes rather than simply be used to execute specified tasks make them a highly attractive algorithmic method for composers. The capability of the evolutionary process itself to form a musical structure, from one stage of evolution to the next, and possibly a composition as a result is an exciting prospect. In contrast, a genetic algorithm's ability to instantaneously 'evolve' a specified starting point containing a musical phrase or structure many times over, offers a composer a unique method of generating material by an algorithmically-based means.

[93] Berry, R and Dahlstedt, P. 2003 Artificial Life: Why should Musicians Bother?. *Contemporary Music Review* 22(3): 59

The reliance of genetic algorithms on concise fitness functions and the effect this can have on the quality and efficiency of results has implications that must be considered further; the choice between a human critic or an automated fitness assessment and the consequent type of methods that may be used are difficult decisions, with major implications in creative terms, and without full knowledge of the context it is impossible to provide clear guidance. In addition, the effect that the initial population's representation and values can have on the result must be considered very carefully by the composer, as it is the starting point from which an evolution begins; poor quality starting data is more than likely to lead to a poor quality result. Despite these issues, genetic algorithms are a highly useful algorithmic method for composition if applied and constructed in a suitable manner.

As demonstrated, the generative methodologies used in *Genesis* are founded upon a great deal of research into the application of their respective approaches. In addition to the generative methods applied in *Genesis*, implementations of other generative techniques have been applied to musical composition such as artificial neural networks, through which a learning agent adapts to its environment with little or no prior knowledge (Correra et al, 2007. Mozer, 1994. Fiebrink, 2009. Le Groux, 2002. Lee and Wessel, 1991. Todd, 1989), and grammars, which create musical structures that fit within a set of imposed rules (Roads, 1979. Ruwet, 1972. Nattiez, 1975. Winograd, 1968. Lerdahl and Jackendoff (1977). Cope, 1992. Rohrmeier, 2007. Johnson-Laird, 2002).

Considering a fundamental purpose of *Genesis* was to offer real-time interaction between the user and the system consequently generating compositions in real-time, offering instantaneous results to the ongoing dialog between user and computer, the generative processes selected form the desired result. In chapter 6 *Evaluation of the Genesis System*, possible modifications of the current generative methodologies are discussed, alongside the prospect of introducing other methodologies such as artificial neural networks for the construction of reasoned responses to the user's inputs.

3.2 A Brief Summary of Machine Listening

Machine listening, the process of using computers to identify and analyse sonic features from audio sources, is used in *Genesis* to obtain data from which to modify and adapt the outputs of its selected generative processes; data extracted from audio signal inputs provided by the user is mapped to fixed and relatable parameters of the generative algorithms such as pitch, onset and timbre. Therefore, machine listening must apply a variety of disciplines such as digital signal processing, psychoacoustics and musical analysis to extract, identify and represent a desired sonic feature.

Psychoacoustics is the study of *sound* perception, often including psychological and physiological responses to sound events, both musical (such as the identification of a musical instrument or a sound's pitch) and non-musical (such as the awareness of loud 'bang' being a potential danger or classification of a sound's source). However, musical analysis is founded on discussion into the identification of exclusively musical features such as key, genre, tonality, rhythm, mood and metre.

The human ear performs a physiological process that can be quantified, and to a large extent, qualified. However, an individual's perception and arrangement of a sound-object's components is believed to include psychological processes and, as a result, can neither be conclusively quantified or qualified, leading to extensive discussion over the methods with which humans perceive sound and consequently reason it as music (or not), which in itself drives fierce aesthetic and philosophical debate. It is not within the remit of this thesis to definitively state what *is* music, however, it is responsible for highlighting the methods of *how* the perception of music and its qualities *may* be defined through proposed models of music perception, and consequently placed in computational analytical algorithms; as stated, models of sound perception can not be decisively quantified or qualified, and therefore must *not* be accepted as definitive explanations of musical perception, but rather as suggestions of how to form a musical analysis from the information provided, be it as symbolic or subsymbolic data.

The theoretical nature of sound perception models is reflected in the difficulties in choosing suitable perception models and applying them to analytical processes; there are many methods to approach the perception of each musical quality, with the consequent hierarchical arrangement of these perceptions for the purpose of performing a chosen analytical task complicating the selection process further. One major consideration in selecting a model's fitness and suitability for a task is the method with which a sound perception model applies the information it is provided with. This can be divided into two distinct categories: predictive and reactive. Predictive models use a variety of different techniques such as neural networks and search to forecast possible events and their components thereby aiming to achieve an increase in analytical quality and, in some cases, efficiency. In contrast, reactive models respond to data at the relative time, without using external data from that time with which to assess events and as a result provide analysis of the 'here and now'.

To improve further the accuracy of waveform data and its components, auditory scene analysis (Bregman, 1990) is proposed to take place, which is 'the perceptual organization of sounds according to the sound sources that are producing them'[94]. So, the use of auditory scene analysis allows for the deconstruction of waveforms into their individual sources, which potentially offers the opportunity for increased accuracy in the analysis of each source.

Through the principles of the organization of sensory stimuli proposed by Koffka (1935), it is possible to offer a guideline of the minimum from which users can identify the individual sources of an auditory scene, and thereby form a basis to create an optimal method of obtaining data. The principles of the organization for sound are as follows[95]:

> *Similarity* – Sound components that come from the same source are likely to be similar

[94] Plack, C. 2005. *The Sense of Hearing*. New Jersey: LEA: 193
[95] Ibid: 195-196

Good Continuation – Sound components that come from the same source are likely to flow naturally over time from one to the other (without abrupt discontinuities)

Common Fate – Sound components that come from the same source are likely to vary together (for example, will be turned on and off at the same times)

Belongingness – A single sound component is usually associated with a single source: It is unlikely that a single sound component originates from two (or more) different sources simultaneously

Closure – A continuous sound obscured briefly by a second sound (e.g., speech interrupted by a door slam) is likely to be continuous during the interruption unless these is evidence to the contrary

From the above example based on Koffka (1935), it is clear that if we are to obtain accurate data from a source, its features must be clearly distinguishable from other sources in an auditory scene. That is to say, an individual sound source must feature *similarity*, *good continuation*, *common fate*, *belongingness* and *closure* in order to be identified as such. With these features in mind, the deconstruction of the sound-objects within a scene into their individual sources by analytical algorithms is possible, and consequently, the resulting analysis of each source significantly increases its chance of a successful outcome. However, if the above list of features is not available from a scenario, the resulting analysis will be limited from the outset in its capability to isolate individual sources, regardless of the complexity of the analytical process itself; a substantial amount of data will contain information from extraneous sources relative to the analytical process, negatively influencing the result.

It is certainly not possible to definitively divide complex waveforms in to their individual sources; considering the mathematical complexities of such waveforms, even with high performance computers, it is still not achievable to deconstruct waveforms into their individual sources. However, it is possible, through pragmatic approaches of obtaining data, to limit the complexity of a waveform to the sources the

user requires. For example, a contact microphone can be placed on a source, which will minimize any masking by other sources.

In terms of *Genesis*, the isolation of sonic events can greatly impact on the predictability of the response provided by the system; the more the system is able to deconstruct the auditory scene, the more predictable results. The ability to identify effectively the sonic features of the sound source increases the congruency between the nature of the sound source and the consequent mapping of the relatable parameters heard in the resulting product, and therefore also increases the expectedness of the outcome. As a result, the user must ensure that the selected sound sources that control and modify the outputs of the generative algorithms in *Genesis* are suitably obtained causing the feature extraction process to represent better the auditory scene, thereby improving its consequent analysis and resultant outcome relative to the ongoing dialog of the composition.

In relation to the consequent definition of sound-objects and their musical qualities once its source has been successfully identified, no definitive description of their properties can be given. For example, consider a performer in a concert hall with a woodblock. The woodblock is hit once with a beater; how do we describe this event in terms of pitch, loudness, spatialisation and timbre? This raises many subsequent questions such as: What is the primary feature of the sound-object? Do we define it primarily by its pitch? Its timbre? Its loudness? If we are to define it by its timbre, how do we specify the timbre? Do we state its timbre by the material of the source i.e., wooden? Or by the envelope of the sound e.g., fast? How important is this sound-object in relation to other sound-objects generated by this source or other sources? Essentially, how can we conclusively define a sound-object in relation to its musical qualities and value?

The phenomenology of sound-objects (the study of the subjective features of a sound-object) has been researched extensively by Pierre Schaeffer who proposed a listening experience referred to as *acousmatic* or *reduced listening* which can be defined as ‘a situation of pure listening, without attention being distracted or supported by visible

or foreseeable instrumental causes'[96]; *acousmatic* listening results in a requirement of the listener to remove extra-musical and historical contexts from sound-objects, thereby reducing the description of a sound-object to its sonic features only. In Schaeffer's text *Traité des objets musicaux* (1966), a topology of sound-objects was proposed, in which sound-objects could be categorised by their sonic features. *Figure 3* illustrates these groupings, and the relationships between the sonic features that influence the categorization process[97]:

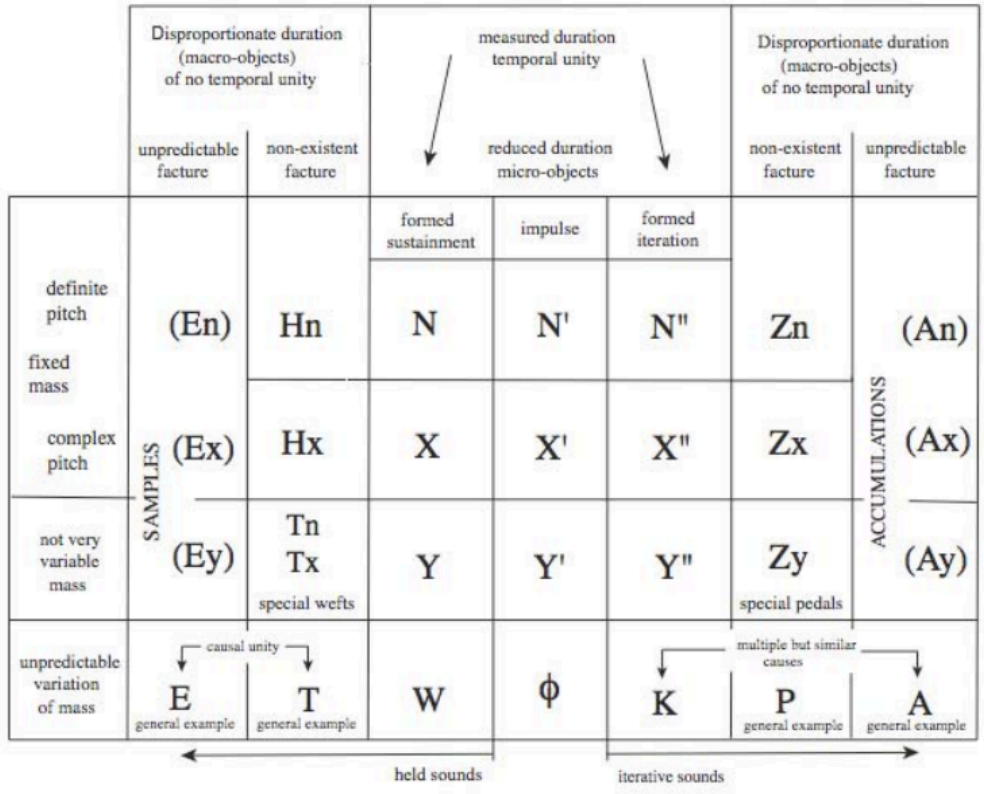


Figure 3. Categorization of sound-objects

From the an electroacoustic composer's viewpoint, Trevor Wishart (1996) states 'the idea of acousmatic listening is easily appreciated by anyone who has worked with sound-materials in the electro-acoustic studio. When working with large numbers of sounds from different sources and particularly when this material is transformed, if only slightly, it becomes difficult to remember from where the various sounds originated and from a compositional point of view such origins need have no special

[96] Bayle, F. 1993. *Musique acousmatique: propositions – positions.* Paris: INA: 179
[97] Chion, M. 1983. *Guide Des Objets Sonores.* Paris. Translation by John Dack and Christine North, 2000

significance'[98]. Therefore, representations of sound-objects through acousmatic descriptions have a strong relationship with techniques used in electroacoustic composition and subsymbolic representations.

In contrast to the view demonstrated by Wishart (1996), the categorization of sound-objects with an *acousmatic* method 'perpetuates an ahistorical view about the nature of musical material. Theodor Adorno argued in the late 1920s, 'the cognitive character of art is defined through its historical actuality' (Adorno and Krenek, 1974; quoted in Paddison, 1993). In other words, it cannot be defined outside of the context of its own historical becoming; rather, the compositional act is engaged, from the very beginning, in a dialectic with history, *in the form of* sonic material'[99]. Therefore, it is argued that the description a sound-object cannot be removed from its historical context, and that it is the attribution of a sound-object to an existing source that defines its properties as a musical device.

Considering the two opposing opinions presented regarding *reduced listening*, they epitomize the difficulties that musical analysis models face in terms of adequate representation of sound-objects; once a source has been identified, there is no unequivocal method for representing its sound-objects. This has implications not only for analytical processes but also for the transcription of sound-objects for musical performance; a performer requires an accurate description of a sound-object in order to be able to adequately represent and perform its prescribed sonic features. If we reflect on the conventional musical score and its transcription technique, the highly symbolic representation of pre-defined, existing sound sources and their musical qualities allows for analysis methods such as Schenkerian analysis, which in itself is not free from criticism of its musical analysis technique (Rosen, 1971; Meyer 1956; Narmour 1977; Kerman 1980).

The ultimate purpose of Schenkerian analysis is to reduce musical works into *Ursatz*, an archetypal progression of a proposed elaboration of a triad. Therefore, implicitly, Schenkerian analysis is only applicable to the Western Tradition, and further still, is

[98] Wishart, T. 1996. *On Sonic Art.* London: Routledge: 67
[99] Kane, B, 2007. L'Objet Sonore Maintenant: Pierre Schaeffer, sound-objects and the phenomenological reduction. *Organised* Sound 12(1): 22

exclusive of works that do not use harmonic rules thus imposed, ruling out an extensive repertoire of work for both the orchestra and/or electroacoustics. As a result of such limitations, even if it were possible to describe a sound-object definitively, the consequent analysis of its relevance and importance in a musical work is still contentious, which strengthens the resolution that analytical models must only be used as *suggestions* of musical perception and not as *absolute representations*.

So, in relation to the compositional process and the role of computational analytical algorithms, the application of sound perception models must be carefully considered; as a primary concern, the sound perception model must be selected in relation to the chosen problem. That is to say, the sound perception model should offer representations of sound-objects that correlate to the generative and analytical techniques required by the composer. For example, juxtaposed to the limitation of Schenkerian analysis, which requires highly symbolic methods of representation and tonal Western Music to successfully complete assessments, Schaeffer's topology of sound-objects into their sonic features, allows for subsymbolic methods of representation and composition which do not exclusively feature Western harmony or orchestration. Therefore, the compositional techniques applied by a composer not only dictate the success of the analytical technique used, but also the models that propose the perception of the sound-objects themselves.

3.2.1 Pitch Perception

Within *Genesis*, extensive electroacoustic compositional techniques are implemented, alongside a variety of tonal precepts, necessitating that the machine listening algorithms reflect these compositional methodologies in their approaches to identifying their respective features and consequently representing them in a relatable format for the desired parameter mapping. Thus, sonic features such as loudness, pitch and timbre are identified through psychoacoustic models with representation methods that provide musical values, thereby assimilating psychoacoustic modelling into both conventional musical features and electroacoustic principles.

Pitch can be considered to be 'that attribute of auditory sensation in terms of which sounds may be ordered on a scale extending from low to high. Pitch depends mainly on the frequency content of the sound stimulus, but it also depends on the sound pressure and the waveform of the stimulus'[100]. Plack (2005) narrows this description of the pitch of stimuli to those 'whose variation is associated with *musical melodies*… this definition is consistent with what some researchers regard as an empirical test of the presence of pitch: If you can show that a sound can produce melodies, then you can be sure it has a pitch (e.g., Burns and Viemeister, 1976) '[101].

So, the presence of pitch allows the consequent creation of melodies, however, the term *melody* implies a tonal structure of music; melodies are not always present in atonal and electroacoustic composition, yet pitch is still perceivable by the listener. Therefore, for the purposes of this thesis, which addresses compositional techniques outside of the Western Tradition such as microsound and granulation, the definition provided by the ANSI is more suitable.

Pitch can be identified through the use of two distinct types of psychoacoustic model: place coding and temporal coding. Through place coding, pitch is defined 'in terms of the place that it is active (for example, on the basilar membrane, or in a neural array'[102] and with temporal coding, pitch is identified 'in terms of the pattern of activity over time, in particular, the phase-locked responses of auditory neurons'[103]. Therefore, both pitch models attempt to emulate the physical response of the human ear to sound-objects through their relative representations of the component frequencies of waveforms.

Both place coding (Licklider, 1958; Moore, Glasberg and Peters, 1985; Dai, 2000; Goldstein, 1973; Terhardt, 1974) and temporal coding (Schouten, 1940, 1970) have demonstrated the applicability of both methods to identify the pitch of sound-objects, with many computational models using the two approaches to successfully detect

[100] American National Standards Institute (ANSI). 1994. *American national standard acoustical terminology.* New York: Acoustical Society of America: 34
[101] Plack, C. 2005. *The Sense of Hearing*. New Jersey: LEA: 133
[102] Ibid: 247
[103] Ibid: 248

pitch (Goldstein, 1973; Therrien, 1989; Wightman, 1973; Terhardt, 1974; Hermes, 1988).

Perhaps the most widely applied pitch perception model is through the use autocorrelation, which is founded in the temporal coding process to pitch classification; an autocorrelation function is a method of describing periodicity, 'computed by correlating a signal with a delayed representation of itself. At times equal to integer multiples of the repetition rate of a waveform, the correlation will be strong. Similarly, if there are common time intervals between waveform features, then this delay will show up strongly in the autocorrelation function'[104].

Autocorrelation has proven to be an effective and efficient method of pitch classification, reflected in its widespread use as a pitch classification tool. However, 'first, autocorrelation models do not provide a satisfactory explanation of why we are so much better at fundamental frequency discrimination, and why pitch is so much stronger for resolved harmonic than for unresolved harmonics... Second, recent experiments with groups of unresolved harmonics suggest that regularity of temporal information may be less important for these stimuli than the gross rate of temporal fluctuations'[105]. As a result, although a computational model used for pitch perception cannot be conclusively defined, autocorrelation methods are suitable for many applications within music.

Considering the perception of the pitch of acoustic sources by computational models and its application within computational algorithmic compositional processes, it is certainly possible to obtain the pitch of sound-objects in many circumstances (assuming a sound-object has been identified clearly in the auditory scene), and therefore describe it in both tonal and microtonal terms, and apply this data for analytical and generative processes. However, the accuracy of the value of a perceived pitch by computational models in relation to our own pitch classification method is distinctly limited, which as a result, will influence a consequent compositional process involving computational algorithms; pitch is defined in terms of our capability to perceive it. Therefore, the results of any analytical or generative

[104] Plack, C. 2005. *The Sense of Hearing*. New Jersey: LEA: 146-147
[105] Ibid: 245

processes are implicitly reliant on the perceived pitch, and thus require an accurate representation of pitch for predictable outcomes.

So, the use of computational models of pitch perception can inadvertently affect the compositional process. For example, if a generative process requires a specific series of pitches in order to trigger and these pitches are not accurately identified the subsequent process will fail. Therefore, the limitations of computational pitch perception models, and the affect this may have on the compositional process, must be carefully considered in relation to the desired task. If this is overlooked, the resulting output of the compositional process may be significantly unexpected and perhaps undesirable (when considering a composer's intentions).

3.2.2 Loudness Perception

Loudness is the 'subjective magnitude of a sound; the perceptual correlate of intensity'[106]. Through experiments involving loudness matching, which require listeners to state their perceived intensity of a sound, it appears that frequency, bandwidth and duration are all factors that influence loudness. In order to obtain a relative value for the loudness of a sound, 'the *loudness level* (in units called *phons*) of a tone at any frequency is taken as the level (in dB SPL) of the 1000-Hz tone to which it is equal in loudness'[107]. Therefore with this method of loudness evaluation, it has been demonstrated that: with frequency 'the growth of loudness with level is greater at low frequencies than at high frequencies'[108], with bandwidth 'if the power of a sound is distributed over a wider region of the cochlea, then the loudness may increase'[109] and with durations 'up to a few hundred milliseconds, the longer the sound, the louder it appears'[110].

It is important to note that despite it being possible to describe how frequency, bandwidth and duration *influence* loudness perception, 'it cannot tell us directly how loudness changes with sound *level*'[111]. So, it is not possible to quantify loudness in

[106] Plack, C. 2005. *The Sense of Hearing*. New Jersey: LEA: 116
[107] Ibid
[108] Ibid
[109] Ibid
[110] Ibid: 145
[111] Ibid: 117

terms of physical values such as dB SPL, which defines the air pressure caused by a waveform. Researchers such as Stevens (1957, 1972) and Schlauch, DiGiovanni and Reis (1998) have attempted to quantify a loudness value by applying loudness scales, which explain 'subjective magnitude with physical magnitude'[112].

Loudness scales are limited in their application due to the very nature of loudness itself; a subjective process cannot be definitively quantified, and as a result, such a task which attempts to do so cannot be regarded as accurate, thereby negating its very purpose. However, the results of using such scales have produced results that appear to fit a logarithmic scale, which may reflect our subjective loudness perception i.e., a sound's intensity climbs more sharply with increases at lower magnitudes, than at higher magnitudes, rendering them applicable to compositional processes.

Considering our listening experience, we are able to describe the loudness of individual sources within an auditory scene, as well as the overall intensity of the sources combined. Our capability to detect the loudness of an individual source is referred to as *intensity discrimination*[113]. It would appear, through experimentation, that our ability to discriminate between the intensity of sources is exceptional at a considerable amount of our dynamic range. It is therefore proposed that, much like pitch perception, place coding and temporal coding are required.

Spread excitation uses place coding to explain the perception of loudness; 'at low levels, only a small region of the basilar membrane is stimulated (the region surrounding the place tuned to the pure tone's frequency), but as the level is increased, a wider area is stimulated'[114]. In addition, it is possible that information is combined from across the excitation pattern to improve performance (Florentine and Buus, 1981). Therefore, the pattern of the firing rates and their respective frequencies strongly present a sound's intensity.

In terms of temporal coding, phase locking may also be necessary for intensity discrimination (phase locking is the 'tendency of an auditory neuron to fire at a

[112] Plack, C. 2005. *The Sense of Hearing*. New Jersey: LEA: 120
[113] Ibid
[114] Ibid

particular time (or phase) during each cycle of vibration on the basilar membrane'[115]). In the case of masking, and in particular masking by noise, phase locking becomes limited, as it is not possible to phase lock to sounds of no fixed periodicity. So, waves within a signal that do contain periodicity can be identified through phase locking, and therefore their intensity can be defined (Sachs and Young, 1980).

Further to the subject of our listening experience, we our able to define the *relative intensities*[116] of frequency components of sound-objects. These are contained in the relative spectrum of a sound-object and enable us to identify a sound's timbre. So, we are able to obtain a spectral envelope, from which the *relative intensities* can be described. Experiments conducted by Green (1988) concluded that changes in the *relative intensities* in the spectrum of only a few dB were noticeable and that the time between changes in these *relative intensities* impacted on the performance of listeners to be able to distinguish variations in loudness, which serves to support the influence of duration on loudness perception.

Zwicker and Scharf (1965) and Moore, Glasberg and Baer (1997) have proposed models of loudness perception that can be applied to computational models. Zwicker and Scharf's model (1965) applies spread excitation of the characteristic frequencies, allowing for *intensity discrimination* as well as the intensity overall level of a signal to be obtained by calculating the sum of the loudness values for each characteristic frequency. The Moore, Glasberg and Baer (1997) model proposes changes to the Zwicker (1965) model, particularly in relation to the masking of sound sources.

When applying computational models of loudness perception within the compositional process, the three assessments possible of overall intensity, *intensity discrimination* and *relative intensity* must be considered; each has a considerably different role in terms of auditory scene analysis, and consequently, on the compositional process. Therefore, the application of loudness models can be to identify the intensity of an overall signal, the intensity of a single source and the intensity of a source's frequency components over time. The use of these evaluations must then be chosen suitably to the compositional process required; for example, the

[115] Plack, C. 2005. *The Sense of Hearing*. New Jersey: LEA: 127
[116] Ibid

intensity of the overall signal comprised of ten sound-objects would be a suitable candidate to signify the dynamic macrostructure of a composition, but not to represent the spectral envelope of a single source. In particular, the classification of timbre is proposed to involve loudness perception, explicitly the *relative intensity*.

In addition to the suitability of a loudness perception model's assessment of overall intensity, *intensity discrimination* and *relative intensity*, the subjective nature of a loudness 'value' must also be taken into account; for tasks involving precise and accurate data such as the triggering of events at specific sound levels, perhaps the dB level provided by an acoustic signal would be more suitable and reliable as it represents a physical, quantifiable value. However, for analytical and generative tasks involving a representation of emotion, such as genre classification or automatic music reviews, the use of loudness perception, and therefore the perceived intensity of a live audio stream and/or audio recording is proposed to be necessary.

3.2.3 Timbre Perception

Considering timbre, a conclusive definition is difficult to qualify, reflected in Plack (2005) who defines timbre as 'that aspect of sensation by which two sounds with the *same* loudness, pitch, duration and ear of presentation can be distinguished. Timbre is often used to refer to the sensations associated with the overall spectral shape of sounds, but timbre is also dependent upon temporal factors such as the envelope. Generally, (and rather vaguely) timbre refers to the "quality" of a sound'[117].

So, timbre can be considered to be that sonic feature which allows us to classify a sound's 'type'. The complexities of expressing a sound-object's 'type' is demonstrated by Dannenberg (1993) who states that 'with many aspects of music, we know what to represent, and the issue is how to represent it. With timbre, we are still learning what to represent'[118]; the difficulties in quantifiably defining the topology of sound-objects, and therefore their timbral representation, is raised in relation to the *reduced listening* method and its requirement of the listener to disassociate sonic

[117] Plack, C. 2005. *The Sense of Hearing*. New Jersey: LEA: 249
[118] Dannenberg, R. 1993. Music Representation Issues, Techniques and Systems. *Computer Music Journal* 17(3): 26

features from their source to produce a sonically pure representation of a sound-object.

Further to the topology of sound-objects and *reduced listening* proposed by Schaeffer (1966), 'topology studies the properties of objects (or spaces) which are not changed by continuous deformations. Roughly speaking, what properties of a rubber object are retained if it is stretched in any conceivable way but not broken torn or pierced?'[119], so the model proposed by Schaeffer (1966), by definition, does not address dynamic changes of a source's state, and therefore modifications in its timbral space.

As a result, Wishart (1996) questions the suitability of such a sound-object classification method 'Does timbral space have a topology? When working with existing musical instruments we may construct a map of the timbral possibilities of the instrument. To do this, rather than merely listing all the possible sound-types which an instrument such as a violin might produce, we would attempt to place these on a map (which might be multidimensional) on which similar sound-objects would be placed close to each other and sound-objects which are quite different from one another would be placed at a greater distance'[120]. Therefore, indeed it may be possible to classify sound-objects within a topological map bound by the timbral space offered by a sound source.

The physical limitations of a performer or the instrument itself impacts on the structure of a topological map however, so despite the possibility of classifying a source's timbre by its sonic features, a definitive and static organization of the topology of a sound-object is not possible. Continuing Wishart's (1996) example of the timbre a violin may produce, 'At least it is relatively easy to get from normal arco sounds to multiphonics played arco sul ponticello by infinitesimal motion in the timbre space (adjacency) but relatively difficult to get from normal arco to percussive effects on the wooden body of the instrument. In fact, to make a 'modulation' in the timbre space from arco sounds to percussion on the wooden body sounds, it is essential to through col legno production or through pizzicato. This means that timbral space viewed as space in which timbral progressions (modulations) will be

[119] Wishart, T. 1996. *On Sonic Art.* London: Routledge: 82
[120] Ibid

made has a distinct structure which, although neither closed nor having a metric, imposes specific limitations on our musical options'[121]. As a result, the structure of a topological map is not only unique to each sound-object, but must also continually change in relation to the physical limitations imposed on the modulation of its timbre.

As with the difficulties of the organization of a topological map and classification of a sound-object, the information required to describe timbral features is also not definitive or reliable. In terms of the component timbral information available in the frequency domains, 'the timbre of a complex tone depends in part on the relative magnitude of the various harmonics of which it is composed… Instruments that produce intense high harmonics (e.g., a trumpet) will tend to sound "bright". Instruments that produce intense low harmonics (e.g., a French horn) will tend to sound "warm" or "dark"'[122]. It is important note to here that the previous example itself demonstrates the very issue of relevant timbral representation of sound-objects: what is "bright", what is "warm" and what is "dark"? These are by no means conclusive descriptions, reflected in the quoted author's use of speech marks to identify each term's ambiguity.

With regards to this verbalization of timbre, verbal scales may be applied, such as a sound-object's perceived '*brightness*, *richness*, *sweetness*, *pleasantness*, *fullness* and *roughness*'[123]. However, 'one of the major disadvantages in using verbal scales to investigate the properties of stimuli, of course is that words may not exist to describe certain perceived differences'[124]. This therefore leads to considerable uncertainty in a verbal description's relevance as a representation of a sound-object's timbre.

The representation of objective values such as frequency and amplitude to describe timbre with subjective scales creates substantial difficulties; subjective processes 'cannot justifiably be treated with the algebra of dimensional analysis that underlies measurement in the physical sciences'[125]. For the perception of timbre, a number of perceptual judgments appear to be combined together such as pitch, loudness and

[121] Wishart, T. 1996. *On Sonic Art*. London: Routledge: 82
[122] Plack, C. 2005. *The Sense of Hearing*. New Jersey: LEA: 25-26
[123] Grey, J. 1975. *An Explanation of Musical Timbre*. Stanford University
[124] Ibid
[125] Wessel, D. 1979. Timbre Space as a Musical Control Structure. *Computer Music Journal* 3(2): 47

spatialisation, forming a multidimensional measurement of sound-objects to form timbral judgments. Therefore, the direct transcription of frequencies and amplitudes to the timbral classification process is not sufficient for concise timbral analysis.

Primarily, it is proposed that the spectral envelope formed of the *relative intensities* of the frequency components (as discussed previously in relation to models of loudness perception) over time are most important for the description of a sound's timbre (Risset, 1966). In particular, three sonic features are important for the classification of a sound-object's timbre; '1) the *relationships* of the attack times of the harmonics, whereby successively higher harmonics take longer to appear and grow more slowly; 2) the fluctuation of the frequency, which is of small amplitude, fast, and quasi-random; and 3) the harmonic content of the tone, which becomes richer in high-frequencies when the over intensity increases'[126].

The spectral envelope is mutually agreed to be part of the timbral perception process (Risset, 1966; Chowning, 1973; Grey, 1975; Wessel, 1974), however, the role of processes outside of this such as spatialisation are disputed, as reflected in Dannenberg's (1993) sentiment; 'as aspects of timbre are isolated and understood, such as spatial location and reverberation, these components come to be regarded separately, leaving timbre as impenetrable as ever'[127].

Representative of the strength of the spectral envelope in the classification of timbre, statistical models have been proposed; Mel-frequency cepstral coefficients (MFCCs) and linear frequency coefficients (LFCs) are suggested to model timbral space (Terasawa, Slaney and Berger, 2005). Both methods are capable of statistically modelling spectral shapes over time, thereby representing a spectral envelope. In addition, the root-mean-square (RMS) of the spectral envelope in combination with the Karhunen-Loève Transform also demonstrates another method for timbral classification for single tones (Kaminsky and Materka, 1995). Another study showed that the constant-Q coefficient could also be applied for the modelling of the spectral envelope of a sound, and therefore the representation of timbre (Brown and Puckette, 1992). Despite the apparent success of statistical models, the statistical data produced

[126] Grey, J. 1975. *An Explanation of Musical Timbre.* Stanford University
[127] Dannenberg, R. 1993. Music Representation Issues, Techniques and Systems. *Computer Music Journal* 17(3): 25

using processes such as MFCCs or LFCs is not perceptual, and is therefore an objective value, contradicting the notion that timbre is formed of multidimensional *perceptions*.

Considering the representation of the perceptual process of timbral classification 'we have not begun to understand the contextual and individual differences involved in timbre perception'[128], so the absence of perceptual process in calculations such as MFCCs is an issue, which as yet, remains unsolved. That is not to say however that the data supplied by such statistical processes cannot be applied for timbral classification roles, albeit perpetually limited ones. As a result, through the use of statistical methods, the topology of sound-objects is strictly limited to the statistical method used. However, the data generated by each method is relative to itself, thereby allowing for an infinite map of a sound-object's spectral shapes in terms of the statistical method. This map of spectral shapes may then be applied to define a unique topology of timbral features relative to the outputs of the statistical approach.

So, with regards to the compositional process, through statistical methods of timbre representation, it is possible to obtain the spectral envelope of a sound-object in terms of the statistical method used (such as MFCCs) and apply this data as a *pseudo-timbre*. That is to say, a statistical method's representation of a sound-object is not what has been previously denoted as timbre, but as a dimension of it, from which comparisons and analysis of a sound-object's spectral shape can be made, therefore representing an aspect of proposed timbral features. As highlighted previously, the very definition of timbre and its temporal features is not unequivocal. Thus, timbre representation cannot be quantified or qualified to a determined value, hence the requirement for such a concept as a pseudo-timbre, involving relative dimensions, which can be adequately represented.

In relation to the organization of the topology of sound-objects, statistical methods can only form static topological structures as the qualitative distinctions required to form dynamic topological structures necessitate perceptual processes outside of defining spectral shapes, which, as discussed, are not present in such models.

[128] Terasawa, H, Slaney, M and Berger, J. 2005. Perceptual Distance in Timbre Space. draft for *ICAD'05*: 8

However, the organization of these static topological structures can use artificial neural networks and in particular, self-organising maps (Cosi et al, 1994) to generate neural nets that automatically define timbral spaces relative to the statistical method use, offering an efficient process of topological organization of static state sound-objects. Timbre perception is therefore limited in its applicability to a computational compositional process, and as demonstrated, the subject itself requires significant progression in terms of definition and resolution over the multidimensional characteristics it may involve before any conclusive use of its proposed features can be applied to composition itself.

3.2.4 Musical Time and Melody Perception

Through the use of a variety of combinations of the perceptual models of pitch, loudness, and timbre described in this section, it is possible to construct models of perception for tasks that require a number of perceptual judgments such as melody, gesture, genre and rhythm classification. In addition, physical values that can be obtained from a waveform, such as its amplitude and frequency, may also contribute to such processes. As a result, analysis may be made of *macro* and *micro* structures of live streams from subsymbolic data represented in the frequency domain through transform techniques such as the Fast Fourier Transform. It is once again important to note the many complexities of representing subsymbolic data for symbolic tasks, such as those previously listed of melody, gesture, genre and rhythm classification. Therefore, the examples presented in the following are *not* fully representative of the respective research area, but serve to show the various tasks and analytical processes that are possible (or not).

The identification of the onset of sound-objects is a primary sonic feature that many analytical tasks require for temporal structuring. The use of onsets as a temporal cue allows the listener to identify the beginning of a sound event, which can either be the initial onset of sound-object or a morphological change in a sustained sound-object's sonic features. If these onsets, when placed in sequence, feature repetition or periodic cues, the sequence forms a structure which may then be used to define a sound-object's rhythm. In terms of the definition of rhythm itself 'in its most generic sense,

the word *rhythm* is used to refer to all the temporal aspects of a musical work, whether represented in a score, measured from a performance, or existing only in the perception of the listener' [129]. Therefore, rhythm may be used for applications such as 'tempo induction, beat tracking, quantization of performed rhythms, meter induction, and characterization of intentional timing variations'[130] and can be applied to other perceptual processes such as gesture and melody recognition.

Before a temporal structure can be formed, the onsets defining the sequence of the structure's temporal features need to be identified. Various features of a sound-object may be used to identify onset information and are listed as follows: onset time, duration, relative amplitude, pitch, harmony, spectral energy, and low-level metrical values (Gouyon and Dixon, 2005). The gathering of onset data from symbolic sources can be obtained simply by using the relevant symbolic data containing the desired onset feature information. However, obtaining onset data from an acoustic signal is a complex task, particularly if more than one sound source is present which causes undesirable interference masking. Assuming that a sound-object can be isolated sufficiently enough to identify an individual sound-object's onset by one of the features listed above, *Figure 4* illustrates how this may then be applied for specific temporal and rhythmic processes[131]:

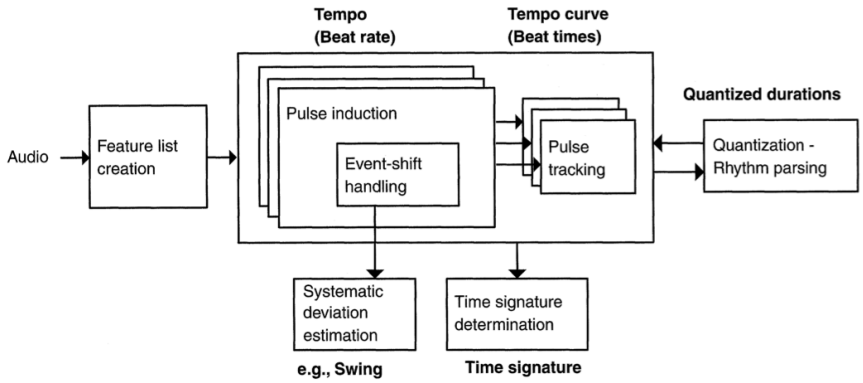


Figure 4. Proposed structure of beat and tempo extraction

[129] Gouyon, F and Dixon, S. 2005. A Review of Automatic Rhythm Description Systems. *Computer Music Journal* 29(1): 34
[130] Ibid: 34
[131] Ibid: 39

It is therefore possible to see in the *Figure 4* the number of rhythmic and temporal features that may be analysed, which are all obtained from the onset of sound-objects. In summary, the following list describes each of the main tasks and outputs available[132]:

> Pulse Selection – The definition of periodic events, which may indicate pulse
> Pulse Induction – The definition of a metrical level or pulse with short-term timings
> Pulse Tracking - The definition of a metrical level or pulse with long-term timings
> Event Shift Handling – The ability to define short-term timing differences that do not affect the long-term timing
> Rhythmic parsing – Placing onsets on a metrical grid, thereby quantizing their values
> Systematic Deviation Estimation – The assessment of short-term timings in relation to defined grids featuring non-metrical timings
> Time Signature Determination – assessing a complete temporal description to define a time signature

Given the number of tasks possible, Povel and Essens (1985) proposed a reactive model that could 'given a sequence of inter-onset intervals as input, identify the clock a listener would associate with it'[133]. In contrast, Desain's predictive model (1992) suggests that rhythm can be 'decomposed into basic expectancy components projected by each time interval implicit to the sequence… The resulting expectancy of complex temporal patterns can be used to model such diverse topics as categorical rhythm perception, clock and meter inducement, rhythmicity, and the similarity of temporal sequences'[134]. However, both models provide adequate representations of temporal cues despite utilising different model structures, indicating there is no default choice between reactive and predictive models.

---

[132] Dannenberg, R. 1993. Music Representation Issues, Techniques and Systems. *Computer Music Journal* 17(3): 20-30
[133] Scheirer, E. 2000. *Music-Listening Systems.* MIT: 43
[134] Desain, P. 1992, A (De)Composable Theory of Rhythm Perception. *Music Perception* 9(4): 439

Combined with the contrasting methods of model organization of reactive or predictive, as with the perceptual models of pitch, loudness, spatialisation and timbre, there is little agreement over how best to represent rhythm; Honing (2001) suggested that absolute onset, metrical structure, tempo and timing are required for a conclusive definition of a musical composition's rhythm, but there is no agreement as to how these features should be represented. Further to this, Gouyon and Dixon (2005) state 'different rhythmic features are relevant at each step in the music communication chain, at each step where rhythmic content is produced, transmitted, or received… A second reason for lack of consensus is that the diverse media used for rhythm transmission suffer a trade-off between the level of abstraction and the comprehensiveness of the representation'[135]. This reflects again the complexities of transcribing perceptual, subjective processes into objective values required by symbolic structures.

In addition, adequate pitch, loudness and timbre perception models are required to identify the various features that may be used to more accurately identify onset, thereby forming a more accurate representation of temporal structures. To demonstrate, at its most fundamental level, the perception of a sound-object's onset is attributed to sound level, which may be described in amplitude. So that is to say, if the amplitude of a sound-object is above a certain amplitude level, it is recognised to be the onset of a sound-object. However, as demonstrated by the research into loudness perception, experiments have shown that bandwidth and frequency affect our perception of a sound's intensity but do not affect the overall amplitude. Therefore, onsets will be misrepresented unless adequate weightings are applied to the amplitudes and the onsets that occur at their value, relative to a loudness perception model.

This highlights a significant issue of this area of musical research; 'there is no definitions or evaluation criteria, because rhythm description systems have been built for diverse applications with diverse data sets' [136]. For example, rhythm perception models that attempt to identify definite and periodic measures of metre or tempo have limited applicability in compositional structures that do not pertain to such

[135] Gouyon, F and Dixon, S. 2005. A Review of Automatic Rhythm Description Systems. *Computer Music Journal* 29(1): 34
[136] Ibid: 49

conventional formalism as those rhythmic organisations are null and void. As a result, the temporal or rhythmic model applied to a compositional process *must* be directly related to the systems and data required by that specific model, rendering the evaluations of many temporal and rhythmic models insufficient to each exclusive musical composition unless extensive steps are made by the composer to accommodate for the requirements of such a model, which may lead to significant changes in a resulting composition.

As mentioned, the use of pitch can be used to indicate the onset of a sound event. More importantly however, the pitch of a sound-object can be applied to processes that specifically require pitch information such as pitch contour recognition. As described previously, pitch is defined to be 'that attribute of auditory sensation in terms of which sounds may be ordered on a scale extending from low to high'[137], hence the necessity to define pitch in terms of pitch contours and not *melody*, which is a strictly tonal formalism. So, pitch contours can of course represent pitch in terms of their respective tones or semitones, but also in microtones, allowing them to be inclusive of contemporary and electroacoustic compositional techniques such as those found in *Genesis*.

Pitch contour recognition requires adequate pitch perception models from which a pitch can be ascertained, which may then be placed sequentially in relation to its position on the scale of low to high, relative to the sound-objects preceding it. In terms of the mapping of this structure, the scale used dictates this, which as a result allows for both tonal and microtonal organization; the mapping can be set to specific frequencies corresponding to chosen pitches, which can include those of a major or minor scale. Much of the research into this area has been completed with a disposition towards tonal structures, which limits its application to many of the compositional approaches such as microsound composition. Despite this, it is still possible to use the *frameworks* of pitch classification models to represent the progression of pitch over time by modifying the mappings the models apply to organise pitch information.

[137] American National Standards Institute (ANSI). 1994. *American national standard acoustical terminology.* New York: Acoustical Society of America: 34

In terms of structuring the mappings of pitch contour recognition process, a *frequency* centre is required, that is to say a tonal centre of *key* if applying tonal formalisms, in order to define a scale with which to place pitches on. This can be defined prior to the execution of a pitch contour model, resulting in a static frequency centre for the duration of its implementation. However, from the pitches themselves, it is possible to dynamically change the frequency centre by applying an array of potential scales. Krumhansl and Kessler (1982) proposed a successful model that can identify dynamic key changes by correlating pitches to a tonal hierarchy of the 24 major and minor keys. Therefore, the pitches can be used not only to denote the shape of pitch contours, but also the frequency centres with which they are perceptually associated.

Once a frequency centre has been obtained, it is then possible to use a further hierarchy to classify the relatedness of a pitch within a pitch contour. The notion of relatedness between pitches can be defined through Narmour's model (1990) which 'proposed several rules that describe what listeners prefer to hear in melodies, based on the principles of good continuation, closure and return-to-origin'[138]. This is based on the principles of the organization of sensory stimuli proposed by Koffka (1935), presented previously. As a result, the model views melody on a note-to-note basis, thereby negating the necessity to consider the influence of macrostructures on the perception of a melody. Such a model therefore allows the analysis of music, which does not consist of tonal formalisms and could be considered the antithesis to Schenkerian analysis.

In contrast to Narmour's model (1990), and perhaps more commonly applied for the purpose of defining the relatedness of pitch values, are formal grammars using the fundamental rules of harmony. For example, the model proposed by Longuet-Higgens (1994) 'developed several models of musical melody and rhythm around phrase-structure grammars'[139] which are based on tonal structures. In addition, Martin (1996) proposed a model for the purpose of transcribing polyphonic music presented as an acoustic signal, which in itself requires a substantial level of pre-processing through auditory scene analysis for the deconstruction of the waveform into its individual

[138] Scheirer, E. 2000. *Music-Listening Systems.* MIT: 29
[139] Ibid

sound sources. The model uses extensive pitch perception models and auditory scene analysis based upon auto-correlation for the isolation of each source's pitch.

With regards to the organization of the pitch values by Martin's model (1996), a blackboard system is used which 'consists of a central dataspace (the blackboard), a set of so-called *knowledge sources* (KSs), and a scheduler'[140]. In summary, the pitches obtained through the extensive pitch perception models are placed on the blackboard, which are then assessed by the knowledge sources that 'fall under three broad areas of knowledge: garbage collection, knowledge from physics, and knowledge from musical practices'[141]. In relation to the organization by the system of the pitches in to tonal structures, the musical practices centre on features such as *intervals*, *octaves* and *chords*; distinctly tonal concepts. As stated, this significantly restricts the application of such models to music, which applies such formalisms, but the frameworks can be modified to incorporate alternative principles such as microtones. So, Martin's model (1996) may apply such techniques by the revision of the musical practices implemented to include contemporary structures of pitch classification, while still using adequate pitch perception models.

3.2.5 Gesture Perception

The expressiveness of a sound-object is described by De Poli (2004) as 'the means used by the performer to convey the composer's message and his/her own contribution to enrich the musical message'[142] with the types of expression and gesture categorised in to: performative, communicative and ancillary (Cadoz and Wanderley, 2000). In summary, 'performative gestures produce sound, and communicative gestures (nods, eye contact, and similar cues) direct other performers. Ancillary gestures – intuitive body movements of the performer while playing – are expressive or emotive gestures that communicate musical meaning to the observer'[143]. Therefore, both auditory and visual cues may be used to identify gestures.

---

[140] Martin, K. 1996b. *Automatic Transcription of Simple Polyphonic Music: Robust Front End Processing*. MIT: 4
[141] Martin, K. 1996a. *A Blackboard System for Automatic Transcription of Simple Polyphonic Music.* MIT: 5
[142] De Poli, G. 2004. Methodologies for Expressiveness Modelling of and for Music Performance. *Journal of New Music Research* 33(3): 191
[143] Overholt et al. 2009. A Multimodal System for Gesture Recognition in Interactive Music Performance. *Computer Music Journal* 33(4): 72

The auditory cues available are 'related to timing of musical events and tempo, dynamics (loudness variation), and articulation (the way the successive notes are connected)'[144]. So, there is a requirement of suitable loudness, pitch and timbre perception models in order to adequately represent the sonic features required for gestural identification. However, as demonstrated with perceptual processes that necessitate a combination of musical perceptions, the interrelation between them, and their consequent influence on a complex perceptual task is not comprehensive; De Poli states 'the understanding of the expressive information is still vague. While its importance is generally acknowledged, the basic constituents are less clear'[145]. Similarly, for the gathering of visual cues, there is no conclusive hierarchy of communicative or ancillary gestures, and their consequent influence on the perceived overall gesture.

It is important to note that the inclusion of visual cues significantly increases the complexity of an analysis, yet holds the potential to yield better results. For example, the model proposed by Overholt et al. (2009) makes use of computer-vision techniques that require digital cameras to visually stream the performer, in synchronization with the audio signal. Therefore, sufficient algorithms are required to identify the components of the visual scene, in a similar vein to auditory scene analysis, which brings with it many complications of adequate deconstruction of the visual environment. In addition, the inclusion of visual cues requires sufficient definition of their perceived visual gestures in combination with description of their relativity to the auditory cues; the interplay between visual and auditory gestures cannot be ignored, but it must be succinctly addressed before such models may conclusively represent musical gesture better than using auditory cues only.

Gestural models often use a method of deviation, from which comparisons can be made between the performance and the score. Deviation is used to identify 'where, how and why a performer modifies, sometimes unconsciously, what is indicated in the notation in the score'[146]. The use of a reference point from which a deviation can be made does not exclusively need to be a score and can indeed use approaches

[144] De Poli, G. 2004. Methodologies for Expressiveness Modelling of and for Music Performance. *Journal of New Music Research* 33(3): 192
[145] Ibid
[146] Ibid: 194

similar to a note-by-note method, such as that suggested by Narmour (1990) for pitch relatedness; 'the idea is that from structural description of a music piece, we can individuate units which can act as a reference at that level. Its sub-units will act as atomic parts whose internal details will be ignored. The expression is defined as the deviation from the norm as given by a higher level unit. For example, the expressive variations of the durations of beats are expressed with reference to the bar duration (as a ratio)'[147]. As a result, such a model has substantial suitability to temporal and pitch structures of contemporary techniques which do not follow Western Art tradition, and indeed the convention of a highly symbolic musical score.

Considering the application to the compositional process of the perceptual models presented in this chapter, it is clear that their use must be carefully considered. In particular, the model's structure in terms of the musical formalisms it is based upon will have a significant, if not a detrimental impact on any compositional process that relies on such models to analyse sound-objects. In addition, and equally as noteworthy, is the absence of qualifiable and quantifiable methods for approaching perceptual modelling; it is indeed useful to attempt to explain the listening experience, but it must be recognised that the explanations thus far are bound by the limited understanding of their respective processes. Therefore, the use of perceptual models *must* be regarded as presenting a particular perspective, which may or may not be representative of our listening experience.

The usefulness of perceptual models must lie in their ability to represent a particular perspective, which must be acknowledged by a composer, in order for a compositional process that applies perceptual models to successfully implement the composer's intentions. With the prospect of systems that offer mood-classification (Meyers, 2004) and automatic record reviews (Ellis and Whitman, 2004) which combine a substantial number of perceptual processes, this acknowledgement of a perceptual model's perspective becomes ever more important; the interrelation between the perspectives must not be overlooked, otherwise, the ultimate perspective of the source may become so abstract that it cannot be understood, and therefore, inapplicable within a compositional process.

[147] Scheirer, E. 2000. *Music-Listening Systems.* MIT: 29

Chapter 4

Interactivity in Digital Music Systems

4.1 Interaction with Creative Systems

Interactivity, in the context of computer music systems, can be achieved through a broad range of approaches including 'installations, networked music ensembles, new instrument designs and collaborations with robotic performers (Eigenfeldt and Kapur, 2008)'[148]. The *Genesis* system offers many potential interactive methods such as networked instances, audience participation and performer-driven control of its generative outputs. However, in terms of a conclusive definition of interactivity, there is much discourse surrounding the issues of *what* makes a computer music system interactive, and *how* interactivity can be realised relative to the approach taken.

Therefore, in designing and implementing *Genesis*, the research required considerable investigation into proposed methods of interactivity and how these can be applied to the fundamental principle of *Genesis* to allow composers and performers to interact with real-time sound-objects. Furthermore, the models and proposed implementations of interactivity discussed in this chapter are applied directly to the evaluation of the system in chapter 6 *Evaluation of the Genesis System*.

Primarily, it is necessary to propose *what* interactivity is. At its most fundamental level, 'interactivity comes from a feeling of participation, where the range of possible actions is known or intuited, and have significant and obvious effects, yet there is enough mystery to spark a curiosity and exploration'[149]. Therefore, an interactive system, such as *Genesis*, must be able to form a reasoned response to an action provided by a user, which provides interest and 'novel circumstance'[150]. Thus, music software that offers an interaction method proposed by Winkler (2001) would result in a resolutely *interactive music system*. But, what constitutes a *reasoned* response? If a computer music system is to be defined as interactive, the responses provided must bear relevance to the inputs of the user, allowing their actions to form an unfolding dialog that can be understood by both parties.

---

[148] Drummond, J. 2009. Understanding Interactive Systems. *Organised Sound.* 14: 125
[149] Winkler, T. 2001. *Composing Interactive Music: Techniques and Ideas Using Max.* Massachusetts: MIT Press: 3
[150] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49

A reasoned response implies that cognition is present, and therefore a cognitive response is necessary in both the human user *and* the computer music system in order to interact, resulting in differences of opinion in *how* interactivity can be implemented. Paine (2002) considers that the use of the term interactivity has been 'abused' by the new media arts due to the perception that most systems 'are not interactive, but simply reactive or responsive because they lack a level of cognition'[151]. Such a supposition is based on a semantic definition of interaction; there is a reciprocal process between the actions of a human and a computer, and for this to occur, *both* parties must 'think' to achieve interactivity.

In contrast, Rowe (1993) defines three response types: *transformative*, *generative* or *sequenced* - 'the *transformative* and *generative* classifications imply an underlying model of algorithmic processing and generation. Transformations can include techniques such as inversion, retrograde, filtering, transposing, delay, re-synthesis, distortion and granulating. *Generative* implies the system's self-creation of responses… *sequenced* response is the playback of pre-constructed and stored materials'[152]. Consequently, although Rowe (1993) does not exclude cognitive processes in his categorization of an interactive music system's responses, he does make acknowledgment of those approaches that do not demonstrate observable cognition.

So, with regard to the cognitive abilities of computer systems, as highlighted in chapter 2.3 *Computers and Algorithms*, regarding McCarthy's (2007) comments on the limitations and confines of artificial intelligence, the idea that a computer must demonstrate a level of cognition in order to form an interactive process, as supposed by Paine (2002), is perhaps flawed from the outset; in the absence of clearly observable cognition, how are we to conclude that a computer music system such as *Genesis* can create reasoned responses, thereby forming an *interactive* computer music system?

Considering the many methods of generative algorithms presented in chapter 3.1 *An Introduction to Real-time Generative Algorithmic Systems*, it is generally accepted

[151] Paine, G. 2002. Interactivity: Where to from Here?. *Organised Sound* 7(3): 295
[152] Drummond, J. 2009. Understanding Interactive Systems. *Organised Sound.* 14: 128

that generative processes can form part of a creative process *without* explicit cognitive ability. Therefore, if a reasoned response can be generated by such an algorithm without the need for perceptible cognitive processes to be present, it is proposed that it is possible to form reasoned responses from *Genesis* through the use of generative algorithms such as Markov chains, genetic algorithms and fractals.

Indeed, generative algorithmic implementations reflect the absence of unequivocal machine cognition, and the realisation by composers that 'a mechanical thinking brain is very far from realisation'[153]. For instance, Blackwell et al (2012) constructed the *Live Algorithm* with the aim to 'emulate human performance convincingly enough that companion improvisers, and listeners, would accept the Live Algorithm as a contributing and creative group member with the same musical status as any other performer' [154]. Similarly, with *Genesis*, a significant challenge is to ensure communication between a human and the machine is through a language that allows both parties to create positively and effectively, with the prospect that the system's *creativity* may be considered with the same regard as a human performer, thereby increasing the perceived level of interaction.

Yet, how are we to define the outputs of a machine as creative, be it from *Genesis* or otherwise, if cognition is unobservable and therefore unreasoned? Considering that many models of creativity and those presented in chapter 2 *An Introduction to Algorithmic Composition* are founded upon psychological phenomena and implied cognition, perhaps as Bown (2012) suggests 'we require a broader view of creativity as the process of creating novel things, not limited to a suite of psychological capacities'[155].

Indeed, Bown (2012) proposes two forms of creativity: *generative* ('an instance of a system creating new patterns or behaviours regardless of the benefit to that system. There is an explanation for the creative outcome, but not a reason'[156]) and *adaptive* ('an instance of a system creating new patterns or behaviours to the benefit of that

[153] Blackwell et al. 2012. 'Live Algorithms: Towards Autonomous Computer Improvisers' in *Computers and Creativity*, eds J McCormack & M d'Inverno, Springer, Berlin: 148
[154] Ibid
[155] Bown, O. 2012. 'Generative and Adaptive Creativity: A Unified Approach to Creativity in Nature, Humans and Machines' in *Computers and Creativity*, eds J McCormack & M d'Inverno, Springer, Berlin: 362
[156] Ibid: 364

system. The creative outcome can be explained in terms of its ability to satisfy a function'[157]). So, it could be argued that *generative* creativity is machine-based and *adaptive* creativity is human.

Considering machine-based *generative* creativity, due to its absence of reason, it is value-free, creating for an unknown purpose, following its algorithmic iterations with no intention or goal. In contrast, *adaptive* creativity is only observable in humans, as reason and cognition dominate the process giving value and purpose to the human creative process; as Bown states 'adaptive creativity is… intended to describe the familiar understanding of human creativity as a cognitive capacity'[158].

However, *generative* and *adaptive* creativity is not a duality. Due to sociological factors such as style and genre, along with chance (as detailed in chapter 2.4 *Unpredictability and Randomness in the Creative Process*), *generative* creativity may be observed in humans, whereas conversely, *adaptive* creativity cannot be achieved in machines due to the absence of cognition and reason. As a result, it is proposed that a hybridisation of *generative* and *adaptive* creativity methods is implemented to successfully interact with machines.

So, if creativity is approached without reason and observable cognition in machines, this does not render any product of the machine to be void of creativity; machines create *generatively*, through which a creative outcome can be explained, with its reason accountable to the *adaptive*/*generative* creativity of the human user. Recalling the Lovelace Test[159], introduced in chapter 2.3 *Computers and Algorithms*, such a creative method confirms the results of the test thus far, in which a system's actions can be explained without their reason responsible to the machine, and instead to their human designer.

The implementation of *adaptive* creativity 'is the more traditional goal of arts-based computationally creative systems, but faces the challenge that the embodiment and

[157] Bown, O. 2012. 'Generative and Adaptive Creativity: A Unified Approach to Creativity in Nature, Humans and Machines' in *Computers and Creativity*, eds J McCormack & M d'Inverno, Springer, Berlin: 364
[158] Ibid
[159] Bringsjord et al. 2001. Creativity, the Turing test and the (Better) Lovelace Test. *Mind and Machines* 11: 3-27

situatedness of the artificial system is a poor reproduction of that of the human'[160]. Therefore, if the objective is to represent solely human creativity through a machine, the absence of reason and cognition make such a goal unattainable. Conversely, a 'generative creativity approach seems equally problematic since generative systems are not adapted to goals and so cannot perform functions similar to human adaptive creativity'[161].

However, collaborative systems, in which the human and machine interact with each other through a chosen paradigm such as *AARON* (McCorduck, 1990) and *Voyager* (Lewis, 2000) have demonstrated successful examples of combining the principles of *generative* creativity and *adaptive* creativity to form interactive, creative machines. For example, the interactive music system *Voyager* (Lewis, 2000) is designed to analyse a real-time human improvisation and 'generates both complex responses to the musician's playing and independent behaviour that arises from its own internal processes'[162], thereby *generatively* creating outputs relative to its received *adaptive* inputs from the human performer and *Voyager's* inherent *generative* behaviours.

As a result, a combination of *generative* and *adaptive* methods of creativity must be considered when constructing creative machines, allowing the computer to create *generatively* and the human to create *adaptively*, with the interaction of the two parties forming a unified creative method. So, with machine creativity, due to its inability to qualify a *generatively* creative process in the real world, 'such systems can only be involved in adaptively creative processes with an adaptively creative individual masterminding this process'[163]. Therefore, if we are to generate creative machines, interactivity between a human and a system is necessary. Indeed, the interaction methods of the *Genesis* system (described in detail in chapter 5 *The Genesis System*) is designed to encourage collaboration between the human and the machine, with the aim of allowing the human performer to ultimately oversee and validate the ongoing creative process.

---

[160] Bown, O. 2012. 'Generative and Adaptive Creativity: A Unified Approach to Creativity in Nature, Humans and Machines' in *Computers and Creativity*, eds J McCormack & M d'Inverno, Springer, Berlin: 374
[161] Ibid
[162] Lewis, G. 2000. Too Many Notes: Complexity and Culture in Voyager. *Leonardo Music Journal*. 10: 33
[163] Bown, O. 2012. 'Generative and Adaptive Creativity: A Unified Approach to Creativity in Nature, Humans and Machines' in *Computers and Creativity*, eds J McCormack & M d'Inverno, Springer, Berlin: 374

So, in contrast to Paine's (2002) supposition regarding the abuse of the term *interactivity* in the new media arts, and considering the need for human supervision of creative machines, it is proposed that the use of the term *interactivity* must be flexible. Wanderley (2001) suggests five interpretations of interaction in musical context which are as follows[164]: *instrument manipulation*, *device manipulation in the context of score-level control*, *other interaction contexts related to traditional HCI interaction styles*, *device manipulation in the context of post-production activities* and *interaction in the context of multimedia installations*. Through application of such interactive methods, *generative* and *adaptive* creativity is achievable in *Genesis*.

In selecting the interactive methods proposed by Wanderley (2001) for *Genesis*, it was necessary to consider the relationships formed with a collaborative approach between a human user and the system primarily communicating through real-time sound-objects, and how this may impact on the creative process. Chadabe (1997) noted, with reference to early examples of interactive musical instruments such as his own CEMS System developed in the early 1970s, '…these instruments were interactive in the same sense that performer and instrument were mutually influential. The performer was influenced by the music produced by the instrument, and the instrument was influenced by the performer's controls'[165]. Thus, 'in interactive music systems the performer can influence, affect and alter the underlying compositional structures, the instruments can take on performer-like qualities, and the evolution of the instrument itself may form the basis of a composition'[166].

So, in relation to the application of generative algorithms in *Genesis*, it is possible for the user to influence an ongoing compositional process, such as the real-time modification of a stochastic model's probability distribution, the fundamental frequency from which a fractal process is to begin or the triggering of granular synthesisers by the onsets of real-time sound-objects, with the outcomes of the algorithmic processes influencing the user's following actions. As a result, the user influences the system and the system influences the user, forming a shared

[164] Wanderley, M. 2001. *Gestural Control of Music*. IRCAM, Paris: 2
[165] Chadabe, J. 1997. *Electric Sound: The Past and Promise of Electric Music*. New Jersey. Prentice Hall: 291
[166] Drummond, J. 2009. Understanding Interactive Systems. *Organised Sound*. 14: 125

interaction, with outputs creating 'novel circumstance'[167] relative to the applied algorithmic process and the adjustments of the user.

It must be noted however, that the influence of the actions provided between the user and system are variable; Chadabe's (1997) observations indicate that influence in an interactive process remains constant, that each action from both parties is accepted with the same degree of acknowledgment. Therefore, following Chadabe's (1997) interactivity method, resulting interactions in an interactive computer music system will always have the same influence on the actions of the user and the system.

In contrast, a variety of interaction models (Rowe, 1993; Winkler, 2001; Paine, 2002) have been proposed which attempt to describe the relative levels of influence between users and interactive computer music systems. For example, Rowe (1993) states 'interactive computer music systems are those whose behaviour changes in response to musical input'[168]. Thus, the behavioural changes will cause variations in influence between the interactions of the user and the system. However, 'the emphasis in Rowe's definition is on the response of the system; the effect the system has on the performer is secondary'[169], thereby suggesting that a system's actions are less influential than the users, while still having a variable influence relative to the hierarchy of user followed by system.

Winkler (2001) extends the fundamental principle that Rowe (1993) established by acknowledging variations in hierarchy between user and system - *the Conductor Model*, *the Chamber Music Model, The Improvisational Model* and *Free Improvisation*[170]. So, Winkler (2001) makes full acknowledgement of the different levels of influence achievable between user and system. Despite this, both Rowe's (1993) and Winkler's (2001) models have limited applicability to methods of interaction that are not driven by instrumental performance; 'in discussing the types of input that can be interpreted, the focus is restricted to event-based parameters such as notes, dynamics, tempo, rhythm and orchestration'[171]. Therefore, both models are

[167] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49
[168] Rowe, R. 1993. *Interactive Music Systems: Machine Listening and Composing*, Cambridge, MA: MIT Press: 1
[169] Drummond, J. 2009. Understanding Interactive Systems. *Organised Sound.* 14: 125
[170] Winkler, T. 2001. *Composing Interactive Music: Techniques and Ideas Using Max.* Massachusetts: MIT Press: 23-27
[171] Drummond, J. 2009. Understanding Interactive Systems. *Organised Sound.* 14: 125

founded on established musical theory, whereas 'interactivity may offer an entirely new approach to music-making, and so in order to avoid getting stuck in the current musical paradigms, we should question not only the nature of the system input…, but we should pay equal attention to the output of the system, and the qualitative relationship between the two'[172].

Paine (2002) addresses the suggested limitations of Rowe's (1993) and Winkler's (2001) models by proposing an interaction model based upon the process of human conversation, described in the following[173]:

1. Unique and personal to those individuals
2. Unique to that moment of interaction, varying in accordance with the unfolding dialog, but is
3. Maintained within a common understood paradigm (both parties speak the same language, and address the same topic)

As a result, Paine's model (2002) forms a dynamic method of interaction, 'with each of the parties constantly monitoring the responses of the other and using their interpretation of the parties' input to make alterations to their own response strategy'[174]. With such a model, the responses are therefore proposed to be more appropriate to interactions that are not based on conventional musical formalisms. For example, 'when the input to the interactive system is a human gesture, it is questionable whether a musical construct, constrained by the precedents of historical musical practice (chromatic music for instance), is an appropriate response'[175]. Indeed, there is no perquisite for the *Genesis* system to use musical instrument-based sound-objects, therefore implicating that an approach which offers gestural communication from *any* sound source and indeed gestures that are not defined in musical formalisms is necessary.

[172] Paine, G. 2002. Interactivity: Where to from Here?. *Organised Sound* 7(3): 297
[173] Ibid
[174] Ibid
[175] Ibid: 298

Therefore, *Genesis* must be able to create 'novel circumstance'[176], unique and relative to its generative and analytical processes. Of most importance however is a dialog between human and computer that must allow the two parties to communicate gestures through a common language, resulting in an understanding of each other's responses; if a common understanding is absent, the generative and analytical processes that define either party's interaction will be irrelevant to the musical context defined by the received responses. This understanding of each other's responses is dependent on the representation of sonic features and the perceptual models that define their musical values. Through these commonly understood representations of a sound-object it is then possible to construct the desired mappings for the required model of interaction between the human and the computer, as described by the four models proposed by Winkler (2001).

When considering interaction with interactive music systems that conforms to a musical instrument paradigm such as a live instrumentalist, which *Genesis* allows, interaction of gestures can be categorised into performative, communicative and ancillary. In summary, 'performative gestures produce sound, and communicative gestures (nods, eye contact, and similar cues) direct other performers. Ancillary gestures – intuitive body movements of the performer while playing – are expressive or emotive gestures that communicate musical meaning to the observer'[177]. Therefore, auditory, physical and visual actions may be used to communicate gestures and interact with digital music systems. Through such gestural classifications, relative mappings can be defined to their respective characteristics, increasing the potential understanding of communication between a human user and a system, thereby improving the interactive method.

Within the proposed gestural categories associated with sound generation and control, a variety of methods have been applied to transfer analogous signals into the digital domain. However, there is a clear division between controllers that are founded on existing musical instruments and innovative ones (Sapir, 2002). For example VideoHarp (Rubine and McAvinney, 1990), Radio Drum (Matthews and Schloss,

[176] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49
[177] Overholt et al. 2009. A Multimodal System for Gesture Recognition in Interactive Music Performance. *Computer Music Journal* 33(4): 72

1989) and Hyperinstruments (Machover and Chung, 1989) are extensions of conventional musical instruments, adapted to offer increased gestural control of existing instruments. In contrast, Bodycoder (Wilson-Bokowiec and Bokowiec, 1995), GloveTalk (Fels and Hinton, 1993) and GAMS (Bauer and Foss, 1992) make use of physical body movements to dictate interactions. Furthermore, bioelectronics has been applied in systems such as Biomuse (Knapp and Lusted, 1990), which uses the electric signals in the brain to interact with digital music software.

Indeed, many generic methods of physical interaction are commercially available, such as the Korg NanoKontrol or the AKAI MPD series, alongside many piano keyboard-based controllers, which offer a series of knobs, sliders, keys and pads that can be assigned to MIDI CC numbers for limited control of sonic features within an interactive system such as note onset, amplitude and pitch. Moreover, the computer keyboard and mouse can also be assigned to trigger events or manipulate values through the X/Y axis of the mouse input. However, such interactions are often bound to a limited instrumental paradigm due to the restricted level of gestural control available; in comparison to Bodycoder (Wilson-Bokowiec and Bokowiec, 1995) in which a glove is worn in combination with sensors placed on the user's body to form a multidimensional control system, the click of a mouse, press of a MIDI piano key or turn of a knob would appear rather arbitrary.

However, it is necessary to consider the relative nature of the interactive system to the interface method. Commercially available software such as Ableton, Logic, ProTools and Cubase incorporate sequencing principles, through which 'audio signals or MIDI messages from an external instrument are captured in real-time, after a record button is pressed'[178]. Therefore a tape recorder metaphor can be applied in which the actions of the user are recorded, ready for playback and manipulation. Furthermore, in order to perform edits and alterations to the sequenced material, many window-based metaphors of physical instruments and equipment are employed, which can be controlled by keyboard, mouse and MIDI CC controllers, resulting in the user being able to interact with the sequenced data by moving virtual sliders, knobs and keys through their MIDI devices and/or computer keyboard and mouse.

[178] Nash, C and Blackwell, A. 2011. Tracking Virtuosity and Flow in Computer Music. *Proceedings of the ICMC'11*: 578

Nevertheless, when applying such metaphors for the physical manifestations of musical instruments and equipment 'a gulf opens up between the user's concept of music and what is easily encapsulated in the notation… The overly metaphorical correspondence to physical music equipment also means that interacting through generic devices, like the mouse, become cumbersome'[179]. However, using such metaphors of existing physical manifestations of musical equipment offers a significant degree of accessibility, allowing the user to engage with known conventional and generic parameters, validating the application of such devices for consumer use.

With *Genesis*, the primary method of control is through the sonic features of real-time sound-objects, through which its generative processes create outputs relative to its auditory inputs. Considering that human instrumental performers can provide such inputs, it is absolutely necessary to use a familiar musical paradigm, combined with an interface that captures and generates gestures satisfactorily; the implementation of musical paradigms will enable instrumentalists to better understand the generative processes of *Genesis* thereby ensuring a commonly understood paradigm between human performer and machine. Moreover, the design of the *Genesis* system must exceed the limitations of MIDI and associated software in order to form an extensively interactive music system while still maintaining accessibility to associated human performers.

Wessel and Wright (2002) consider that interactive systems should have a 'low entry fee', thereby acknowledging the need for accessibility to be a key focus in the design of interactive music systems. Yet, standardised and generic MIDI controllers and physical interfaces 'seem – after even a brief period of use – to have a toy-like character… one quickly "out-grows" the interface by discovering the limits of how it can be used'[180]. Therefore, not only do such devices often not reflect the true nature of the physical parameter they are controlling, as proposed by Nash and Blackwell (2011), they are also limited in their ability to convincingly offer intimate, interesting

[179] Nash, C and Blackwell, A. 2011. Tracking Virtuosity and Flow in Computer Music. *Proceedings of the ICMC'11*: 578
[180] Wessel, D and Wright, M. 2002. Problems and Prospects for Intimate Music Control of Computers. *Computer Music Journal*. 26(3): 12

and expert-level control of musical parameters and unsuitable for systems such as *Genesis*.

In contrast, considering extensive interaction methods, which offer significantly more gestural and intimate control of interactive systems than standardised and generic controllers, 'most traditional acoustical musical instruments are not easy to play at first but afford the development of a high degree of musicality'[181]. So, such a notion would affirm that extensive interaction methods should also offer such degrees of musicality. However, in contrast to the supposed steep learning curve of existing musical instruments, it is proposed that for such interaction methods 'a high degree of control intimacy can be attained with compelling control metaphors, reactive low latency variance systems, and proper treatment of gestures that are continuous functions of time'[182]. Therefore, such an approach is vital to ensure the success of interaction between a human user's real-time sound-objects and the responses of the *Genesis* system.

Despite the proposition of a 'low entry fee' for interaction with *Genesis*, considering the complex learning process associated with existing acoustic musical instruments, such a pedagogical process is necessary for a user to master new, extensive interactive methods; although the learning curve may not be as steep for novel gestural control of interactive music systems, the ability to obtain an expert-level of control and interaction still requires a significant amount of learning from the user to develop their musicality and understanding of the system's interactive properties, leading to a perceived virtuosity.

Though, when considering interactive music systems 'the primary virtuosity is not at the level of the instrument itself, but rather below the instrument at the strata of hardware and code… Virtuosity in contemporary musical composition can therefore be defined as the skill of designing and understanding constraints'[183]. So, it has been suggested that for truly virtuosic performance with interactive music systems, there is

[181] Wessel, D and Wright, M. 2002. Problems and Prospects for Intimate Music Control of Computers. *Computer Music Journal*. 26(3): 12
[182] Ibid: 12
[183] Magnusson, T. 2010a. Designing Constraints: Composing and Performing with Digital Music Systems. *Computer Music Journal*. 34(4): 70

a necessity for the user to also be the designer, thereby constructing and coding the methods through which interaction and sound generation can be achieved, relative to their desired conceptual constraints.

Furthermore, Magnusson (2010a) proposes that 'virtuosity in new digital instruments thus relates to the understanding of the system's core, an understanding typically achieved from the process of being its designer'[184]. Therefore, much commercially available software in which the user is *not* the designer limits the degree of virtuosity; in order to achieve such virtuosic performance, the user must follow the provided musical paradigms and metaphors that must be fully understood *and* be relevant to their desired compositional approach. So, considering that 'sadly but understandably, the electronic music instrument industry, with its insistence on standard keyboard controllers, maintains the traditional paradigm'[185], this inherently limits virtuosic potential of such systems and indeed, the musical style to those bound by conventional music theory.

Therefore, the implications of using a 'conventional' approach to interaction design through the use of a musical instrument paradigm, present in much commercial software, are that 'musicians are already familiar with them and can easily exploit their performance skills learnt over years of practice'[186]; recognisable musical values and conventions are required such as pitch, duration and onset through which the user can readily associate their interactions with the musical formalism applied within such an interactive system. Consequently, a familiar musical relationship can be formed between the user and the system, with the possibility that such an approach may yield better understanding by the user of the techniques applied in the system's responses. So, a more successful outcome is possible as the user and system are interacting through a method that is commonly understood.

In contrast, such instrumental approaches may lead to confusion; 'new themes of reflection arise when gesture is no more linked to sound production and when traditional expressions of virtuosity hardly find place in the music which is

[184] Magnusson, T. 2010a. Designing Constraints: Composing and Performing with Digital Music Systems. *Computer Music Journal*. 34(4): 70
[185] Drummond, J. 2009. Understanding Interactive Systems. *Organised Sound*. 14: 124-133
[186] Sapir, S. 2002. Gestural Control of Digital Audio Environments. *Journal of New Music Research*. 31(2): 120

performed'[187]. As a result, it is also possible that through the use of instrumental approaches, gestures and actions may be lost in translation due to the inherent nature of an instrument's inability to demonstrate successfully the user-perceived action, which may fall outside of musical formalisms such as pitch, duration and onset. Therefore, a primary focus in designing and implementing *Genesis* was to limit the degree of confusion between a gesture and the system's response while still presenting a significantly complex system that may have virtuosic potential.

It is necessary to consider the relatedness of a gesture to a resulting response from an interactive music system. Overholt (2009) poses three key questions regarding the relationship of gesture and the outputs of interactive music systems[188]; *How intuitive are the gestures?*, *How perceptible are the gestures?* and *How physical/powerful are the gestures?* As a result, the challenge with *Genesis* is to obtain relevant gestural information from real-time sound-objects and apply it successfully to relatable mappings which are intuitive, perceivable, and relative to the effort at source.

Primarily, in order to achieve congruency between the perceivable gestures of a real-time sound-object and *Genesis*, a method of interfacing such communication is necessary. MIDI, as noted, is not suitable for such a process; 'MIDI as a musical representation afforded interactive music systems access to very high-level, symbolic description of the music being played into the system and manipulated within. Because 'notes' were already clearly defined in terms of pitch, amplitude, onset and offset times, high level analyses including beat tracking (Desain and Honing, 1999), key induction (Toiviainen and Krumhansl, 2003), segmentation (Cambouropoulos et al., 2001), style identification (Dannenberg et al., 1997) and more could be performed from a relatively secure foundation'[189].

As a result, a number of communication protocols have been developed that allow considerably more intimate control of interactive music systems and are therefore implemented in the *Genesis* system; 'the Open Sound Control standard (OSC) is one of the most direct approaches to resolving the networking and representational

[187] Sapir, S. 2002. Gestural Control of Digital Audio Environments. *Journal of New Music Research*. 31(2): 120
[188] Overholt, D. 2009. The Musical Interface Technology Design Space. *Organised Sound*. 14(2): 218
[189] Rowe, R. 2009. Split Levels: Symbolic to Sub-Symbolic Interactive Music Systems. *Contemporary Music Review* 28(1): 32

limitations of MIDI (Wright and Freed, 1997). Other platforms have been shaped by international standards organizations, or by their connection to existing languages. Two of these are the Structured Audio Orchestra Language (SAOL) (Vercoe et al., 1999) and JSyn (Burk, 1998)'[190]. OSC is perhaps the most commonly applied alternative to MIDI, offering the application of extensive subsymbolic representation of sonic features, and indeed, also highly symbolic methods of representation if required.

Through the OSC communication protocol, it is possible to communicate significant levels of subsymbolic data such as the timbral changes over time of an acoustic instrument through subsymbolic representations of timbre (for example MFCCs) obtained from machine listening algorithms or the finger movement of a glove such as SoniMime (Fox and Carlile, 2005) for the intimate manipulation of timbral mappings within a synthesiser. With this level of gestural control data, considerable expressiveness unattainable through MIDI is achievable, thereby increasing the perceptible physical interaction with an interactive music system, consequently making performances more spectacular, as proposed by Sapir (2002), and significantly increasing the degree of virtuosity.

Open source software environments such as SuperCollider, CSound, Pure Data and Chuck and a limited selection of commercial software such as Max/MSP offer users the opportunity to build their own interactive systems using the OSC communication protocol, through which innovative and complex interaction controllers can be applied, thereby offering the potential to create virtuosic interactive music systems. In such software, the user is often presented with a modular method of interactive system design, in which individual modules such as sound generators, envelopes, generative algorithms and filters can be patched together.

Consequently, through the programming languages exclusive to the software environments, the user can create unique interactive systems for use with interaction methods of their choice and design. For example, Phalanger (Kiefer, Collins and Fitzpatrick, 2009), ixi lang (Magnusson, 2010a), iXiQuarks (Magnusson, 2007) and

[190] Rowe, R. 2009. Split Levels: Symbolic to Sub-Symbolic Interactive Music Systems. *Contemporary Music Review* 28(1): 33

Squeezables (Weinberg and Gan, 2001) use such software to create novel interaction systems, extending interaction far beyond that found in most commercial software. Therefore, *Genesis* required also a suitable programming language to implement and design the desired interactive, generative and analytical processes of real-time manipulation of sound-objects through an OSC interface. The SuperCollider[191] programming language offers the required interfacing methods of OSC combined with extensive GUI objects and generative/analytical unit generators making it an appropriate choice to realise the *Genesis* system.

In terms of designing the *Genesis* system relative to the fundamental principle of using the sonic features of real-time sound-objects for control of its generative processes, it was necessary to consider how its design should be approached. For successful design in interactive music systems, a variety of considerations were made with *Genesis* relative to the field of Human-Computer Interaction (HCI), which uses the concepts of *affordances*, *constraints* and *mappings* (Magnusson, 2010a). Both the concepts of *affordances* and *constraints* are based on ecological psychology, and thus have no conclusive definition, owing to many varied interpretations. Therefore, for the purposes of this thesis, *affordances* are considered to be the properties that an interactive music system offers, with *constraints* being the constructed limitations of the system.

Affordances of complex interactive music systems are often imperceptible, or at the very least, unpredictable, limiting the applicability of affordances in the design process in such instances. Instead, the design process of such systems should prioritise *constraints* as a method of forming an instrument's design strategy (Magnusson, 2010a). Such an approach indeed counters the method of design to conventional acoustic instrumentation, and simple interactive systems; 'instrument makers actively design affordances according to their understanding of musical performance and composition'[192].

[191] McCartney, J. 2002. Rethinking the Computer Music Language: SuperCollider. *Computer Music Journal* 26(4): 61-68
[192] Magnusson, T. 2010a. Designing Constraints: Composing and Performing with Digital Music Systems. *Computer Music Journal*. 34(4): 64

But, considering the relative complexity of intimate sound generation and control, and the data bandwidths required for complex interactive music systems such as the extensive full-body gestural controller GAMS (Bauer and Foss, 1992), the bow vibrating a string on a violin or beater hitting a skin on a drum represents the significantly limited application of constraints in acoustic instruments, and therefore the use of high-level affordances in their design process.

So, with regards to interactive music systems, a *mapping* 'is the location where constraints are defined and the instrument's functionality constructed'[193]. In terms of acoustic instruments, mappings are the physical gesture that connects the performer and instrument, which, as demonstrated, feature few constraints and therefore few mappings. With complex interactive music systems, the constraints are increased considerably, thereby necessitating many mappings to subsymbolic features such as timbre. As noted, most commercial software features an instrumental paradigm, often implemented through MIDI, which offers a relatively unified approach to mapping design such as key note number to pitch or key Note *On* to onset.

Indeed, Paine (2009) proposes a specific unified approach to the mappings of new interactive music systems, which provides guidelines for the design of novel gestural controllers and their consequent constraints and subsequent mappings. Paine's (2009) research demonstrates that through representing gesture in models outside of the instrumental/MIDI paradigm with the Nintendo WiiMote and the Intuos3 Wacom Tablet, including the physical mappings of pressure, speed, angle and position for the control of selected systems, it is indeed possible to move toward a unified approach to interface and interactive music system design. Yet, when considering the complexities of mapping with extensive interactive music systems, such a unified approach is currently unachievable.

If we are to consider the perceptual spaces of defining sound parameters through mappings, a determined level of ambiguity arises; which gesture should be applied to which parameter mapping? For example, through the use of novel gestural controllers such as the WiiMote, which gesture available in the three-dimensional gesture space

[193] Magnusson, T. 2010a. Designing Constraints: Composing and Performing with Digital Music Systems. *Computer Music Journal*. 34(4): 65

of an X/Y/Z axis, accelerometers and trigger buttons should be mapped to timbre? Indeed, perhaps all three-dimensions, with the X/Y/Z axis allowing for acute adjustment of timbral qualities, the accelerometers providing intense timbral shifts and the trigger buttons 'freezing' the current timbral space.

But this is one such possibility, and therefore other methods may provide a more successful gestural control space. And what of other constraints that may need to be controlled at the same time? How are they also to be manipulated by this three-dimensional gesture space in tandem with timbre? Such suppositions reiterate the relative complexity in designing constraints and their associated mappings within extensive interactive music systems.

Arfib et al (2003) propose criteria for the catergorisation of mappings, defined as 'explicit/implicit, simple/complex, and dynamic/static'[194]. In summary, explicit mappings provide definitive links between the input and the output, with implicit mappings being 'considered a black box for which we define behaviour rules but not precise values'[195]. Complex mappings are defined as many gestural parameters to many mappings, while simple mappings are one gestural parameter to one mapping. Finally, a dynamic mapping evolves and adapts over time, modifying its mapping hierarchy and parameters, while a static mapping remains constant, continually applying the same mappings throughout its application.

Therefore, if we consider Paine's (2002) model of interaction, in which a dynamic response strategy is proposed to be necessary to form reasoned responses to the actions of the user, a combination of all criteria outlined by Arfib et al (2003) may be required; such an implementation, particularly through the use of dynamic mappings, could enable the system to make alterations to its response strategy, relative to the data supplied though the explicit/implicit and simple/complex mappings, which could offer an intimate gestural control stream.

[194] Arfib et al. 2003. Strategies of mapping between gesture data and synthesis model parameters using perceptual spaces. *Organised Sound* 7: 130
[195] Ibid

However, with regard to the requirement of a low entry fee (Moreover, Wessel and Wright, 2002) for interactive music systems, a system that has the ability to change dynamically accompanied with extensive implicit and complex mappings may hinder the pedagogical process; responses may differ substantially from one interaction to the next, confusing a new user and limiting their understanding of a system's generative processes. Therefore, in order to achieve accessibility, and maintain continuity and predictability in *Genesis's* outputs, a combination of static, explicit and simple mappings are applied to aid a user to engage and learn the response types to the user's inputs.

In addition to Arfib et al's (2003) criteria regarding mapping categorisation, Magnusson (2010b) considers an epistemic dimension space in which mappings can be applied relative to the constraints of autonomy, music theory, explorability, required knowledge, improvisation, generality, creative simulation and expressive constraints, shown in *Figure 5* below[196]:

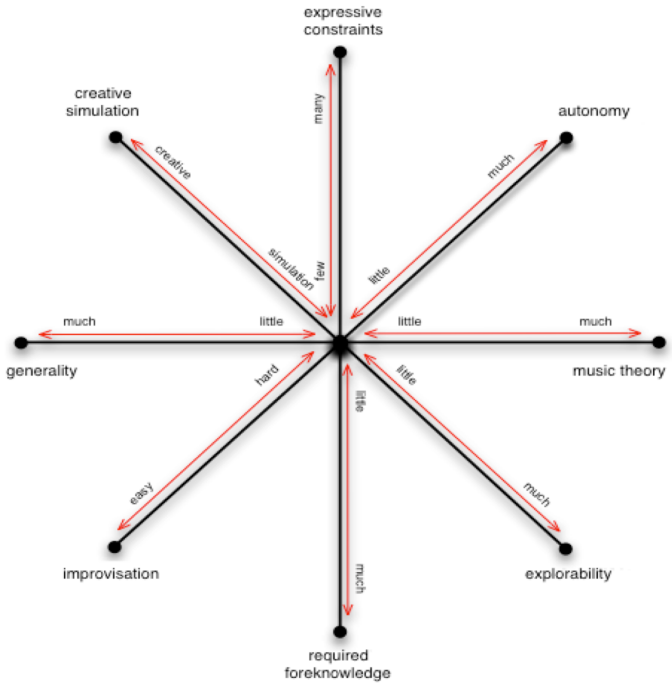


Figure 5: *An Epistemic Dimension Space for Musical Devices*

[196] Magnusson, T. 2010b. An Epistemic Space for Musical Devices. *Proceedings of NIME' 2010*: 44

Magnusson (2010b) describes the constraints illustrated in *Figure 5* as 'parameters that are unique to heavily abstract, conceptualized and symbolically designed musical tools'[197]. Such a supposition again reflects the relative complexity of interactive music systems in comparison to standardised musical instruments, and the necessity to implement high-level constraints in the design process for *Genesis*.

Yet, with the implementation of high-level constraints, the greater the variety between each interactive system's interactive method. Consequently, this results in difficulties in the pedagogical approach to performing with such systems and increases the limit on the number of users, outside of the designer, to successfully perform with the system. Indeed, significant attempts have been made to offer a 'low entry fee' (Moreover, Wessel and Wright, 2002) in systems that are formed of high-level constraints. For example, ixi lang (Magnusson, 2010a), which centres on creating expressive constraints through a 'musical live coding programming language that frees performers from having to think at the level of computer science'[198], still requires substantial commitment from the user to understand an overview of the system's constraints, severely limiting its accessibility and applicability to interactive methods outside of itself.

The fundamental method of interfacing with an interactive music system such as ixi lang (Magnusson, 2010a), the process of musical live coding, is an increasingly popular approach to interacting with interactive music systems; software environments such as SuperCollider, implemented in *Genesis*, allow the user to define through computer code, executed in real-time, a wide range of musical phenomena, such as sound-objects that are generated in real-time through a selected synthesis method, visual projections of such sonifications, and unique GUI interfaces for control of an ongoing musical process.

Therefore, musical live coding practices allow the user to create interfaces in real-time for musical performance through the abstraction of computer programming code. Such an interaction process is perhaps the absolute antithesis to an instrumental

[197] Magnusson, T. 2010b. An Epistemic Space for Musical Devices. *Proceedings of NIME' 2010*: 43 - 46
[198] Magnusson, T. 2010a. Designing Constraints: Composing and Performing with Digital Music Systems. *Computer Music Journal*. 34(4): 69

paradigm for interaction with interactive music systems; 'yet we do not wish to be restricted by existing instrumental practice, but to make a true computer music that exalts the position of the programming language, that exults in the act of programming as an expressive force for music closer to the potential of the machine'[199]. So, it is considered that musical live coding affords a form of expression that is fundamentally relative to the mechanisms of the machine, without the necessity for musical descriptors, and instead, within the syntax of the programming language.

Indeed, with the implementation of musical live coding as an interaction method, constraints and mapping categorizations can be explored in real-time, during the performance process, permitting the user to generate interactive music systems of their design as part of the compositional process. Such a process therefore encapsulates the essence of Magnusson's (2010a) supposition regarding the necessity for truly virtuosic performance with interactive music systems to require the performer to also be the designer.

Yet, the accessibility of systems constructed through musical live coding for others outside of the designer requires an understanding of the significant abstraction from commonly applied musical terminology, and the constraints through which the abstraction is implemented, posing a serious detraction for users without such computer programming knowledge, or those requiring musical metaphor and analogy for the descriptions of affordances, constraints and mappings in an interactive music system.

Moreover, live coders such as *slub* (Alex McLean and Adrian Ward), a duo who write their own software languages for live coding 'control music using user interfaces created by and for themselves'[200], resulting in systems which are absolutely not designed for use with or by others. Of course, it is the prerogative of the designer who is to use (or not) their system but it is necessary to demonstrate that interactive music systems have been constructed without the acceptance of other users, other than designers themselves. Such an approach may indeed prove advantageous, as the system can be formed representing explicitly the designer's own perceptions and

[199] Collins et al. 2003. Live coding in laptop performance. *Organised Sound* 8(3): 322
[200] Ibid: 323

characterisations of sonic features, thereby furthering the degree of possible virtuosity shown through a performer and their interactive music system.

However, in an instance whereby users other than the designer are to perform with an interactive music system, 'newcomers are very cautious when exploring a new instrument: the first gestures allow them to 'get an idea', to make a mental map'[201]. So, if a gesture cannot be communicated by the user through such an abstraction as computer code, then such an interaction method as live coding is not only absent of a low entry fee (Moreover, Wessel and Wright, 2002) but also a commonly understood paradigm through which to exchange interactions.

Considering Overholt's (2009) key questions regarding the relationship of gesture to the outputs of interactive music systems, through a live coding methodology, it is proposed that in reference to the intuitiveness of a gesture, this is directly relatable to the knowledge and understanding of the user of computer programming for musical composition and the software applied. For the perceptibility of a gesture by an audience, unless significant attempts are made to project the interfaces of a live coder's system, it is *not* possible to make links between the user's inputs and the outputs of the system as the audience are simply witnessing a performer typing on a laptop.

Moreover, if a projection of the coding is used, understanding by the audience of the computer programming code is a necessity, as the musical abstraction into code bears few musical descriptors from which to make such links. Indeed, the physicality of the gesture in live coding methods remains constant, and minimal; the user can only communicate gesture through the computer keyboard and mouse. Therefore, the physical effort does not match the generation of a complex, evolving and dynamic sound-objects made possible through live coding.

Due to live coding's limited accessibility to instrumentalists, combined with its restricted gestural capabilities that further hinder its approachability to instrumentalists, *Genesis* extends the instrumental paradigm with the *option* of using

[201] Arfib et al. 2003. Strategies of mapping between gesture data and synthesis model parameters using perceptual spaces. *Organised Sound* 7: 141

live coding techniques for those users who wish to generate novel sound-objects to pass through the *Genesis* system. As a result, the fundamental interactive method of *Genesis* is approached through conventional musical values in combination with perceptual spaces such as psychoacoustic data with acute modification and toggling of its various generative and analytical processes through a familiar and manageable graphical user interface space. Through such an approach, it is proposed *Genesis* can implement high-level constraints, alongside an accessible method of musical narrative that offers reasoned responses to the interactions between the user and the system, which increases the user's understanding of the interactive methods, thereby enabling potential virtuosity.

Other systems which apply similar fundamental principles of sound-object control for selected generative processes, as found in *Genesis*, follow similar interactive approaches. The imitative synthesis method (Grey, 1975; Wessel, 1979; Beauchamp, 1982) establishes such an extension of an instrumental paradigm through the reinterpretation of the perceptual spaces of harmonic instruments via musical and psychoacoustic descriptors; a 'musical excerpt is first analysed and then represented according to perceptual and signal features, keeping a description of links between the kinds of features. Then, we can move into the perceptual spaces representing the sound and use gestures to synthesise the sound from perceptual features'[202]. With such an implementation explicit and implicit mappings can be applied, through direct, linear links between the perceptual spaces and their associated synthesis values, in combination with generative algorithms such as artificial neural networks for adaption of these mappings, which allows the user to 'warp, change, invent an instruments from another one'[203]. Therefore, such a system affords considerable creativity and exploration of timbral parameters through an extended instrumental paradigm.

Adaptive digital audio effects (Verfaille and Arfib, 2001) present a variation in the principle of the imitative synthesis method; features are extracted from sound-objects for consequent mapping to selected parameters of chosen audio effects such as pitch shifters, phase vocoders, filters and time stretchers. Therefore, many low-level sonic

[202] Arfib et al. 2003. Strategies of mapping between gesture data and synthesis model parameters using perceptual spaces. *Organised Sound* 7: 133
[203] Ibid

features such as 'the RMS energy, the spectrum centroid, the fundamental frequency, and the voiced/unvoiced status'[204] can be isolated, relative to the applied machine listening algorithms.

The outputs of the machine listeners, and the musical gestures they identify, can then be explicitly mapped to their audio effect counterpart for macro-level manipulation, such as the fundamental frequency to a pitch shifter's *fundamental pitch* parameter assigned to the overall pitch of the response, or to the micro-level, such as a timbral modification achieved through adjustment of a granulation process's density relative to the perceived spectral density of a signal. Indeed, such mappings may also be implicitly linked through generative processes such as an artificial neural network, enabling considerable adaptivity when engaging with such a system.

Furthermore, methods such as concatenative synthesis (Schwarz, 2006; Casey, 2004; Lazier and Cook, 2003; Momeni and Mandel, 2005) use feature extraction from auditory sources (a *target*) to identify sonic features that match the sonic features of sound-objects represented within a *database*, with 'the best match' used as the output. Such an approach aims to remove the need to manipulate a resulting sound-object through external digital signal processing such as filters, pitch shifters and time stretchers. However, the concatenative synthesis approach still requires extensive mapping of identified sonic features in the *target* relative to prescribed descriptors of the sonic features within the *database*. As a result, congruency between the sonic characteristics and gestures within both the *target* and *database* is paramount, necessitating extensive consideration of the perceptual and musical spaces through which to control the 'best match' algorithms.

Therefore, when auditory sources are used as sound-objects to control algorithmic processes, as in the case of *Genesis*, such as a live instrumentalist, in conjunction with an interactive music system, the intuitiveness of the system is thereby relative to the applied perceptual spaces of the auditory source and their consequent mappings and constraints. Furthermore, a well-founded mapping will increase the perceptibility for the audience of the relationship between the gestures and the system's responses, with

[204] Arfib et al. 2003. Strategies of mapping between gesture data and synthesis model parameters using perceptual spaces. *Organised Sound* 7: 139

such a logical mapping also being relative to the physicality of the gesture, resulting in an increase in the perceived virtuosity in the performance by the audience and is thus implemented in *Genesis*. In chapter 6 *Evaluation of the Genesis System*, the methodology is tested relative to its ability to achieve successful interaction with real-time sound-objects and its capability to form a unified real-time creative process between human and machine.

4.2 Composition with Real-time Interactive Music Systems

The *Genesis* system is designed to be applied in real-time. Therefore is it is necessary to consider the compositional methods that can used in real-time, which can be categorised into score-driven and performance-driven (Rowe, 1993). A score-driven system has 'embedded knowledge of the overall predefined compositional structure. A performer's progress through the composition can be tracked by the system in real-time, accommodating subtle performance variations such as a variation in tempo'[205] applied by composers such as Manoury, Boulez, Lippe and Settle (Cont, 2011). In contrast, a performance-driven system has 'no preconstructed knowledge of the compositional structure or score and can only respond based on the analysis of what the system hears'[206] as demonstrated in Lewis' *Voyager* (Lewis, 2000).

So, in the case of an interactive system that is score-driven, a compositional structure is provided before musical performance, through which the system is able to monitor the user's interactions relative to the score. Consequently, the system's outputs can be generated relative to the predefined compositional structures present in the score and the generative processes applied to create its responses. Conversely, a performance-driven system is unaware of a predefined compositional structure, only using the data provided in real-time from the interaction device to form its responses, which are generated by the applied generative algorithms in the system.

In terms of the products of such interactive methods, score-driven systems, due to their reliance on predefined compositional structures that are known prior to

[205] Drummond, J. 2009. Understanding Interactive Systems. *Organised Sound.* 14: 127
[206] Ibid: 128

performance, are often highly predictable in their development over time. Furthermore, score-driven systems are 'typically programmed to follow the performer faithfully'[207], as the interactions of the user must be formed of symbolic data, which is directly relatable to the musical representations in the score. Thus, the outputs generated by the system are often similar to the interactions of the user.

On the other hand, performance-driven systems are proposed to be improvisatory by default in their interactions, as their outcomes are not relative to a predefined compositional structure, thereby forming a significantly more unpredictable response to the user relative to the data it is provided with and the level of unpredictability applied in its generative algorithms. In addition, due to the absence of a score, the use of symbolic data to communicate interactions is not a necessity. Therefore, subsymbolic methods of representation are more applicable in such a method, offering users an increased level of gestural interaction with the system.

However, it has been proposed that indeed both score-driven and performance driven compositional methods can be combined to form virtual scores (Manoury, 1990); 'a virtual score is a musical organisation in which we know the nature of the parameters that will be processed but not their exact outcome at runtime since they're expressed as a function of live performance'[208]. Therefore, a virtual score (Manoury, 1990) 'consists of electronic programs with fixed or relative values/outcomes to an outside environment'[209]. Consequently, the real-time interactions of a system, and the real-time generative processes that define the responses of the real-time interactions, exist within musical time and must form the musical score, thus generating the resulting composition. Indeed, *Genesis* is implemented to accommodate such a compositional method through its use of generative and analytical algorithms, which respond in real-time to the musical and psychoacoustic values provided in real-time sound-objects.

Though, a distinction must be made regarding the perceived differences between *composition* and *improvisation*, as the real-time generation of musical material during performance is often considered *improvisation*. Francois (2006) states 'to give a

[207] Drummond, J. 2009. Understanding Interactive Systems. *Organised Sound*. 14: 127
[208] Cont, A. 2011. On the Creative Use of Score Following and its Impact on Research. *published in SMC 2011: 8th Sound and Music Computing Conference 2011*: 2
[209] Ibid

definition of the term 'improvisation' is a perilous matter. The three definitions most often mentioned are not able to catch the complexity of the question: a) a musical practice without notation; b) an oral practice of direct communication, in an immediate manner, without any intermediary; c) a spontaneous expression of liberated musicians'[210]. So, this would imply that composition is the contrary, a process in which music is practiced *with* notation, *with* intermediaries and *without* spontaneity.

However, considering the compositional process, as defined in chapter 2.1 *Algorithms in the Compositional Process*, the compositional process itself includes the requirement of spontaneity for the incubation of ideas. It is also not intrinsically reliant on determinant intermediaries as illustrated by the application of stochastic and non-linear processes described in chapter 3.1 *An Introduction to Real-time Generative Algorithmic Systems*. In addition, should it really be considered that if a musical phrase is not notated, then it is by definition *not* a composition, and by default, it is an improvisation? Surely if this statement were true, as demonstrated by the absence of definitively representing and consequently notating a sound-object's sonic features and defining their musical values in chapter 3.2 *A Brief Summary of Machine Listening*, *all* compositions have improvisational components by their very nature, and as a result could be described as an *improvisation*.

Therefore, it is still necessary to define what improvisation *is*. Francois (2006) suggests that 'the art of improvisation seems to be centered on a) the ability to free oneself of the strictness of the framework or gestural technique, in order to concentrate on the globality of what is occurring in the moment; b) the ability to invent along the way of the performance new sound combinations; and c) the ability to concentrate on the present instant without having to plan ahead the musical form in a self-conscious way'[211]. This is to say that *improvisation* is music, which is *not* formed in relation to the temporal macrostructure of a *composition*, but of the present, of the *now*, of a 'novel circumstance'[212], relative and accepting of the macrostructure from which it is contextualized by, but not prescriptive of it.

---

[210] Francois, JC. 2006. Improvisation Today, Between Orality and Writing. *Contemporary Music Review* 25(5/6): 624
[211] Ibid
[212] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49

So, it must be concluded that for the purposes of this thesis, *composition* and *improvisation* are not mutually exclusive. That composition is inherent in improvisation, and that improvisation is inherent in composition. Within digital music systems, the number of improvisational techniques available to the compositional process and the number of compositional techniques available to the improvisational process is relative to the compositional and improvisational abilities of the human and the computer respectively. Therefore, these abilities are reliant on the generative and analytical processes of the two parties. The model of interaction dictates the method with which these processes can be communicated, and as a result, the level of influence of either party on the improvisational and compositional techniques.

Considering the proposed compositional and improvisational processes, and the models of interaction with digital music systems, the realization of composition in real-time is a possibility. Risset (1999) concludes however 'Composition is not – or should not be – a real-time process. Musical notation applies time over space. It refers the reality of the music to a representation – the score – which is out of time. This representation suggested transformations that could not be conceived or performed in real-time – such as symmetries with respect to the pitch or the time axis used in counterpoint. Non real-time operation is necessary to free oneself of the arrow of time and its tyranny, of the dictates of haste, instance, habits, reflexes. Writing music implies prediction and elaboration. The construction of the piece may take a lot of patience: but one should also be able to conceive it in a synoptic way, at a glance much faster than the flow of musical time'[213].

In contention of Risset's (1999) deductions, there is an absence of a conclusive representation of sound-objects and their sonic features for composition that are *not* constructed as part of a real-time process. Therefore, compositions that are notated, that can exist 'out of time'[214], do not truly represent the temporal changes of a sound object's sonic features. As a result, in the realization of compositions that exist 'out of time'[215], the performer is required to interpret and represent the sonic features absent

[213] Risset, JC. 1999. Composing in Real-time?, *Contemporary Music Review* 19(3): 37
[214] Ibid
[215] Ibid

from the musical score, *resolving* the compositional process in real-time, thereby incorporating the 'arrow of time'[216] in the performance process of a notated composition. Thus, it is possible to conclude that the realization and completion of a notated compositional process itself is reliant on real-time processes through its performance.

Further to the notion of real-time processes being part of the realization of compositional processes, indeterminate compositional techniques influence the flow of musical time and the sound-objects that occur within a composition's duration; the events within an indeterminate composition do not occur 'out of time'[217]. That is to say, the events of such a compositional method are exclusive of the composition's notated musical time, but are still accepted as circumstantial events that are resultant of the compositional process, thereby forming the compositional material in real-time.

In addition, the capability of digital music systems to modify, in real-time, the time-scales with which sound-objects can be played back through time-stretching and rearrangement, negates the requirement for 'haste, instance, habits, reflexes'[218]; musical time becomes *elastic* within digital music systems, allowing composers to sustain or hasten musical events at their control. As a result, the idea that 'writing music implies prediction and elaboration'[219] is perhaps unfounded; the capability to modify the temporal morphology of sound-objects through real-time interaction still allows inevitable and predictable events to be defined *without* notation whilst the 'novel circumstances' that occur from applied indeterminate techniques permits the elaboration of such inevitable and predicable events.

The placement of music 'out of time'[220] also assumes a finite compositional process; the resultant composition is expected to exist as a fixed entity. The use of generative techniques in interactive digital music systems 'is best appreciated when studied closely, when run many times, and that true appreciation can place you in the role of understanding everything the composer created'[221]. So, the 'novel circumstance'[222]

---

[216] Risset, JC. 1999. Composing in Real-time?, *Contemporary Music Review* 19(3): 37
[217] Ibid
[218] Ibid
[219] Ibid
[220] Ibid
[221] Collins. N. 2003. Generative Music and Laptop Performance. *Contemporary Music Review* 22(4): 71

inherent in many of the generative techniques indicates that such a finite description of such compositional methods is not satisfactory and that indeed, real-time composition represents the compositional processes of the *now*. Therefore the understanding of a composer's use of real-time compositional processes is bound to the comparison of many performances, relative to the overall compositional goal/s of the composer.

It is possible to conclude then, that composition *can* be a real-time process, challenging Risset's (1999) suppositions. The utility of interactive digital music systems demonstrates the dynamic nature of the relationship between composition and performance; through interactive music systems and the real-time processes they can generate and analyse, the composer *becomes* the performer. So, a real-time composition is indeed a real-time performance, the result of which is a composition of the *now*; 'the composer becomes at the same time the performer, while the performance, or realization, takes on a primary importance. The musician becomes a sort of painter: he acts directly on the quality of the realization'[223].

The acceptance of a real-time compositional technique is founded in a Constructivist approach to the analysis of music; this is to say that 'when analysing audio art and electronic music, technology, technique and musical style are to be taken in account'[224]. This approach is not accepted universally however as reflected in Risset's statement that 'our epoch is too keen on immediate satisfaction. Impatience favors hasty, blind, reflex reaction rather than documented and thoughtful action'[225]. Such a deduction implies that there is an imminent satisfaction through real-time compositional methods, and indeed composition itself.

It is perhaps fair to state that through interactive digital music systems that offer real-time interactive techniques, it is possible to *explore* musical ideas in real-time through methods such as genetic algorithms with the *possibility* of satisfaction achievable by such a process, but this does not guarantee the *imminence* of that satisfaction.

---

[222] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49
[223] Schaeffer, P. 1952. *A la recherche d'une musique concrete*. Paris: Seuil
[224] Battier, M. 2003. A Constructivist Approach to the Analysis of Electronic Music and Audio Art – between instruments and faktura. *Organised Sound* 8(3): 249
[225] Risset, JC. 1999. Composing in Real-time?, *Contemporary Music Review* 19(3): 37

Therefore, a real-time environment must proffer predominantly satisfying sonic outcomes, warranting the application of a real-time compositional process and the acceptance of process by a composer as inclusive to the success of a compositional product.

Chapter 5

The *Genesis* System

5.1 An Overview of the *Genesis* System

The *Genesis* system is a standalone application for Mac OS X 10.6 or higher, written in the object-oriented programming language of SuperCollider version 3.5.3 included on the accompanying DVD in the *Genesis* folder (along with the source code) and is demonstrated in live performance scenarios in the folder *Genesis Performances*.

Genesis is comprised of a series of *SynthDefs*, stored on the SuperCollider Server, which perform specific interactive, generative and analytical algorithmic functions. The *SynthDefs* are themselves formed of UGens, which dictate the parametric values that may be modified by the interactive, generative and analytical algorithmic processes. The UGens define either auditory signals or control signals, which can be sent and received internally within the respective *SynthDef* or routed via *Busses* to the SuperCollider Client.

Within the SuperCollider Client, the auditory signals are routed between the *SynthDefs*, while the control signals are wrapped as OSC messages, which are collected and modified by real-time tasks, routines and human-defined interactions executed on a local or networked SuperCollider Client. The modified control signals are then sent back and received by the respective *SynthDef* to alter its designated UGen parameters in real-time. In addition, much of the bussed control signal data is abstracted and represented within the extensive graphical user interface, thereby visually representing many of the interactive, generative and analytical processes taking place in real-time within the *Genesis* system.

The fundamental principle of the *Genesis* system is to apply the sonic features of real-time audio signals for modification, manipulation and arrangement of real-time sound-objects. The real-time audio signals can be live acoustic signals generated by instrumentalists or any other source, a pre-defined ‘Sample’ reader comprised of

UGens reading buffered audio or live-coded *SynthDefs* defining specific synthesized sound-objects with their respective modulatable parameters. These real-time audio signals can be placed into one of three auditory input sources within *Genesis*, each of which features controls within the GUI such as amplitude and pitch adjustment, relative to the formatting of the input source.

The three inputs each have specific sonic features extracted such as onset, pitch, loudness and pseudo-timbral data, which are used to represent the sonic characteristics of their respective sound-object. The purpose of each of the three inputs is to form three *control* sources of which one is also a *slave* source; the sonic features of the *control* sources dictate or influence selected interactive, generative and analytical processes with the *slave* input source forming the sound-object that is to be modified, manipulated and arranged by the interactive, generative and analytical processes.

In addition to the sonic features of the real-time audio signal input sources dictating or influencing selected interactive, generative and analytical processes, the *Genesis* system features an extensive graphical user interface for the control of many of the parameter settings and inclusion of particular interactive, generative and analytical processes through the computer keyboard, mouse and optional MIDI functionality. The parameter modifications that have been made for *any* interaction within the graphical user interface are scored in real-time as live code, and stored as their respective computer code, permitting the consequent recalling and repetition of a particular interaction during the composition process or for use in other compositional tasks.

In terms of the application of the real-time input audio signals, the GUI offers control for live sampling of these inputs for consequent placement within the Sample UGens of each *control* source, permitting the modification through the GUI of the newly created recordings by the parameters of the Sample UGens. *Genesis* also features a post window that offers a composer the opportunity to use live coding to generate *SynthDefs*, modify the parameters of an instance of *Genesis* and display the current values of the parameters controlled by many of the GUI objects.

Furthermore, MIDI has been implemented for basic interface control of selected arbitrary parameter changes such as the overall amplitudes of the sound-objects. In combination with the capability to control many of the interactive, generative and analytical processes, the graphical user interface also provides a visualisation of the various processes taking place within the *Genesis* system in real-time. Moreover, a dynamic scoring method has been implemented to abstract and represent the results for many of the interactive, generative and analytical processes modifying the *slave* source. The dynamic scoring method is intended to visually represent the current state of the stereo sound space generated by the instance of *Genesis*.

Figure 6 illustrates the architecture of the *Genesis* system's interactive, generative and analytical processes. It is important to note that *all* of the interaction is colour-coded within the graphical user interface of *Genesis*, with yellow being *control* source 1, red being *control* source 2, and blue being both *control* source 3 and the *slave* source. This aids the user to promptly identify *which* real-time input source they are adjusting and is reflected through all diagrams of *Genesis* in this thesis.

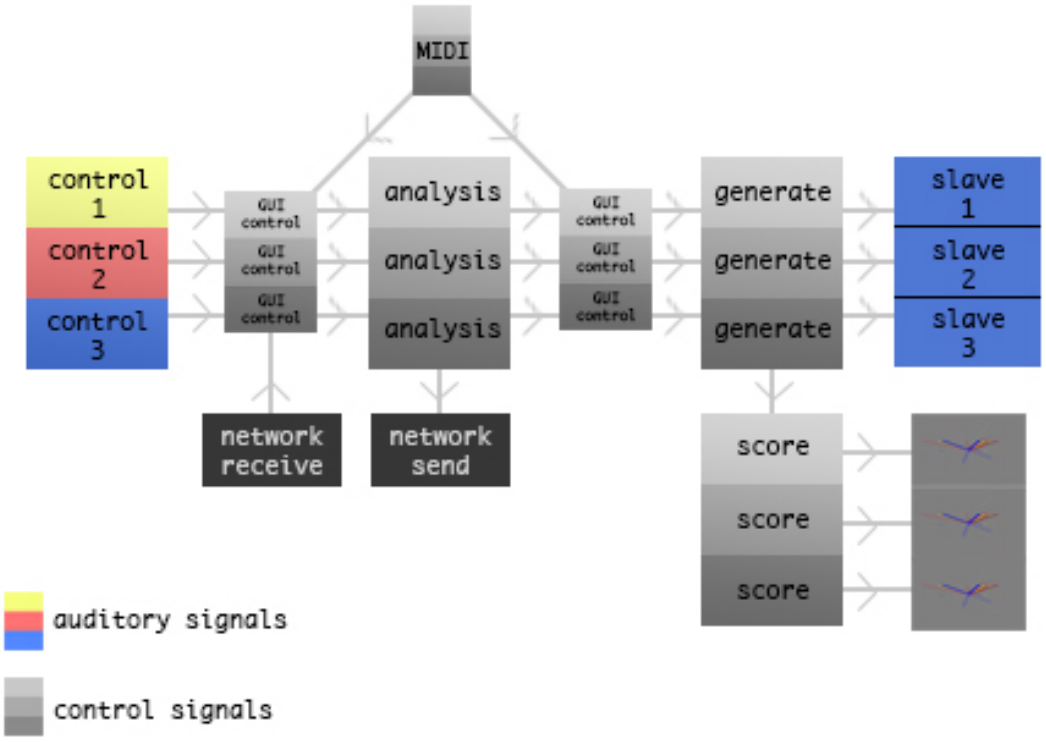


Figure 6. Genesis Architecture

In terms of the representation of sound-objects' sonic features in *Genesis*, the real-time audio signal's sonic features for each input are represented equally; each of the inputs, irrespective of their input source type and source purpose, have the *same* analytical processes applied at the point of input. The onsets, MFCCs, pitch, loudness and tempo are extracted from each input source for consequent application to the analytical and generative processes of *Genesis*, in addition to their visual representation within the GUI and the dynamic scoring system.

In order for many of the *generative* processes to be applied to the *slave* source, the *slave* is recorded to a series of buffers in real-time and consequently played back through a series of granular synthesisers. The granular synthesisers applied to the buffered *slave* source feature trigger, buffer position, playback rate, pan, amplitude envelope and grain length parameters, which are controlled by the *generative* processes, dictated by the real-time audio signal input sources and the graphical user interface.

Therefore, the *slave* source's resulting output is defined by the auditory input sources' and graphical user interface modifications, with many of the *generative* processes that are applied to the *slave* input abstracted and visually represented in the dynamic scoring system. In addition, the *slave* sound-object is also recorded to a single audio buffer *prior* to its recording for the granular synthesiser buffers permitting the modification of the *slave* sound-object's pitch, tempo, envelope and playback position relative (or not) to sonic features extracted from the *control* sources and values defined within the GUI.

In summary, the *interactive* processes are divided between real-time audio signal interactions and graphical user interface interactions; the real-time audio signals of the real-time audio signal input sources can modify the pitch, onset, spectral shape, amplitude envelope and tempo of the *slave* input source, with the graphical user interface controlling many of the *Genesis* system's parameters such as the envelope times of the *slave* input's granular synthesisers, the amplitudes of all of the real-time input sources within the auditory output mix and the execution of many of the

generative processes. All *interactive* processes are detailed fully in section 5.3 *Interactive Processes in Genesis*.

The *generative* processes that can be applied to the *slave* input feature a modified genetic algorithm, fractal noise, Markov chains and random search, as detailed in chapter 3.1 *An Introduction to Real-time Generative Algorithmic Systems*. The modified genetic algorithm is used to explore novel settings of particular parameters of the *slave* input's granular synthesizers, fractal noise defines the buffer position, playback rate and duration of the *slave* input's granular synthesizers' parameters, 0^{th}-order Markov chains control selection of random arrays created relative to the current state of selected variables for the generation of instant parameter settings for *control* sources one and two, and random search dictates the pitch, tempo, duration, onset and arrangement of particular interactions such as the buffer position of the *slave* sound-object prior to its recording for the granular synthesizers. In addition, live coding practices can be applied to generate parameter modifications and *SynthDefs* for use through the *Genesis* system. All *generative* processes are detailed fully in section 5.4 *Generative Processes in Genesis*.

The *analytical* processes within *Genesis* are applied through the Fast Fourier Transform to represent each real-time input source in the frequency domain permitting the extraction of their pitch, onsets, loudness, and tempos. In addition, Mel-frequency cepstral coefficients (MFCCs) are applied for the representation of pseudo-timbral features for each of the real-time input sources. It is important to note that due to the challenges of representing perceptual processes within computational analytical algorithms, as described in chapter 3.2 *A Brief Summary of Machine Listening*, the system's representation of such perceptual features is highly reflective of the applied analytical UGens. All *analytical* processes and their influence on the resulting compositional process are detailed fully in section 5.5 *Analytical Processes in Genesis*.

Considering the application of real-time auditory sources within *Genesis*, it is possible to apply both determinate and indeterminate processes within their auditory signals for the control of the *slave* sound-object; the conditional structure of the real-time

input sources are reflected in the system by its method of *reaction* to the sonic features that are present. That is to say that at the point of analysis, the *Genesis* system represents the selected sonic features without explicit prejudice towards a particular conditional structure, with the ultimate control of whether a particular sonic feature is to be applied to a interactive, generative or analytical process dictated through the graphical user interface.

Through the combination of the real-time interactive, generative and analytical processes within the *SynthDefs* and the consequent modification of their parameters by the real-time tasks, routines and human-defined interactions, an instance of *Genesis* is capable of operating within the four different models of interaction as proposed by Winkler (2001), as described in chapter 4.1 *Interaction with Creative Systems*. For example, a Conductor Model (Winkler, 2001) may be applied through the use of an instrumentalist's sonic features as a *control* source, defining *all* resulting amplitude, pitch, onset, temporal and pseudo-timbral features of a *slave* source relative to the sonic features of the *control* sources. In contrast, a Free Improvisation Model (Winkler, 2001) may be applied through the use of the *Genesis*'s 'Call and Response' function, which records a *control* source's audio to an audio buffer, and generates a response by analysing prescribed sonic features of the *control* source to define a formalist response, which is constructed through modification of the audio recording's pitch and temporal features. All methods of interaction are detailed fully in section 5.3 *Interactive Processes in Genesis*.

As a result, *Genesis* forms a real-time composition system, offering the composer the option of implementing different methods of local interaction on-the-fly, as well as various interactive, generative and analytical processes to define the compositional processes in real-time. Furthermore, the interactive, generative and analytical processes can be communicated via a computer network using the Internet Protocol address (IP) of the desired computer running an instance of *Genesis* to form a networked global model of interaction, primarily based on the Chamber Music Model (Winkler, 2001) through which one instance of *Genesis* acts as the "leader" of a compositional process, sending selected local control data to external instances of *Genesis*. The control data from the "leader" may then be then applied to dictate

selected local interactive, generative and analytical processes on the external instances, which allows the external instances to control particular sonic features of its auditory output, reflecting the nature of the Chamber Music Model's (Winkler, 2001) interplay between performers.

5.2 A Quick Start Guide to *Genesis*

*Important notes ***BEFORE*** starting *Genesis**

1. Create a folder within your computer user's Music folder named 'SuperCollider Recordings' for live sampling functionality. The folder directory listing is as follows:

/Users/*your computer's username*/Music/SuperCollider Recordings

2. Ensure any audio file applied to the Sample UGens is ***STEREO***, formed of two interleaved audio channels of either .wav or .aiff format. Mono files will result in buffer errors, and cause the system to crash.

3. Check that the sample rate of any analog inputs is the same as the sample rate of any analog outputs. The default audio analog in and out of computers running Mac OS X are both 44.1kHz, and are known to function correctly with *Genesis*

4. Ensure your computer is running Mac OS 10.6+. Systems below this are *not* compatible with *Genesis*

5. For optimum network functionality, it is recommended to use local networks to broadcast data between systems over ad hoc Ethernet cabling or ad hoc wireless networks. The method offered within Mac OS X in the 'Create Network…' option in the Airport Menu tab has been tested and offers suitable connection speeds for wireless broadcasting between systems.

6. Due to the requirement of around 350kps for *Genesis* to send its network data, this may inadvertently disrupt the network connection of any computers connected to the same network. Therefore, it is advised to ***DISCONNECT*** the computer from any network during use of *Genesis unless* network functionally is required.

7. Copy *Genesis*.dmg to your hard disk. It is not recommended to run the program from the DVD.

8. *All* MIDI devices must be connected and switched on *prior* to *Genesis* intialisation.

9. Audiovisual examples of each function within *Genesis*, along with their respective implementation method are detailed in section '*5.6 Genesis Methodology with Audiovisual Demonstrations'*

Audiovisual example *1. Quick Start Guide* in the 'Audiovisual Examples' Folder on the accompanying DVD demonstrates each step in the following quick start guide.

Step One: Setting GUI resolution and Performance options

After the loading screen has disappeared, you can select the scaling of the GUI objects to fit your computer's video resolution by typing in your computer's native resolution in the relative boxes.

In addition, if the system is showing a high peak CPU value (above 50%), click the "Performance Hi" button. Consequently, the button will show "Performance Lo". (This removes the Grain Freeze process, but significantly reduces CPU demand) shown in *Figure 7*:

Figure 7. Control of GUI scaling and performance

Step Two: Selection of input for sources and optional placement of audio files in Sample UGens

1. Select desired input sources between an audio file ("Sample"), analog input ("Mic") and *synthDef* ("Synth") through the input PopUpMenu objects shown in *Figure 8*.

* NOTE * Yellow object is *control* source one, red object is *control* source two and blue object is *control* source three/*slave.*

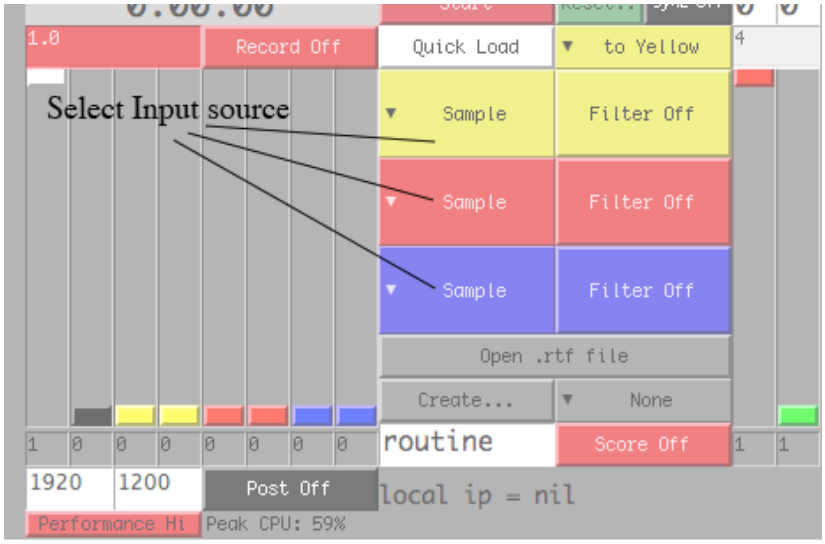


Figure 8. Input source selectors

2. If "Sample" is selected, use the "Add file…" button of the relative input source to open a path dialog for selection of a ***STEREO*** audio file from disk shown in *Figure 9*.

3. Use 'Click to Trigger…' to output the sound-object through auditory mix, if desired. When triggered ***on***, button changes to grey with text "Click to Trigger On" shown in *Figure 9*.

4. Adjust volume with Slider. Ensure trigger is ***on*** fi you wish to place the sound-object into auditory mix shown in *Figure 9*.

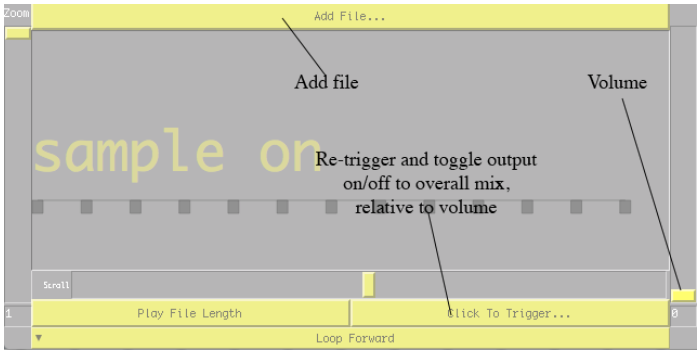


Figure 9. Add file, trigger and volume controls of Control source one

Step Three: Placing *slave* granular synthesizers into the auditory mix

1. Load desired *control* and *slave* sources as demonstrated in Step Two.

2. Ensure a *slave* sound-object is passed into the granular synthesizers by manually clicking the "Click to Trigger…" toggle button for the *slave* sound-object. The button will turn grey, containing the text "Click Trigger On". Leave ***on***.

3. Ensure a *control* source is triggering the granular synthesizers, represented in the GUI buttons below each *control* source's MFCC display shown in *Figure 10*.

Figure 10. Trigger buttons relative to onsets of control source one

4. Adjust 'Threshold' parameters of granular synthesizers to modify each trigger's threshold shown in *Figure 11*.

5. Adjust 'Amplitude' parameters of granular synthesizers controlled by chosen *control* source shown in *Figure 11*.

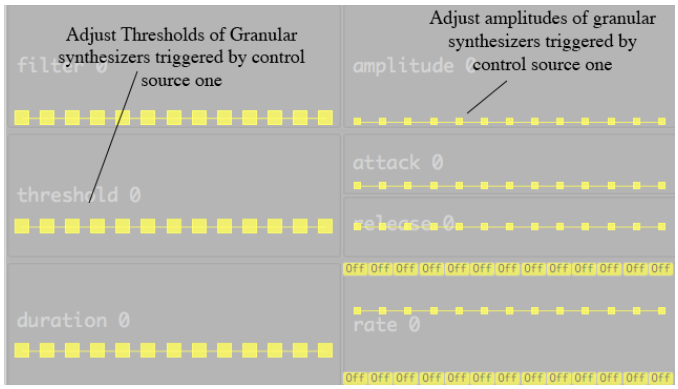


Figure 11. Adjustment of Threshold and Amplitude of granular synthesizers triggered by onsets of control source one

6. Adjust master volume of granular synthesizers, to place them in auditory mix shown in *Figure 12.*

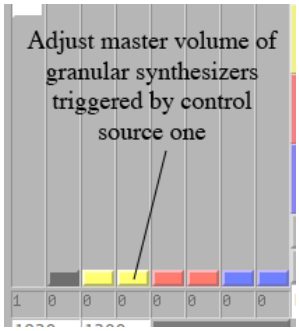


Figure 12. Slider for master volume of granular synthesizers triggered by onsets of control source one

An Overview of the GUI Objects within Genesis

The following figures detail the functionality of each GUI element within the *Genesis* user interface:

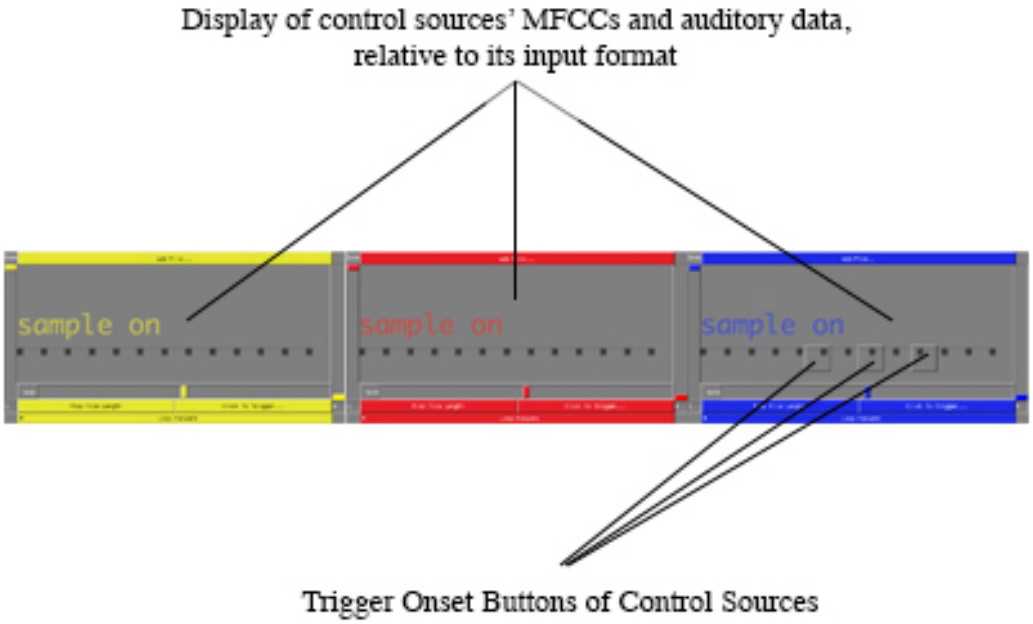


Figure 13. Input Source Display

- Display of current pseudo-timbral data, the current input source and visual representation of *control* source triggers

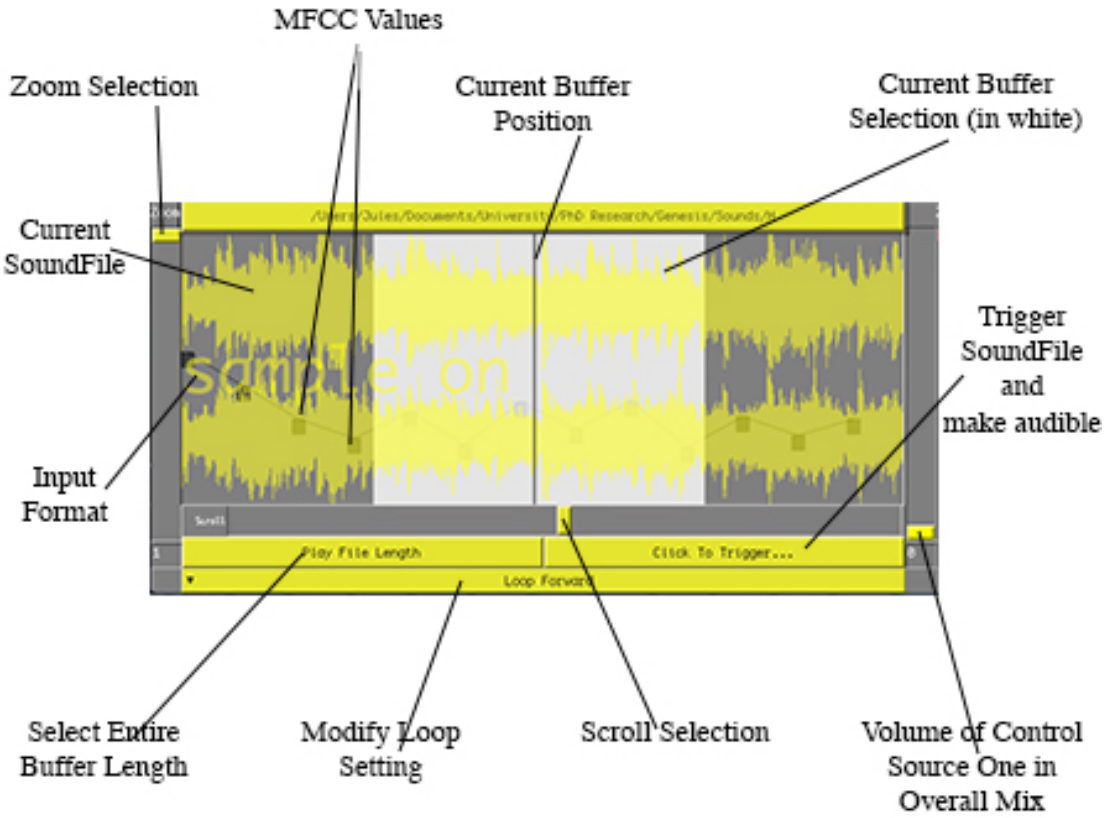


Figure 14. Control Source One Display and GUI controls

- Arbitrary controls for input source parameters (sample file loaded in *Figure 14*)
- Same for each *control* source

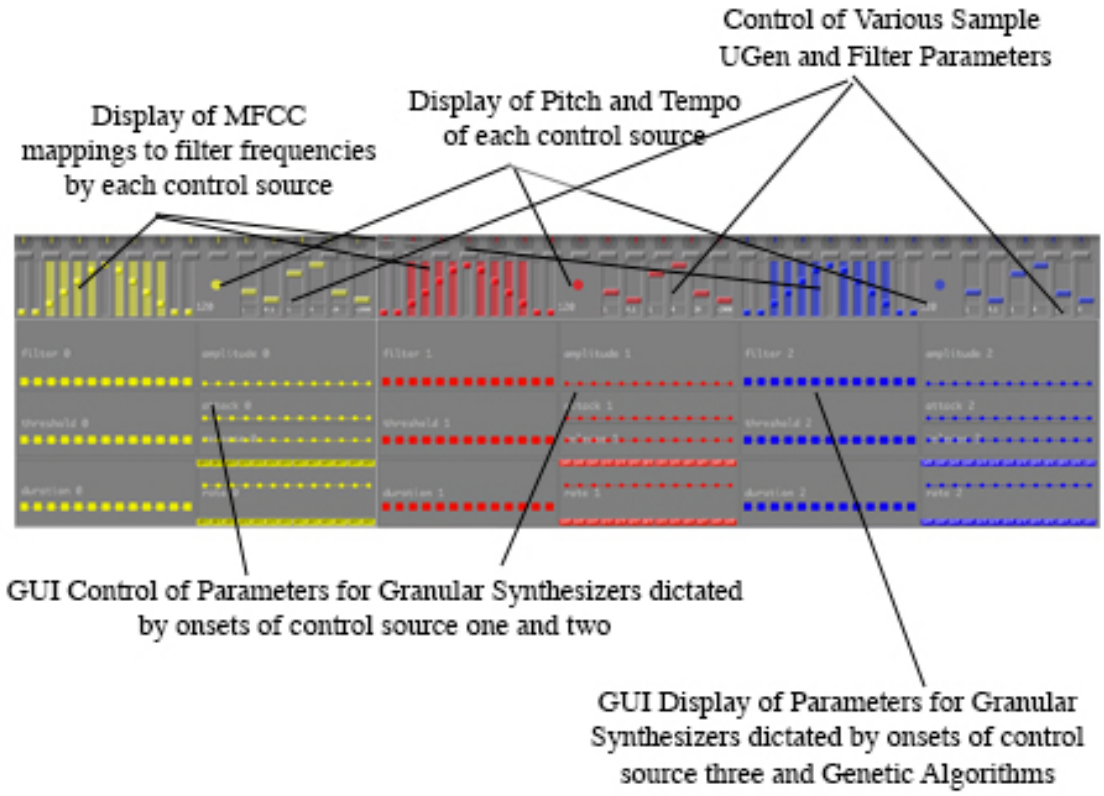


Figure 15. Granular Synthesizer display and GUI controls

- Parametric controls of the granular synthesisers
- Further modification sliders for the input sources

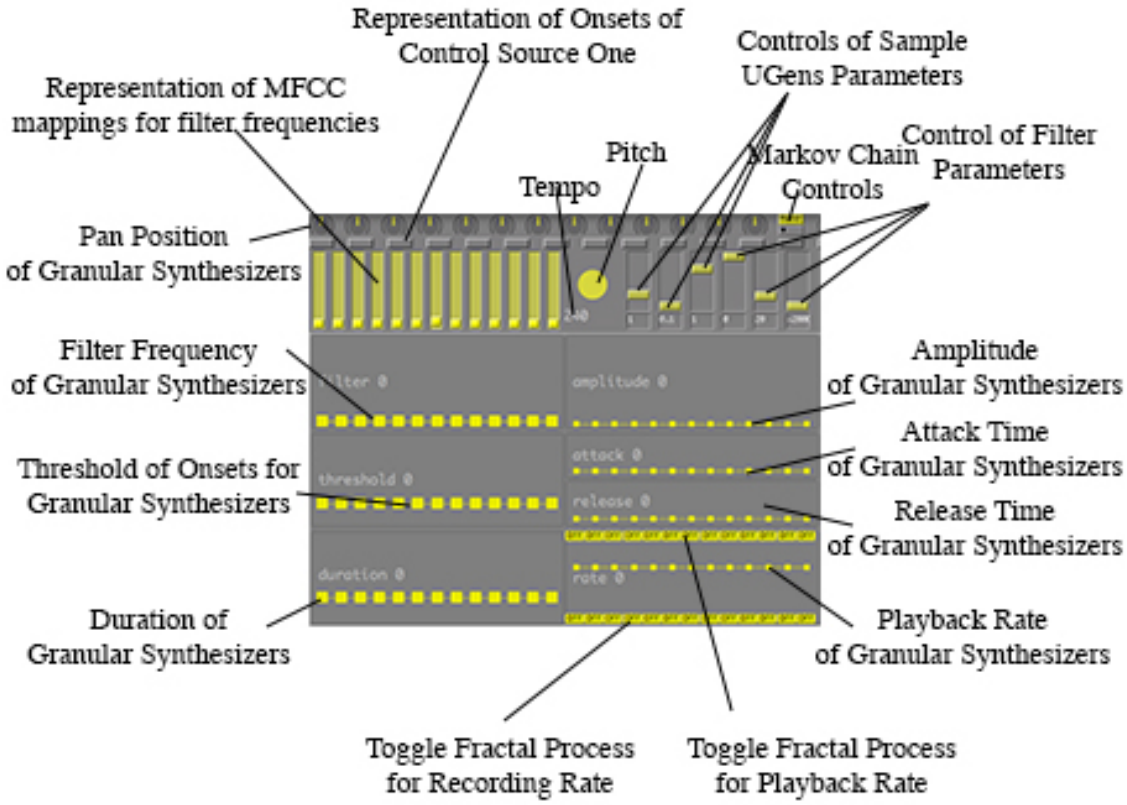


Figure 16. Further Control Source one display and GUI controls

- Parametric controls for the pan, filter, threshold, amplitude, duration, attack, release and rate of the granular synthesiser
- Toggle buttons for initiating fractal noise process on the playback and recording rates of the granular synthesisers
- Buttons representing current onsets triggered by the *control* source
- Display of spectral following current filter frequencies
- Visual display of perceived pitch and tempo
- Toggling and probability distribution control of Markov chain for generation of random arrays relative to current state of granular synthesiser parameter settings
- Parametric control of input sources pitch, time stretch and grain length (relative to input type)
- Modification of spectral following filters' bit rate, update speed and base frequency

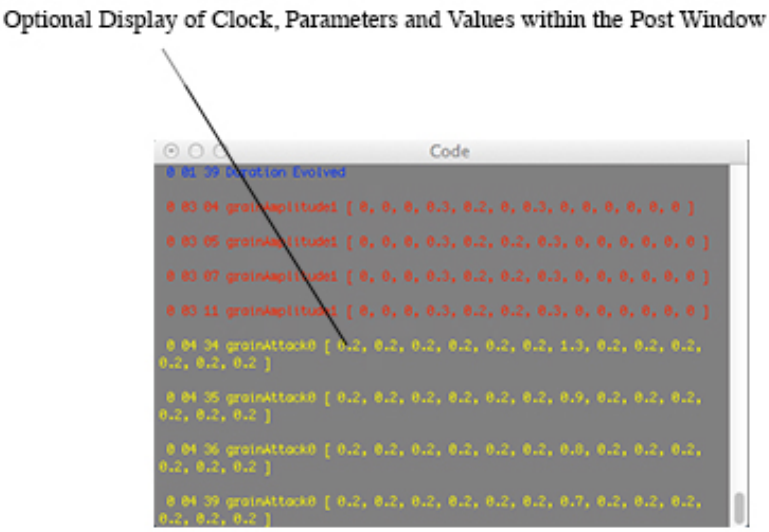


Figure 17. Example of Post Window output

- Optionally displays current numerical values of GUI objects. Toggle on/off for this functionality shown in *Figure 19*

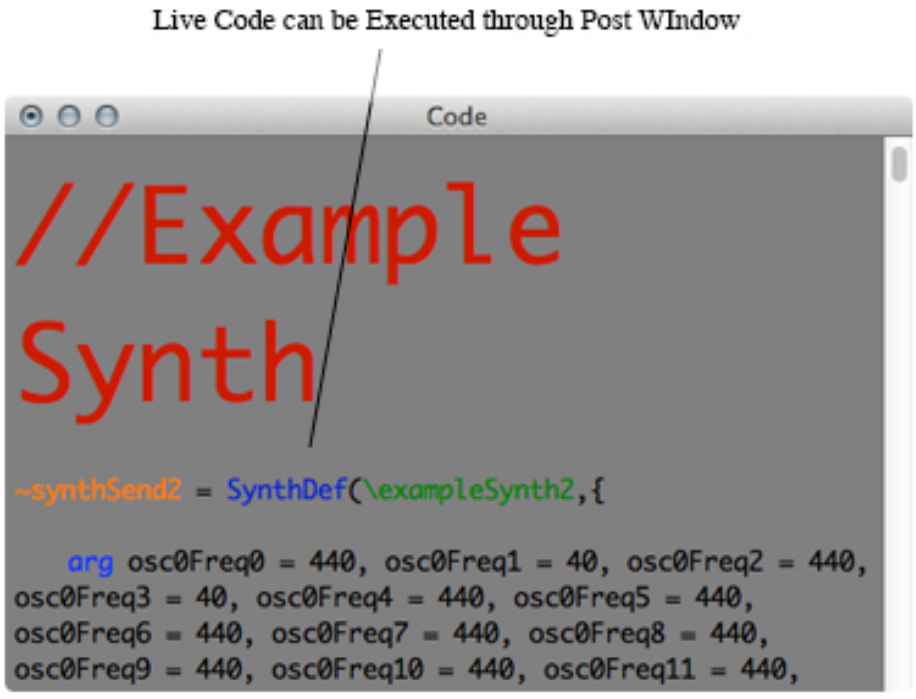


Figure 18. Example of Live Coding in Post Window

- Post window can be used to type live code

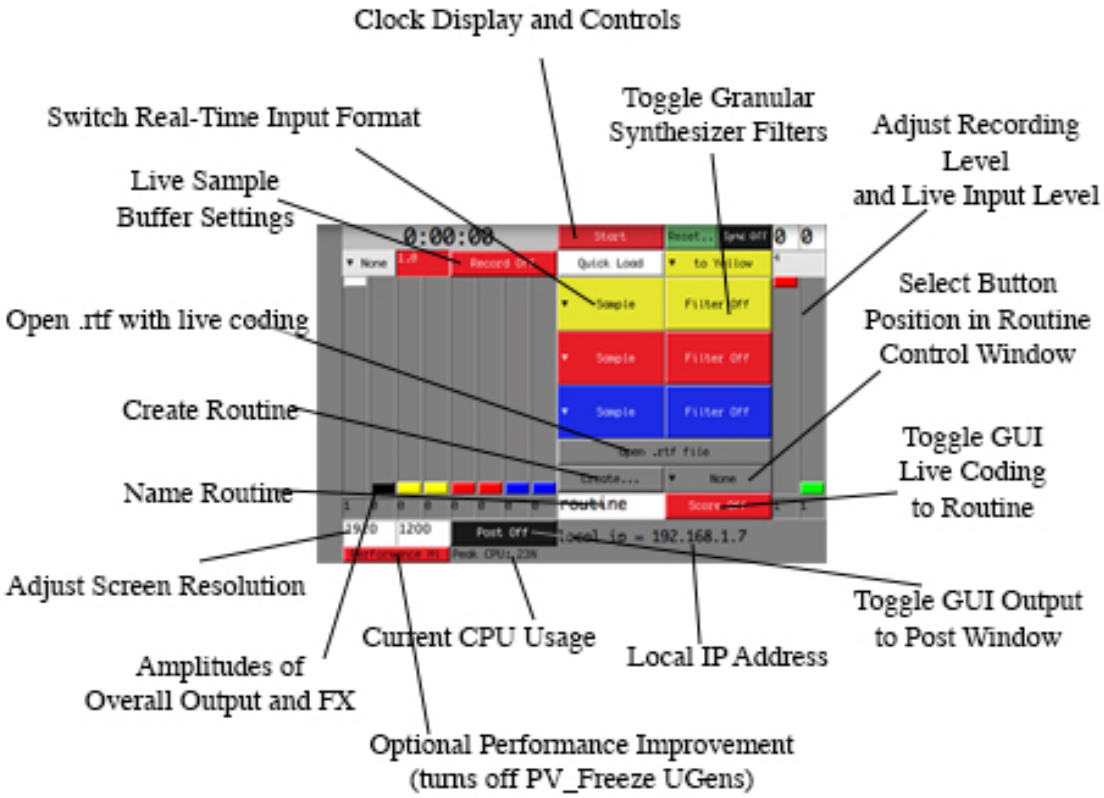


Figure 19. Arbitrary GUI Controls of Genesis

- Main controls for overall amplitudes of the full auditory mix, each bank of granular synthesisers and their associated freeze processes, and the input level of the 'mic' input
- Clock controls which can be synced over network
- Controls for the GUI live coding method which records GUI changes and wraps them as live code
- Input source selection
- Live sampling controls
- Filter toggles for each bank of granular synthesisers
- Editing of GUI scale
- Performance modifier option (*Performance Lo* recommended for machines running about 50% peak CPU)
- Current IP address display for network set-up

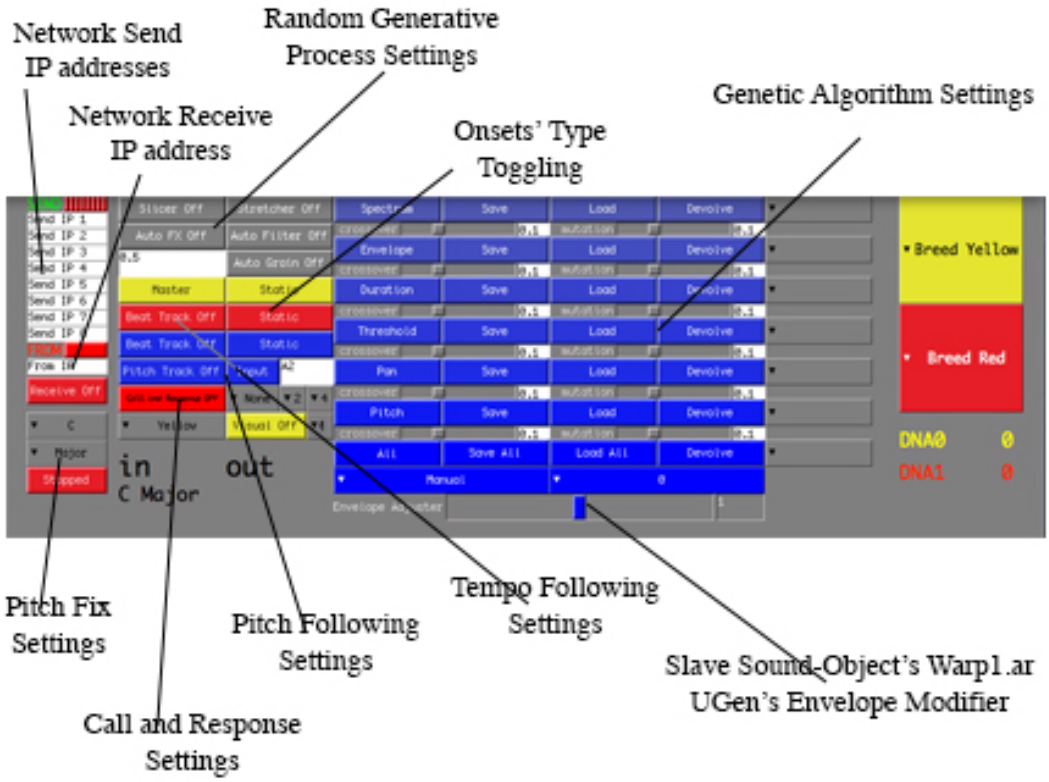


Figure 20. Further Arbitrary GUI Controls of Genesis and Genetic Algorithm controls

- Network send/receive set-up controls
- Pitch fixing controls and display
- Call and Response toggle and controls
- Beat tracking toggles
- Random search functions' toggles
- Static/Dynamic onset toggles
- Dynamic Scoring visualiser on/off toggle with option to make full screen or mini
- Pitch following of *control* source one controls
- Modified GAs controls for the spectrum, envelope, duration, threshold, pan and pitch of *control* source three/*slave*
- PopUp menu for selection of trigger for playback of *control* source three prior to allocation to granular synthesiser buffers
- Envelope time modifier for *control* source three prior to allocation to granular synthesiser buffers

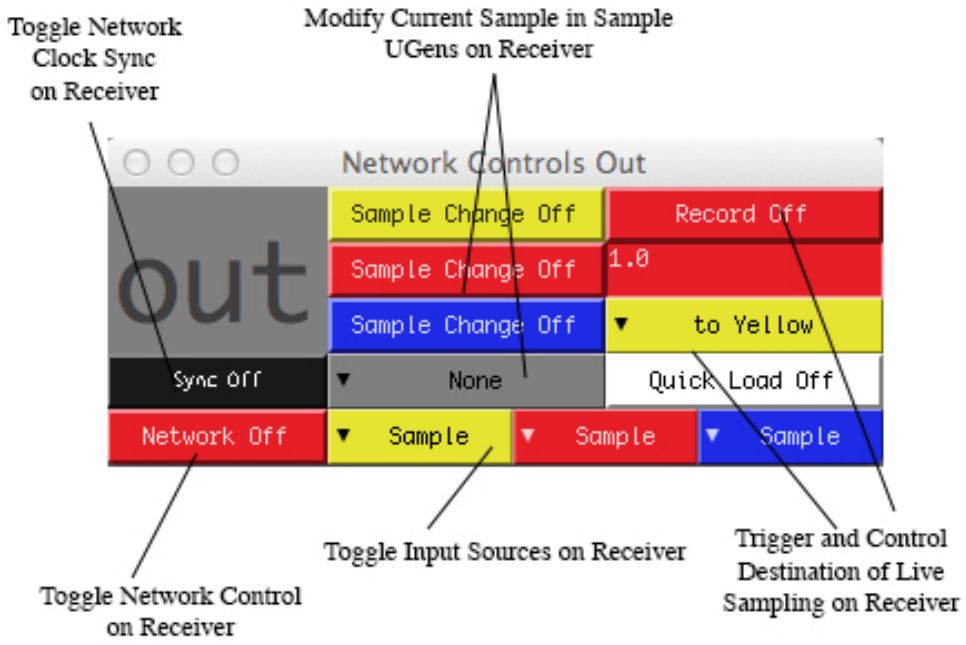


Figure 21. Network OUT window and GUI controls

- Network controls for a *sender* instance of *Genesis*; controls the options presented on all connected *receiver* instances of *Genesis*
- Toggle network control on/off on *receiver*
- Toggle clock sync on/off on *receiver*
- Change sample on *receiver* (bank of samples must be created prior to execution)
- Change *control* source on *receiver*
- Live sampling controls for *receiver*

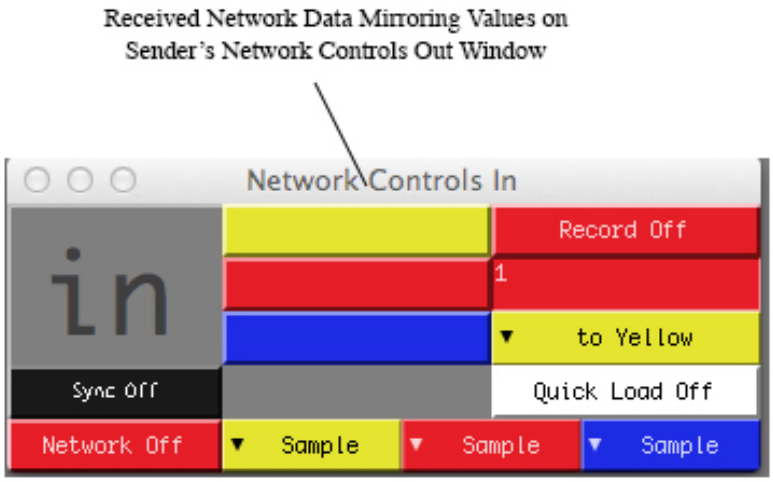


Figure 22. Network IN window and GUI display

- Network controls for a *receiver* instance of *Genesis*; shows the selections set by the *sender*
- Window is not interactive
- Displays network control on/off set by *sender*
- Displays clock sync on/off set by *sender*
- Displays changed samples set by *sender* (bank of samples must be created prior to execution)
- Displays changed *control* source set by *sender*
- Displays live sampling controls set by *sender*

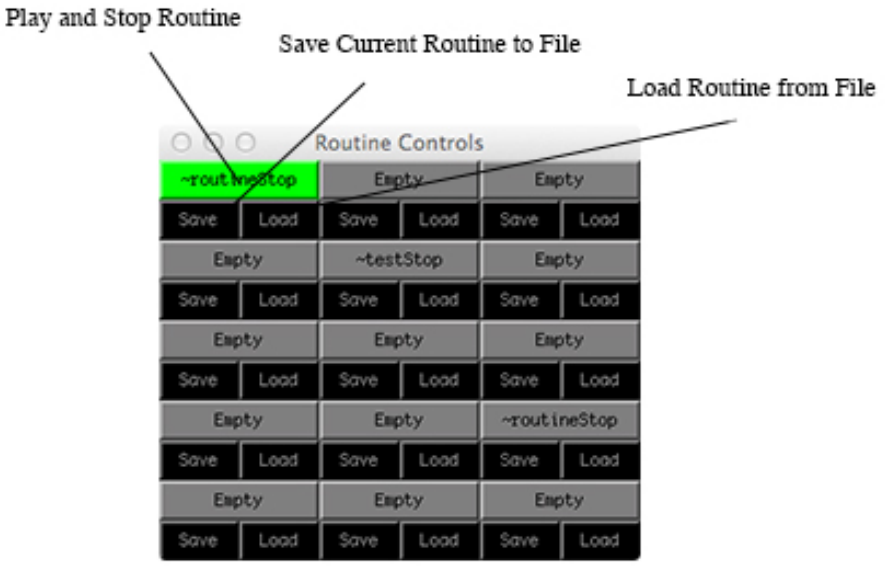


Figure 23. GUI Live Coded routines' window and controls

- Window for when GUI live coding is initiated
- Current file name is displayed. When clicked *on*, routine will play. When clicked *off*, routine will stop and reset
- Option to save and load files

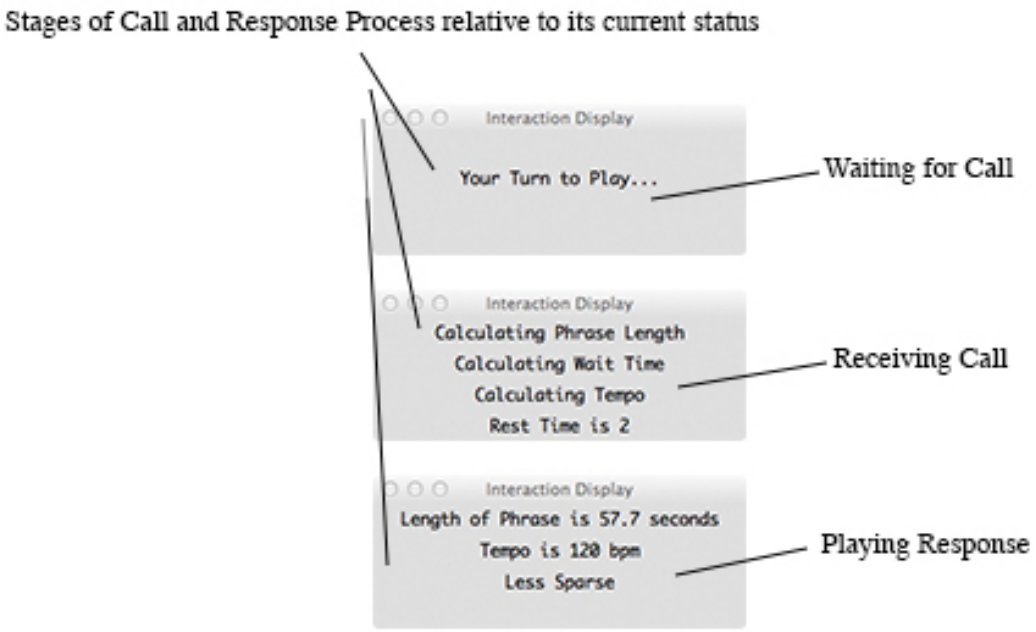


Figure 24. Call and Response displays

- Window for when Call and Response is initiated
- Three key states: waiting for call, receiving call and playing response
- When waiting for call, user must present the system with sound-object
- When sound-object begins and is above a set loudness threshold, receiving call is executed
- When sound-object's loudness falls below a set threshold, the system will generate a response

An Overview of the MIDI Implementation for the Control of Arbitrary Parameters within Genesis

Audiovisual example on the DVD *2. MIDI Implementation.mov* demonstrates the application of MIDI for controlling arbitrary parameters within *Genesis*. The real-time audio input from the Sample UGens forms *control* source one, with all modifications to its parameters controlled through the Korg nanoKontrol.

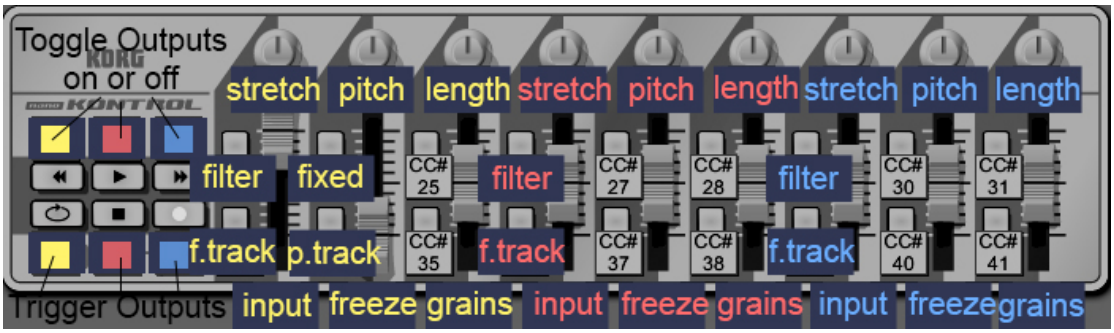


Figure 25. CC numbers attributed to a Korg nanoKontrol Scene One

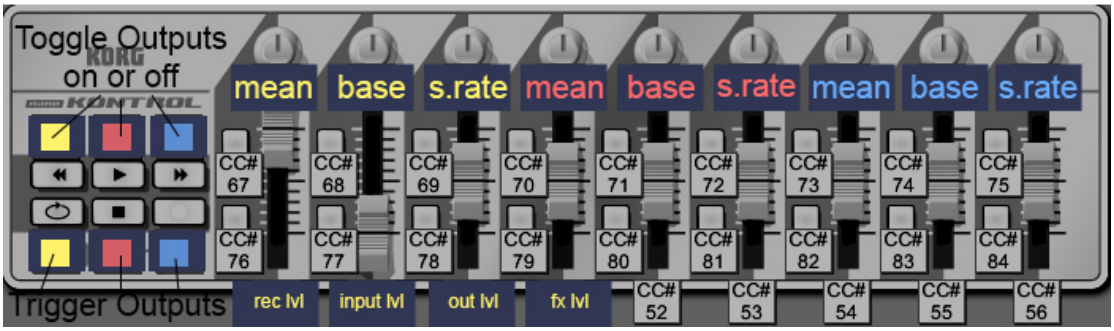


Figure 26. CC numbers attributed to a Korg nanoKontrol Scene Two

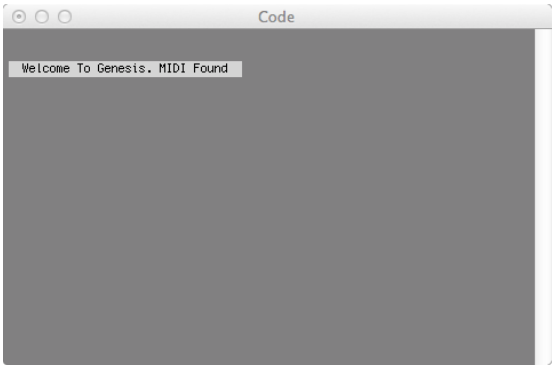


Figure 27. Notification of MIDI connection found posted at Genesis Initiation

The following code represents the parameter within *Genesis* and the attributed MIDI CC Number that can optionally control its value. Listed here are the MIDI CC numbers along with their attributed mapping to a Korg nanoKontrol, although any MIDI compatible device with CC functionality can be assigned to the described mappings.

```
                                    // Korg Knob 1 Scene 1
                                     (num == 14 && chan == 0, {{~osc0StretchSlider0.valueAction =
~rate0Spec0.map(value/127);}.defer});
                                    // Korg Knob 2 Scene 1
                                    if (num == 15 && chan == 0, {{~osc0PitchSlider0.valueAction =
~pitch0spec0.map(value/127);}.defer});
                                    // Korg Knob 3 Scene 1
                                    if (num == 16 && chan == 0, {{~grainLengthSlider0.valueAction =
~grainLengthSpec0.map(value/127);}.defer});
                                    // Korg Knob 4 Scene 1
                                    if (num == 17 && chan == 0, {{~osc1StretchSlider0.valueAction =
~rate0Spec0.map(value/127);}.defer});
                                    // Korg Knob 5 Scene 1
                                    if (num == 18 && chan == 0, {{~osc1PitchSlider0.valueAction =
~pitch0spec0.map(value/127);}.defer});
                                    // Korg Knob 6 Scene 1
                                    if (num == 19 && chan == 0, {{~grainLengthSlider1.valueAction =
~grainLengthSpec0.map(value/127);}.defer});
                                    // Korg Knob 7 Scene 1
                                    if (num == 20 && chan == 0, {{~osc2StretchSlider0.valueAction =
~rate0Spec0.map(value/127);}.defer});
                                    // Korg Knob 8 Scene 1
                                    if (num == 21 && chan == 0, {{~osc2PitchSlider0.valueAction =
~pitch0spec0.map(value/127);}.defer});
                                    // Korg Knob 9 Scene 1
                                    if (num == 22 && chan == 0, {{~grainLengthSlider2.valueAction =
~grainLengthSpec0.map(value/127);}.defer});

                                    // Korg Knob 1 Scene 2
                                    if (num == 57 && chan == 0, {{~trackingUpdateSlider1.valueAction =
~meanSpec1.map(value/127);}.defer});
                                    // Korg Knob 2 Scene 2
                                    if (num == 58 && chan == 0, {{~filterAdjuster0.valueAction =
~adjusterSpec0.map(value/127);}.defer});
                                    // Korg Knob 3 Scene 2
                                    if (num == 59 && chan == 0, {{~samplerateSlider0.valueAction =
~sampleRateSpec0.map(value/127);}.defer});
                                    // Korg Knob 4 Scene 2
                                    if (num == 60 && chan == 0, {{~trackingUpdateSlider2.valueAction =
~meanSpec1.map(value/127);}.defer});
                                    // Korg Knob 5 Scene 2
                                    if (num == 61 && chan == 0, {{~filterAdjuster1.valueAction =
~adjusterSpec0.map(value/127);}.defer});
                                    // Korg Knob 6 Scene 2
                                    if (num == 62 && chan == 0, {{~samplerateSlider1.valueAction =
~sampleRateSpec0.map(value/127);}.defer});
                                    // Korg Knob 7 Scene 2
```

```
                                    if (num == 63 && chan == 0, {{~trackingUpdateSlider3.valueAction =
~meanSpec1.map(value/127);}.defer});
                                    // Korg Knob 8 Scene 2
                                    if (num == 65 && chan == 0, {{~filterAdjuster2.valueAction =
~adjusterSpec1.map(value/127);}.defer});
                                    // Korg Knob 9 Scene 2
                                    if (num == 66 && chan == 0, {{~samplerateSlider2.valueAction =
~sampleRateSpec0.map(value/127);}.defer});


                                    // Korg Fader 1 Scene 1
                                    if (num == 2 && chan == 0, {{~osc0LevelSlider0.valueAction =
~volumeSpec0.map(value/127);}.defer});
                                    // Korg Fader 2 Scene 1
                                    if (num == 3 && chan == 0, {{~grainLevel0Slider.valueAction =
~volumeSpec0.map(value/127);}.defer});
                                    // Korg Fader 3 Scene 1
                                    if (num == 4 && chan == 0, {{~filterSlider0.valueAction =
~volumeSpec0.map(value/127);}.defer});
                                    // Korg Fader 4 Scene 1
                                    if (num == 5 && chan == 0, {{~osc1LevelSlider0.valueAction =
~volumeSpec0.map(value/127);}.defer});
                                    // Korg Fader 5 Scene 1
                                    if (num == 6 && chan == 0, {{~grainLevel1Slider.valueAction =
~volumeSpec0.map(value/127);}.defer});
                                    // Korg Fader 6 Scene 1
                                    if (num == 8 && chan == 0, {{~filterSlider1.valueAction =
~volumeSpec0.map(value/127);}.defer});
                                    // Korg Fader 7 Scene 1
                                    if (num == 9 && chan == 0, {{~osc2LevelSlider0.valueAction =
~volumeSpec0.map(value/127);}.defer});
                                    // Korg Fader 8 Scene 1
                                    if (num == 12 && chan == 0, {{~grainLevel2Slider.valueAction =
~volumeSpec0.map(value/127);}.defer});
                                    // Korg Fader 9 Scene 1
                                    if (num == 13 && chan == 0, {{~filterSlider2.valueAction =
~volumeSpec0.map(value/127);}.defer});


                                    // Korg Fader 1 Scene 2
                                    if (num == 42 && chan == 0, {{~mainOutSlider0.valueAction =
~volumeSpec0.map(value/127);}.defer});
                                    // Korg Fader 2 Scene 2
                                    if (num == 43 && chan == 0, {{~micLevelSlider0.valueAction =
~micLevelSpec0.map(value/127);}.defer});
                                    // Korg Fader 3 Scene 2
                                    if (num == 50 && chan == 0, {{~dryLevelSlider0.valueAction =
~volumeSpec0.map(value/127);}.defer});
                                    // Korg Fader 3 Scene 2
                                    if (num == 51 && chan == 0, {{~fxLevelSlider0.valueAction =
~volumeSpec0.map(value/127);}.defer});



                                    // Korg Top Button 1 Scene 1
                                    if (num == 23 && chan == 0, {{~filterOn0.valueAction =
(value/127);}.defer});
                                    // Korg Bottom Button 1 Scene 1
```

```
                                        if (num == 33 && chan == 0, {{~onsetChooser0.valueAction =
(value/127);}.defer});
                                        // Korg Top Button 4 Scene 1
                                        if (num == 26 && chan == 0, {{~filterOn1.valueAction =
(value/127);}.defer});
                                        // Korg Bottom Button 4 Scene 1
                                        if (num == 36 && chan == 0, {{~onsetChooser1.valueAction =
(value/127);}.defer});
                                        // Korg Top Button 7 Scene 1
                                        if (num == 29 && chan == 0, {{~filterOn2.valueAction =
(value/127);}.defer});
                                        // Korg Bottom Button 7 Scene 1
                                        if (num == 39 && chan == 0, {{~onsetChooser2.valueAction =
(value/127);}.defer});
                                        // Korg Top Button 2 Scene 1
                                        if (num == 24 && chan == 0 && value == 0, {{~pitchFixedButton0.valueAction
= (value/127);}.defer});
                                        // Korg Top Button 2 Scene 1
                                        if (num == 24 && chan == 0 && value == 127,
{{~pitchFixedButton0.valueAction = (value/127);}.defer});
                                        // Korg Bottom Button 2 Scene 1
                                        if (num == 34 && chan == 0 && value == 0, {{~pitchTrackButton0.valueAction
= (value/127);}.defer});
                                        // Korg Bottom Button 2 Scene 1
                                        if (num == 34 && chan == 0 && value == 127,
{{~pitchTrackButton0.valueAction = (value/127);}.defer});

                                        // Korg Bottom Button 7 Scene 1
                                        if (num == 31 && chan == 0, {{~recordSwitch0.valueAction =
(value/127);}.defer});
                                        if (num == 75 && chan == 0, {{~recordSwitch0.valueAction =
(value/127);}.defer});
                                        if (num == 115 && chan == 0, {{~recordSwitch0.valueAction =
(value/127);}.defer});
                                        //Korg 'rewind' button
                                        if (num == 47 && chan == 0 && value == 127, {{~triggerButton0.valueAction =
(value/127);}.defer});
                                        if (num == 47 && chan == 0 && value == 0, {{~triggerButton0.valueAction =
(value/127);}.defer});
                                        //Korg 'play' button
                                        if (num == 45 && chan == 0 && value == 127, {{~triggerButton1.valueAction =
(value/127);}.defer});
                                        if (num == 45 && chan == 0 && value == 0, {{~triggerButton1.valueAction =
(value/127);}.defer});

                                        //Korg 'forward' button
                                        if (num == 48 && chan == 0 && value == 127, {{~triggerButton2.valueAction =
(value/127);}.defer});
                                        if (num == 48 && chan == 0 && value == 0, {{~triggerButton2.valueAction =
(value/127);}.defer});

                                        //Korg 'loop' button
                                        if (num == 49 && chan == 0 && value == 127, {{~triggerButton0.valueAction =
(value/127);}.defer});
                                        if (num == 49 && chan == 0 && value == 0, {{~triggerButton0.valueAction =
(value/127);}.defer});
```

```
                                        //Korg 'stop' button
                                        if (num == 46 && chan == 0 && value == 127, {{~triggerButton1.valueAction =
(value/127);}.defer});
                                        if (num == 46 && chan == 0 && value == 0, {{~triggerButton1.valueAction =
(value/127);}.defer});

                                        //Korg 'record' button
                                        if (num == 44 && chan == 0 && value == 127, {{~triggerButton2.valueAction =
(value/127);}.defer});
                                        if (num == 44 && chan == 0 && value == 0, {{~triggerButton2.valueAction =
(value/127);}.defer});
                                });
```

5.3 Interactive Processes in *Genesis*

The interaction of the generative and analytical processes within *Genesis* are defined through OSC Messages that are divided between the sonic features of the real-time input sources' audio signals and the graphical user interface which is controlled through the computer keyboard, mouse and MIDI, as highlighted in the previous section 5.1 *An Overview of the Genesis System*. *Figure 28* illustrates the flow of interaction between the real-time audio input sources and the graphical user interface:

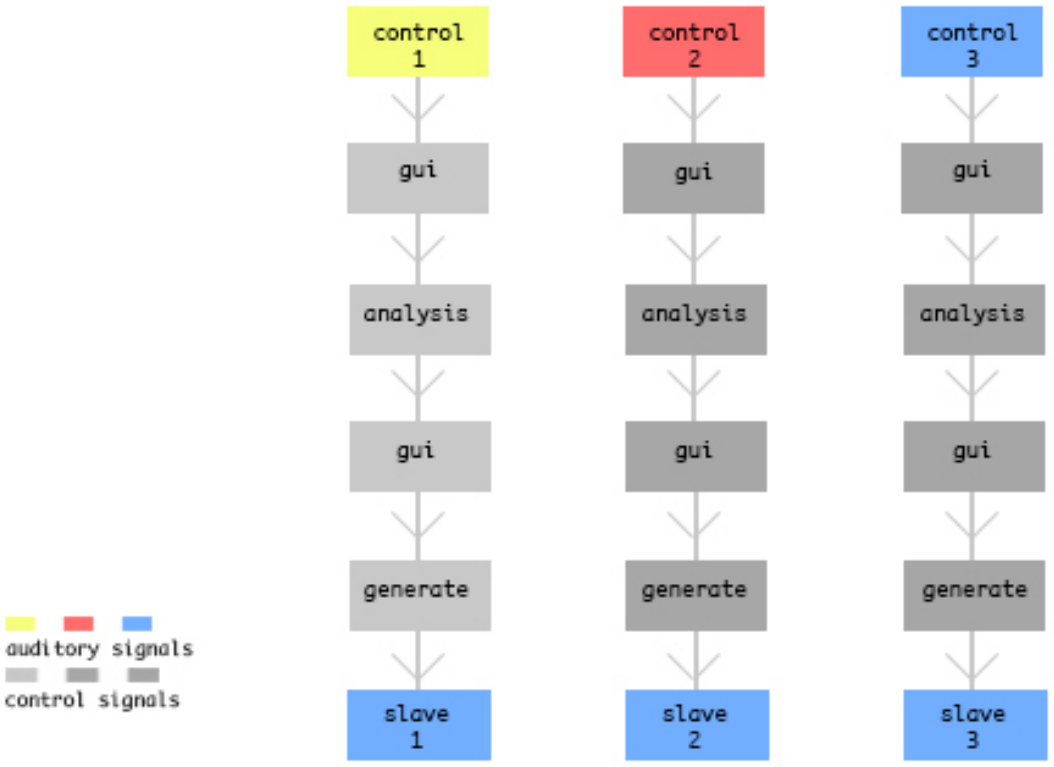


Figure 28. Flow of interaction in Genesis

Figure 28 demonstrates the capability of the graphical user interface to adjust specific sonic features of the real-time input sources' audio signals (relative to their formatting) *prior* to analysis and also *after* the analysis process for acute adjustment of the results for their consequent application to the selected generative processes. In addition, the GUI allows for the selection of *which* generative and analytical processes to use and, if applicable, the adjustment of their GUI modifiable parameters. *Figure 29* lists all major generative and analytical interactions defined by

the graphical user interface and the real-time input sources' audio signals, and how they are combined:

Graphical User Interface Interaction		**Real-time Input Source Interaction**	
Generative Process	*Analytical Process Adjustment*	*Generative Process*	*Analytical Process*
Buttons triggering the modified Genetic Algorithms (GAs) for the modification of the Granular synthesizers of the *control 3*/*slave* sound-object controlled by the *slave* input source	----------	Onsets of *slave* source define the onset and envelope of its Granular synthesizers, relative to the settings of the GAs	Onsets of the *control 3*/*slave* source are monitored
MultiSliderViews for the modification of the Granular synthesizers of the *slave* sound-object controlled by the *control 1* and *2* input sources	----------	Onsets of *slave* source define the onset and envelope of its Granular synthesizers, relative to the settings of the MultiSliderViews	Onsets of the *control 1* and *control 2* sources are monitored
----------	----------	Onsets of selected source define the onset of the grain freeze function	Onsets of the selected sources are monitored
Buttons triggering the fractal noise processes of the Granular synthesizers for *all slave* sound-objects	----------	----------	----------

Buttons triggering the pitch following of a *control* source by the *slave source*	Adjustment of the relative pitch of the *slave* source and playback rates of the granular synthesisers	A *control* source defines the pitch for the *slave* source to follow	Pitch of a *control* source and the *slave* source are monitored
Buttons triggering the tempo following of a *control* source by the *slave source*	Adjustment of which *control* source's tempo to follow by the *slave* source	A *control* source defines the tempo for the *slave* source to follow	Tempo of a *control* source and the *slave* source are monitored
Buttons triggering the spectral shape following of a *control* source by the *slave source*	Adjustment of the base frequencies of the spectrum of the *control* source	A *control* source defines the spectral shape for the *slave* source to follow	MFCCs of a *control* source and the *slave* source are monitored
Buttons triggering the pitch fixing of the *slave* sound-object's pitch to a chosen pitch structure such as a major scale	Adjustment of which pitch structure to apply	---	Pitch of the *slave* sound-object output is monitored
Buttons triggering the use of random buffer positions, filter frequencies, reverb times, time stretching and panning	---	---	---
Buttons triggering the Call and Response function	Adjustment of the time signature, input source, pitch structure and wait time of the response	---	Pitch, duration and onset of a selected input source are monitored
Buttons triggering the saving of interactions within the GUI	---	---	---

----------	Adjustment of envelope time	Loudness and onsets of selected input source defines the envelope time of *slave* sound-object and its triggering	Loudness of selected input source is monitored through sum of MFCCs along with its number of onsets
Live coding for the addition of new *SynthDefs*	Live coding for the modification of *all* defined parameters	----------	Analysis of signals, relative to any additions defined by the live coded *SynthDefs*
MIDI control for the triggering of selected generative processes	MIDI control for the adjustment of selected analytical processes	----------	----------
Pitch, MFCCs, tempo, spectral shape, onsets displayed in main *Genesis* GUI	----------	----------	Pitch, MFCCs, tempo, spectral shape, onsets of sources are monitored
Parameters of *slave*'s granular synthesizers' buffer positions, playback rate, durations, amplitudes, spatialisation and grain freezes displayed in dynamic scoring window	----------	----------	----------
Triggering of live-sampling and placement in to chosen *control* source	----------	Auditory signals of real-time input sources form content of sampled audio	----------
Button triggering the 0^{th}-order Markov chains for selected parameter changes to	----------	----------	

control source one, two and *slave*			

Figure 29. Table of Methods of Interaction in Genesis

In terms of interaction through the real-time input sources' audio signals, as shown in the *Figure 29*, these are controlled through the sonic features of loudness, pitch, spectral shape, tempo and onset, which are extracted through the analytical processes described in section 5.5 *Analytical Processes in Genesis* and represented as OSC Messages. These sonic features are then applied to dictate relative mappings within the generative processes, which are described in 5.4 *Generative Processes in Genesis*. So, through the real-time input sources' audio signals' sonic features, it is possible to interact with the pitch, onsets, duration, pseudo-timbre and amplitude of the *slave* sound-object. This is illustrated in the *Figure 30* for the interaction between *control* source one and the *slave* source:

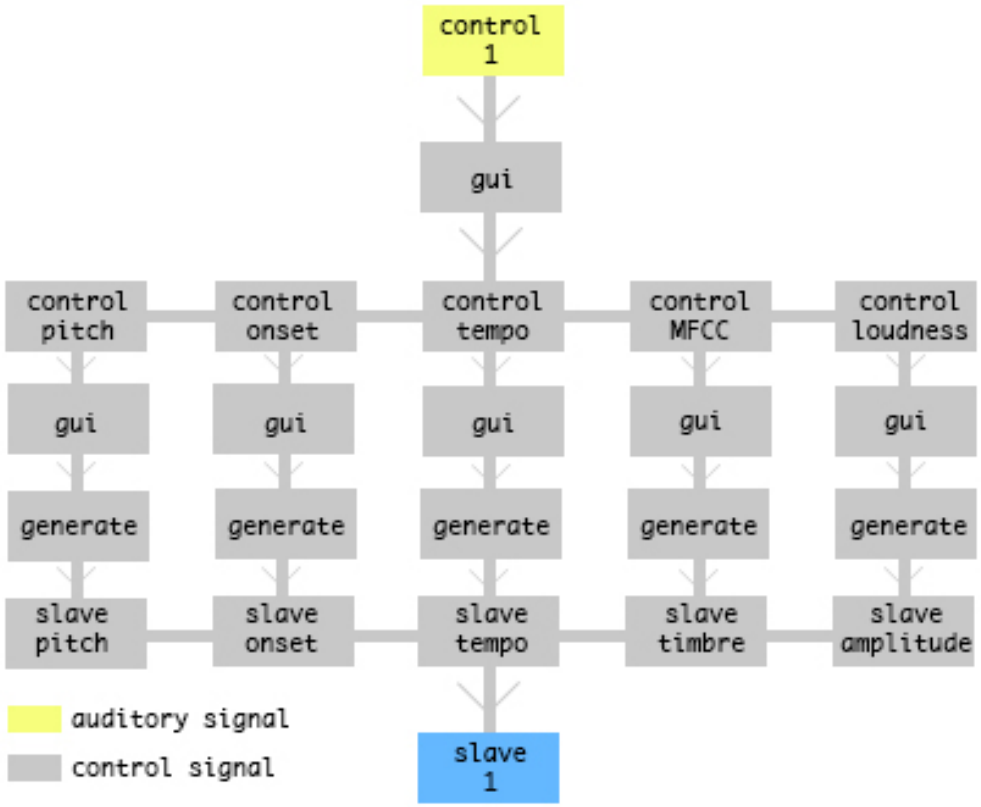


Figure 30. Interaction between Control source one and the Slave

For certain generative processes such as pitch following, the sonic features of the *slave* are compared to the *control* source. Therefore, the interactions of the *control* source are placed relative to the sonic features of the *slave* source and as a result, the pitch of the *slave* source follows the pitch of the *control* source. This is illustrated in *Figure 31* for the comparison of a *control* source's pitch to the *slave* source's pitch:

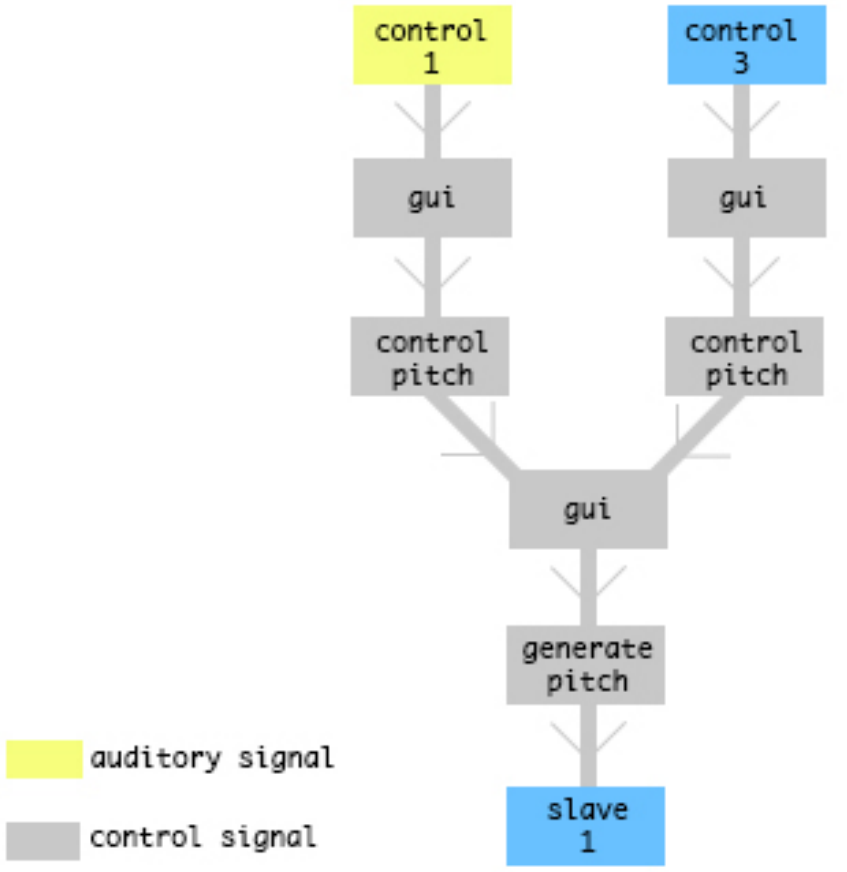


Figure 31. Method of pitch control of slave via control source's pitch

With regards to the interaction communicated through the graphical user interface of the *Genesis* system, these are defined through many different SuperCollider GUI classes listed in the following: Button, PopUpMenu, EZSlider, Slider, TextField, NumberBox, MultiSliderView, StaticText, UserView, Window and SoundFileView. The culmination of these GUI classes is then arranged as shown on the following page in *Figure 32*:

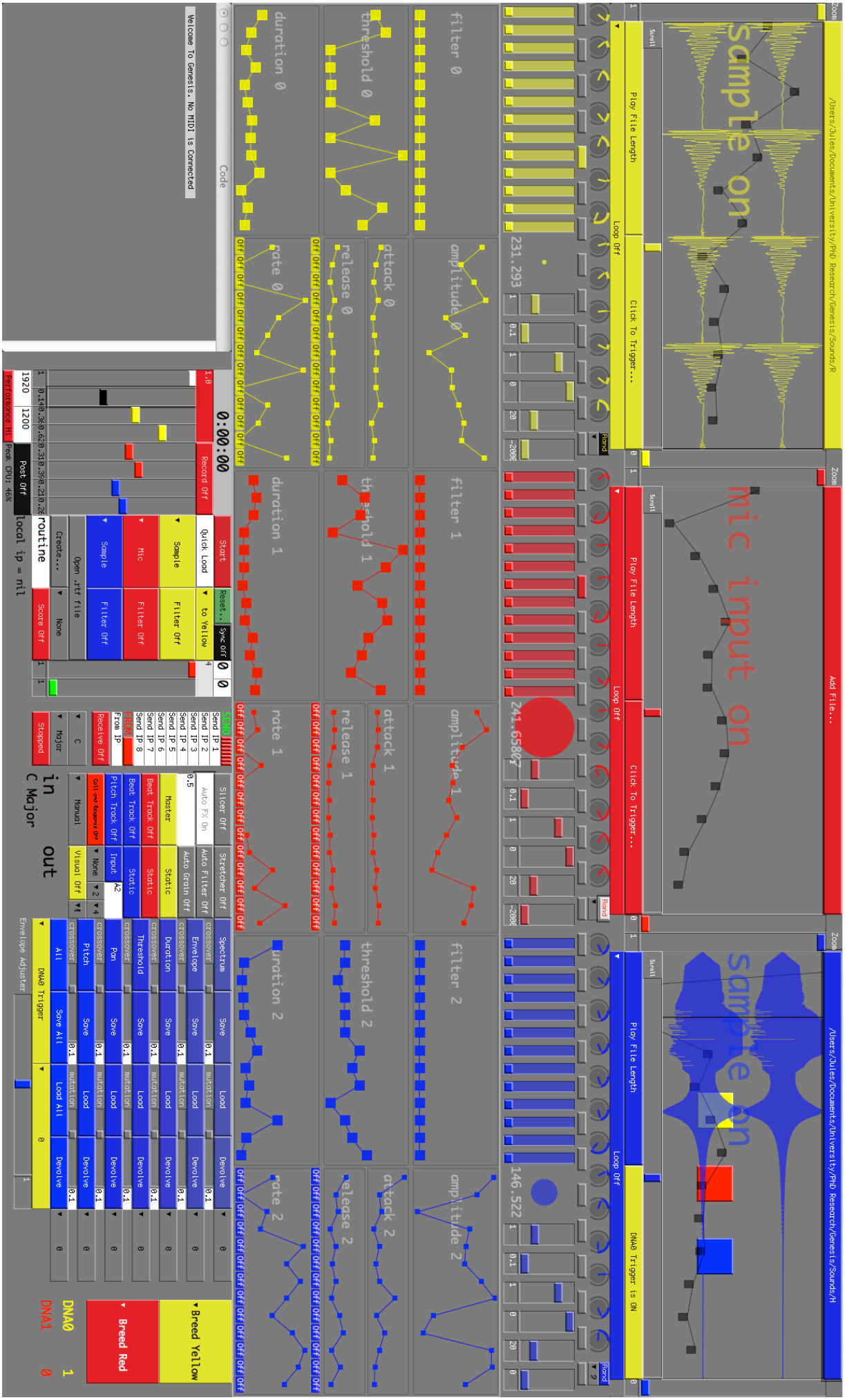


Figure 32. Genesis GUI

So, relative to the generative or analytical process, the graphical user interface permits the acute modification of many parameters within the *Genesis* system. For example, *Figure 33* demonstrates the method through which the threshold of the onsets of *control* source one can be modified within a MultiSliderView to dictate the triggers of the *slave* source's granular synthesizers:

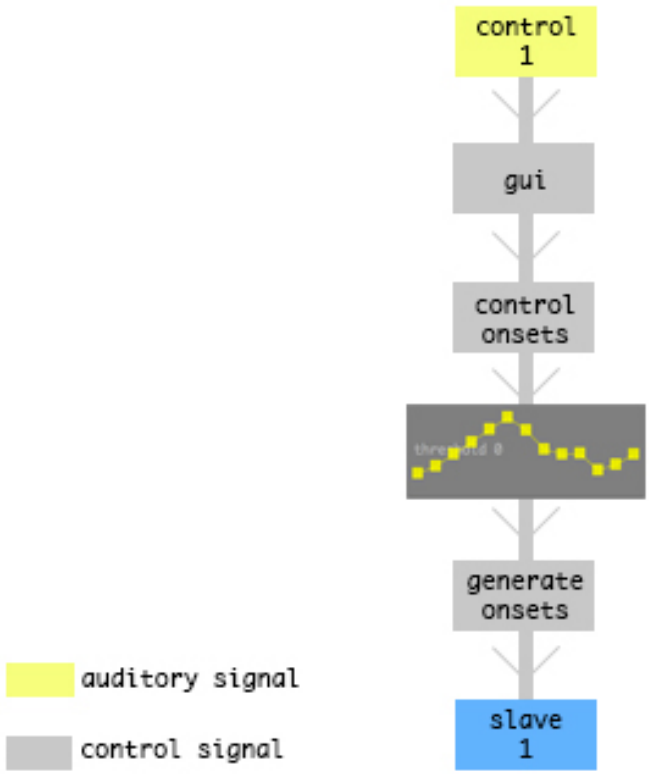


Figure 33. GUI modification of Thresholds for granular synthesizers' onsets

In addition to the modification of many parameters within the *Genesis* system, the graphical user interface is also used to abstract and represent the modifications and particular sonic features of the generative and analytical processes. *Figure 34* shows a screen shot of a single frame of the dynamic scoring system, which represents the buffer position, filter frequency, playback rate, duration, amplitude, spatialisation and grain freeze processes for each of the granular synthesisers of the *slave* sound-object, relative to system's perceived status of the real-time input audio signals:

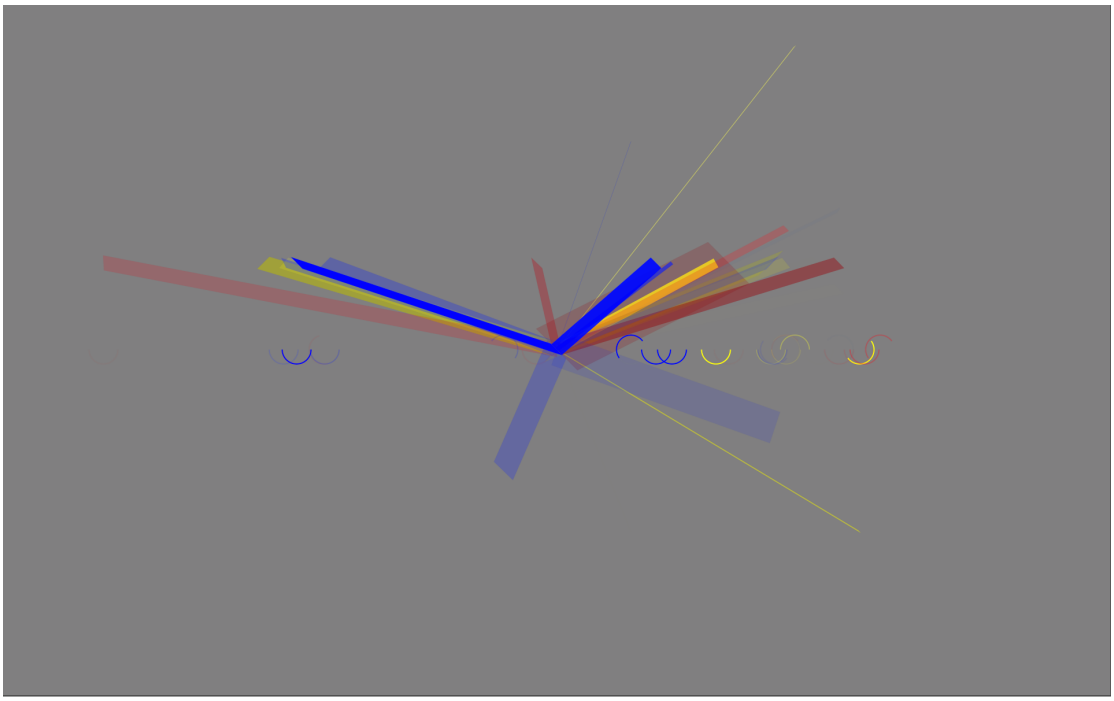


Figure 34. Screenshot of Dynamic Scoring System

Due to the nature of the fractal noise processes, which are explained in further detail in section 5.4 *Generative Processes in Genesis*, many of the modifications that are represented in the dynamic scoring system reflect the 'novel circumstance'[226] of each composition and the interactions that dictate their compositional processes. As a result, even if the same *control* and *slave* sound-objects are analysed in two consequent compositions, with the same compositional techniques, the resulting composition and its dynamic score will feature nuances that indicate the presence of the indeterminate fractal noise processes, which define various parameters of the granular synthesisers of the *slave* sound-object.

Therefore, the purpose of the dynamic score system is to represent to the composer *which* generative process is affecting *which* granular synthesizer of the *slave* sound-object in real-time and of that particular composition of the *now*, simplifying consequent adjustment of the *slave* sound-object's granular synthesizers within the graphical user interface and real-time input sources' audio signals to the relative scoring representations as illustrated in *Figure 35*:

[226] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49

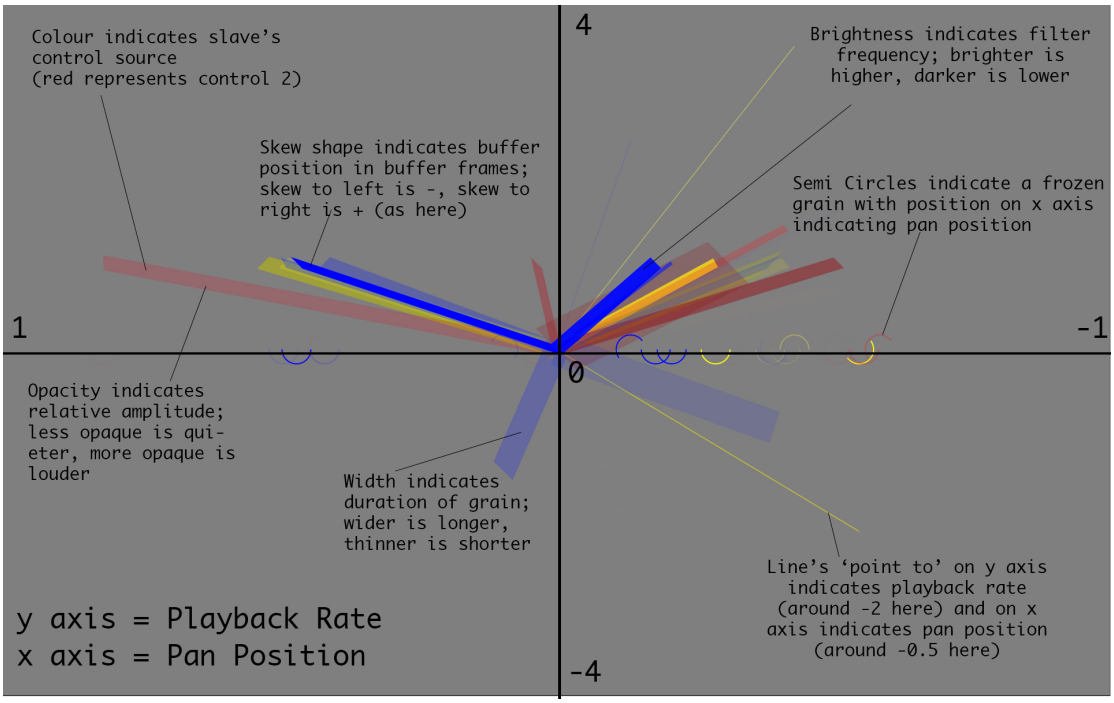


Figure 35. Annotation of Dynamic Scoring System

The dynamic scoring system's abstraction of the interaction between the real-time input sources' audio signals, the graphical user interfaces parameter settings and the generative processes that control the *slave* sound-object's granular synthesizers' sonic output in to the dynamic scoring system demonstrates the communication between the human performer and the computer performer; the dynamic score displays the unique interactions of the real-time input audio source, the graphical user interface and the generative processes, in accordance with the unfolding dialog in real-time, through a common paradigm, which in this case is the playback speed, buffer position, amplitude, trigger, spatialisation, freeze process, relative input source, filter frequency and duration.

The communication method in *Genesis* is achieved through the application of OSC messages for the symbolic representation of the playback speed, buffer position, amplitude, trigger, spatialisation, freeze process, relative input source, filter frequency and duration of the *slave* sound-object. However, collectively, the symbolic features create a holistic representation of the real-time processes within the *Genesis* system. So, for example, through the combination of the filter frequency, relative control source's triggers, pitch, and amplitude of a *slave*'s granular synthesizers, it is possible

to determine a *slave* output's pseudo-timbral features. Therefore, the ability to place composer-prescribed variables within the OSC Message's arguments, as afforded by the SuperCollider programming language, enables *Genesis* to form a unique *and* comprehensible paradigm for the communication of the interactions between the real-time input audio source, the graphical user interface and the generative processes for both symbolic and subsymbolic representations of sonic features.

The application of OSC messaging also permits the use of computer networks to communicate data between computer systems. In *Genesis*, the pitch, MFCCs and onsets of the *control* sources that are present on a local system can be sent, via the computer network to other instances of *Genesis*. As a result, the real-time input audio source's present on one instance of *Genesis* can control a number of networked instances of *Genesis*. In addition, arbitrary controls can be sent to networked instances of *Genesis* such as the triggering of recording buffers and toggling of the network's functionality. The flow of network communication between the instances of *Genesis* is one-way; one computer acts as a 'leader', sending the relevant data and controlling the particular sonic features of pitch, MFCCs and onsets of the relevant *control* sources on the networked instances of *Genesis*. Therefore, the *control* sources present on the 'leader' modify and manipulate the chosen *slave* source on the networked instances of *Genesis*, which in effect synchronizes the network's pitch, MFCCs and onsets for each *slave* source.

In order to set-up a network between *Genesis* instances, the user *must* connect the relative computers together, either wirelessly or through cable. Once a network has been established, and the relative Internet Protocol (IP) addresses have been allocated, the *Genesis* instance that is to *control* the particular features of the computers on the network (the *sender)* must input the relative IP addresses of the *receiver*. Then, on the *receiver*, the IP address of the *sender* must be input (the IP address of the local *Genesis* instance is displayed in the *Genesis* GUI along with a NumberBox for the placement of the relative IP address of the networked instance/s). The flow of communication is illustrated in *Figure 36*:

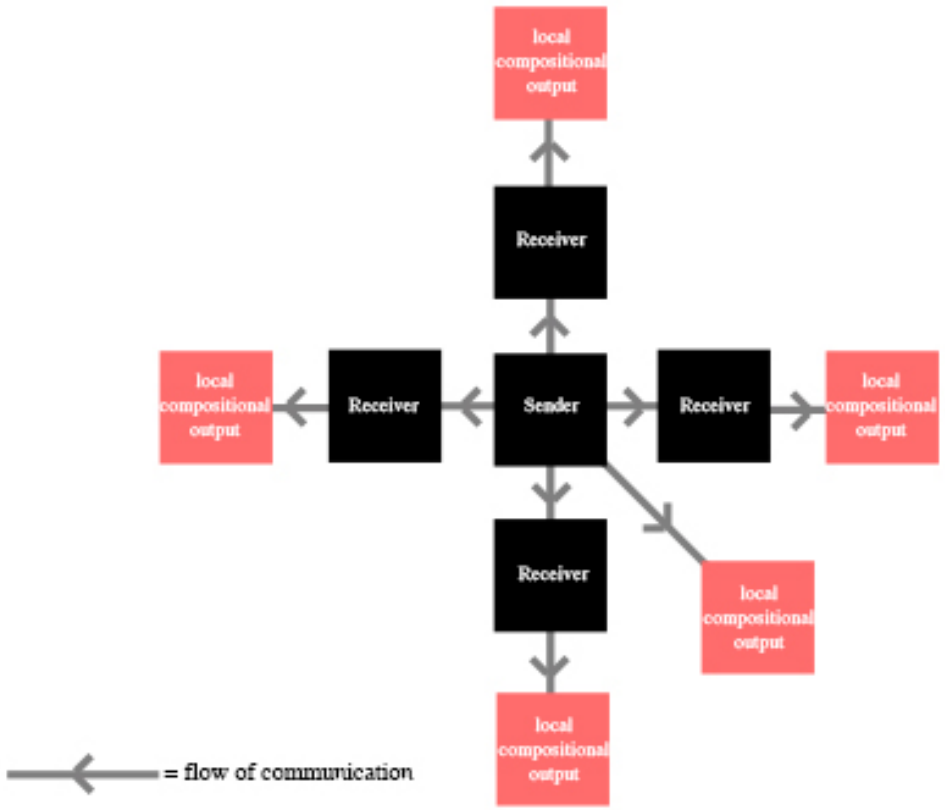


Figure 36. Network Interaction in Genesis

In terms of interaction between the *control* sources and the generative and analytical processes within *Genesis* that modify, manipulate and arrange the *slave* source, this can be approached through different methods, relative to the *control* source's format. So, for example, if using the live acoustic signals generated by an instrumentalist, it is possible to use a fixed score for the instrumentalist, perhaps featuring various monophonic melodies. The system can then follow the pitch, onset, MFCCs, loudness and tempo of the instrumentalist, forming a meta-instrument, which is a 'musician-machine interface and a gesture transducer intended for electro-acoustic music, multimedia work, and, more generally, for controlling algorithms in real-time'[227]. This thereby forms a Conductor Model of interaction, as proposed by Winkler (2001), through which the instrumentalist fulfills the role of a conductor 'acting as the single source for coordinating players' actions by directing the time flow, shaping the dynamics, and adjusting the acoustical balance'[228]. Therefore, the *Genesis* system follows the actions of the instrumentalist, in combination with any networked

[227] de Laubier, S. 1998. The Meta-Instrument. *Computer Music Journal* 22(1): 25
[228] Winkler, T. 2001. *Composing Interactive Music: Techniques and Ideas Using Max*. Massachusetts: MIT Press: 23-27

instances of *Genesis*, forming an ensemble of meta-instruments shown in *Figure 37* below:

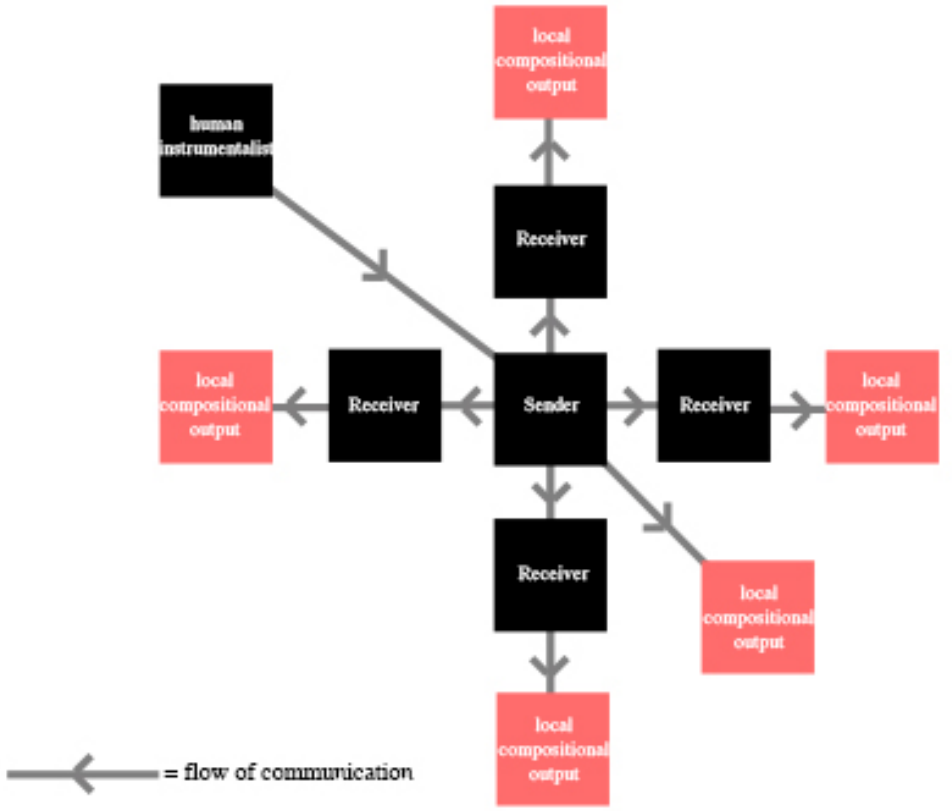


Figure 37. Ensemble of meta-instruments controlled by a live instrumentalist

Alternatively, if using the live acoustic signals generated by an instrumentalist, in the absence of a notated score for the instrumentalist but the application of the predefined sonic structures, the system can follow the pitch, onset, MFCCs, loudness and tempo of the instrumentalist, with the instrumentalist explicitly applying the resulting outputs of the *Genesis* system to influence their performance due to the lack of a determined notated score prescribing the compositional material. Therefore, the Improvisational Model as proposed by Winkler (2001) can be applied, through which the interaction of the performers influences both performers resulting compositional output as shown in *Figure 38* overleaf:

Figure 38. An improvisation model with a live instrumentalist and Genesis

Moreover, a human supervisor of *Genesis* can also be implemented to adjust and modify its outputs in the GUI relative to a live instrumentalist or any sound-object placed within the *control* sources. In a circumstance that involves a live instrumentalist, both performers have the option of both improvising, one improvising while the other follows a set programme or both following a score. *Figure 39* below shows an example of a human performer improvising with the *Genesis* GUI while the human instrumentalist follows a score, thereby dictating the onset, loudness, pitch and timbral parameters of supervised *Genesis* system, relative to the parameter settings entered by the *Genesis* supervisor:

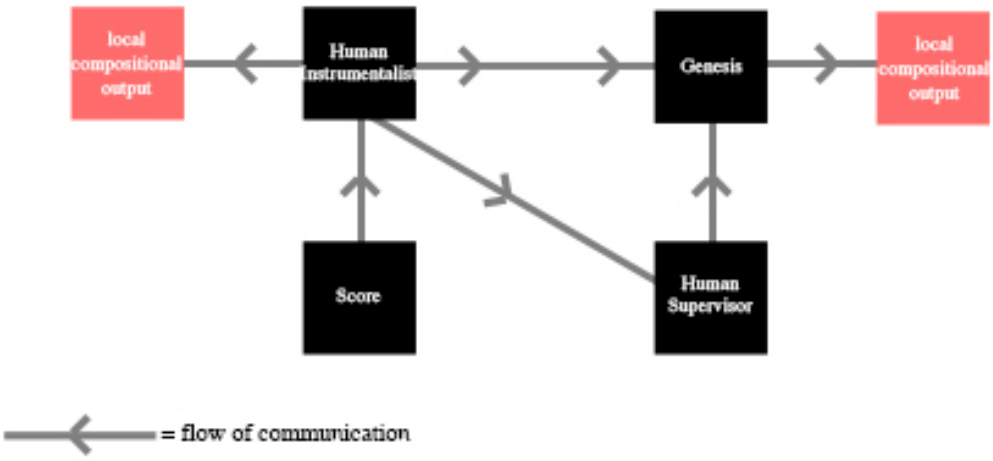


Figure 39. Human Supervised implementation of Genesis with a live Instrumentalist

Genesis can also be used as an unsupervised music system in which it can generate compositions relative to live streams such as train station ambience or to recorded audio material allocated to its *control* or *slave* inputs. In addition, multiple instances can be used using the network functionality thereby syncing the control of the

instances' outputs to a central system as described in *Figure 36. Network Interaction in Genesis*. Once input sources have seen selected, the system requires no supervision unless prescribed by a composer. *Figure 40* below demonstrates such a scenario whereby recorded samples and train station ambience are used to control a series of *Genesis* instances, each with their own *slave* sound source, generating four distinct sonic outputs:

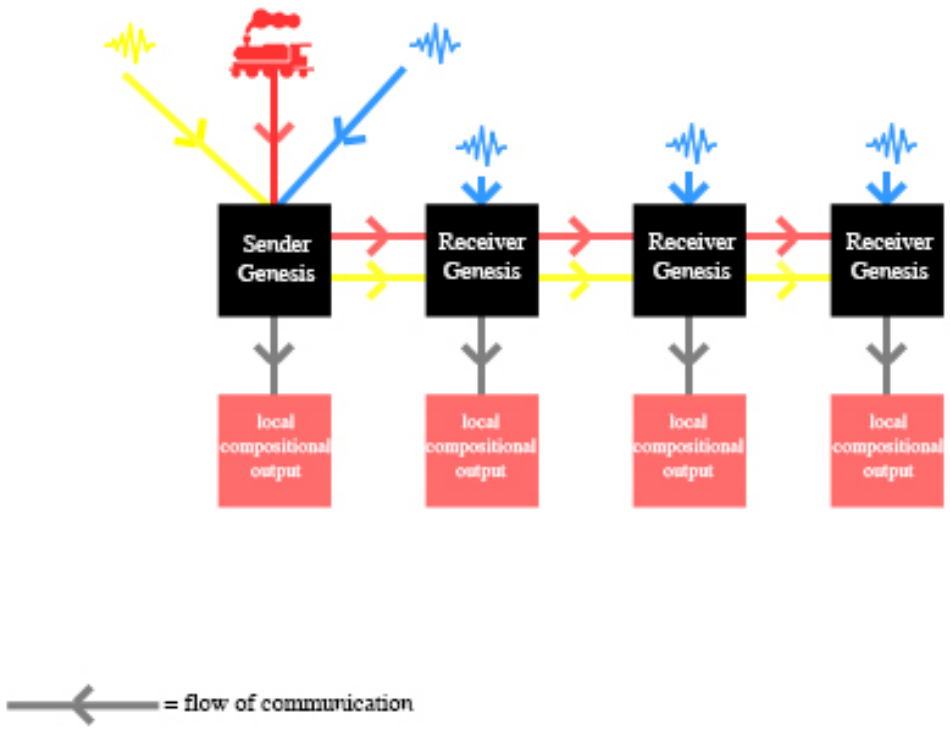


Figure 40. Unsupervised network of Genesis systems

Therefore, *Genesis* hybridizes the *generative* and *adaptive* methods of creativity discussed in chapter 4.1 *Interaction with Creative Systems* by requiring (even in an unsupervised circumstance) a human performer/composer to instigate a scenario through which *Genesis* can interact with sound-objects. Consequently, the resulting composition is predominantly attributable to the *adaptive* of a creativity human composer, with the explicit *generative* creative outputs of *Genesis* credited to the system.

As a result of the methods of interaction available within *Genesis* through the combination of the GUI and the real-time input studio sources, an instance of a

Genesis system has six main modes of interaction, which can be adjusted on-the-fly, through the relevant application of the generative, analytical and interactive processes. These are described in *Figure 41*, along with the level of influence that the *Genesis* system may have on the resulting compositional process:

***Genesis* Mode**	**Description**	**Level of Influence**
Unsupervised	• *Control* sources of any type • Required generative processes are toggled or routined within the GUI *prior* to composition • The result is output with no explicit external modification by a human performer of the GUI parameters in real-time • No implicit requirement for a human supervisor (apart from the set-up) • Human instrumentalist may improvise relative to the system's outputs	High (if no human present in compositional process) Low, Mid, High (if Human improviser applied)
Supervised Score-Following	• *Control* sources of fixed structures such as a live instrumentalist following a score • The required generative processes are toggled and adjusted within the GUI *prior* and *during* composition relative to a score • The result is output with explicit modification via GUI in real-time • Requirement of a human supervisor modifying parameters of *Genesis* within GUI, relative to a predefined score/structure	Low, Mid, High
Supervised Improvisation	• *Control* sources *may* include an improvisatory human instrumentalist, who may wish to improvise relative to the outputs of *Genesis*	Low, Mid, High

	• The required generative processes are toggled and adjusted within the GUI *prior* and *during* composition • The result is output with explicit external modification and improvisation in real-time by a human supervisor • Requirement of at least *one* human improviser (one human supervisor modifying the parameters of *Genesis* within GUI)	
Unsupervised Ensemble	• *Control* sources of any type sent to *sender* instance of *Genesis* forwarded to networked *receiver* instances • Required generative processes are toggled or routined on each instance within the GUI *prior* to composition • The result is output with no explicit external modification by human supervisors/s in real-time • No requirement for a human supervisor (apart from the set-up) • Human instrumentalist may improvise relative to the system's outputs	High (if no human present in compositional process) Low, Mid, High (if Human improviser applied)
Supervised Ensemble Score-Following	• *Control* sources of fixed structures sent to *sender* instance of *Genesis* forwarded to networked *receiver* instances • Required generative processes are toggled and adjusted within the GUI on each instance *prior* and *during* composition relative to a score • The result is output with explicit external modification in real-time by human supervisors for each instance • Requirement of a human supervisor modifying parameters of *Genesis* within	Low, Mid, High

	GUI	
Supervised Improvisation Ensemble Network	• *Control* sources *may* include an improvisatory human instrumentalist sent to *sender* instance of *Genesis* forwarded to networked *receiver* instances • The required generative processes are toggled and adjusted within the GUI on each instance *prior* and *during* composition • The result is output with explicit external modification and improvisation in real-time by human supervisors for each instance • Requirement of at least *one* human improviser per *Genesis* instance (one human supervisor modifying the parameters of *Genesis* within GUI)	Low, Mid, High

Figure 41. Table of Modes of Interaction with Genesis

So, considering *Figure 41*, it is certainly evident that a number of models of interaction are feasible. Indeed, these models can be combined together, thereby integrating a variety of models of interaction within a performance such as an *Unsupervised* instance of *Genesis* using the auditory outputs of a *Supervised Improvisation* to contribute to the ongoing compositional process. In addition, the models of interaction can be initiated in real-time, allowing a composer to switch between interaction methods during a performance.

Therefore, interaction within the *Genesis* system combines sonic features obtained from real-time input audio sources in combination with the modification of these sonic features and the generative processes within *Genesis* through the graphical user interface. As demonstrated, an instance of *Genesis* can perform unsupervised during the real-time compositional process resulting in a highly influential algorithmic compositional process. In contrast, an extensive amount of compositional values can be input by a human composer through the GUI or explicit instrumentalists during the

compositional process dictating specific generative parameters resulting in a low or mid or high level influence by *Genesis* on the algorithmic compositional process.

5.4 Generative Processes in *Genesis*

The primary generative processes within *Genesis* centre on the modification of a series of granular synthesisers for each *slave* source, as introduced in section 5.1 *An Overview of the Genesis System*. To elaborate, the *slave* sound-object is written to a series of audio buffers in real-time, with the capability to modify the length of these buffers through the GUI. There are thirty-nine granular synthesizers for each *slave* sound-object, divided between the three *control* sources, resulting in three sets of thirteen granular synthesizers; each granular synthesizer of each set represents one of the thirteen Mel-frequency cepstral coefficients (MFCCs) obtained from the relative *control* source. Therefore, this forms three modified *slave* sound-object's applying the buffered *slave* sound-object as an auditory source, with each *slave* sound-object's granular synthesizers' relative to each of the *control* source's MFCCs. Moreover, the application of thirteen granular synthesisers for each *control* sources results in a manageable GUI mapping for each of the granular synthesisers' parameters, and serves to increase CPU efficiency by limiting the total number of processes that can occur at any one time while still generating the desired sonic output.

Each granular synthesizer uses the GrainBuf.ar UGen, which reads the respective audio buffer of the buffered *slave* sound-object. The GrainBuf.ar UGen features a number of parameters: number of channels, trigger, duration, sound buffer, playback rate, buffer position, interpolation, pan, grain envelope and maximum number of grains. The parameters that are modified by the generative processes within *Genesis* are the trigger, duration, playback rate, buffer position, pan and grain envelope (the grain envelope and pan are placed outside of the UGen as they are also applied to processes outside of the granular synthesizers, but they still serve the same purpose as the parameter within the UGen).

The triggering of the grains in the GrainBuf.ar UGen is set by the onsets of the *control* sources. In order to obtain the onsets, each *control* source is filtered through a

series of thirteen band-pass filters, with two modes of functionality: static and dynamic. These modes of functionality can be toggled in real-time within the GUI using the respective 'Static/Dynamic' toggle button for the *control* source of the granular synthesizers' onsets. The static functionality fixes the filter frequency relative to the position of the filter defined in the MultiSliderView of the GUI, while the dynamic functionality applies the MFCC values of the *control* source to multiply the filter frequencies (MFCCs are a subsymbolic representation of timbre as discussed in chapter 3.2.3 *Timbre Perception*). So, with dynamic functionality, each of the band-pass filters' frequencies dynamically changes relative to the respective value of the MFCC mapped to that particular band-pass filter.

The auditory signals that pass through the filters are individually assessed to measure the onset of events, with the thresholds of the amplitudes that are to be considered an onset modifiable within the GUI. The onsets that are above the threshold are used to trigger a series of thirteen envelopes for their respective of granular synthesizer, as well as triggering a grain within the GrainBuf.ar UGen's 'trigger' parameter. As a result, the application of the dynamic functionality represents the dynamically changing spectral onsets of each *control* source. In contrast, the static functionality can be used to create a composer-defined spectral shape within the GUI, reflecting the onsets at the specified static frequencies. *Figure 42* illustrates this process for *control* source one and the thirteen granular synthesizers of *slave* 1:

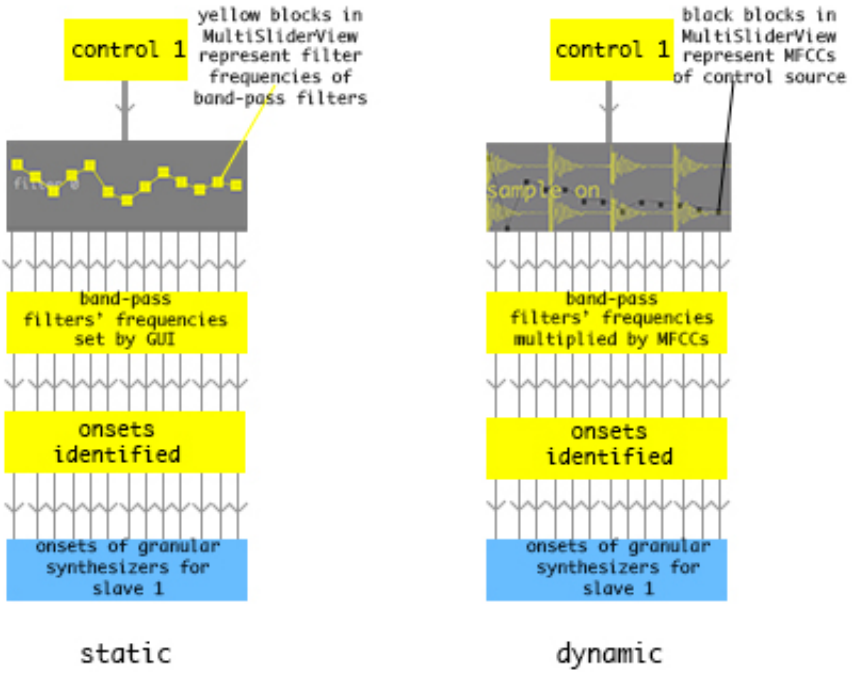


Figure 42. Static and Dynamic control of onsets of granular synthesizers

The envelope of each grain is defined through the use of the EnvGen.kr UGen, which features an envelope parameter and a gate. The gate is triggered by the respective *control* source's onsets, which initiates the beginning of a grain's envelope. The envelope parameter for each of the granular synthesizers uses the Env.perc UGen, which dictates the attack time, release time, peak amplitude level and curve. The curve is set at the time of *Genesis* startup to a *sine* in order to limit clipping, with the attack, release and amplitude modifiable within the GUI. So, the onsets of the *control* sources trigger a grain *and* its envelope, while the GUI sets the attack time, release time and peak amplitude of the envelope itself. Furthermore, the durations of the grains can be modified by the GUI, which can be adjusted relative to the attack time and release time of a grain's envelope.

As with the extraction of the onsets from the *control* source, a series of band-pass filters can be multiplied by the values of the MFCCs of the *slave*'s chosen onset controller to form a dynamic modification of the filter frequencies relative to the MFCCs of the *control* source. However, instead of placing the filters on a *control* source's auditory signal for consequent analysis to extract the onsets relative to the MFCCs of the *control* source, a band-pass filter is assigned to the auditory output for

each of the granular synthesizers. Therefore, a representation of a *control* source's spectral shape can be used to modify the spectral shape of a respective *slave* source through the dynamic modification of the granular synthesizers' outputs' auditory signals via their respective band-pass filters. Furthermore, the use of the GUI's MultiSliderView filter frequency filter frequency may also be applied to each of the band-pass filters, forming a static spectral shape of the granular synthesizers' band-pass filters.

The shape of the spectrum of the *control* source is represented within the GUI, along with the ability to modify the mapping of the fundamental frequencies of the filter frequencies through a GUI Slider. This is illustrated in *Figure 43* for the *slave* sound-object controlled by the onsets of *control* source one.

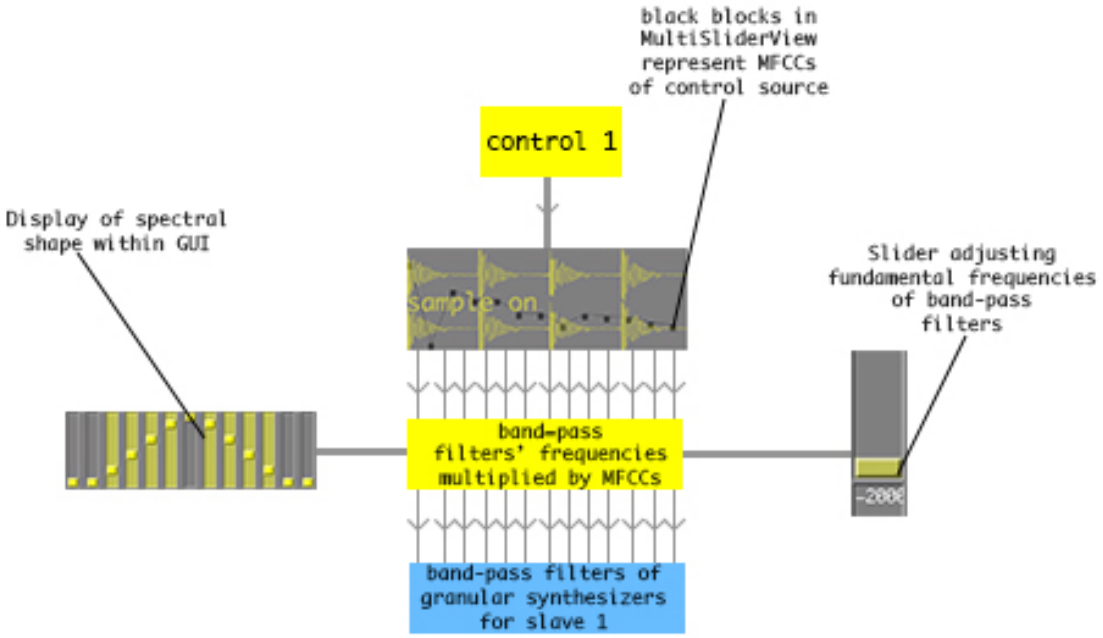


Figure 43. Mapping of MFCCs to Filter Frequencies of Granular Synthesizers

Due to the nature of the spectral shape following process task, which may cause excessive clipping within the auditory signal as a consequence of multiplying filter frequencies by the values of MFCCs mappings, a task runs alongside the process to

monitor the maximum and minimum values of the filter frequencies; the task modifies the amplitude of the granular synthesizers to 0 if the process's filter frequency is above 4000Hz or below 40Hz. This serves to limit the occurrence of clipping within the signal when the filters and dynamic spectral shape following process are toggled *on* within the GUI.

The onsets obtained from the *control* sources are also used to trigger other generative processes. The fractal noise process, which defines the buffer position of each granular synthesizer, uses the respective onset to trigger a new value. The fractal process itself applies the PinkNoise.kr UGen, which outputs values that are mapped relative to the size of the granular synthesizer's audio buffer. So, if a buffer is chosen in real-time through the GUI of one second in duration at a sample rate of 44100kHz, the buffer position may be between -44100 and 44100, following the nature of a pink noise fractal process as described in chapter 3.1 *An Introduction to Real-time Generative Algorithmic Systems*. The value of the PinkNoise.kr UGen is consequently output *only* when an onset triggers a new grain, thereby syncing the modification of buffer position, with the triggering of a grain and its envelope. The following data output represents the buffer position values generated over time, relative to its triggering by its associated *control* source's onset:

```
UGen(Gate): 29623
UGen(Gate): 37330.2
UGen(Gate): 40837.7
UGen(Gate): 21464
UGen(Gate): -5418.71
UGen(Gate): 7520.01
UGen(Gate): 19262.1
UGen(Gate): -6.47678
UGen(Gate): -32154.5
UGen(Gate): -14030.1
UGen(Gate): 4310.17
UGen(Gate): -4342.77
UGen(Gate): 24993.4
UGen(Gate): 38844
UGen(Gate): 10643.5
UGen(Gate): 30402.7
UGen(Gate): 11447.3
UGen(Gate): 19468.2
UGen(Gate): 5513.38
UGen(Gate): 34995.2
UGen(Gate): 14772.4
UGen(Gate): 27338.1
UGen(Gate): -23407.4
```

The resulting output of the fractal noise process causes the buffer position of each of the granular synthesizers triggered by a particular *control* source to differ, forming a collage of buffer positions, generating a collaged representation of the *slave* sound-object's auditory signal. This process is illustrated in *Figure 44*, supposing *control*

source one (yellow), *control* source two (red) and *control* source three/*slave* (blue) have triggered the onsets for ten of their granular synthesizers:

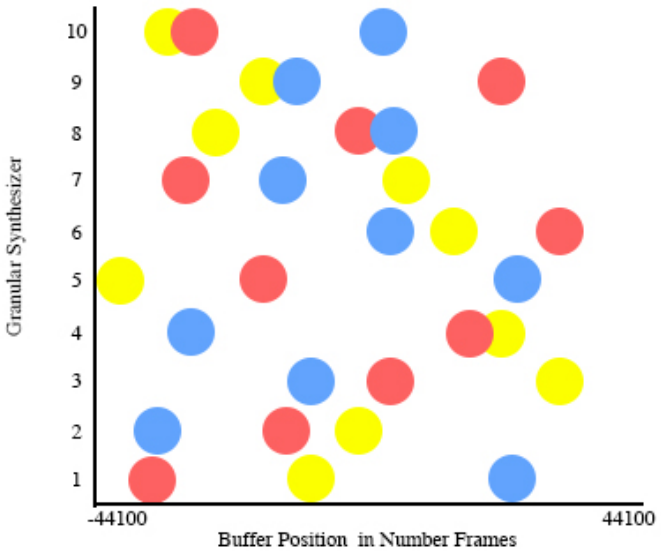


Figure 44. Collage of Buffer Positions

Similar to the generative process defining buffer position, the playback rate can also be dictated by the fractal noise process (this process can be toggled on or off for each granular synthesiser through the graphical user interface). The PinkNoise.kr UGen is applied, but instead is mapped to values between -4 and 4, thereby allowing playback rates of between to -4x or 4x the recorded speed, with the duration of the grain mapped relative to the playback rate (and the pitch output by the pitch tracking process if pitch tracking is toggled *on*).

The pitch following process applies the pitch data extracted from *control* source one, and compares it to either a composer-defined pitch value for the *slave* object or the pitch data extracted from the *slave* sound-object prior to any pitch adjustments by the pitch following process. (The pitch extraction method is detailed further in section 5.5 *Analytical Processes in Genesis*). The difference between the two pitches is then calculated, with the output of the pitch following process placed in to the rate parameter each GrainBuf.ar, altering the output pitch of each grain's *slave* sound-object relative to *control* source one, thereby reflecting the pitch contour of *control* source one. This is illustrated in *Figure 31 Method of pitch control of slave via*

control source's pitch in section 5.3 *Interactive processes in Genesis*. In addition, the pitch contour that results from the modification of each granular synthesiser is represented in the dynamic scoring system.

The fractal noise process can also modify the recording rate of the slave sound-object to the audio buffers, which can be toggled on or off in the GUI. Furthermore, the playback rates and recording rates can be multiplied via the rate MultiSliderView in the GUI, offering real-time adjustment of each fractal process's bounds. The process is triggered relative to the onsets of the *control* sources for each grain of the granular synthesizers, which, as with the buffer position, is consequently output *only* when an onset triggers a new grain, thereby syncing the modification of playback rate and/or recording rate, with the triggering of a grain and its envelope. This process is illustrated for the playback rate of a single granular synthesiser triggered by *control* source one in *Figure 45* below (the process is also the same for selection of the recording rate and selection of buffer position, although selection of the buffer position is not modifiable by the output of the pitch tracking algorithm):

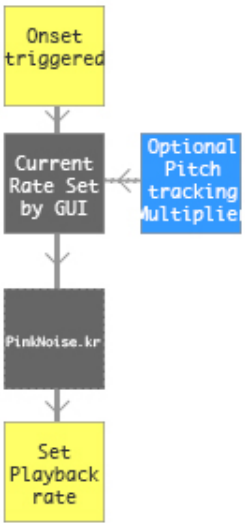


Figure 45. Flow of fractal noise modification of playback rate

In addition to the fractal noise processes, the onsets, and more specifically, the values of the envelopes the onsets triggered by the *control* sources are applied to the grain freezing process; the PV_Freeze UGen holds a grain of sound when triggered, with any value above 0.5 causing a grain to be held. There are 39 PV_Freeze UGens, each

assigned to each granular synthesizer's output, with the triggering of the process relative to the envelope of the granular synthesizer it is assigned to. As a result, grains of sound are 'frozen' when the granular synthesizer's envelope is above a value of 0.5 and held until the falls below the threshold. The amplitude of the grains freeze process for each *control* source's granular synthesizers can be collectively adjusted via the GUI and/or the arbitrary MIDI controls. The freeze grain process is illustrated in *Figure 46* for a single freeze grain controlled by *control* source one:

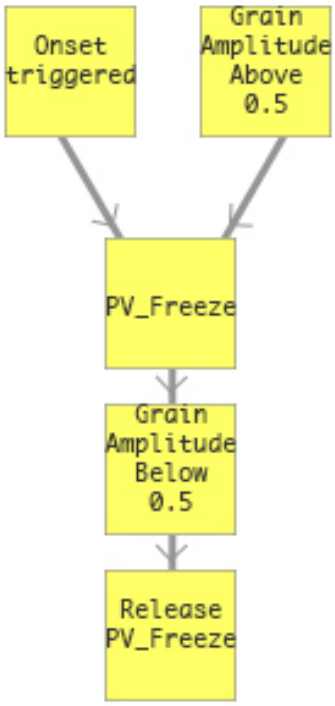


Figure 46. Freeze grain process

With regards to the interaction of the GUI processes that modify the parameters of the granular synthesisers, these are approached relative to the *control* source; for *control* sources one and two, *all* modification of the granular synthesizers' parameters is through MultiSliderViews and Knobs for each specific parameter of filter frequency, threshold, duration, amplitude, attack, release and playback rate, while for *control* source three/*slave*, modification of the granular synthesizers' parameters is through a modified genetic algorithm *except* for the filter frequencies (this is to limit sudden changes in frequency which may generate clipping). In order to apply the modified

genetic algorithms to *control* source three, the values of the parameters from *control* source one and two are placed within the RedGA UGens for the consequent exploration of the defined values.

In terms of the MultiSliderViews and Knobs used for defining the parameters of each *slave* sound-object's granular synthesizers triggered by the onsets of *control* source one and two, these are mapped to specified bounds listed in *Figure 47*:

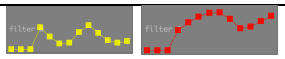
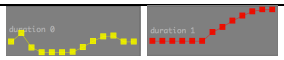
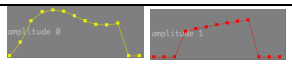
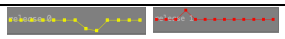
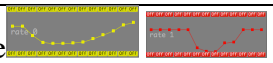
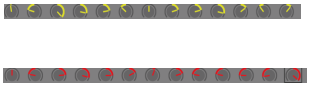
***Genesis* Parameter**	**Bounds and Mapping**
Band-Pass Filter 	Filter Frequency between 40 – 4000 Linear
Onset Threshold 	Threshold between 0 - 1 Linear
Grain Duration 	Time in seconds between 0.05 - 4 Linear
Grain Amplitude 	Amplitude between 0 - 1 Decibels
Grain Attack Time 	Time in seconds between 0.05 - 2 Linear
Grain Release Time 	Time in seconds between 0.05 - 2 Linear
Grain Playback Rate 	Values between -4 to 4 Linear
Grain Pan 	Values between -1 to 1 Linear

Figure 47. Mappings and Bounds of selected granular synthesizer GUI objects

The values present within the MultiSliderViews and Knobs for the *slave* source's controlled by the onsets of *control* source one and two define the initial data for the parameters of the modified GA for the granular synthesizers of the *slave* source controlled by *control* source three. In addition, the MFCC values of *control* source one and two are interpolated at the time of the relevant GA execution, thereby

creating a modified spectral shape for the application of the band-pass filters for the *slave 3* sound-object.

The modified GA interpolates the data provided by the MultiSliderViews, Knobs and MFCC values using an editable random mutation function (set relative to the bounds of the its respective parameter) with a changeable crossover determining the amount of data to be swapped between the two data sets of *control* source one and two. As a result, a single offspring is always generated based on the current status of the parameters of *control* source one and two that is immediately attributable to the granular synthesis parameters of the *slave*. Furthermore, should the latest offspring be deemed unsuitable, the data of all previous generations is automatically cached and can be saved at any time to a .txt file by clicking the 'Save' button for reloading via the 'Devolve' or 'Load' buttons respectively.

Figure 48 demonstrates the interpolating of the data from the parameters of the granular synthesizers triggered by *control* source one and two for consequent application to the parameter settings of the granular synthesizers triggered by *control* source three:

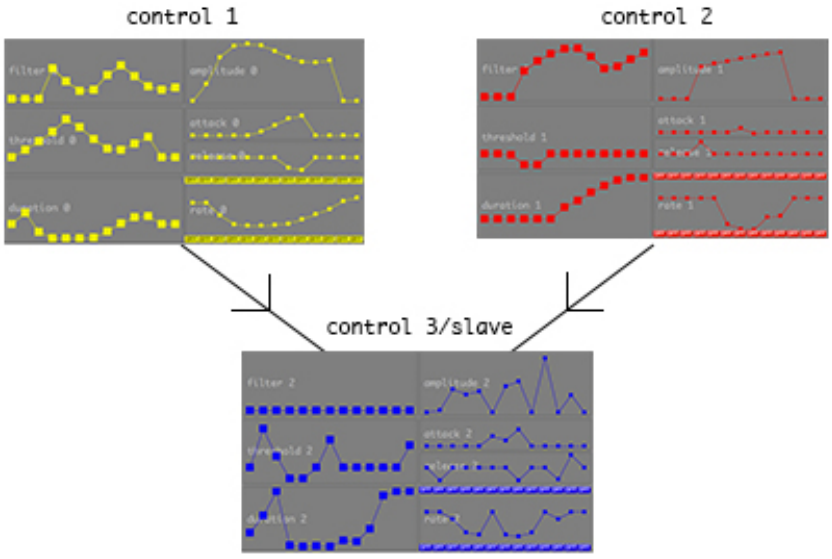


Figure 48. Gathering of data for the modified Genetic Algorithm

The modified genetic algorithms are controlled through Buttons within the UI as displayed in *Figure 49*:

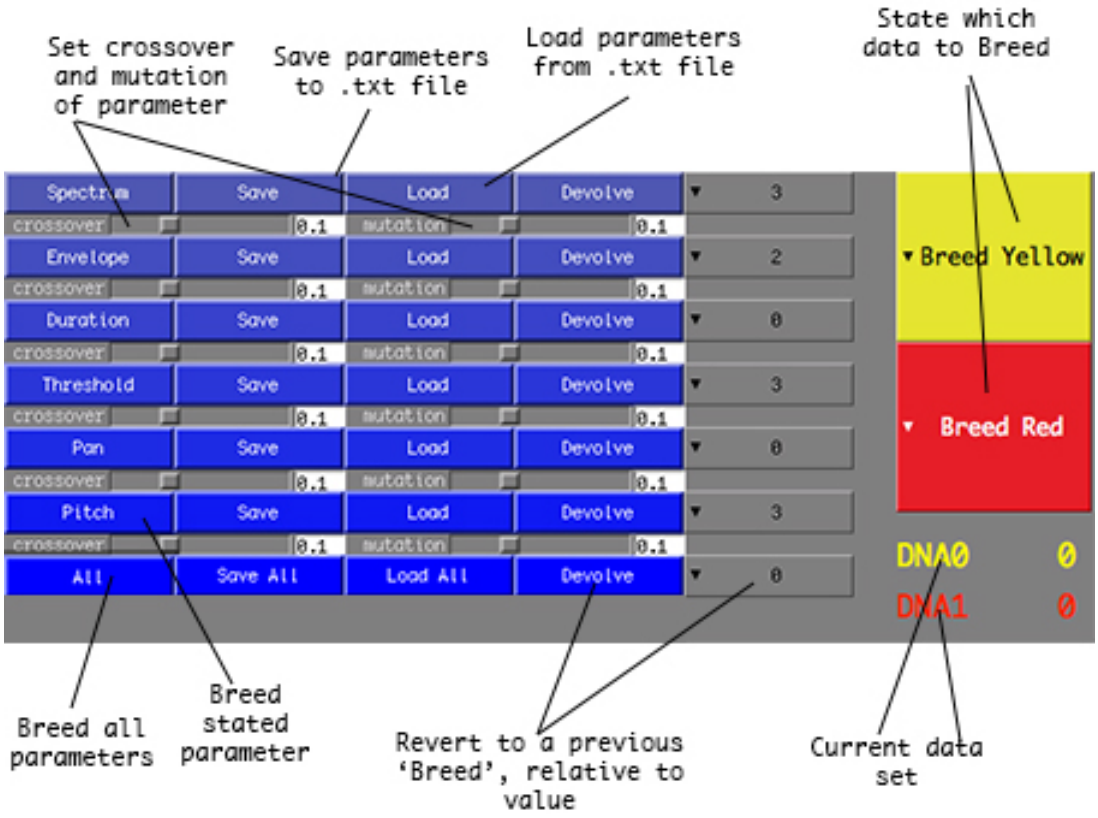


Figure 49. Modified Genetic Algorithm GUI controls

Once an initial data set has been obtained, it is then possible to interpolate different data sets by selecting which data set to breed through PopUpMenus within the GUI as demonstrated in *Figure 50*:

Figure 50. Population selection for Genetic Algorithms through GUI

As a result of the selection via the GUI of which data set to interpolate, the parameter to interpolate, the possibility to 'devolve' to previous states, modification of the crossover and mutation and the loading of previous parameter settings, the 'fitness function' of the modified genetic algorithm within *Genesis* is executed through the use of a human critique that must decide, control and evaluate each set of parameter data for the granular synthesizers whose triggers are controlled by *control* source three/*slave*.

With regards to the modification of the playback rates and recording rates, as stated previously, these can be controlled with the fractal noise process for each of the *slave* sound-object granular synthesisers, which can be toggled on or off within the GUI. This is illustrated in *Figure 51* for the granular synthesisers whose triggers are controlled by *control* source one: the MultiSliderView adjusts the playback/recording rate for each grain, the toggle Buttons above switch *on* or *off* the fractal process for each grain, and the toggle Buttons below switch *on* or *off* the adjustment of the recording rates for the *slave* sound-object's recording buffers of each grain relative to the playback rates:

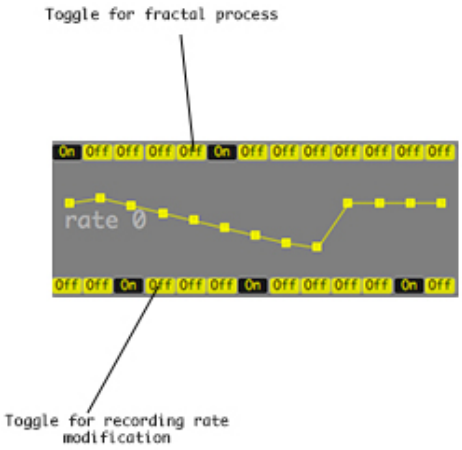


Figure 51. GUI control of Playback and Recording rates of granular synthesizers

In order to modify a number of granular synthesiser parameter settings simultaneously, a 0^{th}-order Markov chain can be toggled *on* via the 'Rand' button for *control* sources one and two using three sets of random arrays created relative to the current state of selected variables each set to different bounds of their respective

parameters of pan, attack, release, duration, rate, amplitude and threshold: set one features relatively small bounds, set two is slightly larger and set three being the largest with the larger the bounds, the greater the randomness of the results. The user can select the probability distribution of the Markov chain to increase the likelihood of assigning either set one, set two or set three to be the predominant random array applied. Furthermore, the random arrays are applied relative to the tempo of the respective *control* source, thereby generating new data sets at time specific intervals. As a result, the granular synthesisers of *control* sources one and two continue to change over time with minimal intervention from the user.

Considering the generative processes that modify the *slave* sound-object *outside* of the granular synthesisers, this centres on the real-time modification of the parameters of the Warp1.ar UGen, which reads an audio buffer and consequently permits time stretching through granular processes along with arbitrary buffer adjustments such as the buffer position and playback triggers. As described in section 5.1 *An Overview of Genesis*, the real-time audio signals can be live acoustic signals generated by instrumentalists, a pre-defined 'Sample' reader comprised of UGens reading buffered audio and live-coded *SynthDefs* defining specific synthesized sound-objects with their respective modulatable parameters. These real-time audio signals can be placed into one of three auditory input sources within *Genesis*, each of which features simple controls such as amplitude, relative to the formatting of the input source. In terms of the *slave* sound-object, the selected input source, regardless of its formatting, is placed in to a single audio buffer, which is then read by the Warp1.ar UGen enabling real-time modification of live streams.

The playback of the *slave* object can therefore be modified, manipulated and rearranged within *Genesis prior* to its recording for the granular synthesizer buffers. As a result of the parameter changes possible within the Warp1.ar UGen, the generative processes of tempo following, envelope following and random allocation of the values for the arbitrary parameters of the Warp1.ar UGen can be applied in real-time. In addition, the GUI also permits modification of the selected individual parameters as well as the toggling *on* or *off* of the various generative processes.

The tempo following method uses a principle equivalent to the pitch following process; the tempo of each *control* source is extracted and compared to the tempo of the *control* source three/*slave* sound-object. (The tempo extraction method is detailed further in section 5.5 *Analytical Processes in Genesis*). The difference is then calculated with the result placed into the Warp1.ar UGen's Phasor.ar control input, which results in the *slave* sound-object's tempo matching to a *control* source. With regards to the selection of which *control* source's tempo to apply, the GUI permits the selection of either *control* source one or two for modification of the *slave* sound-object's tempo.

The envelope following process applies the sum of the onsets for the *slave* sound-object every second; the fewer the number of onsets, the slower the envelope time, the greater the number of onsets, the faster the envelope time. The envelope itself is an Env.perc UGen, featuring an attack time, release time and peak amplitude parameter, which is wrapped within an EnvGen.kr UGen, defining the gate, thereby dictating the triggering of the envelope. This envelope is placed over the Warp1.ar UGen for the *slave* sound-object to control its amplitude over time.

In order to trigger the envelope, a variety of trigger methods can be selected through the GUI: manual, selected onsets of *control* source one, selected onsets of *control* source two and selected onsets of *control* source three. For manual operation, the composer must mouse click a toggle button within the GUI or the associated MIDI mapping to trigger the envelope. In contrast, a selected onset, relative to one of the thirteen MFCCs of each *control* source can be selected in the GUI, offering the triggering of the envelope via the real-time input audio sources. Due to the nature of the onset tracking process (as described in section 5.5 *Analytical Processes in Genesis*), onsets appear instantaneously, so in order to sustain an onset, its value is placed in addition to the perceived loudness of the *control* signal. Therefore, an onset is held until the loudness of the *control* signal has dropped below the specified threshold. As a result, the envelope's gate remains open relative to the loudness of the *control* signal and its selected onset, thereby sustaining the *slave* sound-object's auditory output for the duration.

In addition to triggering the envelope of the Warp1.ar UGen for the *slave* sound-object, the triggers that can be selected through the GUI also trigger the playback of the buffered audio. The playback position can be modified within the GUI for the pre-recorded audio files placed within the 'Sample' UGens through selection of the buffer frames in the GUI SoundFileView for the *slave* sound-object. However, for live acoustic sources and any live coded *SynthDef* whose auditory signals are buffered in real-time, it is not possible to present this data within a SoundFileView. So, the playback position can be defined within the GUI post window or via the real-time random generative process, which selects random values within the bounds of the buffer length.

The real-time random search process, which can randomly create values of the playback position relative to the *slave* sound-object's buffer's length, is a task that also can be used to randomly generate time stretching values for the Warp1.ar UGen of the *slave* sound-object (cannot be used in conjunction with tempo following), filter frequency values for a RHPF.ar UGen (high-pass resonant filter) / RLPF.ar UGen (low-pass resonant filter) for the *overall* auditory *Genesis* mix, reverb parameters for the GVerb.ar UGen for the *overall* auditory *Genesis* mix and pan parameters for two MonoGrain.ar UGens for the *overall* auditory *Genesis* mix. The time between new outputs can be adjusted in real-time, and is set to a default of 0.5 seconds, resulting in a new set of values every 0.5 seconds. Furthermore, each parameter adjustment can be toggled on or off via the GUI, along with the inclusion of the real-time random generative process itself.

Due to the nature of the real-time random value generative process and the possibility to trigger the Warp1.ar UGen with the onsets of the *control* sources, the buffer playback position can consistently change, relative to the time between outputs of the process. That is to say, each time an onset triggers the Warp1.ar UGen to playback, the corresponding value of the playback position defined by the real-time random value search process is applied, causing the consequent playback to begin from the stated value. As a result of the processes that modify the Warp1.ar UGen of the *slave* sound-object, the pitch, tempo, retriggering, and envelope of the *slave* sound-object

can be modified relative to the sonic features of the real-time auditory sources with its playback position generated randomly *prior* to its recording for the granular buffers.

Further to the real-time random value generative process, which can be applied to process values of the UGens modifying the *overall* auditory *Genesis* mix, the 'Call and Response' process is used to modify the *overall* auditory output mix of *Genesis*. The 'Call and Response' function can be toggled on or off in the GUI and relies on the real-time auditory signal from one of the *control* sources to dictate a 'Call'; this call must be an auditory signal with a loudness above a prescribed threshold, set at default to 15 phons. Once the loudness has fallen below the threshold, a wait time is added, which prevents the *Genesis* system from instantaneously creating a 'Response' whenever the loudness falls below the threshold.

While the 'Call's' auditory signal's perceived loudness is above the threshold, and within the wait time, *all* auditory signals in the audible *Genesis* mix are recorded to an audio buffer. If no further events occur above the threshold within the wait time, the recording to the buffer is stopped. The buffered audio recording is played back through a Warp1.ar UGen, forming a 'Response', with the values of the playback position, rhythm and pitch of the audio recording chosen through random selection of predefined rhythms, buffer frames and pitch structures relative to features identified within the 'Call'.

So, the number of onsets within the 'Call' signal defines the amount of playback position edits to the recorded buffer, thereby structuring the rhythm of the 'Response': the more edits, the shorter the rhythmic values, resulting in faster rhythms, the fewer edits, the longer the rhythmic values, resulting in slower rhythms. The speeds at which the rhythms are played back are also relative to the tempo of the 'Call', adjusting the tempo of the defined rhythms. In terms of pitch, the 'Call's' last note's pitch defines the fundamental pitch value of the recorded audio, through which all consequent modification of the recordings pitch by the generative process will be made relative to. The duration of the 'Response' itself is decided in relation to the duration of the recorded audio buffer. Furthermore, within the GUI, a number of alterations can be made to adjust the performance in real-time of the 'Response'; the

wait time, *control* source, pitch structure and time signature can be modified relative to the requirements of the composition.

With regards to the pitch fixer process which can be toggled on or off within the GUI, the current pitch of the *slave* sound-object can be fixed to a particular pitch structure such as C Major scale. As a result, if applying the C Major scale, a Bb will be corrected to either a B or a C within its perceived octave. The process requires the pitch tracking of the overall *Genesis* mix, for comparison with the defined pitch structure. The difference between the two values is then assessed and any adjustment required is generated in real-time and placed within a PitchShift.ar UGen's 'pitch shift' parameter that passes the *slave* sound-object's auditory signal. In order to modify the available pitch structures, PopUpMenus are placed within the GUI to select predefined structures, which can be adjusted within the *Genesis* programming code.

In terms of the application of live coding within *Genesis*, as stated, *SynthDefs* can be coded within the post window and passed into a selected *control* source, forming the live-coded *SynthDef* input source. In order to pass the live-coded *SynthDef* through *Genesis*, the *SynthDef's* Out.ar UGen's *Bus* parameter, which designates the audio bus to send the auditory signal/s of the *SynthDef* to, *must* match one of the three pre-allocated audio busses assigned for the live-coded *SynthDef* input sources, titled ~synthBus0 for *control* source one, ~synthBus1 for *control* source two and ~synthBus2 for *control* source three/*slave*. As a result, *any* auditory output generated by a live-coded *SynthDef* can be used as a *control* or *slave* source within *Genesis*. Furthermore, as highlighted with the modification of the buffer position for a *slave* sound-object formed of a live acoustic signal or live-coded *SynthDef*'s audio signal, this can be adjusted through the post window with the relative live coding. This is also true for *all* pre-defined parameters within *Genesis*, offering the composer the capability to generate patterns and tasks for the control of *Genesis* within the post window through live coding.

Many of the pre-defined parameters within *Genesis* are of course modifiable through the various GUI objects present in the *Genesis* GUI, in addition to the ability of

altering the parameters through live coding. However, live coding is applied to the output of each GUI object that modifies the GUI controlled the parameters within *Genesis*. So, the parameters changes specified by the respective GUI object are written as a string within a hidden post window (this functionality can be toggled *on* or *off* within the GUI). These changes can then be wrapped as a task and consequently executed through GUI. In order to allow dynamic modification of the parameters, a clock can be applied, which can be controlled and synced via the network. The values of the clock are placed alongside the string output from the GUI object, with the task applying *if* statements to permit the modification of the respective *Genesis* parameter *if* the current clock value is equal to the clock value defined in the task.

To exemplify the method of GUI live coding within *Genesis*, if the clock is running and the GUI live coding functionality is toggled *on*, modifications within the GUI are written as a string in the hidden post window, along with the current value of the clock. Once the modifications have been made, the task can be allocated a name through a GUI TextField and consequently loaded to a GUI Window offering the capability to hold sixteen such tasks. If the loaded task is triggered within the GUI 'Routine Controls' window, the modifications will be made relative to the values of the *Genesis* clock. These tasks can be saved to disk as a .txt file and loaded in future compositions. The method of GUI live coding within *Genesis* is further illustrated in *Figure 52*:

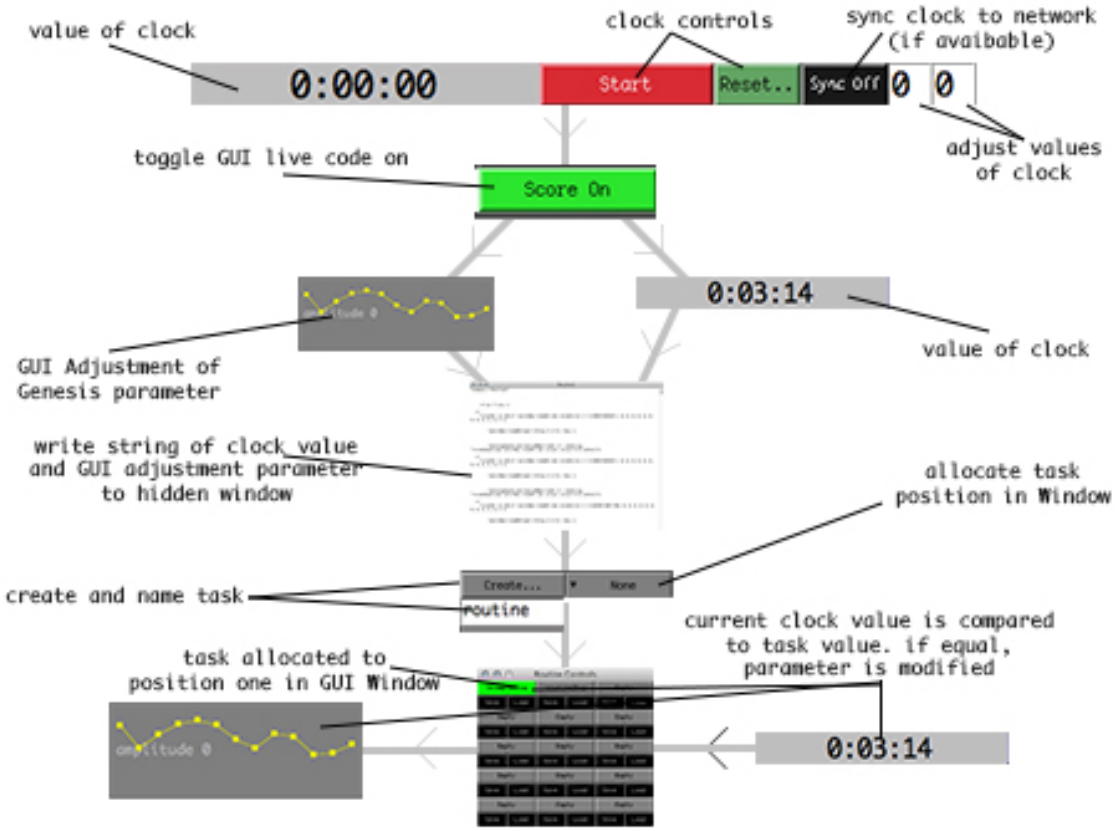


Figure 52. GUI Live Coding Method

With the method of GUI live coding within *Genesis*, as described above, it is possible to generate tasks in real-time through the alterations made within the GUI, which can consequently be applied in real-time via the GUI control of the newly created task; the wrapping, naming and creation of a task are executed in real-time, thereby allowing its immediate application to a compositional process through its execution in the GUI Window.

The live-sampling process offers the composer the option of generating sound-objects formed of the real-time inputs' auditory signals, thereby permitting the consequent control of a sound-object's pitch and temporal structure through the Sample UGens. As a result, the newly generated audio recording can be applied to form a *control* and/or *slave* sound-object; the real-time input source's auditory signal is saved to the computer's hard disk allowing it to be immediately placed in to a selected *control* source's Sample UGens. The duration of the recording, its *control* source destination and its triggering are defined within the GUI for the local system, along with the

capability to control the process on any networked system via a network control window.

Therefore, generative processes that control *Genesis* are applied through the symbolic or subsymbolic representation of the sonic features of real-time auditory signals, fractal noise processes, search processes, GUI object modification, live coding, and a unique GUI live coding approach, written specifically for *Genesis*. The combination of pre-defined generative processes and live coding results is a system that offers the composer the capability to apply many conceivable real-time generative processes (such as the three purposes proposed by Supper (2001) of generative processes for modeling traditional, non-algorithmic procedures, modelling new original compositional procedures and selecting algorithms from extra-musical disciplines), for the control of the pre-defined parameters within *Genesis* relative to the UGens and classes presented within the *Genesis* system itself as well as the many SuperCollider classes that are contained within the *Genesis* application's package. In addition, the generative processes that dictate the auditory signals within *Genesis* can also be applied in real-time through the use of live coded *SynthDefs* or live acoustic signals as input sources, which can be live-sampled and placed within the system's Sample UGens. This permits the composer to generate or apply, in real-time, many feasible sound-objects for the *control* or *slave* sound-objects within *Genesis*.

5.5 Analytical Processes in *Genesis*

All analytical processes within *Genesis* apply values obtained from the Fast Fourier Transform (FFT) of the real-time input sources with the outputs of the FFT analyzed relative to the defined analytical process. The results of the analytical processes are then used within particular generative processes (as described in section 5.4 *Generative Processes in Genesis*) and/or represented within the GUI. As stated in section 5.1 *An Overview of the Genesis System*, each real-time input source's auditory signals are represented equally, irrespective of their input source type, be it a live acoustic signal, the pre-defined 'Sample' reader UGens or a live-coded *SynthDef*'s auditory output. As a result, the sonic features of pitch, onset, amplitude, tempo and

MFCCs are extracted from each source, ready for application to the various analytical processes, with the analytical processes themselves defining the importance and role of an individual source's sonic feature/s, relative to the analytical process's required output.

The role of Mel-frequency cepstral coefficients (MFCCs) is highly prevalent in the control of the onsets for the granular synthesizers of the slave sound-object (as detailed in 5.4 *Generative Processes in Genesis*) and the dynamic spectral following process. In order to extract the MFCCs (as described in chapter 3.2.3 *Timbre Perception*), each *control* input source is assigned an FFT chain with a buffer size of 1024 frames and a sine window, which is consequently read by the MFCC.kr UGen. The MFCC.kr UGen offers a parameter to define the number of coefficients, which is selected to thirteen. This results in thirty-nine MFCC coefficients in total for the three input sources. In consideration of the processing limitations, GUI scale and feasibility of interaction, thirty-nine MFCCs triggering the onsets of thirty-nine granular synthesisers and associated band-pass filters creates a manageable environment successfully demonstrates the functionality of the fundamental principle of *Genesis* to use the sonic features of real-time audio signals for modification, manipulation and arrangement of real-time sound-objects.

The values of the MFCCs are 'somewhat'[229] normalized by the MFCC.kr UGen within a range of around 0.0 to 1.0, which helps to restrict anomalous values from occurring. This serves to create a more stable set of values for the consequent mapping of the MFCCs to the generative processes within *Genesis*, which, in terms of the dynamic spectral following process, can prevent clipping in the auditory signal. In addition, due to the increased stability of the MFCC values, their representation within the GUI can be mapped relatively simply to the values between 0.0 and 1.0, limiting the possibility that their values will be misrepresented within the allocated GUI MultiSliderViews for each *control* source.

The onsets of the real-time input sources, which trigger the grain trigger parameter and the envelope of the granular synthesisers, along with optional re-triggering of the

[229] MFCC.kr UGen Class Help File. 2012. SuperCollider Version 3.

Warp1.ar UGen of the *slave* sound-object and the gates of the fractal processes' outputs, are extracted by applying the Onsets.kr UGen *after* the band-pass filters of the *control* sources whose filter frequencies are controlled by the static or dynamic methods as described in section 5.4 *Generative Processes in Genesis*. The Onsets.kr UGen detects the onset of sonic events defined by the power of the auditory outputs of the band-pass filters using a control signal of 0 or 1 to indicate the onset state: 0 being no onset detected, 1 being an onset detected.

As with the MFCC FFT chain, a buffer size of 1024 frames and a sine window are applied to ensure a high temporal resolution, necessary for an onset task which represents the change in amplitude events over time; the more temporal data, the more onsets can be detected. The Onsets.kr UGen itself features various parameters of which all but the threshold and onset detection function are set to default; the threshold is modifiable within the GUI, with the onset detection function set to \phase, which is 'generally good, especially for tonal input, medium efficiency'[230]. With regards to the onset detection function, this is applied in consideration of the use of tonal sound-objects with the *Genesis* system.

The onsets are represented within the GUI through the use of GUI Buttons which toggle *on* or *off* relative to the value of its corresponding Onsets.kr UGen's control signal output: *off* is gray, *on* is coloured relative to the *control* source. This allows the composer to view which processes are triggered by which onsets of the *control* source, simplifying consequent modification of the generative process controlled by the relevant onsets of the *control* sources. Furthermore, the capability to visualize the detection of onsets aids the process of the threshold adjustment within the GUI. For example, if the *overall* amplitude of a granular synthesizer is set to 0, the triggering of the grain and its envelope will not be heard. However, as the onsets of the *control* sources, which triggers these parameters, can be observed visually, it is possible to adjust the threshold, relative to the state of the onset GUI buttons, in preparation for a consequent increase in the granular synthesizer's *overall* amplitude.

[230] Onsets.kr UGen Class Help File. 2012. SuperCollider Version 3.

The pitch of the real-time input sources and the overall mix are extracted through the Pitch.kr UGen. This applies an autocorrelation function to obtain periodicity within the signal, adapted from the temporal coding model proposed by Licklider (1951). The result is output as a frequency value, relative to the fundamental frequency defined by the Pitch.kr UGen. For the purposes of *Genesis*, the default general settings of the Pitch.kr UGen have been applied due to the variance of sound-object types that can be applied to the *Genesis* system.

The pitch is represented within the GUI through a UserView and TextField, relative to the analytical process. For the pitch of the real-time input sources, UserViews are applied. The frequency output by the Pitch.kr UGen of the relative source is mapped to the UserView's drawFunc method, with the frequency dictating the Pen class's addWedge radius parameter; the higher the frequency, the longer the radius, the lower the frequency, the shorter the radius. However, for the pitch of the *slave* sound-object and its application to the pitch fixing process detailed in section 5.4 *Generative Processes in Genesis*, the pitch is extracted *prior* to modification and *after* modification in order to represent the occurrence of any modification by the process within the GUI.

The UserView method is applied in combination with two TextFields each using the cpsmidi.midname methods, stating Pitch.kr UGen's frequency as MIDI Note Names. The pitch *prior* to any modification of the pitch fixing process is displayed through one UserView (in yellow) using the drawFunc method described previously, with the this frequency also represented as a MIDI Note Name in a TextField. The relative difference between the pitches *prior* to any modification and the defined pitch structure is represented in a UserView (in red) placed atop of the overall mix's *prior* pitch UserView, with higher levels of difference increasing the radius of the red UserView. The resulting pitch of the process is displayed in another TextField, visually representing the modification made by the pitch fixing process. This process is illustrated in *Figure 53*:

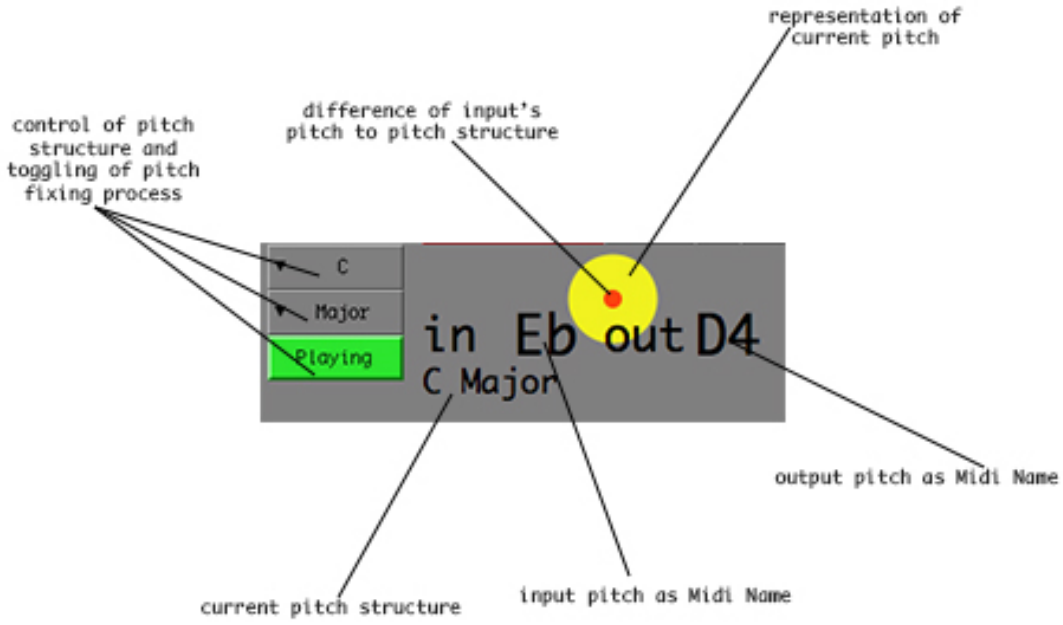


Figure 53. GUI display of pitch fixing process

The loudness of the real-time input sources, which is applied to measure the duration of the 'Call' for the 'Call and Response' function and sustain of the enveloping process for the Warp1.ar UGen of the *slave* sound-object uses the Loudness.kr UGen to represent the perceived loudness of the real-time input signals in phons.

The output of the Loundess.kr UGen for each real-time input source is represented in the GUI through the use of Buttons, which toggle on or off relative to the value of the output, similar to the Buttons applied to represent the values output by the Onsets.kr UGens. However, the loudness Buttons only toggle on if the value of the respective Loundess.kr UGen is above a defined threshold, set to a default of 15 phons, matching the threshold at which the envelope of the *slave* sound-object's Warp1.ar is sustained.

The extraction of an input source's tempo is measured through the BeatTrack.kr UGen. As with the Pitch.kr UGen, an autocorrelation function is applied to identify periodicity within the signal. The BeatTrack.kr UGen 'determines the beat, biased to the midtempo range by weighting functions. It does not determine the measure level,

only a tactus'[231]. So, the outputs of the BeatTrack.kr UGen are weighted towards tempos of between 100-120bpm and are relative to the signal's onsets divided between the rhythmic values of crotchet, quaver and semiquaver. As a result, its application is more suited for sound-objects that feature distinct and distinguishable rhythmic onsets that can be classified in to crotchets, quavers and semiquavers as well as being of midtempo range. The tempo output of the BeatTrack.kr UGen represents the duration of the identified crotchet onsets and must be multiplied by 60 in order to obtain a value of beats per minute. The tempo output by the BeatTrack.kr UGen is applied within *Genesis* to the tempo following process and the 0-th order Markov chain execution described in section 5.4 *Generative Processes in Genesis* and displayed within the GUI as its beats-per-minute (bpm) value for each of the real-time input sources.

As demonstrated in chapter 3.2 *A Brief Summary of Machine Listening*, models describing the perceptual processes which form the values of perceived sonic features cannot be unequivocally defined. Therefore, the application of such models within computational processes reflects the idiosyncrasies and limitations of the models of sound perception. With regards to the analytical processes that are applied in *Genesis*, it is possible to observe the constraints of the UGen classes in the consequent application of the outputs forming what may be perceived as glitches in the auditory signals and their representation within the GUI.

Most notably, the pitch following and tempo following processes highlight the limitations of the Pitch.kr and BeatTrack.kr UGens. For example, the functionality of pitch tracking through the Pitch.kr UGen decreases significantly if the periodicity of the input source's wave becomes less discernable. This can be caused by two key factors: high levels of noise in the input source and/or the occurrence of polyphony, which both cause interference masking of the signal's frequency components. The consequent application of the Pitch.kr UGen's output, if high levels of interference masking are present, may cause the pitch following process to compare inaccurately the pitches of the *control* source and the *slave* source, relative to the pitch perceived by a human.

[231] BeatTrack.kr UGen Class Help Files. 2012. SuperCollider 3.5.3

In terms of the tempo following process, the BeatTrack.kr UGen struggles to identify tempos outside of the mid tempo range due to its weighting bias towards 100 - 120 bpm. In addition, the BeatTrack.kr UGen requires loud, fast attacking onsets to define the relative tactus of the real-time input signal. As a result, the tempos applied to the tempo following process are not necessarily truly representative of the perceived tempo of the input source, causing the output of the process to generate an audible signal that noticeably disproportions the tempos of the relative *control* source and the *slave* source; the resulting tempo of the *slave* source does not match the *control* source.

Despite the limitations of the analytical UGens as described above, the constraints of the analytical processes form a system which influences the compositional output of *Genesis*; acknowledgment and awareness by the composer of these system's characteristics and their consequent behaviours relative to the real-time input source's sound-object's frequency components may generate unique methods of sound-object control, distinctive of *Genesis* and the UGens that form its analytical suppositions.

For example, the resulting output of a pitch following process with a *control* signal that features a significant level of interference masking may produce a pitch output, which generates a novel pitch contour, relative to the pitch frequencies identified by the Pitch.kr UGen. As a result, the output of the process becomes an representation of *Genesis*'s interpretation of a real-time input source's pitch. Furthermore, this principle can of course be applied to *all* other analytical processes and their consequent application to generative processes; the compositional outputs of the *Genesis* system, which apply the sonic features obtained from the real-time auditory signals to generate modifications to its auditory outputs, reflect its UGens' perception of the sonic features applied relative to the generative process.

5.6.1 Granular Synthesis control

NOTE *For clarity and ease-of-reading, the computer code's associated mappings and UGens for* ***one*** *of the thirty-nine granular synthesisers are represented in the following (unless otherwise stated). In execution, thirty-nine instances are run in real-time of each code example relating to the granular synthesisers.*

5.6.1.1 Static Onsets of Control Sources Triggering Onsets of Granular Synthesizers

Audiovisual example on the DVD *3. Local Static Onsets.mov* in the *Audiovisual Examples* folder demonstrates the control of the static onsets for the granular synthesizers dictated by *control* source one. The real-time audio input from a piano keyboard forms *control* source one and *control* source three/*slave* sound-object. Only *slave* output audible.

The following code represents the process for triggering the onsets and envelopes of the granular synthesizers controlled by *control* source one.

Control source one filtered by bank of band-pass filters with static filter frequencies, relative to those defined in GUI:

```
osc0Filter0 = BBandPass.ar(osc0, Select.kr(feedback0[46], [osc0Freq0, Select.kr(feedback0[60],
[filterData0[0], networkFilters0[0]])]), feedback0[57]);
```

Onsets of *control* source one analysed by a bank Onsets.kr UGens, with thresholds defined in GUI:

```
osc0Onset0 = Onsets.kr(osc0OnsetChain0, threshold0Array0[0], \phase);
```

Position of buffer for each granular synthesizer defined by fractal noise process triggered by output onsets:

```
position0 = Gate.kr(PinkNoise.kr(~warpBuffer0[0].numFrames)/(~warpBuffer0[0].numFrames),
Select.kr(networkChooser0,[onsets0[0], networkOnsets0[0]]));
```

Outputs of bank of Onsets.kr UGens trigger granular synthesizers reading buffered audio recording of *slave* sound-object:

```
onset0GrainMacro0 = GrainBuf.ar(1, Select.kr(networkChooser0,[onsets0[0], networkOnsets0[0]]), duration0,
~warpBuffer0[0].bufnum, BufRateScale.kr(~warpBuffer0[0].bufnum) * pitchChooser0, position0,
interpolation) * envelopes0[0];
```

Outputs of bank of Onsets.kr UGens trigger envelopes of granular synthesizers with envelope times adjusted by GUI:

```
onset0Envelope0 = Env.perc(grain0Attack0, grain0Release0, grain0Amplitude0, 'sine');
onset0Env0 = EnvGen.kr(onset0Envelope0, Select.kr(networkChooser0,[onsets0[0], networkOnsets0[0]]), 1);
```

5.6.1.2 Dynamic Onsets of Control Sources Triggering Onsets of Granular Synthesizers

Audiovisual example on the DVD *4. Local Dynamic Onsets.mov* in the *Audiovisual Examples* folder demonstrates the control of the dynamic onsets for the granular synthesizers dictated by *control* source one. The real-time audio input from a piano keyboard forms *control* source one and *control* source three/*slave* sound-object. Only *slave* output audible.

The following code represents the process for the onsets of the granular synthesizers controlled by *control* source one.

Control source one's MFCCs extracted:

```
mfccAnalysis0 = FFT(~mfccBuffer0, osc0Array1);
mfcc0 = MFCC.kr(mfccAnalysis0);
```

Control source one filtered by bank of band-pass filters with dynamic filter frequencies, relative to those defined by MFCCs of *control* source one:

```
osc0Filter0 = BBandPass.ar(osc0, Select.kr(feedback0[46], [osc0Freq0, Select.kr(feedback0[60],
```

```
[filterData0[0], networkFilters0[0]])]), feedback0[57]);
```

Onsets of *control* source one analysed by a bank Onsets.kr UGens, with thresholds defined in GUI:

```
osc0Onset0 = Onsets.kr(osc0OnsetChain0, threshold0Array0[0], \phase);
```

Position of buffer for each granular synthesizer defined by the fractal noise process triggered by output onsets:

```
position0 = Gate.kr(PinkNoise.kr(~warpBuffer0[0].numFrames)/(~warpBuffer0[0].numFrames),
Select.kr(networkChooser0,[onsets0[0], networkOnsets0[0]]));
```

Outputs of bank of Onsets.kr UGens trigger granular synthesizers reading buffered audio recording of *slave* sound-object:

```
onset0GrainMacro0 = GrainBuf.ar(1, Select.kr(networkChooser0,[onsets0[0], networkOnsets0[0]]), duration0,
~warpBuffer0[0].bufnum, BufRateScale.kr(~warpBuffer0[0].bufnum) * pitchChooser0, position0,
interpolation) * envelopes0[0];
```

Outputs of bank of Onsets.kr UGens trigger envelopes of granular synthesizers with envelope times adjusted by GUI:

```
onset0Envelope0 = Env.perc(grain0Attack0, grain0Release0, grain0Amplitude0, 'sine');
onset0Env0 = EnvGen.kr(onset0Envelope0, Select.kr(networkChooser0,[onsets0[0], networkOnsets0[0]]), 1);
```

5.6.1.3 Genetic Algorithm Modification of Granular Synthesizers' Parameters

Audiovisual example on the DVD *9. Local GAs.mov* in the *Audiovisual Examples* folder demonstrates the control of the parameters for the granular synthesizers' whose onsets are dictated by *control* source three through the genetic algorithms executed in the GUI. The real-time audio input from the Sample UGens forms *control* source one, *control* source two and *control* source three/*slave*. Initially, the audible outputs of the *control* sources are played, followed by the granular synthesizers that are controlled by their onsets.

The following code represents the process for controlling the parameter of the granular synthesizers of *control* source three via genetic algorithms.

The parameters of the granular synthesizers whose onsets are dictated by *control* one and two are collected from control busses, such as MFCCs:

```
~mfccBus0.getn(13, {arg value; {mfccData0 = value;}.defer});
```

Values collected from busses are placed in RedGA classes and executed via GUI Buttons with the crossover and mutation modified by GUI EZSliders. The resulting outputs are allocated to the relevant parameter. The following code represents the process for MFCC data:

```
//MFCC
//Crossover EZSlider
~crossoverSlider0[0] = EZSlider(~w5, Rect(300, 25, 233, 12.5), "crossover", ~volumeSpec0, unitWidth:30,
initVal:0.1,numberWidth:30, layout:\horz);
~crossoverSlider0[0].font_(Font("Monaco", 10));
~crossoverSlider0[0].setColors(Color.clear,Color.white.alpha_(0.7));

~crossoverSlider0[0].action_({ |ez|
crossover0 = ez.value;
});

//Mutation EZSlider
~mutationSlider0[0] = EZSlider(~w5, Rect(500, 25, 233, 12.5), "mutation",       ~volumeSpec0,
unitWidth:30, numberWidth:30, initVal:0.1,layout:\horz);
~mutationSlider0[0].font_(Font("Monaco", 10));
~mutationSlider0[0].setColors(Color.clear,Color.white.alpha_(0.7));
~mutationSlider0[0].action_({|ez|
mutation0 = ez.value;
});

//Execute algorithm
~algorithmButton0[0] = Button(~w5, Rect(300, 0, 100, 25));
~algorithmButton0[0].states_([["Spectrum", Color.white, Color.blue.alpha_(0.4)]]);
~algorithmButton0[0].font_(Font("Monaco", 10));
~algorithmButton0[0].action_({|butt|
if(butt.value == 0,
{
RedGA.mutationFunc = {rrand(0, 2000);};
mfccGenomeA = RedGAGenome.new(~mfccDataSelector0);
mfccGenomeB = RedGAGenome.new(~mfccDataSelector1);
RedGA.crossOverRate = crossover0;
mfccCrossover = RedGA.breedMultiPoint(mfccGenomeA, mfccGenomeB);
mfccCrossover.do{|x| x.chromosome};
RedGA.mutationRate = mutation0;
mfccBreed = RedGA.mutate(mfccCrossover[0]).chromosome;
~trackerSynth2.set(\mfccFeedback, mfccBreed);
~mfccArrayOut.addFirst(mfccBreed.max(0.005));
});
});
```

5.6.1.4 Fractal Noise Modification of Granular Synthesizers' Playback Rate

Audiovisual example on the DVD *10. Fractal Static.mov* in the *Audiovisual Examples* folder demonstrates the control of the playback rate for the granular synthesizers' whose static onsets are dictated by *control* source one and *control* source two. The real-time audio input from the Sample UGens forms *control* source one, *control* source two and *control* source three/*slave*. Initially, the audible outputs of the *control* sources are played, followed by the granular synthesizers that are controlled by their onsets. The recording rate of the granular synthesizers whose onsets are controlled by *control* source one are also modified by the process.

Audiovisual example on the DVD *11. Fractal Dynamic.mov* demonstrates the control of the playback rate for the granular synthesizers' whose dynamic onsets are dictated by *control* source one and *control* source two. The real-time audio input from the Sample UGens forms *control* source one, *control* source two and *control* source three/*slave*. Initially, the audible outputs of the *control* sources are played, followed by the granular synthesizers that are controlled by their onsets. The recording rate of the granular synthesizers whose onsets are controlled by *control* source one are also modified by the process.

The following code represents the process of controlling the pitches for the granular synthesizers dictated by the onsets of *control* source one via fractal noise values.

Control source one filtered by bank of band-pass filters with static or dynamic filter frequencies:

```
osc0Filter0 = BBandPass.ar(osc0, Select.kr(feedback0[46], [osc0Freq0, Select.kr(feedback0[60], [filterData0[0], networkFilters0[0]])]), feedback0[57]);
```

Onsets of *control* source one analysed by a bank Onsets.kr UGens, with thresholds defined in GUI:

```
osc0Onset0 = Onsets.kr(osc0OnsetChain0, threshold0Array0[0], \phase);
```

Outputs of bank of Onsets.kr UGens trigger fractal noise outputs of playback rate PinkNoise.kr UGens:

```
pitchChooser0 = Gate.kr(Select.kr(fractal0Grain0, [grain0Pitch0, grain0Pitch0 * PinkNoise.kr(4)]),
Select.kr(networkChooser0,[onsets0[0], networkOnsets0[0]]));
```

Outputs of fractal noise playback rate optionally routed to the recording rate of the granular synthesizers' audio buffers:

```
bufferFilterHi0 = BufWr.ar(osc2, ~warpBuffer0[0],  Phasor.ar(0, BufRateScale.kr(~warpBuffer0[0].bufnum) *
Select.kr(naturalChooser0, [1, rates[0]]), 0, BufFrames.kr(~warpBuffer0[0])));
```

Outputs of bank of PinkNoise.kr UGens define rate of granular synthesizers:

```
onset0GrainMacro0 = GrainBuf.ar(1, Select.kr(networkChooser0,[onsets0[0], networkOnsets0[0]]), duration0,
~warpBuffer0[0].bufnum, BufRateScale.kr(~warpBuffer0[0].bufnum) * pitchChooser0, position0,
interpolation) * envelopes0[0];
```

5.6.1.5 Spectral Following of Control Source for Application to Each Granular Synthesizer's Filter Frequencies

Audiovisual example on the DVD *12. Local Spectral Follwing.mov* in the *Audiovisual Examples* folder demonstrates the control of the filter frequencies for the granular synthesizers mapped to the MFCC values of *control* source one and *control* source two. The real-time audio input from the Sample UGens forms *control* source one, *control* source two and *control* source three/*slave*. Initially, the audible outputs of the *control* sources are played, followed by the granular synthesizers that are controlled by their onsets.

The following code represents the process for the spectral following of *control* source one's MFCC values for application to the granular synthesizers controlled by *control* source one.

Control source one's MFCCs extracted:

```
mfccAnalysis0 = FFT(~mfccBuffer0, osc0Array1);
mfcc0 = MFCC.kr(mfccAnalysis0);
```

Control source one filtered by bank of band-pass filters with dynamic filter frequencies, relative to those defined by MFCCs of *control* source one:

```
osc0Filter0 = BBandPass.ar(osc0, Select.kr(feedback0[46], [osc0Freq0, Select.kr(feedback0[60],
[filterData0[0], networkFilters0[0]])]), feedback0[57]);
```

MFCCs of *control* source one mapped to filter frequency values:

```
filterControlOut0 = Out.kr(~filterTracker0, [
((multiplier0*(mfccData0.sum**3)**1)+(feedback0[51])).min(20001).max(19),
}).send(s);
```

Output values of mappings are checked for values above 4000Hz and below 20Hz, with those values resulting in a granular synthesizer with an amplitude of 0:

```
~filterCutterRoutine = Routine.new({
inf.do({ arg i;

if ( (~filterCutter0[0] >4000) || (~filterCutter0[0] <20),

          {~granularMacroSynth0.set(\amplitude0, 0);
          {~filterTrackerSlider0[0].setColors(Color.grey,Color.white.alpha_(0), Color.white.alpha_(0)
          ,Color.grey,Color.white.alpha_(0), Color.yellow,nil, Color.yellow.alpha_(0.8),
          Color.white.alpha_(0));}.defer; ~filterCutterGUI0[0] = 0;},

          {~granularMacroSynth0.set(\amplitude0, 0.5);
          {~filterTrackerSlider0[0].setColors(Color.grey,Color.white.alpha_(0),
          Color.yellow.alpha_(0.5),Color.grey,Color.white.alpha_(0), Color.yellow,nil,
          Color.yellow.alpha_(0.8), Color.white.alpha_(0));}.defer; ~filterCutterGUI0[0] = 1;}
);
);
```

Resulting outputs define filter frequency of each granular synthesizer's filter frequency:

```
osc0Filter0 = BBandPass.ar(onset0GrainMacro0, Select.kr(feedback0[46], [osc0Freq0, Select.kr(feedback0[60],
[filterData0[0], networkFilters0[0]])]), feedback0[57]);
```

5.6.1.6 Markov Chain manipulation of Granular Synthesiser Parameters

Audiovisual example on the DVD *23. Markov Chain.mov* in the *Audiovisual Examples* folder demonstrates the Markov chain control of random arrays created

relative to current data for the progressive development of the granular synthesiser parameter settings of amplitude, rate, pan, duration, threshold, attack and release for control source *one* and *two*. The outputs of the granular synthesisers controlled by control source *one* and *two* are audible.

Create an initial random array based on current state for the parameters of amplitude, rate, pan, duration, threshold, attack and release with the option to set 'small changes more likely', 'medium changes more likely, and 'large changes more likely'.

```
//Create difference Array and PopUp Menu Array Selector
          ~differenceArray0 = [0.7, 0.2, 0.1];

          ~differenceButton0 = PopUpMenu(~w7, Rect(1920/3.25, 15, (1920/3)/20, 15));
          ~differenceButton0.items = ["small change more likely", "med change more likely", "large change more
          likely"];

          ~differenceButton0.value = 0;

          ~differenceButton0.action = {arg menu;

          if(menu.value == 0,
          {
          ~differenceArray0 = [0.7, 0.2, 0.1];
          }
          );

          if(menu.value == 1,
          {
          ~differenceArray0 = [0.1, 0.7, 0.2];
          }
          );

          if(menu.value == 2,
          {
          ~differenceArray0 = [0.1, 0.2, 0.7];
          }
          );

          };

//Create data arrays for each feature (Pan shown here)

          ~randPanArray0 = [rrand(-0.50,0.50), rrand(-0.50,0.50), rrand(-0.50,0.50), rrand(-0.50,0.50), rrand(-
          0.50,0.50), rrand(-0.50,0.50), rrand(-0.50,0.50), rrand(-0.50,0.50), rrand(-0.50,0.50), rrand(-
          0.50,0.50), rrand(-0.50,0.50), rrand(-0.50,0.50), rrand(-0.50,0.50)];

          ~randPanDiff0 = pan0Data0 - ~randPanArray0;
```

Start a routine, with intervals set relative to the perceived tempo, ensuring changes are applicable to modification. If not, generate a new random array. This limits extraneous variables.

```
//random routine
        ~randRoutine0 = Routine.new({
         inf.do({ arg i;

//ensure valid values (Only Pan shown here)
        ~checkRandPanDiff0 = Array.fill(13, {arg i;
          if(~randPanDiff0[i] <= -1,
          {
          ~randPanDiff0[i] = rrand(-0.50, 0.50);
          }
          );
          if(~randPanDiff0[i] >= 1,
          {
          ~randPanDiff0[i] = rrand(-0.50, 0.50);
          }
          );
          });
```

```
        });

//send values to synths and GUI (Only Pan shown here)

        {
        ~panners0Array = Array.fill(13, {arg i;
        ~panKnob0[i].valueAction = \pan.asSpec.unmap(~randPanDiff0[i]);
        };
        );

        }.defer;


//create a selected difference array (Small Change presented here)
        ~smallChange0 = [rrand(-0.1, 0.1), rrand(-0.1, 0.1), rrand(-0.1, 0.1), rrand(-0.1, 0.1), rrand(-0.1,
        0.1), rrand(-0.1, 0.1), rrand(-0.1, 0.1), rrand(-0.1, 0.1), rrand(-0.1, 0.1), rrand(-0.1, 0.1), rrand(-
        0.1, 0.1), rrand(-0.1, 0.1), rrand(-0.1, 0.1)];
```

Add the newly created arrays to the current data and then send it to the granular synthesisers controlled by control source *one* and *two*, relative to the current probability distribution setting.

```
//wchoose amount of change (Only Pan shown here)
        ~randPanDiff0 = [~randPanDiff0 + ~smallChange0, ~randPanDiff0 + ~medChange0, ~randPanDiff0 +
        ~largeChange0].wchoose(~differenceArray0);

//set duration between each change relative to control source one tempo
        ~randDurationChooser0 = [0.1, 0.2, 0.33, 0.5, 0.66, 1, 2].choose;

        (~tempo0Message0*~randDurationChooser0).wait;

         });
        });

        ~randRoutine0.reset;
        ~randRoutine0.play;

        }

        );
```

5.6.2 Real-time Digital Audio Effects' Control

5.6.2.1 Onsets of Control Sources Triggering Grain Freeze Process

Audiovisual example on the DVD *5. Grain Freeze.mov* in the *Audiovisual Examples* folder demonstrates the control of the dynamic onset triggering for the PV_Freeze UGen, which freezes the audio to a grain triggered by *control* source one. The real-time audio input from the Sample UGens forms *control* source one, *control* source two and *control* source three/*slave*. *Control* source one audible, followed by *control* source two and *control* source three/*slave* sound-object, then by the grain freeze process.

The following code represents the process for the freezing of grains controlled by the onsets of *control* source one.

Control source one filtered by bank of band-pass filters with static/dynamic filter frequencies:

```
        osc0Filter0 = BBandPass.ar(osc0, Select.kr(feedback0[46], [osc0Freq0, Select.kr(feedback0[60],
[filterData0[0], networkFilters0[0]])]), feedback0[57]);
```

Onsets of *control* source one analysed by a bank Onsets.kr UGens, with thresholds defined in GUI:

```
        osc0Onset0 = Onsets.kr(osc0OnsetChain0, threshold0Array0[0], \phase);
```

Outputs of bank of Onsets.kr UGens trigger PV_Freeze UGen relative to its allocated granular synthesizer:

```
        grain0Shift0 = FFT(~pitchBuffer0[0], triggersIn0[0]);
        grain0Shift0 = PV_Freeze(grain0Shift0, freezeOn0);
        grain0Pan0 = Pan2.ar(IFFT(grain0Shift0), grain0PanArray0[0], 1) * envelopes0[0];
```

5.6.2.2 Onsets of Control Sources Dictating Envelope Trigger and Time for Slave Sound-Object Prior to Buffering for Granular Synthesizers

Audiovisual example on the DVD *6. Slave Sound-Object Enveloping.mov* in the *Audiovisual Examples* folder demonstrates the control of the dynamic onset triggering of the envelope for the Warp1.ar UGen *prior* to the *slave* sound-object's placement in the granular synthesizers' audio buffers. The number of triggers per second dictate the envelope time. The real-time audio input from the Sample UGens forms *control* source one and *control* source three/*slave*. *Slave* output always audible with *control* source one faded in and out.

The following code represents the process for controlling the envelope of the Warp1.ar UGen of the *slave* sound-object by *control* source one, prior to its placement within the granular synthesizer buffers.

Control source one filtered by bank of band-pass filters with static/dynamic filter frequencies:

```
osc0Filter0 = BBandPass.ar(osc0, Select.kr(feedback0[46], [osc0Freq0, Select.kr(feedback0[60],
[filterData0[0], networkFilters0[0]])]), feedback0[57]);
```

Onsets of *control* source one analysed by a bank Onsets.kr UGens, with thresholds defined in GUI:

```
osc0Onset0 = Onsets.kr(osc0OnsetChain0, threshold0Array0[0], \phase);
```

Number of onsets per second from the *control* source one are counted, with the result dictating the attack and release time of the *slave* sound-object's envelope. Attack and Release values for number of onsets per second >= 750, >=687.5 and >=625 shown here:

```
~envelopeRoutine1 = Routine.new({
inf.do({ arg i;
        ~g00 = ~g00 - g0;
        ~g00 = g0;
        case
        {~g00 >= 750}
        {~bufferSynth.set(\attack2, 0.01); ~bufferSynth.set(\release2, 0.01);}
        {~g00 >= 687.5}
        {~bufferSynth.set(\attack2, 0.025); ~bufferSynth.set(\release2, 0.025);}
        {~g00 >= 625}
        {~bufferSynth.set(\attack2, 0.05); ~bufferSynth.set(\release2, 0.05);}
        g0 = 0;
        1.wait;
        });
        });
```

Onsets, attack time and release time placed within relevant Env.adsr and EnvGen.kr parameters, with envelope time modifiable via the GUI:

```
osc2 = (Warp1.ar(1, b, (materialPosition2/BufFrames.kr(b)), feedback[25], grainLength2, -1, 8, 0.1, 2) *
EnvGen.kr(Env.adsr(attack2Mean, attack2Mean, Select.kr(resetChooser0, [1, spliceDuration.max(0.5)]),
release2 * attackMultiplier0, 1, 'sine'), mat3Trigger + Select.kr(amplitudeChooser0, [0, mfccReturn0,
mfccReturn1, mfccReturn2]), 0.1));
```

5.6.2.3 Pitch Following of Control Source One by Slave Sound-Object

Audiovisual example on the DVD *13. Pitch Track Both Inputs.mov* in the *Audiovisual Examples* folder demonstrates the pitch following of the *slave* sound-object to *control* source one's pitch by pitch-tracking and comparing both auditory signals' pitch. The real-time audio input from a sampled major scale is provided as *control* source one

with a sampled, monophonic synthesiser loop applied as *control* source three/*slave*. Both input sources audible.

Audiovisual example on the DVD *14. Pitch Track Slave Pitch Fixed.mov* demonstrates the pitch following of the *slave* sound-object to *control* source one's pitch by pitch-tracking and comparing the analysed pitch of *control* source one to a fixed value of the *slave* sound-object, defined in the GUI to a C. The real-time audio input from a sampled major scale is provided as *control* source one with a sampled, monophonic synthesiser loop applied as *control* source three/*slave*. Both input sources audible.

The following code represents the process for pitch following.

Control source one's pitch extracted:

```
# inputPitcher0, hasinputPitcher0 = Pitch.kr(inputPitch0);
```

Slave source's pitch extracted:

```
# pitchOut0, hasPitchOut0 = Pitch.kr(osc2Array2);
```

Alternatively, the pitch of the *slave* source can be defined via a GUI TextField:

```
~pitchFixed0 = TextField(~w5, Rect(250, 150, 50, 25));
~pitchFixed0.string = "A2";
~pitchFixed0.action = {arg field;
        ~bufferSynth.set(\fixedPitch, field.value.namemidi.midicps);
                };
```

Control source one's pitch divided by *slave* source's pitch:

```
ratePitches = Select.kr(fixedChooser, [Select.kr(networkChooser0, [meanInputPitcher0/meanPitcher2,
networkPitch0[0]/meanPitcher2]), Select.kr(networkChooser0, [meanInputPitcher0/meanPitcher2,
networkPitch0[0]/meanPitcher2])]);
```

Control source one's GrainBuf.ar rate parameter multiplied by result:

```
pitchChooser0 = Gate.kr(Select.kr(fractal1Grain0, [grain1Pitch0, grain1Pitch0 * PinkNoise.kr(4)]) *
Select.kr(pitchTrackOn, [1, pitchTracker]), Select.kr(networkChooser0,[onsets1[0], networkOnsets1[0]]));
```

5.6.2.4 Tempo Following of Control Source One by Control Source Two

Audiovisual example on the DVD *15. Tempo Following.mov* in the *Audiovisual Examples* folder demonstrates the tempo following of *control* source two to *control* source one's tempo by beat-tracking and comparing both auditory signals' tempo. The real-time audio input from the Sample UGens forms *control* source one and *control* source two. Initially, the audible outputs of the *control* sources are played, followed by result of the tempo following process.

The following code represents the process for tempo following.

Control source one's tempo extracted:

```
#crotchetTick0, quaverTick0, semiquaverTick0, tempo0 = BeatTrack.kr(fftAnalysis0, 0);
```

Control source two's tempo extracted:

```
#crotchetTick2, quaverTick2, semiquaverTick2, tempo2 = BeatTrack.kr(beatTrack2, 0);
```

Control source one's tempo divided by *control* source two's tempo, with *control* source two's Warp1.ar UGens Phasor.ar UGen multiplied by result:

```
controlRate0 = tempo0/tempo1;

materialPosition1 = RedPhasor2.ar(midiIn1, BufRateScale.kr(d.bufnum) * controlRate0, start1, end1,
loopOn1, start1, end1);
osc1 = (Warp1.ar(1, d, (materialPosition1/BufFrames.kr(d)), grainPitcher1, grainLength1, -1, 8, 0.1, 2) *
EnvGen.kr(Env.adsr(attack1Mean, attack1Mean, 1, release0 * attackMultiplier0, 1, 'sine'), midiIn1, 0.1))
* osc1Level0;
```

5.6.2.5 Pitch Fixing of Slave Sound-Object

Audiovisual example on the DVD *17. Pitch Fix with Original.mov* in the *Audiovisual Examples* folder demonstrates the fixing of the *slave* sound-object's output to a C Major scale. The real-time audio input from a piano keyboard forms *control* source three/*slave* sound-object. The simultaneous outputs *pre* and *post* modification by the pitch fixing process are audible.

Audiovisual example on the DVD *18. Pitch Fix without Original.mov* demonstrates the fixing of the *slave* sound-object's output to a C Major scale. The real-time audio input from a piano keyboard forms *control* source three/*slave* sound-object. The output *post* modification by the pitch fixing process is audible.

The following code represents the process for pitch fixing the *slave* sound-object.

Slave source's pitch extracted:

```
# pitchOut0, hasPitchOut0 = Pitch.kr(osc2Array2);
```

Pitch structure is defined in GUI and placed in a control buffer:

```
//Select scale root note
~scaleFunctions = [(0..10).collect({|n| (Scale.major.degrees+(12 * n))}).flatten, (0..10).collect({|n|
        (Scale.minor.degrees+(12 * n))}).flatten, (0..10).collect({|n| (Scale.chromatic.degrees+(12 *
        n))}).flatten];
        ~chosenScale = ~scaleFunctions.at(0);
        ~chosenScaleAdjust = ~chosenScale - 1;

        ~noteText = PopUpMenu(~w5, Rect(15, 200, 75, 25));
        ~noteText.font_(Font("Monaco", 10));
        ~noteText.items = ["C", "C#", "D", "D#", "E", "F", "F#", "G", "G#", "A", "A#", "B"];
          ~noteText.action = {arg menu;
          ~chosenScaleAdjust = (~chosenScale + menu.value);
        };

//Select scale type
        ~scaleSelect0 = PopUpMenu(~w5, Rect(15, 225, 75, 25));
        ~scaleSelect0.font_(Font("Monaco", 10));
        ~scaleSelect0.items = ["Major", "Minor", "Chromatic"];

        ~scaleSelect0.action = {arg menu;
        ~chosenScale = (~scaleFunctions.at(menu.value)) + ~noteText.value;
        ~chosenScaleAdjust = ~chosenScale - 1;
        };

//Toggle on Pitch Fixing
          ~scaleOnButton.action_({arg butt;
          if ( (butt.value == 1),
          {         ~bufferSynth.set(\scaleChooser, 1);
                    ~scaleBuffer = Buffer(s, ~chosenScaleAdjust.size, 1, bufnum:900);
               s.listSendMsg(~scaleBuffer.allocMsg(~scaleBuffer.setnMsg(0, ~chosenScaleAdjust.midicps)));
               ~pitchTrackerSynth = Synth(\scaler);
               ~pitchTrackerSynth.set(\bufnum, ~scaleBuffer);
               ~scalerUpdateRoutine0.reset;
               ~scalerUpdateRoutine0.play;},
          {~bufferSynth.set(\scaleChooser, 0); ~scalerUpdateRoutine0.stop; ~pitchTrackerSynth.free;
          }
```

```
);
});
```

Slave source's pitch compared to pitches defined within buffer:

```
index = IndexInBetween.kr(bufnum, freqAdjust);
frequencyDiff = index.frac * (Index.kr(bufnum, index + 1) - Index.kr(bufnum, index));
```

Result is placed in PitchShift.kr UGen, making adjustment audible:

```
out = PitchShift.ar(in, grainSize, 1 - (frequencyDiff / freqAdjust), 0.00001, 0.01) * 2;
```

5.6.2.6 Random Search Process for Control of Reverb, Filter, Panning and the Buffer Position and Time Stretching of the Slave Sound-Object's Warp1.ar UGen

Audiovisual example on the DVD *22. Random Search Processes.mov* in the *Audiovisual Examples* folder demonstrates the control of the reverb, filter and panning of the overall auditory mix in combination with the random selection of the *slave* sound-object's Warp1.ar UGen's buffer position and time stretching. The real-time audio input from the Sample UGens forms *control* source one and *control* source three/*slave*. Both sources are audible.

The following code represents the process for random search control of the overall auditory mix's reverb, filter and panning in combination with the buffer position and time stretching of the *slave* sound-object's Warp1.ar UGen.

Execute random generative process via GUI Button:

```
~autoFXButton0 = Button(~w5, Rect(100, 25, 100, 25));
~autoFXButton0.action_({arg butt;
if ( butt.value == 1,
{
~generativeRoutine = Routine.new({
inf.do({ arg i;
var offset, duration, cutlength, barposition, stretch, roomSize, damper, volume, filterFreq0, filterRes0,
filterFreq1, filterRes1, grainsize0, grainsize1, grainpan0, grainpan1, grainvol0, grainvol1, reverbTime,
revSpread, earlyRef, tailLev;

//Randomly select values for features
        stretch = (-2..2).choose;
        roomSize = (10..300).choose;
        damper = (0..1).choose;
        filterFreq0 = (40..20000).choose;
        filterRes0 = (0.01..1).choose;
```

```
                filterFreq1 = (40..440).choose;
                filterRes1 = (0.01..1).choose;
                grainsize0 = (0.05..0.15).choose;
                grainsize1 = (0.05..0.15).choose;
                grainpan0 = (-1..1).choose;
                grainpan1 = (-1..1).choose;
                grainvol0 = (0.25..1).choose;
                grainvol1 = (0.25..1).choose;
                offset = ((~start)..(~length)).choose;
                cutlength = (0.01..1).choose;
                reverbTime = (0.1..5).choose;
                revSpread = (10..100).choose;
                earlyRef = (0.5..1).choose;
                tailLev = (0.5..1).choose;

//Send values to SynthDefs
                ~bufferSynth.set(\spliceDuration, cutlength);
                ~bufferSynth.set(\reset2, offset);
                ~bufferSynth.set(\stretcher, stretch);
                ~fxSynth.set(\roomSize0, roomSize);
                ~fxSynth.set(\damper, damper);
                ~fxSynth.set(\lpfFreq, filterFreq0);
                ~fxSynth.set(\lpfRes, filterRes0);
                ~fxSynth.set(\hpfFreq, filterFreq1);
                ~fxSynth.set(\hpfRes, filterRes1);
                ~fxSynth.set(\grainSize0, grainsize0);
                ~fxSynth.set(\grainSize1, grainsize1);
                ~fxSynth.set(\grainPan0, grainpan0);
                ~fxSynth.set(\grainPan1, grainpan1);
                ~fxSynth.set(\grainVol0, grainvol0);
                ~fxSynth.set(\grainVol1, grainvol1);
                ~fxSynth.set(\revTime1, reverbTime);
                ~fxSynth.set(\spread, revSpread);
                ~fxSynth.set(\earlyRf, earlyRef);
                ~fxSynth.set(\tailLev, tailLev);

                ~algoTime.wait;
                });
                });
                ~generativeRoutine.reset;
                ~generativeRoutine.play;
        });
```

5.6.2.7 Call and Response

Audiovisual example on the DVD *20. Call and Response.mov* in the *Audiovisual Examples* folder demonstrates the application of a Call followed by a Response. The Call is provided by the real-time audio input from a subtractive synthesizer placed in to *control* source one. Initially the Call is audible, followed by the Response.

The following code represents the process for Call and Response.

Loudness of Call signal measured:

```
fft0 = FFT(~powerBuffer0, buffer0);
power0 = Loudness.kr(fft0);
```

Output is recorded with Loudness values above 5 triggering the recording of the Call:

```
//Loudness trigger for recording toggle on/off
		~waitCounter0 = Routine {
			inf.do({ arg i;

			if((~interactInput0[~interactInputChooser0] < 5),
			{
			~waitCount0 = ~waitCount0 + 0.5;
			~stopInteractGUI = ~waitCount0;
			});

			if((~interactInput0[~interactInputChooser0] >= 5),
			{
			~waitCount0 = 0;
			~stopInteractGUI = 0;
			});

			0.5.wait
			});
			 };

// count time of Call
		~interactCounter0 = Routine {
			inf.do({ arg i;

			if((~interactCounterOn == 1),
			{
			~interactCount0 = ~interactCount0 + 1;
			~interactEnvelope = ~interactCount0;
			~interactTimeOut = ~interactTempo0;
			~interactOnsetCalculator = (~interactOnsetCounterOut/~interactEnvelope);
			~interactPitch0 = ~grainPitch0Message0.cpsmidi/50;
			~interactPitch1 = ~grainPitch0Message0.cpsmidi/50;

			});

			if((~interactCounterOn == 0),
			{
			~interactCount0 = 0;
			});

			1.wait

			});
			 };

//When recording toggled on, record audio out, If toggled off, playback recorded audio
	~interactRoutine1 = Routine {
			inf.do({ arg i;

			if((~waitCount0 < ~waitCountAdjuster0),
			{
			~fxSynth.set(\interactBufferOn0, 1);
```

```
~interactSynth.set(\triggerOn0, 0, \volumeOut0, 0);
~interactCounterOn = 1;

 if((~waitCount0 >= ~waitCountAdjuster0),
{
~fxSynth.set(\interactBufferOn0, 0);
~interactSynth.set(\triggerOn0, 1, \volumeOut0, 1);
~interactCounterOn = 0;
~interactOnsetCounter = 0;
});

0.01.wait

});
 };
```

When the Call's Loudness falls below 5, the Response is triggered. The Response's audio is formed of the recorded Call, which is played back through the Warp1.ar UGen, with its parameters defined by the values of the interactRoutine0:

```
        ~interactRoutine0 = Routine {
                inf.do({ arg i, interactPos, interactPitch, interactVol, interactTime0, interactTime1,
                interactTime2, interactTime3, interactTime4, interactSustain, interactEnd, interactDivision,
                interactStretch;

//Specify buffer playback start/end position, volume, pitch and envelope sustain
                interactPos = rrand(0, ((s.sampleRate * ~interactEnvelope) - (s.sampleRate +
                (~waitCountAdjuster0 * s.sampleRate)))).round(s.sampleRate/~interactTimeOut);
                interactEnd = rrand(interactPos, ((s.sampleRate * ~interactEnvelope) - (~waitCountAdjuster0 *
                s.sampleRate))).round(s.sampleRate/~interactTimeOut);
                interactPitch = rrand(~interactPitchOut0, ~interactPitchOut1).round(~interactPitchRound0);
                interactVol = rrand(0.8, 1);
                interactSustain = ~interactEnvelope.max(1);

//Create arrays of various playback durations
                interactTime0 = [[1/~interactTimeOut, 1], [((1/~interactTimeOut) * 2), 0.5],
                [((1/~interactTimeOut) * 0.5), 2], [((1/~interactTimeOut) * 1.5), 0.75]].choose;

                interactTime1 = [[1/~interactTimeOut, 1], [1/~interactTimeOut, 1], [((1/~interactTimeOut) * 2),
                0.5], [((1/~interactTimeOut) * 0.5), 2], [((1/~interactTimeOut) * 0.5), 2]].choose;

                interactTime2 = [[((1/~interactTimeOut) * 0.5), 2], [((1/~interactTimeOut) * 0.3), 3],
                [((1/~interactTimeOut) * 0.25), 4], [((1/~interactTimeOut) * 0.5), 2], [((1/~interactTimeOut) *
                0.25), 4], [((1/~interactTimeOut) * 0.5), 2]].choose;

                interactTime3 = [[((1/~interactTimeOut) * 0.125), 8], [((1/~interactTimeOut) * 0.125), 4],
                [((1/~interactTimeOut) * 0.25), 4], [((1/~interactTimeOut) * 0.125), 8]].choose;

                interactTime4 = [[((1/~interactTimeOut) * 4), 0.25], [((1/~interactTimeOut) * 2), 0.5],
                [((1/~interactTimeOut) * 2), 0.5], [((1/~interactTimeOut) * 4), 0.25], [((1/~interactTimeOut) *
                2), 0.5], [((1/~interactTimeOut) * 1.5), 0.75]].choose;


//if conditions are met (performer is ready for response), play back recorded audio applying relative duration
```

```
array and envelope. Example shown here is max. 25 onsets collated over 1 second:
                                if((~interactCounterOn == 0) && ((~stopInteractGUI) > (~waitCountAdjuster0/2)) &&
                                ((~stopInteractGUI) < (interactSustain + (~interactEnvelope/100.max(0.1)))),
                                {
                                {
                                ~candenceOn = 0;

                                if(((~interactOnsetCalculator) >= 0) && ((~interactOnsetCalculator) < 25),

                                {
                                ~interactTimeSelector = interactTime4;
                                ~interactSpeedText0.string = "Sparse";
                                });

                                ~interactStartPos = interactPos;
                                ~interactFinishPos = interactEnd;

                                });

                                }.defer;
                                },

//If the envelope time is nearing a close, tidy up with a time stretched buffer sample

                                if((~interactCounterOn == 0) && (~candenceOn == 1)
                                && ((~stopInteractGUI) >= (interactSustain + (~interactEnvelope/100.max(0.1))))
                                && ((~stopInteractGUI) <= (interactSustain + (~interactEnvelope/100.max(0.1)) +
                                ~waitCountAdjuster0)),
                                {
                                {

                                ~interactSpeedText0.string = "Cadence";

                                ~interactStartPos = 0;
                                ~interactFinishPos = s.sampleRate/~interactTimeOut;

                                ~interactTimeAdjust = [~waitCountAdjuster0, ~waitCountAdjuster0];

                                }.defer;
                                });

//Set parameters in Interact Synth relative to output of interact Routine0

                                ~interactSynth.set(
                                \triggerOn0, 1,
                                \start0, ~interactStartPos,
                                \end0, ~interactFinishPos,
                                \pitch0, interactPitch,
                                \timeStretcher0, ~interactTimeAdjust[1],
                                \sustainTime0, interactSustain.max(1) + ~interactTimeAdjust,
                                \attackTime0, ~interactEnvelope/100.max(0.1),
                                \amplitude0, interactVol,
                                \releaseTime0, 0.1,
                                );

                                ~interactTimeAdjust[0].wait;

                                });
```

```
};
```

5.6.3 Network Control

5.6.3.1 Set-Up of Networked Instances of Genesis

Audiovisual example on the DVD *7. Network Set Up.mov* in the *Audiovisual Examples* folder demonstrates the set-up of two networked instances *Genesis*. No sound.

The following code represents the process for setting up the two instances of *Genesis*.

Network sender IP address defined by IP of Receiver:

```
~ipOfSender0 = TextField(~w5, Rect(15, 15, 75, 15));
~ipOfSender0.font_(Font("Monaco", 10));
~ipOfSender0.string = "Send IP 1";
~ipOfSender0.action = {arg field;
~networkSender0 = NetAddr(field.value, 57120);
```

Data is sent via network senders using OSC (Only pitch of control source one shown here):

```
guiUpdateRoutine0 = Task {
inf.do{
          ~gui0Bus0.getn(62, {arg value; {
          ~networkSender0.sendMsg(*(["pitch0"] ++ [value[4], value[18], ~networkOut0]));
          }.defer});
```

Network Receiver address defined by IP of Sender:

```
~ipOfReceiver = TextField(~w5, Rect(15, 150, 75, 15));
~ipOfReceiver.font_(Font("Monaco", 10));
~ipOfReceiver.string = "From IP";
~ipOfReceiver.action = {arg field;
~networkReceiver0 = NetAddr(field.value, 57120);
```

Data is collected by network responders to IP of Sender (Only pitch of control source one shown here):

```
~pitchResponder0 = OSCresponder(~networkReceiver0, '/pitch0', {| t, r, msg|

{
~pitchReply0 = [msg[1], msg[2], msg[3]];
}.defer;
}).add;
```

Data is allocated to relevant *SynthDefs* via task:

```
~networkUpdate0 = Task({
loop{
        (1/60).wait;
        ~bufferSynth.set(\networkPitch0, ~pitchReply0);
        };
        }).start;
```

5.6.3.2 Networking of Control Sources Triggering Onsets of a local Slave Sound-Object's Warp1.ar UGen on a Networked System

Audiovisual example on the DVD *8. Network Onsets.mov* in the *Audiovisual Examples* folder demonstrates the control of the dynamic onset triggering of the envelope via a network for the Warp1.ar UGen *prior* to the *slave* sound-object's placement in the granular synthesizers' audio buffers. The number of triggers per second dictate the envelope time. The real-time audio input from a piano keyboard forms the local *control* source one of the Sender, with Sample UGens forming the *control* source three/*slave* on the Sender and the Receiver. Sender and Receiver *slave* sound-objects audible.

The following code represents the process for triggering the onsets and envelopes of the *slave* local sound-object controlled by a networked *control* source one.

Control source one on Sender filtered by bank of band-pass filters with static/dynamic filter frequencies:

```
osc0Filter0 = BBandPass.ar(osc0, Select.kr(feedback0[46], [osc0Freq0, Select.kr(feedback0[60],
[filterData0[0], networkFilters0[0]])]), feedback0[57]);
```

Onsets of *control* source one on Sender analysed by a bank Onsets.kr UGens, with thresholds defined in GUI:

```
osc0Onset0 = Onsets.kr(osc0OnsetChain0, threshold0Array0[0], \phase);
```

Onsets of *control* source one on Sender sent via network to specified IP address using a routine:

```
~trigger0Bus0.getn(13, {|vals| {
		~networkSender0.sendMsg(*(["triggers0"] ++ triggers0Network));
				}.defer});
```

Receiver collects onsets by network responders to IP of Sender:

```
~triggerResponder0 = OSCresponder(~networkReceiver0, '/triggers0', {| t, r, msg|
		{
		~trigger0Reply = [msg[1], msg[2], msg[3], msg[4], msg[5], msg[6], msg[7], msg[8], msg[9],
		msg[10], msg[11], msg[12], msg[13]];
		}.defer;
		}).add;
```

Received onsets allocated to relevant *SynthDefs* via task:

```
~networkUpdate0 = Task({
loop{
		(1/60).wait;
		~granularMacroSynth0.set(\networkOnsets0, ~trigger0Reply);

			};
		}).start;
```

Number of onsets per second from the networked *control* source one are counted, with the result dictating the attack and release time of the *slave* sound-object's envelope. Attack and Release times for number of onsets >= 750, >=687.5 and >=625 shown below:

```
~envelopeRoutine1 = Routine.new({
inf.do({ arg i;
		~g00 = ~g00 - g0;
		~g00 = g0;
		case
		{~g00 >= 750}
		{~bufferSynth.set(\attack2, 0.01); ~bufferSynth.set(\release2, 0.01);}
		{~g00 >= 687.5}
		{~bufferSynth.set(\attack2, 0.025); ~bufferSynth.set(\release2, 0.025);}
		{~g00 >= 625}
		{~bufferSynth.set(\attack2, 0.05); ~bufferSynth.set(\release2, 0.05);}
		g0 = 0;
		1.wait;
		});
		});
```

Onsets, attack time and release time placed within relevant Env.adsr and EnvGen.kr parameters, with envelope time modifiable via the GUI:

```
osc2 = (Warp1.ar(1, b, (materialPosition2/BufFrames.kr(b)), feedback[25], grainLength2, -1, 8, 0.1, 2) *
EnvGen.kr(Env.adsr(attack2Mean, attack2Mean, Select.kr(resetChooser0, [1, spliceDuration.max(0.5)]),
release2 * attackMultiplier0, 1, 'sine'), mat3Trigger + Select.kr(amplitudeChooser0, [0, mfccReturn0,
mfccReturn1, mfccReturn2]), 0.1));
```

5.6.3.3 Networking of Control Sources Triggering Onsets of Granular Synthesizers and the pitch following of the Slave Sound-Object's Warp1.ar UGen

Audiovisual example on the DVD *16. Network Pitch and Onsets.mov* in the *Audiovisual Examples* folder demonstrates the control of the dynamic onset triggering of the envelope via a network for the Warp1.ar UGen *prior* to the *slave* sound-object's placement in the granular synthesizers' audio buffers in addition to the local *slave* sound-object's pitch. The number of triggers per second dictates the envelope time. The real-time audio input from a piano keyboard forms local *control* source one of the Sender with Sample UGens forming the *control* source three/*slave* on the Sender and the Receiver. All sources audible.

The following code represents the process for controlling the onsets of the granular synthesizers and pitch of the *slave* sound-object's Warp1.ar UGen on a networked receiver.

Networked *Control* source one filtered by bank of band-pass filters with static/dynamic filter frequencies (static in Audiovisual example 14):

```
osc0Filter0 = BBandPass.ar(osc0, Select.kr(feedback0[46], [osc0Freq0, Select.kr(feedback0[60],
[filterData0[0], networkFilters0[0]])]), feedback0[57]);
```

Networked *Control* source one's pitch extracted:

```
# inputPitcher0, hasinputPitcher0 = Pitch.kr(inputPitch0);
```

Onsets of *control* source one analysed by a bank Onsets.kr UGens, with thresholds defined in GUI:

```
osc0Onset0 = Onsets.kr(osc0OnsetChain0, threshold0Array0[0], \phase);
```

Onsets and pitch of *control* source one sent via network to specified IP address using a routine:

```
~trigger0Bus0.getn(13, {|vals| {
            ~networkSender0.sendMsg(*(["triggers0"] ++ triggers0Network));
            ~networkSender0.sendMsg(*(["pitch0"] ++ [value[4], value[18], ~networkOut0]));
            }.defer});
```

Receiver collects onsets by network responders to IP of Sender:

```
~triggerResponder0 = OSCresponder(~networkReceiver0, '/triggers0', {| t, r, msg|
          {
          ~trigger0Reply = [msg[1], msg[2], msg[3], msg[4], msg[5], msg[6], msg[7], msg[8], msg[9],
          msg[10], msg[11], msg[12], msg[13]];
          }.defer;
          }).add;
```

Received onsets allocated to relevant *SynthDefs* via task:

```
~networkUpdate0 = Task({
loop{
          (1/60).wait;
          ~granularMacroSynth0.set(\networkOnsets0, ~trigger0Reply);
          };
          }).start;
```

Local *Slave* source's pitch extracted:

```
# pitchOut0, hasPitchOut0 = Pitch.kr(osc2Array2);
```

Networked *Control* source one's pitch divided by local *slave* source's pitch:

```
ratePitches = Select.kr(fixedChooser, [Select.kr(networkChooser0, [meanInputPitcher0/meanPitcher2,
networkPitch0[0]/meanPitcher2]), Select.kr(networkChooser0, [meanInputPitcher0/meanPitcher2,
networkPitch0[0]/meanPitcher2])]);
```

Control source one's GrainBuf.ar rate parameter multiplied by result:

```
pitchChooser0 = Gate.kr(Select.kr(fractal1Grain0, [grain1Pitch0, grain1Pitch0 * PinkNoise.kr(4)]) *
Select.kr(pitchTrackOn, [1, pitchTracker]), Select.kr(networkChooser0,[onsets1[0], networkOnsets1[0]]));
```

5.6.4 Interaction Control and Display

5.6.4.1 Live Routine and Live Sample Generation

Audiovisual example on the DVD *21. Live Routine, Live Sampling.mov* in the *Audiovisual Examples* folder demonstrates the live coding provided by the GUI object, and consequent wrapping as a routine along with the live sampling process. The real-time audio input from the Sample UGens forms *control* source one. Initially, the audible outputs of the *control* sources are played, followed by its repetition through a newly generated routine, then by this routine's output live sampled and played back.

The following code represents the process for ***live routine*** generation:

A hidden post window with a predefined task string is created at the initiation of *Genesis*:

```
        ~saveName = "~routine";
        ~saveText = Document.new("Save Session", makeListener: false);
        ~saveText.bounds_(Rect((1920/3), 456, 1920/3, 230));
        ~saveText.background = Color.gray.alpha_(0);
        ~saveText.editable = true;
        ~saveText.string_(~saveName ++ " = Routine({

      inf.do ({ arg i; ",
        (~saveText.string.size), (~saveText.string.size));
        ~saveText.selectLine(~saveText.string.size);

        ~taskString = " \n\n " ++ "

        0.01.wait;

        });

        });";

~saveText.editable =  false;
```

Each GUI object writes a string with its valueAction and its current value to the hidden post window. The following code shows this for the pitch modifier EZSlider of the Sample UGens for *control* source one:

```
        ~osc0PitchSlider0.action = {|ez|
                ~bufferSynth.set(\grainPitcher0, ez.value);
```

```
        textEditor.stringColor_(Color.yellow, (~textEditor.string.size), (~textEditor.string.size));
        ~textEditor.string_(" \n\n " ++ ~clockOut ++ " osc0Pitch0 = " ++ ez.value.asString ++ " ",
        (~textEditor.string.size), (~textEditor.string.size));

        ~textEditor.selectLine(~textEditor.string.size);

        ~saveText.string_(" \n\n " " if(clockGUI == " ++ ~clockGUI ++  "," ++
        "{{ ~osc0PitchSlider0.valueAction = " ++ ~osc0PitchSlider0.value.asString ++  " }.defer;}); ",
      (~saveText.string.size), (~saveText.string.size));

      ~saveText.selectLine(~saveText.string.size);
      }
```

The current content of the hidden post window can be wrapped as a task and titled through the GUI (the following code demonstrates this process for one task button within the 'Routine Controls' window):

```
        ~newRoutineButton0 = Button(~w6, Rect(225, 225, 100, 25));
        ~newRoutineButton0.states_([["Create...",Color.black,Color.gray],]);
        ~newRoutineButton0.action_({|butt|

        ~saveText.editable = true;

//Use saved text to create a prewrapped task
        if(~allocateTask0.value == 1,
        {
        ~saveText.string_(~taskString, (~saveText.string.size), (~saveText.string.size));
        ~saveText.selectLine(~saveText.string.size);
        ~saveText.syntaxColorize;
        ~taskPositionRoutine0.clear;
        ~taskPositionRoutine0 = ~saveText.string.interpret;
        ~taskPositionString0 = ~saveText.string;
        ~saveText.selectRange(~saveText.string.size - (~taskString.size - 1), ~saveText.string.size);
        ~saveText.selectedString = "";
        ~newTaskButton0.states_([[~saveName ++ "Stop",Color.black,Color.gray], [~saveName ++
        "Stop",Color.black,Color.green],]);

        ~textEditor.stringColor_(Color.black, (~textEditor.string.size), (~textEditor.string.size));

        ~textEditor.string_(" \n\n " ++ ~clockOut ++ " " ++ ~saveName ++ " loaded to 1 ",
        (~textEditor.string.size), (~textEditor.string.size));

        ~textEditor.selectLine(~textEditor.string.size);

        ~newTaskButton0.action_({|butt|

//Make button play/stop routine
        if(~newTaskButton0.value == 0,
        {
        ~taskPositionRoutine0.stop;
        },
        {
        ~taskPositionRoutine0.reset;
        ~taskPositionRoutine0.play;
        });
```

```
    });
});
```

A task can also be loaded from and saved to the computer's hard disk (the process is very similar to that above. The key difference being the wrapped task comes from a saved file, as opposed to the hidden post window).

The following code represents the process for ***live sample*** creation.

Buffer length can be set via the GUI:

```
~recordLengthInput0 = TextField(~w6, Rect(75, 25, 50, 25));
~recordLengthInput0.background = Color.red.alpha_(0.8);
~recordLengthInput0.action = {arg field;
        ~recordBuffer0.free;
        ~recordBuffer0 = Buffer.alloc(s, s.sampleRate * field.value.asInteger.max(1), 2, bufnum:226);
};
```

The recording can be triggered locally or over a network within the GUI (locally demonstrated here):

```
~recordSwitch0 = Button(~w6, Rect(125, 25, 100, 25));
~recordSwitch0.states_([
                                    ["Record Off",Color.white.alpha_(0.8),Color.red.alpha_(0.8)],
                                    ["Record On",Color.red.alpha_(0.8),Color.white.alpha_(0.8)],
                                                ]);


~recordSwitch0.action_({arg butt;
if ( butt.value == 1,
        {~recordBuffer0.free;
        ~recordBuffer0 = Buffer.alloc(s, s.sampleRate * ~recordLengthInput0.value.asInteger.max(1), 2,
        bufnum:226);
        ~fxSynth.set(\recordOn, 1);
        },

        {~fxSynth.set(\recordOn, 0);
        ~recordBuffer0.write(sampleFormat: 'int16');
        ~quickLoadPath0 = (thisProcess.platform.recordingsDir +/+ "SC_" ++ Date.localtime.stamp ++
        ".aiff")
        ;}
        )
    });
```

The destination Sample UGens of the recording is set in the GUI (Shown here for control source one):

```
~quickLoadReceiveSelector0 = PopUpMenu(~w0, Rect(225, 50, 125, 25));
~quickLoadReceiveSelector0.items = ["to Yellow", "to Red", "to Blue"];
~quickLoadReceiveSelector0.action = {arg menu;
        if (menu.value == 0,
        {       ~bufferSelector0 = c;
                ~sampleSelector0 = ~sampleViewer0;
                ~sampleFileSelected = ~sampleFile0;
                ~numFramesSelector = ~cnumFrames;
                ~dnaSelector0 = ~dna0;
                ~loadSelector0 = "Yellow";

        };);
        };
```

5.6.4.2 Dynamic Scoring System

Audiovisual example on the DVD *19. Dynamic Scoring System.mov* in the *Audiovisual Examples* folder demonstrates the visualisation process of the parameters of the granular synthesizers whose onsets are dictated by each *control* source. The real-time audio input from the Sample UGens forms *control* source one, *control* source two and *control* source three/*slave*. Initially, the audible outputs of the *control* sources are played, followed by the granular synthesizers that are controlled by their onsets, in combination with their visualisation.

The following code represents the process for dynamic scoring of the granular synthesizers' parameters.

Control sources' busses defining each granular synthesizer's parameters are collected. (Only MFCC data shown here):

```
~mfccBus0.getn(13, {arg value; {mfccData0 = value;}.defer});
```

Values are mapped to Pen methods (Only control source one shown here):

```
//Create Window
        ~visualiserWindow = Window("Genesis Visualiser", Rect(0, 0, ~visualWidth * 2, ~visualHeight * 2), false,
        ~borderOn);

        ~visualiserWindow.view.background = Color.gray;
        ~visualiserWindow.alwaysOnTop = true;
        ~visualiserWindow.userCanClose = false;
        ~visualiserWindow.front;
```

```
        ~visualiserWindow.drawFunc = {

//Draw with pen, applying parameters obtained from busses
        Pen.use {

                Pen.translate(~visualWidth, ~visualHeight);
                Pen.width = (duration0Data0[0] * 15);
                1.do {
                        Color.yellow([1, [(filterFreqData0[0]/4000) + 0.5, (~filterCutter0[0]/4000) +
                        0.5].at(~visualSpectrum0)].at(~visualFilter0), (envelope0Data0[0] * grainsVolume0) *
                        ([1, [1, ~filterCutterGUI0[0]].at(~visualSpectrum0)].at(~visualFilter0))).setStroke;
                Pen.moveTo(Point((~visualWidth) * pan0Data0[0], (pitch0Data0[0] * (~visualHeight/4)) * -1));
                Pen.lineTo(Point(0, 0));
                Pen.skew(position0Data0[0], position0Data0[0]);
                Pen.stroke;
                };
                };
                };
```

Chapter 6

Evaluation of the *Genesis* System

6.1 Evaluation Methodology

Formal evaluative methodologies for real-time interactive music systems, such as *Genesis*, are significantly limited. Indeed, research has shown that there are a 'consistently low proportion of papers containing formal evaluations'[232]. However, Human Computer Interaction (HCI) evaluation techniques are commonly applied to assess the success of real-time interactive music systems. An HCI evaluation method is 'historically drawn from four complimentary domains - software engineering, software human factors, computer graphics, and cognitive science – that could be grouped into two main foci: methods and software (Carroll, 2002)'[233] resulting in an approach that is founded on objective, quantifiable, task-based interaction, focusing on the *process*.

Considering that the success of *musical* interactions is creative and subjective, they are not quantifiable to a reliable measure. Indeed, the *product* of musical creativity is never unequivocal. As a result, the context of musical interaction poses an issue when applying it to HCI evaluation methods; how do we evaluate a real-time interactive music system's ability to perform a *creative* task? Furthermore, who should complete an evaluation of such a system? The performer, the composer or the audience? Collins (2007) suggests that the evaluation of real-time interactive music systems requires '1) technical criteria related to tracking success or cognitive modelling; 2) The reaction of an audience; 3) The sense of interaction for the musicians who participate'[234] thereby drawing upon HCI techniques to obtain evaluative feedback of the *process* and *product* from different sources.

---

[232] Stowell et al. 2009. Evaluation of live human-computer music-making: quantitative and qualitative approaches. *International Journal of Human-Computer Studies*. 67 (11): 960
[233] Wanderley, M and Orio, N. 2002. Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI. *Computer Music Journal*. 26(3): 62
[234] Hsu, W and Sosnick, M. Evaluating Interactive Music Systems: An HCI Approach. *NIME 2009*: 26

However, in terms of live musical interaction, a predominant feature of *Genesis*, 'the performer has privileged access to both the intention and the act, and their experience of the interaction is a key part of what determines its expressivity'[235]. Moreover, 'another challenging aspect of interface evaluation is that the participant populations are often small (Wanderley and Orio, 2002)'[236], further complicating the issue of finding suitable candidates to assess and evaluate real-time computer music systems.

Therefore, although a variety of sources, including audience-feedback and composer-feedback, may prove useful with regards to broader perspectives of a real-time interactive music system, those who have *engaged* in the act of interaction with such a system are argued to provide the most insight into their success and expressiveness. Indeed, considering the remit of the thesis, a small sample group of performer-based evaluation is applicable and attainable. As a result, evaluation of *Genesis* is primarily performer–centered.

As stated, HCI evaluation techniques are task-oriented, centering on *process*, requiring a specified goal to be set relative to a quantifiable target. Yet, regarding the volume of interactive methods that can be objectified in real-time interactive music systems, description of *every* possible task and attributed goal to be defined as criteria for evaluation is unfeasible; in the context of interaction with musical control interfaces, Wanderley and Orio (2002) state that 'it is nearly impossible to cover all the features of a controller unless an unbearable number of musical tasks is considered'[237]. The solution Wanderley and Orio (2002) provide is to evaluate a handful of low-level basic musical objectives. However, this creates artificial results in the context of musical performance by oversimplifying the range of possibilities a real-time interactive music system may have to an individual *process*, thereby not reflecting the true nature of a system's creative potential and its *product*.

Moreover, the more complex a system is, the more difficult it becomes to evaluate successfully through simple task-based interactions. For example, Hsu and Sosnick

[235] Stowell et al. 2009. Evaluation of live human-computer music-making: quantitative and qualitative approaches. *International Journal of Human-Computer Studies*. 67 (11): 960
[236] Ibid: 961
[237] Wanderley, M and Orio, N. 2002. Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI. *Computer Music Journal*. 26(3): 71

(2009) reflect on their experiences of evaluating interactive music systems by stating 'as the number of system components increased and their interactions increased in complexity, it became difficult to correlate design decisions to improvements in musicality'[238]. As a result, due to the complexity and significant number of interactive methods in *Genesis*, the evaluation of its high-level musical goals is prioritised, such as its musicality, ability to engage with performers and accessibility to instrumentalists.

In order to obtain evaluative feedback of such high-level goals from performers using HCI evaluation techniques, methods range from "talk-aloud" protocols (Ericsson and Simon, 1996), through which performers make statements during performance regarding their experiences, to tests based on human cognition such as GOMS (Card et al, 1983) which measure time taken by a user to achieve a specified goal, and observation of a user's reactions during performative interaction. However, in the case of real-time interactive music systems, such approaches are not reliable: a "talk-aloud" protocol breaks the flow of interaction, requiring a user to disrupt their ongoing creative process; time-based tests are not suitable in the context of musical performance as they bear no context to musical time; observations by a third party of a user's satisfaction in their interactions are highly subjective.

Questionnaires filled in by a user after a performance offer a useful method of reflective evaluation without the limitations of those methods listed above. Through a questionnaire, adequate *quantitative* and *qualitative* results can be obtained. Psychometric scales, such as the Likert scale[239], provide scalable results from the experiences of the user, which can be applied to HCI-based objectives. As a result, a Likert-scale approach will be used to evaluate high-level goals in *Genesis* to provide quantitative data that will be measured and compared between performers, with qualitative data obtained from the responses made by the participants to a number of key questions.

In the context of this thesis, the principle aim of the evaluation is to examine how successful the interactive methods implemented in *Genesis* are perceived to be by

[238] Hsu, W and Sosnick, M. Evaluating Interactive Music Systems: An HCI Approach. *NIME 2009*: 25
[239] Munshi, J. 2014. *A Method for Constructing Likert Scales.* Sonama State University

potential users of such a system. As a result, performers with an expressed interest in live performance with computers/electronics are suitable candidates. Furthermore, considering the remit of this thesis, which focuses on the algorithmic implementations in *Genesis*, and the novel methods of interactivity the system allows, the evaluation will be based upon the proposed trial of 'a single interface with no explicit comparison system'[240].

The trial of the *Genesis* system by the selected performers is based upon the qualitative approach suggested by Stowell et al (2009), through which a participant is invited to try out an interface and engage in *free exploration*, *guided exploration* and a *semi-structured interview* to evaluate a real-time interactive music system; in *free exploration* 'the participant is encouraged to try out the interface for a while and explore it their own way'[241], in *guided exploration* 'the participant is presented with audio examples of recordings created using the interface'[242] and in the *semi-structured* interview 'the interview's main aim is to encourage the participant to discuss their experiences of using the interface in the free and guided exploration phases'[243]. Furthermore, in order to obtain the most congruent, focused and personal evaluation results, solo sessions with each participant are conducted in which the author accompanies the performer, as proposed by Stowell et al (2009).

However, for the evaluation of *Genesis*, this qualitative evaluation method is adapted with the objective to gather further insight into the perceived success of the interactive methodologies in *Genesis*. Instead, *free exploration* includes an improvisatory performance with the system, based upon selected high-level features within *Genesis*, generating an extensive global *product* with which to discuss in the evaluation. Also, the musical outcomes of the *free explorations* with each participant are presented as audiovisual examples in the folder *Evaluation Performances* on the DVD to provide documental evidence of the performances discussed further into this chapter.

[240] Stowell et al. 2009. Evaluation of live human-computer music-making: quantitative and qualitative approaches. *International Journal of Human-Computer Studies*. 67 (11): 962
[241] Ibid: 964
[242] Ibid
[243] Ibid

In addition, the *guided exploration* is completed in real-time, with the author demonstrating and using step-by-step instruction to show the participant an *overview* of the principles of the *Genesis* system. The purpose of the *guided exploration* is to increase a participant's understanding by answering any questions they may have, thereby encouraging them to explore *Genesis* how they see fit.

Furthermore, a *fully-structured* interview was conducted, in the form of the questionnaire approach detailed previously that combines a Likert-scale and a performer-centric commentary, to ensure fairness and balance between evaluation results from each performer. The questionnaire presented to each participant was as follows:

***Genesis* Evaluation Questionnaire**
1) How would you describe your interaction with *Genesis*?
2) Which aspects of performing with the *Genesis* system did you enjoy?
3) In relation to your musical background, how familiar and accessible was your interaction with *Genesis*?
4) How would you describe the performance capabilities of the *Genesis* system to generate reasoned musical responses to your interactions?
5) To what extent do you feel the musical paradigms implemented in *Genesis* are beneficial to your creative approach?
6) In terms of musicality, how successful do you consider the overall output of your performance with *Genesis*?
7) To what extent did you feel *Genesis* introduces a novel method of live performance and why?

8) To what extent did you feel in control of the performance process and why?
9) How would you describe the creativity of the responses generated by *Genesis*?
For the following questions, please state to what extent you agree with the following statements and add a comment justifying your response.
10) I would like to perform again with *Genesis* system:
Strongly Agree Agree Neither Agree/Disagree Disagree Strongly Disagree
Comment:
11) The *Genesis* system provided creative outputs that were relatable to my inputs:
Strongly Agree Agree Neither Agree/Disagree Disagree Strongly Disagree
Comment:
12) The real-time methodology of *Genesis* contributed positively to my performance process:
Strongly Agree Agree Neither Agree/Disagree Disagree Strongly Disagree
Comment:
13) *Genesis* features a significantly large musical exploratory space:
Strongly Agree Agree Neither Agree/Disagree Disagree Strongly Disagree
Comment:
14) The graphical user interface of *Genesis* is intuitive and easy-to-use:
Strongly Agree Agree Neither Agree/Disagree Disagree Strongly Disagree
Comment:
15) *Genesis* adapts its creative outputs over time:
Strongly Agree Agree Neither Agree/Disagree Disagree Strongly Disagree
Comment:

16) I felt engaged with *Genesis* during the performance process:				
Strongly Agree	Agree	Neither Agree/Disagree	Disagree	Strongly Disagree
Comment:				
17) I would liken the outputs of *Genesis* to those of a human being:				
Strongly Agree	Agree	Neither Agree/Disagree	Disagree	Strongly Disagree
Comment:				
18) Please add here any further comments you may have regarding your experience with *Genesis*:				

The questions within the evaluative questionnaire are designed to focus primarily on the key issues raised in chapter 4 *Interactivity with Digital Music Systems*. In terms if the underlying rationale, the following identifies the intended subject and the relationship to research into interactivity with digital music systems for each question. The first nine questions are as follows:

1) How would you describe your interaction with *Genesis*?

In order to categorise the methods of interaction with *Genesis* relative to Winkler's (2001), Paine's (2002) and Rowe's (1993) models of interaction, the responses of the participants should highlight how they perceive *Genesis's* approaches to interaction, thereby enabling a performer-based perspective that can be applied to the proposed models. The question should also highlight the perceived affordances and constraints of the system (as suggested by Magnusson (2012b) by indicating how they considered *Genesis* in the act of performance.

2) Which aspects of performing with the *Genesis* system did you enjoy?

Considering the variety of interactive and generative approaches available in *Genesis*, identifying *which* aspects are most enjoyable to a performer should serve to highlight the relative success of its numerous algorithmic implementations.

3) In relation to your musical background, how familiar and accessible was your interaction with *Genesis*?

A principle feature of the *Genesis* system is to form an extension of an instrumental paradigm, thereby making it relatable to instrumentalists and having a 'low entry fee' (Wessel and Wright, 2002). This question should ascertain to what extent the applied method is successful.

4) How would you describe the performance capabilities of the *Genesis* system to generate reasoned musical responses to your interactions?

Regarding the interpretation by the system of gesture, and the consequent generative processes which are mapped to such gestures modelled on the methods suggested by Arfib el al (2003), this question aims to discover to what extent a gestural input and an interaction by *Genesis* are perceived to be related, as proposed by Overholt et al (2009).

5) To what extent do you feel the musical paradigms implemented in *Genesis* are beneficial to your creative approach?

This question is designed to investigate further the performer's background and how their creative process may impact on successful interaction with *Genesis* through an instrumental paradigm. Moreover, the question is intended to identify if methods through abstraction such as live coding may be considered more successful when interacting with a machine.

6) In terms of musicality, how successful do you consider the overall output of your performance with *Genesis*?

With regards to the global *product* of performance with *Genesis*, it is necessary to consider how it is perceived *musically*, therefore indicating its aesthetic value and potential virtuosity.

7) To what extent did you feel *Genesis* introduces a novel method of live performance and why?

In terms of the approach to interaction within *Genesis*, it is important to identify how this system bears relevance to any other systems the performer may have used, hopefully contextualizing *Genesis* from a performer's perspective.

8) To what extent did you feel in control of the performance process and why?

This question is designed to identify a more descript view of how interaction with *Genesis* is influential, and how the dynamic of interaction over time with the system may be perceived to change. This should provide further evidence for categorisation of *Genesis* into the proposed models of interaction by Winkler (2001), Paine (2002 and Rowe (1993).

9) How would you describe the creativity of the responses generated by *Genesis*?

Relative to the discussion regarding *generative* and *adaptive* creativity (Bown, 2012), and the proposition of a hybridisation of the two approaches, this question attempts to discover how creativity with the *Genesis* system is considered, and whether its outputs are in themselves relatable and 'reasoned' to the performers interactions.

The application of a Likert scale for the following questions is intended to obtain quantifiable evidence to provide measurable comparison between the performers of their experiences with *Genesis*. Each question is accompanied by an optional

comments section in order to obtain further insight into each participant's feedback. The questions are as follows:

10) I would like to perform again with *Genesis* system:

This question is designed to relate directly to the enjoyment the performers may have experienced with the system, hopefully representing a desire by such performers to implement *Genesis* in further study and performances and indicating its value as a real-time interactive music system.

11) The *Genesis* system provided creative outputs that were relatable to my inputs:

Extending question **4** regarding how well the system is perceived to generate reasoned responses to the performer's interactions, this question attempts to quantify that response, demonstrating the success of the system's algorithmic implementations for the generation of musical gestures.

12) The real-time methodology of *Genesis* contributed positively to my performance process:

Considering a principle feature of *Genesis* is to function in real-time, it is necessary to consider to what extent this impacts on a performer's ongoing performance process and whether *Genesis* does indeed form a *real-time* interactive music system. This question is designed to obtain a quantifiable response to such issues.

13) *Genesis* features a significantly large musical exploratory space:

Related to the implementation of a 'low entry fee' (Wessel and Wright, 2002), significant discussion is presented that indicates a 'low entry fee' limits the prospective musicality of interactive music systems. This question attempts to identify if the algorithmic methods in *Genesis* reflect this supposition or if *Genesis* achieves a

'low entry fee' combined with a substantially interesting interactive musical sound space.

14) The graphical user interface of *Genesis* is intuitive and easy-to-use:

Further to the implementation of a 'low entry fee' (Wessel and Wright, 2002), a familiar and accessible interface is a factor within such a requirement. Therefore, this question should indicate how the GUI supplements this prerequisite.

15) *Genesis* adapts its creative outputs over time:

With regard to the propositions by Arfib et al (2003) and Paine (2002) of implementing *dynamic* and *evolving* parameter/interaction spaces, considering *Genesis* does not include such constraints, this question is intended to signify whether its methodology is perceived to feature them, and if so, to what extent. Therefore, this question should indicate if *dynamic* algorithmic design is required to generate such an effect.

16) I felt engaged with *Genesis* during the performance process:

This question is designed to identify how influential the *Genesis* system is in performance, and consequently how involved the performer feels as part of the creative process. As a result, this should suggest how successful the hybridisation of *generative* and *adaptive* creativity with *Genesis* is perceived to be.

17) I would liken the outputs of *Genesis* to those of a human being:

This question extends question **16**, and is designed to allow the performer to reflect on how they considered a machine to represent a human performer. With regards to Blackwell et al's (2012) suggestion of creating live algorithms that can emulate human performance convincingly, the response to this question should indicate to what extent *Genesis* achieves such a notion.

18) Please add here any further comments you may have regarding your experience with *Genesis*:

The performer must be encouraged to contribute any further comments on or concerns they may have with the system to represent any issues that they themselves may consider important as a performer using a real-time interactive music system. Indeed, this may highlight subjects that are personal to those performers or moreover, implications of *Genesis* that may have not been considered.

6.2 Evaluation Results

For each participant, an overview of the interaction method used in each evaluation is presented, along with reference to the audiovisual examples of their *free exploration* with *Genesis* and a figure illustrating the interaction approach. The questionnaire of the respective participant is then directly transcribed, followed by discussion of their feedback. Once each participant's evaluation method and feedback has been shown, comparison of the feedback and discussion regarding the success of the evaluation method are presented. All audiovisual examples are contained in the *Genesis Performances* folder on the accompanying DVD in each participant's respective named folder.

I approached three participants with a variety of musical backgrounds intended to provide a balanced perspective in the evaluation of *Genesis*. As stated in section 6.1 *Evaluation Methodology*, participants were selected dependent on their having an expressed interest in electroacoustic composition and performance techniques. The three participants are John Snijders, Shelly Knotts and Mark Carroll who are all members of the Durham University Music Department, contacted upon recommendation and discussion through conversation with research staff in the department.

In summary, John Snijders is Reader of Performance at Durham University, and an accomplished pianist who has performed with a variety of contemporary composers and performers around the globe. Shelly Knotts is a PhD student at Durham

University, greatly involved in the live coding scene, with a strong interest in networked performance and interaction, having organised a number events such as the Network Music Festival. Mark Carroll, composer-in-residence at the South Bank centre, London, is now a first year PhD student at Durham University, who was completing his MA in Composition at the time of evaluation. During his MA course, Mark began composing with SoundLoom[244], which introduced him to electroacoustic techniques and interaction with computers for composition.

As stated in section 6.1 *Evaluation Methodology*, each participant is invited to participate in *guided exploration*, followed by *free exploration*. After these have been completed, participants are presented with the evaluation questionnaire shortly after their interaction with *Genesis*.

John Snijders

Introduction to the accompanying video documentation

The *John Snijders Free Exploration.mov* video displays all instances of *Genesis* (two of which display the Dynamic Scoring System throughout the performance), the live piano performer and a stereo recording of the output.

The *free exploration* was recorded at Durham University Music Department Concert Hall on the 6th June 2013 in front of a live audience. The performance applies one live piano performer (John Snijders), one *supervised* instance of *Genesis* (Julian Lywood Mulcock) and two *unsupervised* networked instances of *Genesis*.

For each instance of *Genesis*, the granular synthesisers dictated by *control* source one are triggered by the live piano performer, with the granular synthesisers dictated by *control* source two triggered by a microphone placed at the rear of the concert hall. The *supervised* instance of *Genesis* uses a live stream of the piano obtained through a contact microphone for *control* source three/*slave* with the networked instances of

[244] Wishart, T. (n.d.). *Trevor Wishart.* [online] Trevorwishart.co.uk. Available at: http://www.trevorwishart.co.uk/slfull.html [Accessed Mar. 2013]

Genesis applying a pre-selected series of sample banks, broadly categorised into strings, woodwind and bells for their *control* source three/*slave* sound-objects. *Figure 54* illustrates the interaction methodology of the performance:

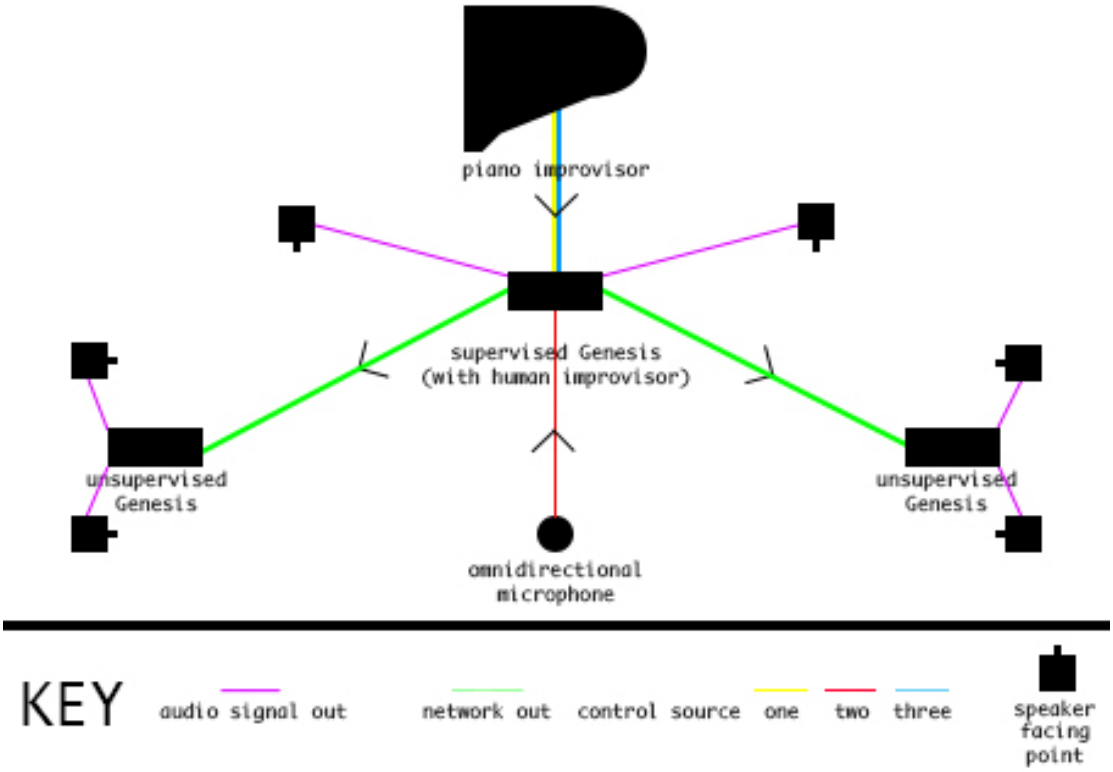


Figure 54. Performance Interaction with John Snijders

Considering the outline of the modes of interaction provided in chapter 5 *The Genesis System* in *Figure 41*, this performance scenario provides a hybrid of different interaction modes between the human performers and *Genesis*; the *supervised* instance of *Genesis* offers a *supervised improvisation* model of interaction, whereas the *unsupervised* networked instances of *Genesis* offer an *unsupervised ensemble* model of interaction, which results in an amalgam of interaction methodologies, generating a composition of a *Supervised/Unsupervised Improvisation Ensemble*; the human controlled instance of *Genesis* applies improvisational techniques for the modification of parameter settings through the GUI relative to the outputs of the improvisatory piano performer, while the unsupervised instances of *Genesis* use a looped routine which selects pre-defined parameters and sound-objects at selected durations, relative to durations decided prior to performance .

The following code defines the looped routine running on each *unsupervised* instance of *Genesis* throughout the performance:

```
//add files
~networkPerformanceFiles = "/Users/*****/Documents/University/PhD Research/Performance Audio Files Set
One/*".pathMatch;

//routine to randomly select sample files
~networkSampleRoutine0 = Routine.new({

    inf.do({ arg i;

                    var durations = rrand(0.5, 2.0);

                    var filesChooser = [0, 1, 2].choose;
//set up default settings
          ~inputSwitch2.valueAction = 0;
          ~loopChoose2.valueAction = 0;
          b.free;
          ~samplePath2 = ~networkPerformanceFiles.choose;
          b.allocRead(~samplePath2.asString);
          ~sampleFile2.openRead(~samplePath2.asString);
          ~sampleViewer2.soundfile = ~sampleFile2;
                ~sampleViewer2.read(0, ~sampleFile2.numFrames);
                ~sampleViewer2.refresh;
                ~bnumFrames = ~sampleFile2.numFrames;
                ~samplePathButton2.states_([[~samplePath2.asString, Color.white, Color.blue.alpha_(0.8)]]);
                ~bufferSynth.set(\start2, 0, \end2, ~sampleFile2.numFrames, \mate3Trigger, 1, \clipAdjust,
                0.05);

          ~start = 0;
          ~length = ~sampleFile2.numFrames;
          ~child.add(1);

//place name of file in DSS window
          if (~visualWindowCheck0 == 1,

          {~visualSlaveSource.string = ~samplePath2; ~visualSlaveSource.stringColor = Color.blue;
~visualSlaveSource.font = Font("Monaco", 25);}

          );

//if random duration is less than 1.25, generate random pan settings for control source one
          if (durations < 1.25,
          {
          ~panners0Array = Array.fill(13, {arg i;
          ~panKnob0[i].valueAction = \pan.asSpec.unmap(rrand(-1.0, 1.0));
          });
          };

          );

//if random duration is greater than 1.25, generate random pan settings for control source two
          if (durations > 1.25,
          {
          ~panners1Array = Array.fill(13, {arg i;
```

```
            ~panKnob1[i].valueAction = \pan.asSpec.unmap(rrand(-1.0, 1.0));
            });
            };

            );

//if clock is less than or equal to 8.5 minutes, select files from set one
            if(~clockGUI <= 83300,

            {
            ~networkPerformanceFiles = "/Users/*****/Documents/University/PhD Research/Performance Audio Files
Set One/*".pathMatch;
            });

//if clock is between 8.5 minures and 16 minutes, select files from set two

            if((~clockGUI > 83300) && (~clockGUI <= 160000),

            {
            ~networkPerformanceFiles = "/Users/*****/Documents/University/PhD Research/Performance Audio Files
Set Two/*".pathMatch;
            });

//if clock is between 8.5 minures and 16 minutes, select files from set three

            if((~clockGUI > 160000) && (~clockGUI <= 250000),

            {
            ~networkPerformanceFiles = "/Users/*****/Documents/University/PhD Research/Performance Audio Files
Set Three/*".pathMatch;
            });


//if clock is greater than 25 minutes, select files from all three sets

            if((~clockGUI > 250000) && (~clockGUI <= 300000) && (filesChooser == 0),

            {
            ~networkPerformanceFiles = "/Users/*****/Documents/University/PhD Research/Performance Audio Files
Set One/*".pathMatch;
            });

            if((~clockGUI > 250000) && (~clockGUI <= 300000) && (filesChooser == 1),

            {
            ~networkPerformanceFiles = "/Users/*****/Documents/University/PhD Research/Performance Audio Files
Set Two/*".pathMatch;
            });

            if((~clockGUI > 250000) && (~clockGUI <= 300000) && (filesChooser == 2),

            {
            ~networkPerformanceFiles = "/Users/*****/Documents/University/PhD Research/Performance Audio Files
Set Three/*".pathMatch;
            });

//if the sample file matches the first file of the folder, set random amplitudes and durations for control
source one
```

```
		if (~samplePath2 == ~networkPerformanceFiles[0],
		{
		~randomAmplitudes0 = [rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0,
1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0),
rrand(0.0, 1.0), rrand(0.0, 1.0)];
		~grainAmplitude0Slider.valueAction = ~randomAmplitudes0;
		~randomAmplitudes0Array = Array.fill(13, {arg i;
		~envelopeSynth.set("grain0Amplitude" ++ i.asString, ~volumeSpec0.map(~randomAmplitudes0[i].value));
		});

		~randomDuration0 = [rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0,
4.0), rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0, 4.0),
rrand(0.0, 4.0), rrand(0.0, 4.0)];
		~onsetDurationSlider0.valueAction = ~randomDuration0/4;
		~randomDuration0Array = Array.fill(13, {arg i;
		~granularMacroSynth0.set("onset0Duration" ++ i.asString,
~durationSpec0.map(~randomDuration0[i].value));
		});

		~onsetChooser0.valueAction = 1;

		};
		);

//if the sample file matches the second file of the folder, set random amplitudes and durations for control
source two

		if (~samplePath2 == ~networkPerformanceFiles[1],
		{
		~randomAmplitudes1 = [rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0,
1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0),
rrand(0.0, 1.0), rrand(0.0, 1.0)];
		~grainAmplitude1Slider.valueAction = ~randomAmplitudes1;
		~randomAmplitudes1Array = Array.fill(13, {arg i;
		~envelopeSynth.set("grain1Amplitude" ++ i.asString, ~volumeSpec0.map(~randomAmplitudes1[i].value));
		});

		~randomDuration1 = [rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0,
4.0), rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0, 4.0), rrand(0.0, 4.0),
rrand(0.0, 4.0), rrand(0.0, 4.0)];
		~onsetDurationSlider1.valueAction = ~randomDuration1/4;
		~randomDuration1Array = Array.fill(13, {arg i;
		~granularMacroSynth1.set("onset1Duration" ++ i.asString,
~durationSpec0.map(~randomDuration1[i].value));
		});

		~onsetChooser1.valueAction = 1;

		};
		);

//if the sample file matches the third file of the folder, set random attack and pitch for control source one

		if (~samplePath2 == ~networkPerformanceFiles[2],
		{
		~randomAttack0 = [rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0,
2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0),
rrand(0.0, 2.0), rrand(0.0, 2.0)];
```

```
            ~grainAttack0Slider.valueAction = ~randomAttack0/2;
            ~randomAttack0Array = Array.fill(13, {arg i;
            ~envelopeSynth.set("grain0Attack" ++ i.asString, ~attackSpec0.map(~randomAttack0[i].value));
            });

            ~randomPitch0 = [rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0,
4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0,
4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0)];
            ~pitch0Slider0.valueAction = (~randomPitch0 + 4)/8;
            ~randomPitch0Array = Array.fill(13, {arg i;
            ~granularMacroSynth0.set("grain0Pitch" ++ i.asString, ~randomPitch0[i].value);
            });

            ~onsetChooser0.valueAction = 0;

            };
            );

//if the sample file matches the fourth file of the folder, set random attack and pitch for control source two

            if (~samplePath2 == ~networkPerformanceFiles[3],
            {
            ~randomAttack1 = [rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0,
2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0),
rrand(0.0, 2.0), rrand(0.0, 2.0)];
            ~grainAttack1Slider.valueAction = ~randomAttack1/2;
            ~randomAttack1Array = Array.fill(13, {arg i;
            ~envelopeSynth.set("grain1Attack" ++ i.asString, ~attackSpec0.map(~randomAttack1[i].value));
            });

            ~randomPitch1 = [rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0,
4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0), rrand(-4.0,
4.0), rrand(-4.0, 4.0), rrand(-4.0, 4.0)];
            ~pitch1Slider0.valueAction = (~randomPitch1 + 4)/8;
            ~randomPitch1Array = Array.fill(13, {arg i;
            ~granularMacroSynth1.set("grain0Pitch" ++ i.asString, ~randomPitch1[i].value);
            });

            ~onsetChooser1.valueAction = 0;

            };
            );

//if the sample file matches the fifth file of the folder, set random release for control source one

            if (~samplePath2 == ~networkPerformanceFiles[4],
            {
            ~randomRelease0 = [rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0,
2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0),
rrand(0.0, 2.0), rrand(0.0, 2.0)];
            ~grainRelease0Slider.valueAction = ~randomRelease0/2;
            ~randomRelease0Array = Array.fill(13, {arg i;
            ~envelopeSynth.set("grain0Release" ++ i.asString, ~attackSpec0.map(~randomRelease0[i].value));
            });

            ~algorithmButton0[6].valueAction = 1;
            ~onsetChooser2.valueAction = 0;
```

```
        };
        );

//if the sample file matches the sixth file of the folder, set random release for control source two and turn
pitch tracking on

        if (~samplePath2 == ~networkPerformanceFiles[5],
        {
        ~randomRelease1 = [rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0,
2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0), rrand(0.0, 2.0),
rrand(0.0, 2.0), rrand(0.0, 2.0)];
        ~grainRelease1Slider.valueAction = ~randomRelease1/2;
        ~randomRelease1Array = Array.fill(13, {arg i;
        ~envelopeSynth.set("grain1Release" ++ i.asString, ~attackSpec0.map(~randomRelease1[i].value));
        });

        ~algorithmButton0[5].valueAction = 1;
        ~pitchTrackButton0.valueAction = 1;

        };
        );

//if the sample file matches the seventh file of the folder, set random threshold for control source one

        if (~samplePath2 == ~networkPerformanceFiles[6],
        {
        ~randomThreshold0 = [rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0,
1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0),
rrand(0.0, 1.0), rrand(0.0, 1.0)];
        ~onsetThresholdSlider0.valueAction = ~randomThreshold0;
        ~randomThreshold0Array = Array.fill(13, {arg i;
        ~analysisSynth.set("osc0OnsetThreshold" ++ i.asString,
~thresholdSpec0.map(~randomThreshold0[i].value));
        });
        };

        ~algorithmButton0[1].valueAction = 1;

        );

//if the sample file matches the eighth file of the folder, set random thresholds for control source two,
reset pitches to 1 for control one and two, and turn of pitch tracking

        if (~samplePath2 == ~networkPerformanceFiles[7],
        {
        ~randomThreshold1 = [rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0,
1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0), rrand(0.0, 1.0),
rrand(0.0, 1.0), rrand(0.0, 1.0)];
        ~onsetThresholdSlider1.valueAction = ~randomThreshold1;
        ~randomThreshold1Array = Array.fill(13, {arg i;
        ~analysisSynth.set("osc1OnsetThreshold" ++ i.asString,
~thresholdSpec0.map(~randomThreshold1[i].value));
        });

        ~onsetChooser2.valueAction = 1;
        ~pitchTrackButton0.valueAction = 0;

        ~resetPitchArray0 = [1,1,1,1,1,1,1,1,1,1,1,1,1];
```

```
        ~pitch0Slider0.valueAction = (~resetPitchArray0 + 4)/8;
        ~resetPitch0Array = Array.fill(13, {arg i;
        ~granularMacroSynth0.set("grain0Pitch" ++ i.asString, ~resetPitchArray0[i].value);
        });

        ~resetPitchArray1 = [1,1,1,1,1,1,1,1,1,1,1,1,1];
        ~pitch1Slider0.valueAction = (~resetPitchArray1 + 4)/8;
        ~resetPitch1Array = Array.fill(13, {arg i;
        ~granularMacroSynth1.set("grain1Pitch" ++ i.asString, ~resetPitchArray1[i].value);
        });

        };
        );

        durations.wait;

});
        });
```

John Snijders' Evaluation Feedback

***Genesis* Evaluation Questionnaire**
1) How would you describe your interaction with *Genesis*?
The interaction was actually fairly limited and more or less one-directional. I influenced what came out of the system, but the system only had a limited way of influencing me. It could only influence me indirectly by making me react musically to what I heard coming from the speakers, but could not do anything about the sounds I made.
2) Which aspects of performing with the *Genesis* system did you enjoy?
It was enjoyable to try and see how different sounds would influence results from the system, but this was more or less a passive thing. Otherwise it was just a pleasant way to improvise with a real time sounds system.
3) In relation to your musical background, how familiar and accessible was your interaction with *Genesis*?
Access was very easy in so far that I just had to play my instrument and was not really involved in any of the technical aspects of the system. As the interaction was more or less one-sided, I did not really experience any difficulties or issues.
4) How would you describe the performance capabilities of the *Genesis* system to generate reasoned musical responses to your interactions?
It seemed to react directly to what I was doing, but in a fairly onedimensional way. It

would rarely, if at all, do something counterintuitive, and I think this shows its origin as a machine. It might be interesting if there could be more AI involved, getting the system to learn as it goes along, and react in different ways, to try things out, as it were.
5) To what extent do you feel the musical paradigms implemented in *Genesis* are beneficial to your creative approach?
In its current state the system is fun to play with, mainly to see what the computer comes up with as a result of the live performer's input. Other than that, the creative approach the live performer needs to take is mostly one that searches for the musical input that generates the most interesting musical output from the system.
6) In terms of musicality, how successful do you consider the overall output of your performance with *Genesis*?
This is hard to say as I could not really hear the overall result very well, due to the fact that the speakers were positioned in such a way that I could only hear a general sound world and the total result was out of my auditory purview.
7) To what extent did you feel *Genesis* introduces a novel method of live performance and why?
It builds on work that has been done at STEIM and other laboratories for sound development, but due to a limited knowledge of that particular part of the music world, I cannot say how novel the approach is.
8) To what extent did you feel in control of the performance process and why?
I controlled my own part, but not the computer's part. It reacted to what I played but beyond my control. I could just react to its reactions.
9) How would you describe the creativity of the responses generated by *Genesis*?
It is in the early stages of being creative. Perhaps by making it more aware, and more experimental in its choices and decisions. Machines tend not to take risks, because they are not aware of the concept. The introduction of AI might be a good step forward in producing a system that can work together with the live performer as either one meta-instrument or two improvisers working closely together.
For the following questions, please state to what extent you agree with the following statements and add a comment justifying your response.
10) I would like to perform again with *Genesis* system:

Neither Agree/Disagree
Comment:
Perhaps if it is more developed it would be good to give it another go. As it stands the result was all right, but not so musically interesting that I would go on tour with it.
11) The *Genesis* system provided creative outputs that were relatable to my inputs:
Disagree
Comment:
The system definitely made decisions that were clearly related to my inputs. Whether or not they can be called creative is a whole other discussion. I would say this was not the case.
12) The real-time methodology of *Genesis* contributed positively to my performance process:
Disagree
Comment:
Basically, as I could not hear the output very well I was only marginally influenced by it and mostly went my own way.
13) *Genesis* features a significantly large musical exploratory space:
Disagree
Comment:
Not yet. It needs a lot more difference in its algorithms, and as far as I am aware, the system does not work well with monophone instruments, and needs more complex sounds to work with to be able to produce good and interesting results.
14) The graphical user interface of *Genesis* is intuitive and easy-to-use:
Neither Agree/Disagree
Comment:
I was not involved with this part of *Genesis*.
15) *Genesis* adapts its creative outputs over time:
Neither Agree/Disagree
Comment:
Probably yes, but not in an intelligent way. It was more like it adapted its algorithms.
16) I felt engaged with *Genesis* during the performance process:
Disagree

Comment:
No, I played, and *Genesis* did what it did, but there was no real sense of engagement. Perhaps over time this can grow, as one gets more used to and familiar with the specifics of the system. Just from one single session this was not yet the case.
17) I would liken the outputs of *Genesis* to those of a human being:
Strongly Disagree
Comment:
Apart from the question if this is desired, it is very clear that the output is generated by a machine that has no real intelligence, cannot make real artistic choices and is governed by algorithms that are not sophisticated enough yet to be comparable to the decisions made by a human being. One big difference is that the machine does not have a concept of risk and adventure and cannot go beyond what it is taught to do. It would rarely, if at all, do something counterintuitive, and I think this shows its origin as a machine.
18) Please add here any further comments you may have regarding your experience with *Genesis*:

Assessment

Following discussion directly after the *free exploration*, both performers (Julian Lywood Mulcock and John Snijders) were satisfied with the resulting composition. Indeed, audience feedback was also encouraging, with many questions raised regarding the intelligence of the *Genesis* system; when I replied that no specific neural network or explicit artificial intelligence methodologies had been implemented, one questioner (Dr Sam Hayden) was very surprised. Considering this was the first time that both performers had worked together, and the first time *Genesis* had been applied outside of a studio setting, the composition demonstrated convincingly the potential of *Genesis* to function, in real-time, with a number of performers, control sources and networked instances.

Prior to the concert performance, a *guided exploration* was carried out, in which John Snijders was introduced to the fundamental principles of *Genesis*, and Julian Lywood

Mulcock discussed the experience John Snijders had with electroacoustic methodologies and his preference in piano performance approaches. During *guided exploration*, John Snijders was very keen to explore *Genesis*; he was genuinely interested in the technology and the algorithmic methodologies involved. The resulting conversations enabled mutual understanding of each other's objectives and musical knowledge. Following the conversation and *guided exploration*, a *free exploration* took place, which allowed both performers to experience *Genesis* in a real-time situation with others for the first time.

It was decided that for the *free exploration* a loose structure would be applied, in which John Snijders would alter the types of sounds he would generate with the piano over a specified duration. In addition, the sound sets for the *unsupervised* instances of *Genesis* were decided, along with the durations they would be enabled for. The resulting composition appears to demonstrate a progression of musical discourse by all of the performers (human and computer), creating a satisfying ebb and flow in musical trajectory. Indeed, during the post-performance discussion with the audience, one comment highlighted the appearance of this phenomenon. This would appear to reflect the structure of the inputs by John Snijders, and highlighting the principle of *Genesis* to follow the sonic features of a real-time input source.

This was the first time *Genesis* was used in a concert scenario and with a live instrumentalist, and the feedback from both the performer and the audience was very positive. Yet, time constraints (four hours including *guided/free exploration* and the concert performance) were a limiting factor in how many *Genesis* features could be discussed and implemented, and should I have the opportunity to work with John Snijders again, I would attempt to arrange significantly more rehearsal time. Indeed, many of the comments in John Snijders' evaluation could have been resolved/tested should time have been on our side. However, I feel this does not detract from the relative success of the *free exploration*.

As noted, not all features in *Genesis* were discussed or implemented, nor would this have been practical in the time available. In particular, due to the time constraints, the *Call and Response* feature was not used as the amplitude level of the piano input from

the contact microphone was very dynamic. In hindsight, this could have been resolved by having a slider in *Genesis* to acutely adjust the input levels, in order to find an optimum amplitude from which to trigger the *Call and Response* feature, and such feedback is invaluable for future research and development. However, a significant number of features were implemented over the course of the performance including *Pitch Track*, *Spectral Following*, network functionality, the modified genetic algorithm, envelope following, and looped routines for the *unsupervised* instances of *Genesis*, the results of which were generally as I expected.

Considering the key issues raised in chapter 4 *Interactivity with Digital Music Systems*, the time constraints did highlight the value of the low-entry fee: John Snijders stated in his evaluation feedback that access to the system was very easy, and that he was not really involved with any of the technical aspects of the system. Furthermore, he states the system was fun to play with, and allows the instrumentalist to see what the computer generates as a result of their input. Moreover, in conversation with John Snijders, we discussed his familiarity of working with algorithmic systems, which he stated was fairly limited, and confirmed by his response to question 7. Therefore, the musical paradigms applied in *Genesis* appear to offer successfully considerable accessibility to performing with the system.

Disappointingly, and due to constraints on time and arrangement of the piano/speakers in the Durham University Concert Hall, we were unable to reposition the speaker/piano set-up, which would have enabled to John to hear the overall output of the performance; an issue he raises when considering how successful the *free exploration* was. However, as stated, in conversation after the performance, he was generally satisfied with the result. I did provide John with a video link to watch the performance, but based on his response to the question 6, it would appear he did not have an opportunity too.

In terms of the real-time functionality, and again due to the limitations of the speaker/piano set-up, John was unable to hear the outputs of *Genesis* so only marginally modified his compositional process during the performance, as stated in his response to question 12. As a result, *Genesis* had a low-level of influence on John

during in this performance. Should we work together again, the placement of speakers would be paramount, thereby allowing John to be immersed in the soundscape, as opposed to being disjointed from it. This difficulty raises a more general question that is central to many live improvisational situations, that is how to achieve an optimum balance between performance and feedback sound levels.

The interaction method that John describes in his responses to questions 1, 2, 4, 6, 8, 12, 16 fits the *Conductor Model* defined by Winkler (2001) in which John was the master, with the other performers globally following his lead. In the concert performance instance it was indeed the case that the limited scope of the responses John was able to hear relative to his input would have significantly impacted on his ability to improvise and interact with *Genesis* in a more dynamic and fluent manner. Interestingly, John states in response to question 1, *Genesis* 'could not do anything about the sounds I made', which considering the current architecture of the system would be an impossibility but, extension of *Genesis* by implementation of hardware that could physically alter an instrument is a very exciting thought (perhaps Arduino[245] could be a start).

With regard to the creativity demonstrated by the system, John states he feels the system is in the 'early stages of being creative'. On elaborating his statement, John considers that 'Machines tend not to take risks, because they are not aware of the concept' concluding that introducing more AI may be a way of introducing further creativity. Furthermore, John also affirms that the system was making decisions that were clearly related to his inputs, so this implies that a form of intelligence, although primitive, is present in *Genesis*. Indeed, considering the current architecture of *Genesis* and the algorithmic methodologies implemented, as discussed in chapter 5 *The Genesis System*, further AI can be applied to increase the perceived level of creativity such as a neural network which learns a performer's style and consequently adapts its outputs. However, I am encouraged that John still perceives that *Genesis* demonstrates a level of intelligence and creativity in its present state.

[245] Arduino.cc, (2015). *Arduino - Home.* [online] Available at: http://www.arduino.cc [Accessed Jun. 2014]

Overall, the time/speaker constraints certainly impacted on the experience John had with the system (as he acknowledges), and in terms of engaging with the system John states 'perhaps over time this can grow, as one gets more used to and familiar with the specifics of the system. Just from one single session, this was not yet the case'. Indeed, although John states that *Genesis* does 'not yet' feature a large exploratory musical space, one performance with a distinct set of extended piano techniques cannot realistically demonstrate the extent of the musicality of the system.

Shelly Knotts

Introduction to the accompanying video documentation

Example 1

The *Shelly Knotts Free Exploration 1.mov* video displays one *unsupervised* instance of *Genesis* and a live electronic violin performer (Shelly Knotts), along with a stereo recording of the output.

The example, recorded in the Durham University Music Department studios on the 5th of March 2014, explicitly implements the *Call and Response*, *Markov Chain* and *fractal process* functionality of the *Genesis* system, with the electronic violin performer providing the initial *Call* material followed by a *Response* generated by *Genesis* in real-time.

The *unsupervised* instance of *Genesis* applies the sonic features of the violin performer's outputs to define the duration of each Markov chain modification to the rate, duration, threshold, attack and release of the granular synthesisers controlled by the *control* source one.

The electronic violin performer toggles the *Call and Response* functionality through the *space bar*, and provides all output audio data, along with the onsets for the granular synthesisers controlled by *control* source one, two and three.

Considering the modes of interaction provided in chapter 5 *The Genesis System*, this example demonstrates an *unsupervised* method of interaction with *Genesis*, with applied improvisation by a live instrumentalist. (*note* the requirement of using a space bar for triggering of *Call and Response* by the human performer was required due to a clicking in the signal from the electronic which was unintentionally triggering the *Call and Response* function). *Figure 55* below illustrates the interaction methodology of the performance:

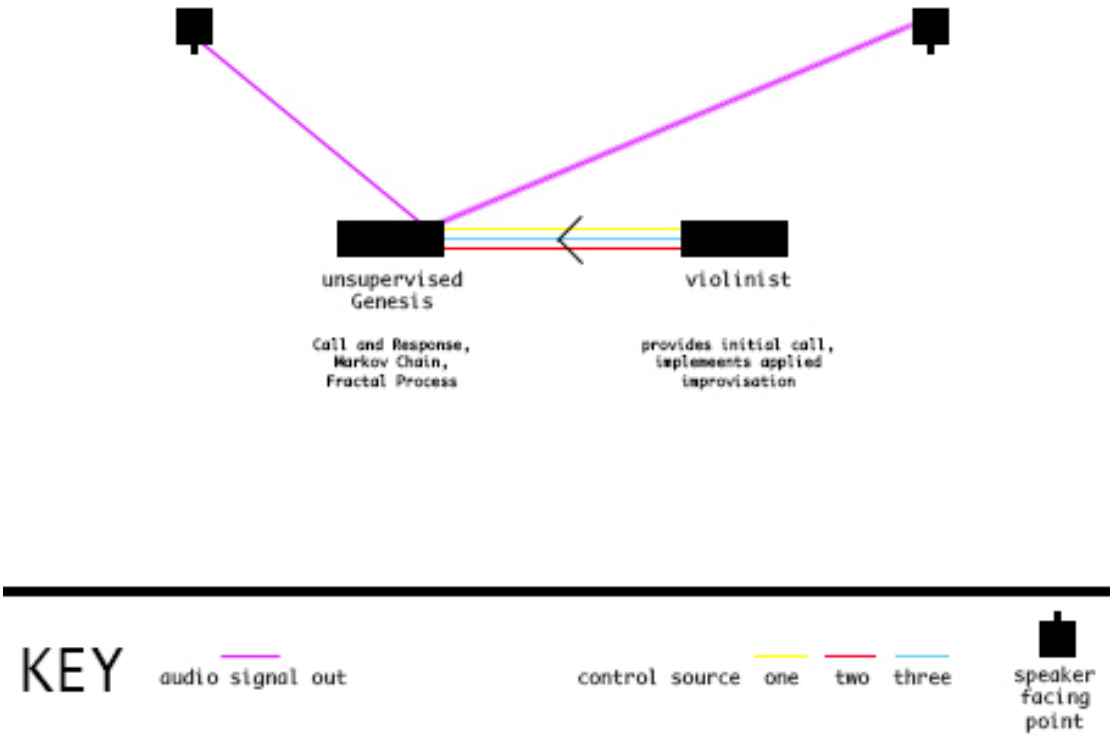


Figure 55. Performance interaction with Shelly Knotts (unsupervised)

Example 2

The *Shelly Knotts Free Exploration 2.mov* video displays one *supervised* instance of *Genesis* (Julian Lywood Mulcock) and a live violin performer (Shelly Knotts), along with a stereo recording of the output.

The example, recorded in the Durham University Music Department studios on the 5th of March 2014. The *supervised* instance of *Genesis* applies the onsets of the

electronic performer's outputs, with manipulation of the fundamental granular synthesiser parameters (rate, duration, threshold, attack and release) applied by the human supervisor.

Considering the modes of interaction provided in chapter 5 *The Genesis System*, this example demonstrates a *supervised improvisation* method of interaction. *Figure 56* below illustrates the interaction methodology of the performance:

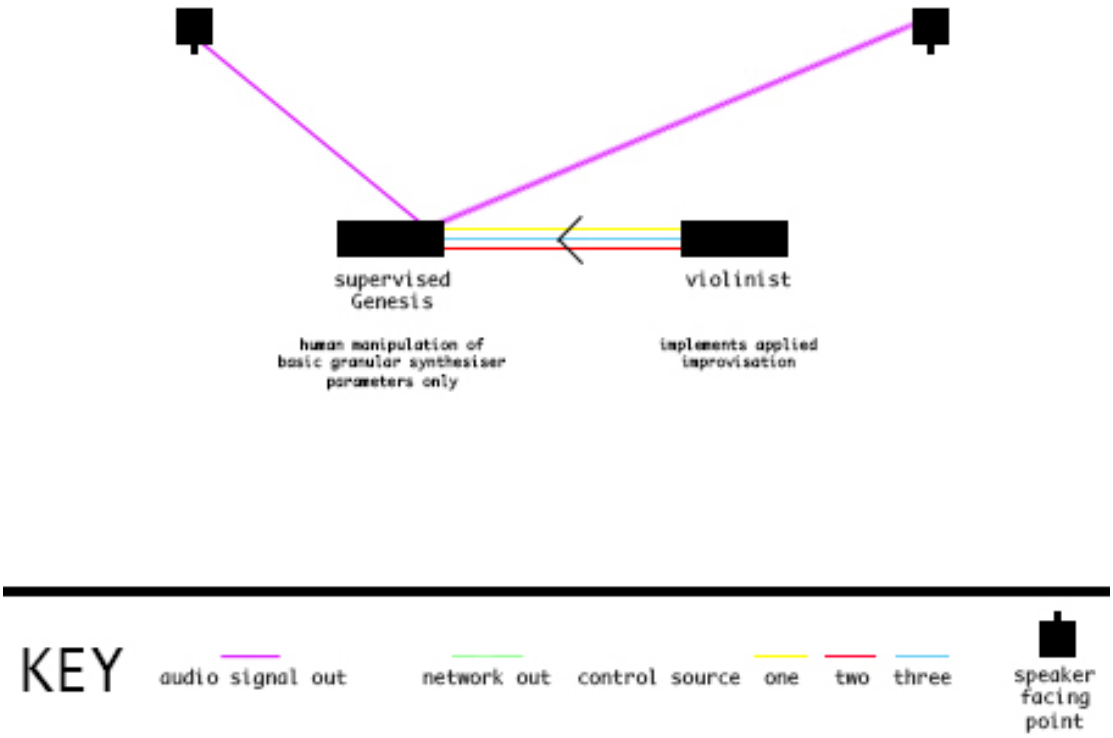


Figure 56. Performance interaction with Shelly Knotts (supervised)

Example 3

The *Shelly Knotts Free Exploration 3.mov* video displays two *networked supervised* instances of *Genesis*, and a stereo recording of the output.

The example, recorded in the Durham University Concert Hall on the 26th of February 2014, makes explicit use of the *Network* functionality. The *Genesis* system supervised

by Julian Lywood Mulcock provides the *sender* data, with the *Genesis* system supervised by Shelly Knotts acting as the *receiver*.

The *sender* instance of *Genesis* provides onset control data for control source one/two on the *receiver* instance, with each supervisor generating their own GUI modifications to their respective systems and each instance holding their local output audio sources for *control* source three/*slave*.

Considering the modes of interaction provided in chapter 5 *The Genesis System*, this example demonstrates a *supervised improvisation ensemble network*. *Figure 57* below illustrates the interaction methodology of the performance:

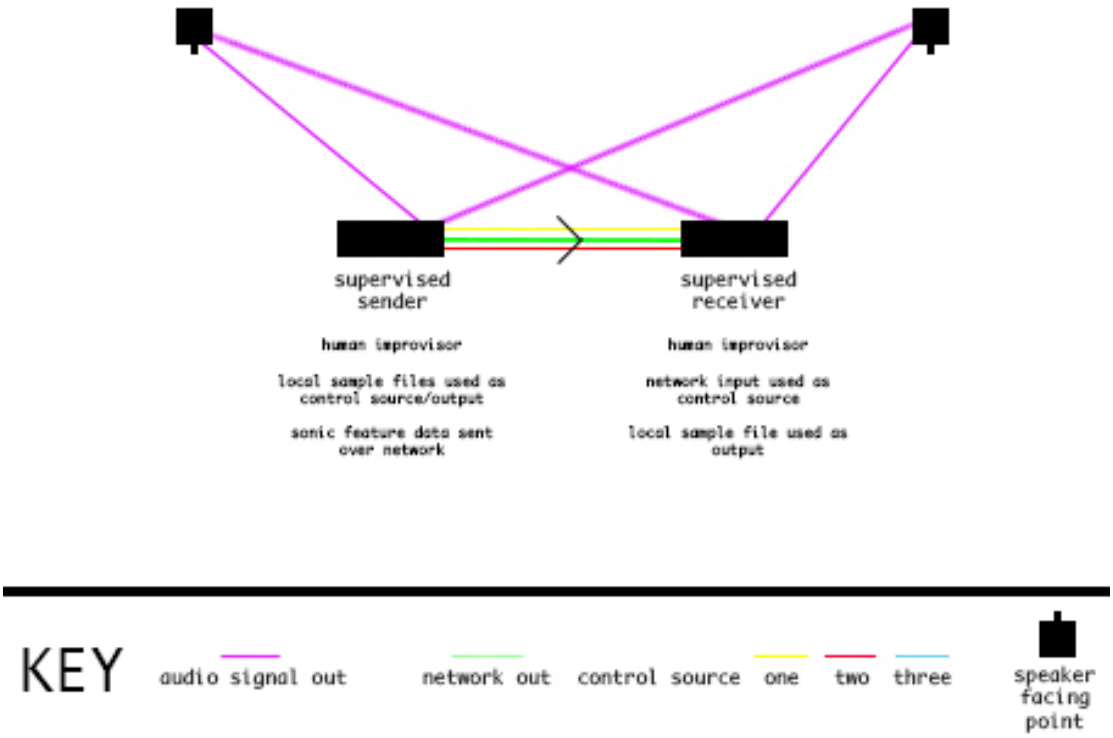


Figure 57. Performance interaction with Shelly Knotts (networked)

Example 4

The *Shelly Knotts Free Exploration 4.mov* video displays two *individual supervised* instances of *Genesis*, and a stereo recording of the output.

The example, recorded in the Durham University Concert Hall on the 26th of February 2014, makes explicit use of the *Pitch Tracking* functionality. Each system is *supervised* individually, with no network functionality. The system supervised by Julian Lywood Mulcock has the *Pitch Follow* function toggled *on*, while the system supervised by Shelly Knotts has the *Pitch Follow* function toggled *off*.

Each system has their own local *control* sources, based upon pre-defined sample material, with only the principle granular synthesiser parameters modified by the two *supervisors*.

Considering the modes of interaction provided in chapter 5 *The Genesis System*, this example demonstrates a *supervised improvisation* by each performer. *Figure 58* below illustrates the interaction methodology of the performance:

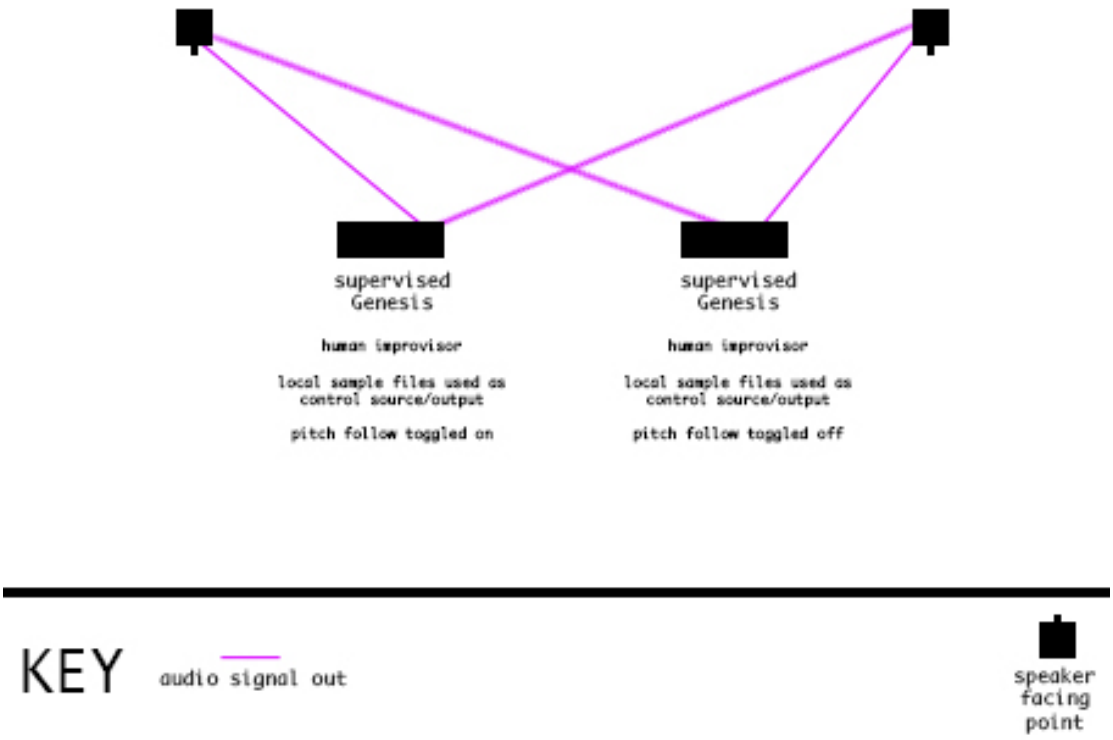


Figure 58. Performance interaction with Shelly Knotts (non-networked)

Shelly Knotts' Evaluation Feedback

***Genesis* Evaluation Questionnaire**
1) How would you describe your interaction with *Genesis*?
Laptop: I found the interface itself a little difficult to use - although there were some tech problems with the screen resolution - it felt quite cluttered and there were a lot of controls which I felt was quite a lot to deal with. Maybe in the next version there could be different pages for different types of controls so it's not all squashed onto one page and which controls do what could be a bit clearer. Violin: In the call and response performance it would have been better to have a foot pedal for controlling the interaction with the laptop.
2) Which aspects of performing with the *Genesis* system did you enjoy?
Laptop: I enjoyed playing around and experimenting with a new system. In the most part I liked the sounds produced. Violin: I felt like the system was reasonably responsive to my playing.
3) In relation to your musical background, how familiar and accessible was your interaction with *Genesis*?
Laptop: My background is in laptop performance, so the types of processes the sound world used where familiar to me. Despite this, the interface is very complex so I feel that it is a system that would take some time and experimentation to learn and isn't necessarily immediately accessible to experienced laptop performers. Violin: Although I am experienced in playing contemporary and improvised music at an amateur level I have not performed as a violinist with generative/live electronic systems on one previous occasion. However the setup was very straight forward and easy to use/access.
4) How would you describe the performance capabilities of the *Genesis* system to generate reasoned musical responses to your interactions?

Laptop: There seemed to be meaningful musical results from the interaction with the instrument but I was not always aware of the causality of resulting sound. Violin: Mixed. The responses were interesting and clearly showed some relation to what I was playing. However I felt like the 'types' of response were somehow limited, the laptop seemed to often play with slowly pitch shifting samples which I felt was a little too directive in terms of influencing what I played in response. Also the computer part was more often than not very dense and on some occasions I felt that it did not respond very well to the type of input I was giving it e.g. playing a simple pitched sound and getting a very dense textural response.
5) To what extent do you feel the musical paradigms implemented in *Genesis* are beneficial to your creative approach?
6) In terms of musicality, how successful do you consider the overall output of your performance with *Genesis*?
Laptop: Listening back to the recordings I wasn't entirely happy with the musical output, the music doesn't have much variation in density and tends to be quite slow in moving to new musical ideas. Having only played a few times with the system I cannot say whether this is due to limitations of the system or my inexperience in performing with it. Violin: The violin improvisations were much more musically satisfying I felt like they had a nice shape and that the system was quicker in moving to new musical spaces than in the laptop improvisations.
7) To what extent did you feel *Genesis* introduces a novel method of live performance and why?
As far as I am aware generative music improvisation systems have been around since the 1980's, but I do not have enough knowledge in this area to say how *Genesis* differs from other systems- perhaps the networked aspect of the system with multiple laptops in combination with the generative system is relatively novel.

8) To what extent did you feel in control of the performance process and why?
Laptop: I wasn't entirely aware of how changing controls impacted the sound output of the system and often could not tell which sounds were mine and which were those of the other performer. In some cases I was able to manipulate the sounds in the way I intended, but mostly I didn't feel in control of the system. In the networked improvisation I remember that I felt limited and not entirely happy with the type of interaction. Violin: I felt as though the system did respond to the way I played but perhaps not as fully as I would have liked and in some cases I felt a little restricted by the system and like I was following the system rather than the other way round.
9) How would you describe the creativity of the responses generated by *Genesis*?
For the following questions, please state to what extent you agree with the following statements and add a comment justifying your response.
10) I would like to perform again with *Genesis* system:
Agree
Comment:
I'd be interested in looking further into the capabilities of the system as a laptop performance tool as I didn't feel like I got to grips with the system during our performance session.
11) The *Genesis* system provided creative outputs that were relatable to my inputs:
Agree
Comment:
In both cases the output of the system had a clear relation to my actions.
12) The real-time methodology of *Genesis* contributed positively to my performance process:
Strongly Agree Agree Neither Agree/Disagree Disagree Strongly Disagree
Comment:

13) *Genesis* features a significantly large musical exploratory space:
Neither Agree/Disagree
Comment:
I felt the musical space had a particular character and, although there were large variations within this space/character, I'm not entirely convinced that significantly different results - in terms of overall impression - would occur in repeat performances.
14) The graphical user interface of *Genesis* is intuitive and easy-to-use:
Disagree
Comment:
I found the interface quite difficult to use, overly cluttered and perhaps could be arranged in a more intuitive way with better work-flow.
15) *Genesis* adapts its creative outputs over time:
Neither Agree/Disagree
Comment:
I would have to explore the system further to be able to answer this question.
16) I felt engaged with *Genesis* during the performance process:
Agree
Comment:
I found the system very engaging to work with in both cases and enjoyed the process of 'working out' how the system would respond to my inputs.
17) I would liken the outputs of *Genesis* to those of a human being:
Neither Agree/Disagree
Comment:

I felt the violin performances in particular had a musical shape, but that a human being would have a more diverse set of responses.
18) Please add here any further comments you may have regarding your experience with *Genesis*:

Assessment

In order to obtain evaluation feedback for specific functionalities of *Genesis*, I approached Shelly Knotts, a live coder and electroacoustic musician. We arranged to test two main interaction methods with *Genesis*, with Shelly as an instrumentalist, and Shelly as a supervisor of a *Genesis* instance. With Shelly as an instrumentalist, we decided to implement a *supervised* (*Shelly Knotts Free Exploration 1.mov)* and *unsupervised* (*Shelly Knotts Free Exploration 2.mov*) studio session, and as a *supervisor,* we decided to implement a networked (*Shelly Knotts Free Exploration 3.mov*) and non-networked (*Shelly Knotts Free Exploration 4.mov*) *Genesis* duo.

In the performances with Shelly as a supervisor during *guided exploration* of the *Genesis* GUI and principles, it became evident that significant time would be needed to detail the low-level algorithmic methodologies in *Genesis* and how they could be controlled through the GUI. Considering the time constraints, only a brief presentation of the GUI and its functionality was possible, which inevitably limited Shelly's level of understanding of the *Genesis* GUI. However, in terms of the principles of *Genesis*, Shelly did show an assured grasp of what the system was doing and how the real-time sound-objects generated the system's outputs. When Shelly performed as an instrumentalist, during the *guided exploration* there was a noticeable difference in the ease at which she was able to interact and play with the system; straightaway, Shelly was creating and generating music with the system.

The aim of the *unsupervised* session (*Shelly Knotts Free Exploration 1.mov*), with Shelly as an instrumentalist, was to test, for the first time, *Genesis* running

unsupervised with a human instrumentalist providing all of its data, resulting in the proposed *unsupervised* interaction methodology, detailed in chapter 5 *The Genesis System*. The *Call and Response* and *Markov Chain* functionalities were turned *on* for the duration of the *free exploration*. The results of this test were very encouraging, demonstrating the ability of *Genesis* to be engaging and musical while *unsupervised*.

For comparison with the *unsupervised* session, a *supervised* session (*Shelly Knotts Free Exploration 2.mov*), with Shelly as an instrumentalist, and Julian as a supervisor was arranged. In the *free exploration*, the *Call and Response* and *Markov Chain* functions are turned *off*, in order to demonstrate their role in the *unsupervised* interaction method. Therefore, all algorithmic modifications are made through the GUI by the human supervisor and the sonic inputs of the instrumentalist. In terms of the resulting composition, I prefer the *unsupervised* example, as I feel the *Call and Response* and *Markov Chain* functionality allow the system to successfully generate convincing compositions *unsupervised* with a human instrumentalist (despite the issue regarding unwanted triggering of the *Call and Response* functionality and requirement to break the flow of interaction by using the *space bar* instead).

As noted, due to the limited time available for the networked (*Shelly Knotts Free Exploration 3.mov*) and non-networked (*Shelly Knotts Free Exploration 4.mov*) examples, these exploratory investigations could not achieve their full potential. However, the networked example does demonstrate that using a *supervised master Genesis* instance can function with a *supervised sender Genesis* instance in real-time for generating composition, despite the limitations of Shelly's understanding of the GUI (which as noted, were primarily due to time constraints).

Although not specified by myself, Shelly conveniently separated her evaluation feedback into her experiences as a supervisor (laptop) and as an instrumentalist (violin). In general, the feedback was more positive when considering her instrumental interaction with system. However, Shelly also acknowledges the time constraints, noting that more time could have increased her understanding of the GUI. Indeed, Shelly has stated she would welcome the opportunity to work with the system again, as she felt that she had only just got to grips with *Genesis* in the allotted time.

As a result, the low-entry fee appears to be in favour of instrumental interaction with the system, as opposed to access through the GUI. Furthermore, Shelly felt the causality of responses to user interactions generated by *Genesis* were clearer when using an instrument, and that they had an explicit link to her inputs. I primarily approached Shelly in consideration that she has a strong familiarity with SuperCollider and believed this would enable her to easily access the GUI. Therefore, as a material action point for future research and development it would appear the *Genesis* GUI needs to be adjusted and simplified, and with regard to instrumentalist interaction, a foot pedal may be a stable solution of triggering the *Call and Response* function.

The way in which Shelly describes interaction with computer music systems suggests she has a personal preference to be a 'leader'. When acting as an instrumentalist, Shelly states 'in some cases I felt restricted by the system and like I was following the system rather than the other way round'. Furthermore, she describes that when *supervising* the system, occasionally she did not feel in control. Her comments appear to demonstrate that the level of influence *Genesis* has on a compositional process is dynamic, and reflects the *free improvisation* model proposed by Winkler (2001) in which 'neither performer nor computer may be "in control" but each will have some influence on how the other responds'[246].

With regard to the creativity demonstrated by the system, Shelly similarly recognizes that the time constraints did not necessarily allow for explicit investigation of the types of responses *Genesis* may create, and notes that more time with the system would potentially show the full creative ability of the system. However, Shelly agrees that creative outputs of *Genesis* were relatable to her actions, but that in the allotted time, the types of responses seemed limited.

Overall, I am pleased with the outcome of the instrumental performances and acknowledge the potential requirement to amend the GUI to enable faster access for future *Genesis* supervisors. Most of all, I am satisfied that Shelly stated "I found the

[246] Winkler, T. 2001. *Composing Interactive Music: Techniques and Ideas Using Max.* Massachusetts: MIT Press: 27

system very engaging to work with in both cases and enjoyed the process of 'working out' how the system would respond to my inputs". Furthermore, as already noted, Shelly would like to investigate further the capabilities of the system as a laptop performance tool, and I would welcome this opportunity.

Mark Carroll

Introduction to the accompanying video documentation

Example 1

The *Mark Carroll Free Exploration 1.mov* video displays one *unsupervised* instance of *Genesis* and a live electronic cello performer (Mark Carroll), along with a stereo recording of the output.

The example, recorded in the Durham University Music Department studios on the 17th of March 2014, explicitly implements the *Call and Response* and *Markov Chain* functionality of the *Genesis* system, with the electronic cello performer providing the initial *Call* material followed by a *Response* generated by *Genesis* in real-time.

The *unsupervised* instance of *Genesis* applies the sonic features of the cello performer's outputs to define the duration of each Markov chain modification to the rate, duration, threshold, attack and release of the granular synthesisers controlled by the *control* source one.

The electronic cello performer toggles the *Call and Response* functionality through a foot pedal, and provides all output audio data, along with the onsets for the granular synthesisers controlled by *control* source one, two and three.

Considering the modes of interaction provided in chapter 5 *The Genesis System*, this example demonstrates an *unsupervised* method of interaction with *Genesis*, with applied improvisation by a live instrumentalist. *Figure 59* below illustrates the interaction methodology of the performance:

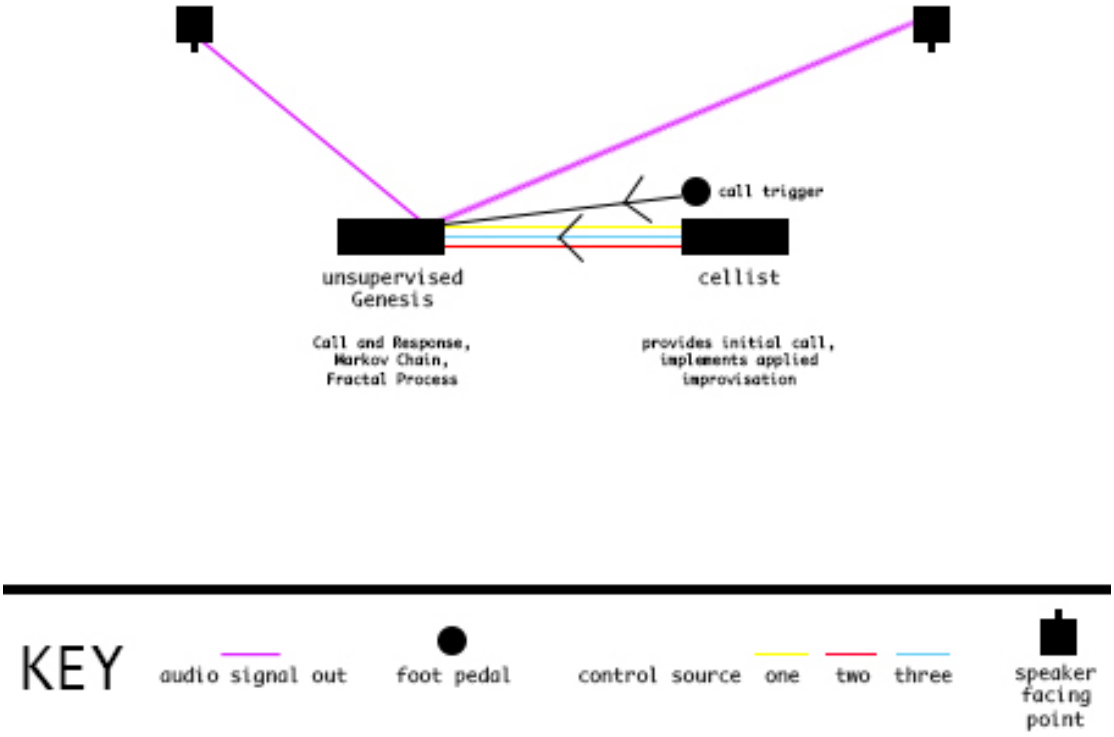


Figure 59. Performance interaction with Mark Carroll (Call and Response)

Example 2

The *Mark Carroll Free Exploration 2.mov* video displays one *supervised* instance of *Genesis* and a live electronic cello performer (Mark Carroll), along with a stereo recording of the output.

The example, recorded in the Durham University Music Department studios on the 17th of March 2014, presents an extended real-time improvisation with the system, with Mark Carroll supervising the instance of *Genesis*, modifying its outputs through the GUI as he sees fit.

The instance of *Genesis* applies the sonic features of the cello performer's outputs to define the onsets of the granular synthesisers controlled by *control* source one, two and three, along with the audio output by the system.

All modifications of the GUI are implemented by the instrumentalist, including the toggling of *Pitch Follow* and *Spectral Follow* functionalities, and overriding of the parameter settings of the granular synthesisers implemented by the *Markov Chain* manipulation of the granular synthesisers' rate, threshold, duration, attack and release. Although, Mark had the option to apply the foot pedal for the *Call and Response* functionality, the feature was not used in this performance.

Considering the modes of interaction provided in chapter 5 *The Genesis System*, this example demonstrates a *supervised improvisation* method of interaction with *Genesis*, with applied improvisation by a live instrumentalist. *Figure 60* below illustrates the interaction methodology of the performance:

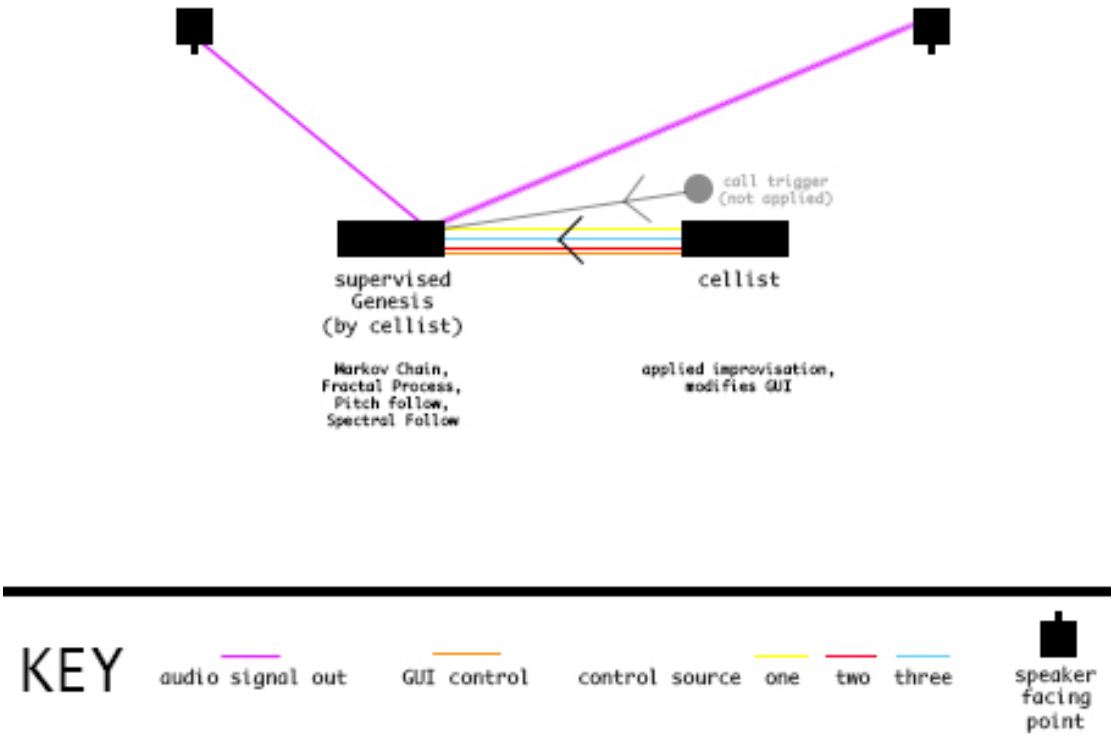


Figure 60. Performance interaction with Mark Carroll (self-supervised)

Mark Carroll's Evaluation Feedback

***Genesis* Evaluation Questionnaire**
1) How would you describe your interaction with *Genesis*?
Great learning experience. I was fairly unfamiliar with playing with an electronic system, and had no idea that they could seem to respond in a creative way.
2) Which aspects of performing with the *Genesis* system did you enjoy?
Particularly the free improvisation – treating it like another performer (but one which fortunately didn't have an opinion on what I was playing), just seeing what it did, and basing my own decisions partly on that.
3) In relation to your musical background, how familiar and accessible was your interaction with *Genesis*?
It was explained well, and the visual display was easy to get to grips with, especially for me as a newcomer to live electronics. It was easy to spot an immediate link between performing an action on the console and the resulting sound.
4) How would you describe the performance capabilities of the *Genesis* system to generate reasoned musical responses to your interactions?
Very imaginative. It genuinely seemed to be 'thinking about' what I played, and offering interesting developments and/or complementary material.
5) To what extent do you feel the musical paradigms implemented in *Genesis* are beneficial to your creative approach?
Like a good improviser, its responses to my material offered all sorts of possibilities for my own development of that material, as such I constantly felt creatively stimulated and 'encouraged' by it.
6) In terms of musicality, how successful do you consider the overall output of your performance with *Genesis*?
Fairly successful. As an inexperienced improviser-with-live-electronics, it's hard for me to judge, but I felt very satisfied.
7) To what extent did you feel *Genesis* introduces a novel method of live performance and why?
I am not terribly familiar with live electronics, but *Genesis* seems to approximate a second human improving performer, and I have not come across this before. I would

hazard a guess that this in itself is novel. It was certainly new to me, and thus exciting.
8) To what extent did you feel in control of the performance process and why?
The console allowed me to change parameters during performance, however I would have preferred to do this with a foot pedal or similar, rather than having to stop playing sometimes to operate the mouse. I felt that this impeded flow to an extent.
9) How would you describe the creativity of the responses generated by *Genesis*?
Refreshing and interesting.
For the following questions, please state to what extent you agree with the following statements and add a comment justifying your response.
10) I would like to perform again with *Genesis* system:
Strongly Agree
Comment:
I thoroughly enjoyed the freedom offered by the combination of what seemed like a 'thinking' machine improvising with me, and the lack of concern on my part about what this improvising machine thought about my own improvising.
11) The *Genesis* system provided creative outputs that were relatable to my inputs:
Agree
Comment:
There were odd times when *Genesis* seemed to offer something different, but I still felt that this fitted within the confines of a 'normal' group improvisation, and instinctively I still felt that there was a relation to my input.
12) The real-time methodology of *Genesis* contributed positively to my performance process:
Strongly Agree
Comment:
As I said before, it nicely approximated a living, improvisation partner, which I found encouraged my own creativity.
13) *Genesis* features a significantly large musical exploratory space:
Agree
Comment:
I certainly enjoyed exploring a large musical space, and felt that (e.g. in the real-time

composition) by the end, the piece had gone on a lengthy and varied journey.
14) The graphical user interface of *Genesis* is intuitive and easy-to-use:
Agree
Comment:
I have limited experience with such systems (my experience is limited to SoundLoom and SONAR), but I found it easy to use, apart from having to stop playing with one hand to operate the mouse. Having a foot pedal or similar would have really improved this.
15) *Genesis* adapts its creative outputs over time:
Agree
Comment:
There were some similarities in delay-effect texture at times, but on the whole I found a pleasing, gradual development of responses.
16) I felt engaged with *Genesis* during the performance process:
Agree
Comment:
Having to stop to use the mouse did feel somewhat like it disrupted the feeling of me improvising with another performer.
17) I would liken the outputs of *Genesis* to those of a human being:
Strongly Agree
Comment:
Interesting, complementary and stimulating.
18) Please add here any further comments you may have regarding your experience with *Genesis*:
Great fun!

Assessment

In order to obtain further evaluative feedback of specific functionalities in *Genesis*, I approached Mark Carroll, a contemporary composer and cellist. We arranged to perform with the *Genesis* system in an *unsupervised* scenario for around two hours, in

which Mark would generate two performances with the system, one explicitly applying the *Call and Response* function (*Mark Carroll Free Exploration 1.mov*), and one with the Mark supervising *Genesis* through modification of GUI during performance (*Mark Carroll Free Exploration 2.mov*). In both instances, the *Markov chain* feature is turned *on*.

During the *guided exploration*, Mark asked many questions about *Genesis*, and algorithmic composition in general. It was clear his familiarity with electroacoustic techniques was limited, however, his appreciation and acknowledgement of their validity as musical phenomena was evident. Therefore, Mark was eager to explore with the system and engage with it. While explaining the fundamental principles of *Genesis*, and as Mark began to play with the system, it was apparent that he understood the musical paradigms and immediately was able to access the system. As a result, I decided that for the *free exploration* I would leave Mark unattended and able to explore with the system in his own time and with out any external influence. On returning, Mark was very happy with the performances and discussed further algorithmic methodologies and his interest in them as compositional devices.

In terms of the low-entry fee design of *Genesis*, considering that Mark was able to work with the system unattended and was satisfied with the output, this indicates that instrumentalists who are not electroacousticians are able to grasp the basic principles of the system with relative ease. Mark acknowledges this by stating that the visual display (including the dynamic scoring system) was easy to get to grips with, despite him being a newcomer to live electronics. Furthermore, his feedback indicates the accessibility of the system's sonic outputs to performers not familiar with live electronics and the sound modifications they can create by declaring he was pleased with them.

Additionally, Mark considered the system's sonic outputs to be large, and he felt that by the end of the piece, he had 'gone on a lengthy and varied journey'. However, Mark did have issues with modifying the UI whilst performing, and suggests extending the functionality of the foot pedal for better connectivity to the system, as the current set up, with interaction required through the GUI, somewhat broke the

flow of performance. This feedback adds further weight to the feedback obtained from the other participants in terms of the desirable improvements to the user interface.

With regard to the interaction methodology with *Genesis*, Mark primarily treated and considered the system as another performer, to the extent that he felt its outputs were like those of a *human* performer. Mark consistently refers to improvisation with the system, implicating that the interaction methodology fits the *free improvisation* model (Winkler, 2001). Indeed, he notes that the system would seem to offer something different, and that he felt this fitted with the confines of a 'normal' group improvisation.

When considering the responses of *Genesis*, he described the system's outputs to be creative, to the extent of human creativity, and that the system appeared to develop its responses. This indicates that Mark felt the system was adaptive, modifying its responses over the course of the performance. Furthermore, he states that he was unaware that a live electronic music system could respond in such a creative way and that the system encouraged his own creativity, a comment that I feel positively reinforces the purpose of *Genesis* as a real-time interactive music system and the associated research.

Overall, I am greatly encouraged by Mark's feedback. Although Mark does not refer directly to the *Call and Response* function in his evaluation, the audiovisual evidence visibly shows his positive reaction to the responses generated in *Mark Carroll Free Exploration 1.mov*. Interestingly, Mark did not consider the system to be judgmental, which allowed his expressive flow to be completely free and his creative process to be positively encouraged, an aspect that makes the work on *Genesis* very rewarding.

6.3 Comparative Analysis of the Evaluative Feedback

The feedback obtained from the three participants has provided honest and, in many respects, congruent opinion of *Genesis* and its functionality as a real-time compositional tool. Where there is a divergence of opinion, there are similarly cogent

reasons why this might be so, in turn highlighting that the questionnaire was integral to obtaining this evaluative data. As stated in 6.1 *Evaluation Methodology*, sample sizes for evaluation of real-time music systems are often small due to the selected audience of such work (Wanderley and Orio, 2002) and I found this to be true when approaching performers and composers to interact with *Genesis*. However, the sample presented in this thesis reflects a diverse range of musicians, with differing experiences of real-time music systems. As a result, a broad range of opinions regarding *Genesis* was acquired.

Considering the intention of providing evidence of higher-level features of *Genesis* (as stated in section 6.1 *Evaluation Methodology*), the audiovisual examples deliver clear and fluent examples of the system in a variety of situations and interactive methodologies, proposed in chapter 5 *The Genesis System*. As noted, the purpose of focusing on higher-level functionalities and products was to act on the notions proposed by Hsu and Sosnick (2009) in order to present interaction samples centering on the creativity achievable with *Genesis* and its musicality. To that extent, the evaluative feedback, and the associated audiovisual examples demonstrate the capabilities of *Genesis* to function in real-time, with live performers, to generate satisfying musical compositions.

With regard to the evaluative feedback provided by the participants, for direct and quantitative comparison of the success of the higher-level interactive methods in *Genesis*, *Figure 61* below illustrates the variety of views expressed from each of the participants, relative to the results of the Likert-scale applied in the questionnaire:

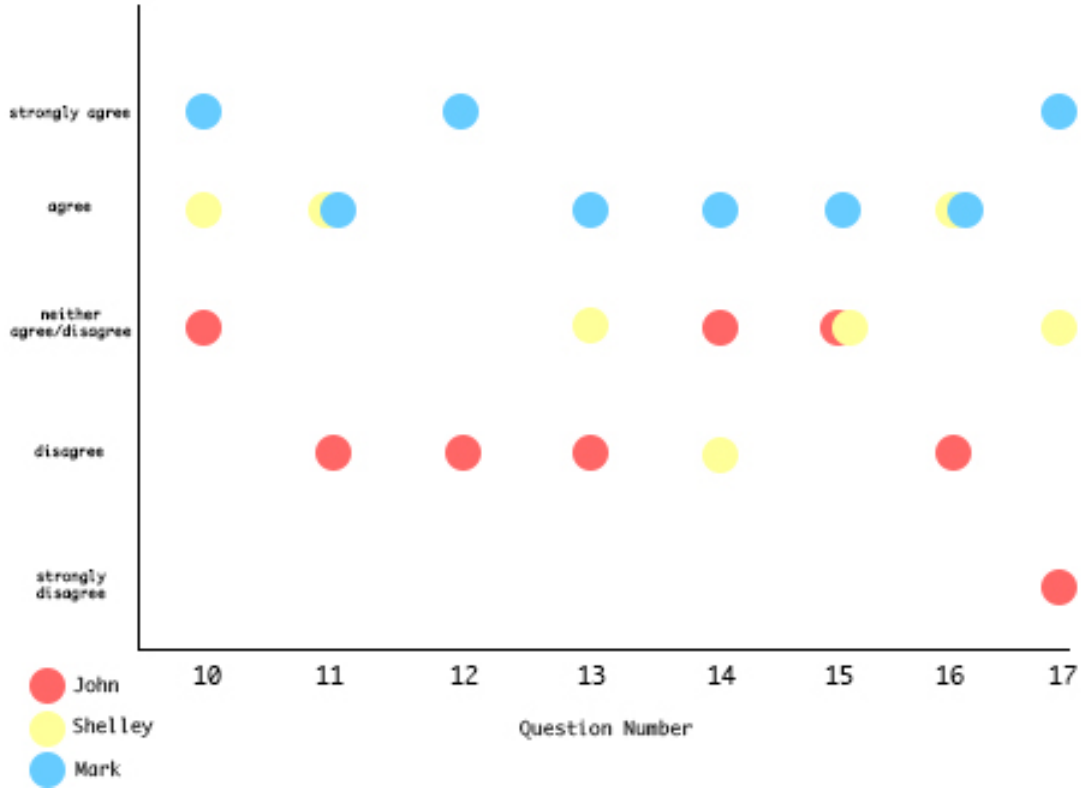


Figure 61. Likert-scale results' comparison

The results of the Likert-scale questions, when compared between participants, demonstrate the diversity of opinion expressed in the evaluation feedback of *Genesis*. In particular, the response to question 17 'I would liken the outputs of *Genesis* to those of a human being' represents great disparity between the participants' experiences with the system. However, each participant was directed through the same evaluation process (*guided exploration*, *free exploration* and the *fully-structured interview*), and each had a chance to interact with *Genesis* in a similar way (each participant interacted with at least a *supervised* and *unsupervised* instance of *Genesis* in the *guided* exploration).

Overall, I would consider the evaluation feedback to the qualitative questions of the questionnaire to be positive, particularly in terms of accessibility to the system for instrumentalists interacting through their respective instruments. The comments regarding the interactive methods and creativity with *Genesis* describe contrasting experiences. However, particularly in the case of John Snijders, there were practical factors that limited the breadth of engagement. Time constraints were a factor with

every participant (for example, explanation of GUI functionalities with Mark and Shelly had to be kept brief in the *guided exploration*). As a result, it is hoped that with increased application of *Genesis* by performers, deeper understanding of features within *Genesis*, such as the GUI and the system's interactive models, can be achieved.

It is important to note the time taken in order to complete the questionnaires; in most cases, feedback was not completed for around four to five months after each participant's interaction with the system, despite candidates receiving the questionnaire soon after performance. These significant delays had not been anticipated, and some allowance has to be made for the recall of experiences some way in the past. Furthermore, although the audiovisual examples (provided in section 6.2 *Evaluation Results*) were available via www.dropbox.com (a free file sharing site) for each participant to easily view their respective performance, it would appear that this was not taken opportunity of in all cases.

Considering further development of the evaluation methodology applied for this thesis, application of discourse analysis would obtain further useful insight into the opinions of participants using *Genesis*. Discourse analysis is the process of dissecting written text 'using a *structured method* which can take apart the language used in discourses (e.g. interviews, written works) and elucidate the connection and implications within, while remaining faithful to the content of the original text (Antaki et al. 2004)'[247]. In terms of the responses to the questionnaires provided by the participants, discourse analysis would require the provision of much more depth on their reflections than proved to be the case. Furthermore, discourse analysis would be highly applicable as part of continuing study of the characteristics of *Genesis* in the context of increasing the number of case studies using feedback on the initial trials to shape and refine the scope of the practical experiments, and also the qualitative and quantitative methods of eliciting more insightful feedback.

As noted in 6.1 *Evaluation Methodology*, there is no formalised method for approaching evaluation of real-time interactive music systems, including recommended time for participants to interact with systems, how best to obtain

[247] Stowell et al. 2009. Evaluation of live human-computer music-making: quantitative and qualitative approaches. *International Journal of Human-Computer Studies.* 67 (11): 961

feedback once performance has been completed and which levels of functionality should be approached. Therefore, with the evidence provided in the audiovisual examples and the feedback obtained from the participants in the questionnaires, a valid and reasonable performer-centric evaluation is presented by applying the methodologies suggested by Stowell at al (2009). Indeed, 'this area is underexplored and needs much more research, such as the further development of structured approaches to analysing user talk (both within and outside the traditions of Discourse Analysis)'[248], and it is intended that the evaluation method presented in this thesis is one such example of further development.

6.4 Evaluation of the *Genesis* System's Methodology

With reference to the research aims, outlined in chapter 1.3 *Aims of the* Research, and in particular, the original research contributions, considering the interactive, generative and analytical process applied within *Genesis* and described in detail in chapter 5 *The Genesis System*, *Genesis* forms a novel method of real-time musical interaction; interactive processes apply the sonic features of real-time auditory signals from any conceivable and attainable auditory source to define the values of the parameters for many generative processes with the sonic features of onset, MFCCs, pitch, tempo and loudness extracted through the analytical processes. Furthermore, the interactive processes apply extensive graphical user interface control for adjustment of the generative and analytical processes relative to a desired compositional process, thereby offering different models of interaction, which are adjustable on-the-fly and in real-time. This section evaluates the method applied in *Genesis* in terms of its efficiency, mappings, real-time interaction and the GUI, and how this relates to the algorithmic systems discussed in chapter 4 *Interactivity in Digital Music Systems*.

6.4.1 Efficiency in *Genesis*

With regards to efficiency of the system, and the importance efficiency has in real-time digital music systems (as highlighted in chapter 2.3 *Computers and Algorithms*),

[248] Stowell et al. 2009. Evaluation of live human-computer music-making: quantitative and qualitative approaches. *International Journal of Human-Computer Studies.* 67 (11): 973

this is relative to the processing power and random access memory (RAM) of the computer running an instance of *Genesis*; when all predefined interactive, generative and analytical processes are applied, the system runs at a peak average of 25% CPU on a 2012 Apple iMac. Although this represents a viable environment for field-testing the integrity of the system under various operating conditions, there are a number of material constraints to be borne in mind when engaging with more demanding situations. With the application of live coding for the generation of tasks and sound-objects, which may incur a high number of CPU processes relative to the generative task, the peak average CPU usage may rise to higher peak CPU percentages and may cause highly undesirable distortion and clipping within the auditory output of the system.

Furthermore, older computer systems run at higher peak averages with the predefined interactive, generative and analytical processes before the application of any live coded generative processes, thereby increasing the chance of errors occurring in the auditory output. As a result, a performance modifier Button has been applied within the GUI, which can be used optionally to considerably reduce the peak CPU usage, by removing the PV_Freeze process for the outputs of *all* granular synthesizers. This results in a reduction of the peak average CPU usage to 17% on a 2012 Apple iMac, substantially increasing efficiency and reducing the possibility of distortion in the auditory signal, while maintaining all other interactive, generative and analytical processes.

In addition to the consideration of different computer systems having varying CPU limitations, which may affect the quality of an auditory output, the video resolution between computers can also differ, with different video resolutions possibly causing GUI objects to be misrepresented, requiring scaling, in order to correct any issues. As the GUI within *Genesis* is an extensive and integral interactive function, it is necessary to ensure the GUI functionality is maintained on systems with different video resolutions. The standard resolution of GUI objects in *Genesis* is 1920 x 1200, which enables all text, modifiable GUI objects and UserViews to be displayed clearly, while allowing manageable interaction between the computer keyboard and mouse.

However, as noted, video resolutions on computer systems vary, and, taking this in to account, the resolution of the *Genesis* GUI objects can be adjusted through a scaler function, executed by inputting the resolution of the current computer system within a GUI NumberBox, which scales the standard resolution of *Genesis* GUI objects to the resolution of the system running the *Genesis* program. Therefore, when scaling is applied, all GUI objects are displayed within the computer system's video resolution relative to the scaling value. Nevertheless, despite the inclusion of the scaling function, the optimum resolution of *Genesis*'s GUI objects is 1920 x 1200, ensuring all GUI objects are displayed clearly and are easily modifiable by the computer keyboard and mouse, with the possibility that the display and GUI control of *Genesis* may be hindered by other video resolutions, as noted by Shelly Knotts in her evaluation feedback.

As discussed in section 4.1 *Interaction with Creative Systems*, Open Sound Control (OSC) permits the application of user-defined parameters to be broadcast locally and over a network for the control of the generative processes. So, due to OSC messaging, a system as comprehensive as *Genesis* can be constructed. The various methods of controlling OSC messages, such as the sonic features of a real-time input source, live coding and GUI objects afforded the inclusion of different models of real-time interaction within *Genesis*. In particular, the symbolic and subsymbolic representations of the sonic features of the real-time input sources, extracted through the relative analytical UGens, are allocated to OSC Message value.

Nonetheless, although analytical data from the real-time input sources is extracted at sample rate and in reference to the impact efficiency on digital music systems, the speed at which this data can be represented on a local system and/or broadcast to networked instances is limited. For example, values that are sent from the Server to the Client require a task to collect and apply the data, in addition to any modifications to the data needed for a generative or analytical process. To apply running tasks within SuperCollider, a clock must be implemented, dictating the interval of time between the function/s of each task.

Considering the GUI, a frame rate of 30 frames per second is an acceptable and efficient update speed, which necessitates a task update speed of 1/30 of a second, far slower than a typical sampling rate of 44.1kHz, which requires an update speed of 1/44100 of a second. Therefore, the representation of instantaneous events, such as the onsets of the *control* sources displayed in the GUI Buttons in the main *Genesis* window, is not fully accurate, and are only represented should the task receive the onset at the time of execution. Despite this limitation, the *Genesis* system adequately represents most values with optimal delay between the event and its representation in the GUI.

The dynamic scoring system, constructed of GUI objects dictated by the values of the *slave* sound-object's granular synthesizers' envelope, pan position, filter frequency, freeze, duration, playback rate, *control* source, *overall* loudness of each bank of granular synthesisers and buffer frame values, abstracts the current status of the interactive, generative and analytical processes. When executed, there is a clear correlation between the real-time auditory inputs' onsets, the values defined within the GUI of the generative and analytical processes and the auditory output of the *slave* sound-object's granular synthesizers, as demonstrated in the audiovisual 19. *Dynamic Scoring System* on the accompanying DVD in the *Audiovisual Example* Folder.

Although the Loudness.kr UGen is applied to measure the overall loudness of each bank of granular synthesisers in order to represent the status of each bank's loudness, further spectral analysis of each granular synthesiser's output could be applied, such as extracting their MFCCs. This would extend the capabilities of the system's representation of its sonic outputs, outside of the parameter values of each sound-object's granular synthesiser's parameter settings and a general loudness value. However, the addition of such dynamic spectral analysis would incur a significant efficiency penalty, and therefore is currently unfeasible.

When applying networked instances of *Genesis*, the latency between the systems may be noticeable in both the visual representation and auditory output of the system. This is relative to the broadcast format and the bandwidth available, with the minimum acceptable bandwidth for sending and receiving *Genesis* specific communication

being 350kps; for Wireless communication over WLAN, the signal can only travel as fast as the radio waves between the computers running the instances of *Genesis*, with optimal communication between systems relying on Ethernet cabling, increasing the maximum potential speed at which the broadcast can be sent to the speed of light.

Currently, the minimizing of latency between networked systems has no conclusive solution, as highlighted in section 4.1 *Interaction with Creative Systems*. For example, the TablaNet system (Sarkar, 2007), which is 'a real-time online musical collaboration system for the tabla'[249], attempts to minimize the effect of latency by introducing predictive algorithms that anticipate incoming network traffic but this was found to 'result in a slightly different musical experience at both ends'[250] which could be considered just as undesirable the latency itself.

As a result, until methods minimizing the occurrence of latency between networked systems can be improved, the implementation of the network functionality within *Genesis* should be carefully considering prior to performance, relative to the bandwidth and distance from the networked instances, with lower bandwidths and further distances increasing latency, and higher bandwidths and shorter distances decreasing latency. The consequence of shorter latency offers the potential for near instantaneous functioning of the interactive, generative and analytical processes between networked instances of *Genesis*, helping to maintain an instantaneous feedback loop between the real-time input source and the resulting auditory output of *Genesis*.

When engaging with a live instrumentalist, it is imperative to ensure clarity in the unfolding dialog between the interactions of the performers (in this case, *Genesis* and a live instrumentalist); considering Paine's interaction model (2002), there *must* be a direct link between the actions of the live performer and the actions of *Genesis*, through the commonly understood paradigm of onset, MFCCs, pitch, tempo and loudness. With the introduction of any unavoidable latency, this direct link may be lost as the consequence of the actions, although commonly understood, is not relative to the moment of interaction.

[249] Sarkar, M. 2007. *TablaNet: A Real-Time Online Musical Collaboration System for Indian Percussion*. MIT: 3
[250] Ibid

6.4.2 Mappings in *Genesis*

6.4.2.1 Fractal Mappings

Within the generative processes in *Genesis*, 'novel circumstance'[251] is certainly prevalent in the auditory outputs of the *slave* sound-object; the fractal processes triggered by the onsets of the *control* sources, which dictate the buffer position, playback rate, recording rate, and duration of the granular synthesizers, as described in detail in chapter 5.4 *Generative Processes in Genesis*, reflects the inclusion of indeterminate processes that generate, in real-time, parameter values mapped to the onsets of *control* sources.

Considering the fractal process of the buffer position for the granular synthesizers, this is *not* optional; due to the nature of recording live streams, the sample rate dictates the length of each buffers audio recording, so to maintain sample rate quality sound of 44.1kHz, a minimum of a one second long buffer must be applied (any shorter than one second at a sample rate of 44.1kHz, and frequency resolution is diminished). Therefore, the temporal resolution of each recording must be a minimum of one second, resulting in each granular synthesizer's buffer updating a minimum of once every second. So, the auditory outputs of the granular synthesizers are relative to their assigned buffer positions over a minimum of 44100 frames. Thus, if a buffer position of 0 is defined, the auditory output will be relative to the signal held in the buffer at its 0 frame. Now, if a buffer position of 0 is selected, and an onset from the *control* source triggers its playback, this does not ensure instantaneous playback of the real-time recorded *slave* sound-object; this is dependent on the buffer recording being at buffer position 0 at the time of the *slave* sound-object's onset. As a result, if the buffer position is any frame higher in value than 0, the resulting output will have a delayed onset relative to the number frames difference from the actual onset of the *slave* sound-object.

In order to minimize the amount of delayed onset between the *control* onsets and the onsets of a *slave* sound-object, the fractal process of the buffer position for the

[251] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49

granular synthesizers selects a buffer position relative to the length of the buffer, with the default value being 44100 frames. As a result, as demonstrated in chapter 5.4 *Generative Processes in Genesis*, the resulting buffer positions form a collage of the current buffered *slave* sound-object, dynamically changing, in real-time, the buffer positions from which the granular synthesizers playback to the bounds of brown noise, as described in section 3.1 *An Introduction to Real-time Generative Algorithmic Systems*. As a consequence, the buffer position of each granular synthesizer is indeterminately selected between the values of the maximum number of buffer frames, increasing the possibility of at least one grain (if triggered by a *control* source) allocated a buffer position close to the onset of *slave* sound-object, thereby minimizing the chance that a delayed onset may occur in comparison to the use of a fixed buffer position value such as 0 frames. This is illustrated in *Figure 62* for a *slave* sound-object onset and its consequent playback buffer positions of thirty granular synthesizers relative to their *control* source with yellow representing *control* source one, red representing *control* source two and blue representing *control* source three/*slave*:

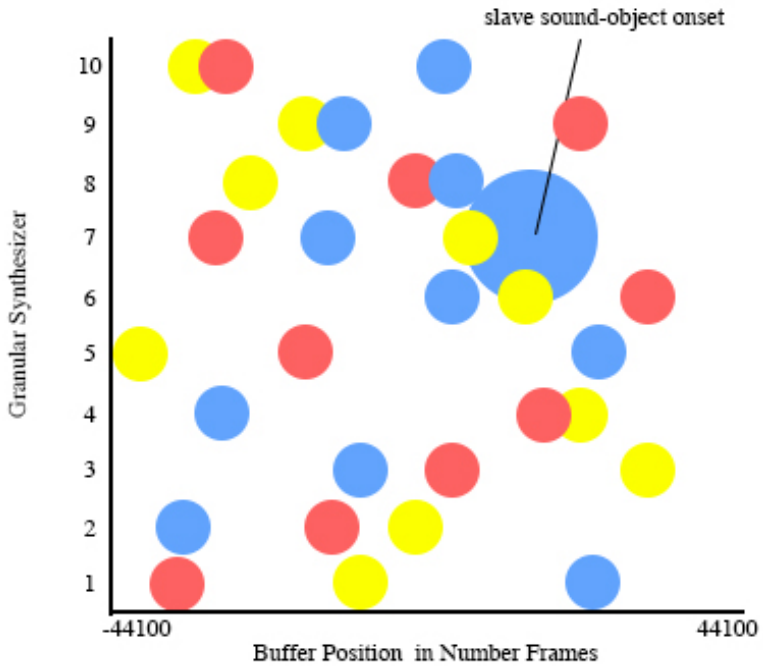


Figure 62. Fractal Buffer Positions

So, the granular synthesizers with the buffer positions closest to the *slave* sound-object onset will playback almost instantaneously the *slave* sound-object within the audio buffer.

The static and dynamic onset approaches within *Genesis*, as described in chapter 5.4 *Generative Processes in Genesis*, create a unique method of triggering generative processes. Through the static onsets, the *overall* loudness level of the *control* source is obtained, triggering the fractal and granular processes relative to the thresholds defined in the GUI. Moreover, the dynamic onsets offer a novel method of mapping the spectral data of a *control* source and applying the extracted onsets, in combination with modification of their thresholds within the GUI, to the respective fractal and granular processes. Therefore, the application of the dynamic onsets generates an auditory output for a *control* source's granular synthesizers, which reflects the spectral components of the *control* source, as opposed to its *overall* loudness level. As a result, granular synthesizers dictated by the dynamic onset method are triggered significantly more fluidly and actively than with the application of static onsets, replicating the dynamic nature of the process. This is illustrated in the *Figure 63* by representing the possible onsets of a *control* source obtained with static and dynamic onset methods over time applying the same real-time input source for each onset extraction method:

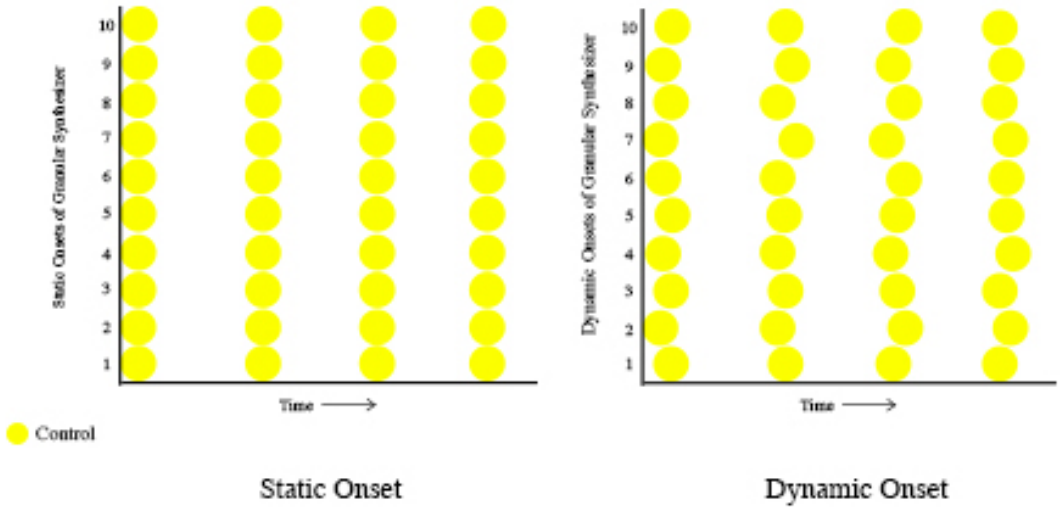


Figure 63. Static and Dynamic onsets over time

In combination with the fractal process of assigning buffer positions of the granular synthesizers, the application of dynamic onsets further alleviates the issue of incurring delayed onsets between the onset of a *slave* sound-object and the consequent instantaneous playback of that *slave* sound-object by the granular synthesizers

dictated by a *control* source; there is an increased possibility of coincidental onsets between the *control* source to the *slave* sound-object when applying the dynamic onset method. This is illustrated in *Figure 69* by representing the possible onsets of a *control* and *slave* source obtained with static and dynamic onset methods over time applying the same real-time input sources for each onset extraction method, along with the possible buffer positions of the granular synthesizers:

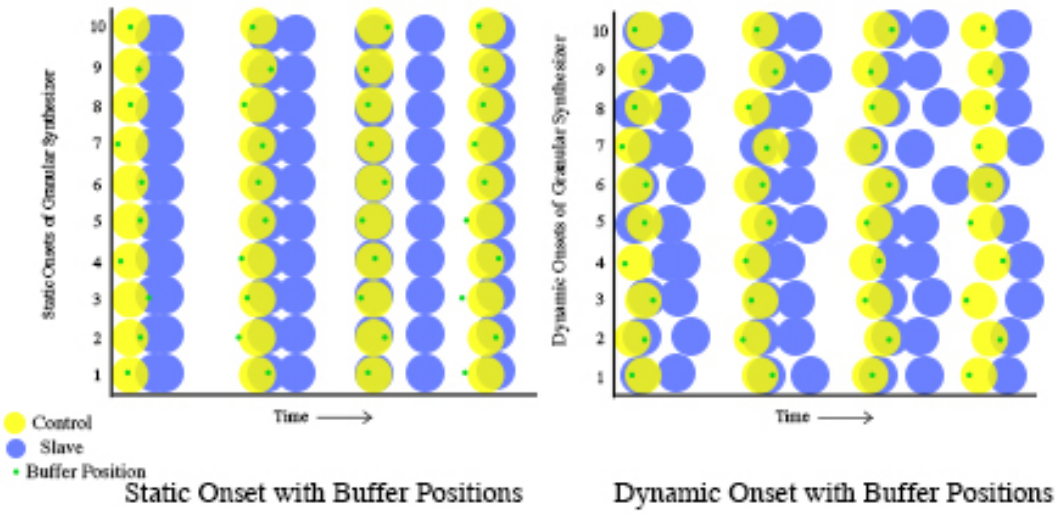


Figure 64. Possible static and dynamic onsets with buffer positions

The fractal process approach to selecting the buffer positions of the granular synthesizers, in conjunction with the optional static or dynamic onset functionality, is applied in consideration of the feasibility of controlling complex mappings, as described in chapter 2.1 *Algorithms in the Compositional Process* and to minimize delayed onsets between *control* sources and the *slave* sound-objects auditory outputs. There are 39 granular synthesizers within *Genesis*, each with ten modulatable parameters, resulting in 390 possible adjustable settings for the granular synthesizers alone, which cannot be realistically controlled in real-time through GUI interaction of each individual parameter. Furthermore, as demonstrated in *Figure 64*, the use of a fixed value for the buffer position of the granular synthesizers can incur a significant delayed onset between *control* source onsets and the onset of the *slave* sound-object. Therefore, an algorithmic process is required to dynamically alter the buffer positions of the 39 granular synthesizers. The use of a fractal process sufficiently and algorithmically controls the buffer positions of the granular synthesisers *without* requiring a significant level of CPU processing power in combination with offering

the capability to permit the real-time modification of the bounds of the process relative to the size of the buffer, in addition to real-time modification of those bounds should the buffer size be changed within a composition.

Similarly, the application of a fractal processes for the real-time generation of playback rates, recording rates and durations also offers an optional efficient and effective method of dynamically modifying selected parameters of the 39 granular synthesizers. In relation to the interaction between the real-time input sources and the triggering of the fractal processes of the granular synthesizers, this ensures the parameter values change relative to the onsets of the real-time input source, thereby helping to maintain a correlation between the interactions of the real-time input source and the resulting output of the *slave* sound-object's granular synthesizers. With regards to values produced by the fractals, they are not mapped to a specific structure other than the bounds of the respective parameters values such as -4 to 4 for playback rate. Therefore, the values of the fractal processes are not restricted by formalist structures (although it is possible to apply extensive mappings such as pitch structures relative to the playback rate to form values only within, for example, a diatonic scale, this restricts the composer to such formalist structures from the outset. If such pitch structures are desired, they can be implemented in real-time through live coding or within the computer code of the *Genesis* system).

Contrary to the use of fractal processes for the modification of selected parameters of the granular synthesizers, other generative processes may be applied, such as cellular automata or stochastic processes (described in chapter 3.1 *An Introduction to Real-time Generative Algorithmic Systems*). However, considering the efficiency with which fractal processes can be applied and their reciprocal nature, the values they produce are manageable and can be mapped with relative ease. In contrast, for example, processes such as cellular automata may generate a significant number of anomalous values, outside of the bounds of the selected parameter such as the playback rate, which may cause undesired clipping or distortion in a granular synthesizer's auditory signal, a consequence of applying computational algorithmic techniques that should be avoided, relative to the compositional process, as highlighted in chapter 2.1 *Algorithms in the Compositional Process*.

Considering the real-time functionality of *Genesis*, not only is efficiency highly important in order to minimize latency in the auditory signal, as discussed previously, but also the quality of the auditory output must remain high; the auditory output of a real-time composition must maintain an acceptable level of audio fidelity in order to ensure a clarity in the interaction between the audio signal and the processes defining any parameters that may be modifying it, otherwise the errors in the auditory signal (unless desired) cannot be removed once an occurrence of such an anomaly has taken place due to the real-time nature of the compositional process, and as a result, may affect the resulting listening experience. The application of the bounds within the fractal processes help to limit the impact of such errors within the auditory signal, while also remaining an effective method of generating novel values for the parameters they control.

With further regard to the necessity to maintain a sufficient level of audio fidelity, the fractal process dictating the buffer positions is triggered relative to the onsets identified within the *control* sources. As the envelopes of the granular synthesizers are also triggered by the relative onsets to those of the fractal process, the envelope limits the occurrence of clipping in the fractal buffer position process; the real-time modification of a buffer position can generate clicks within the audio signal, due to the sudden pressure changes in the buffered audio's waveform. The application of an envelope smooths the transition by reducing the amplitude of each grain to 0 at the time of any change in the buffer position, limiting the occurrence of instantaneous clicks in the auditory signal.

6.4.2.2 GA Mappings

The implementation of genetic algorithms for the manipulation of the settings of one set of granular synthesizers within *Genesis* offers the possibility to explore novel parameter settings, relative to the parameter settings defined by the respective GUI MultiSliderViews of the granular synthesizers controlled by the onsets of *control* source one and *control* source two. As stated in chapter 5.4 *Generative Processes in Genesis*, the fitness function of the genetic algorithms within *Genesis* is executed through the use of a human critique. Therefore, the composer, as opposed to a fitness

function within the *Genesis* system, completes the assessment of the granular synthesizers current parameter settings controlled by the genetic algorithms, with the option to 'Devolve' parameter changes to a chosen point of evolution, should an outcome or series of outcomes be rendered unsuitable to the ongoing compositional process.

As a result, the use of a human critique offers a potentially more qualitative result but less efficient fitness function than the use of an automatic fitness function, as described in chapter 3.1 *An Introduction to Real-time Generative Algorithmic Systems*. However, considering the number of parameters modified by the genetic algorithms, the variables of the parameters possible and the real-time nature of the task necessitating it is relative to the current auditory output, the construction of an automated fitness function is highly complex; with a human critique, extensive analysis of the output can be completed by the composer instantaneously as part of their intrinsic compositional process, as opposed to the requirement to assess and organise an analysis of the current auditory output by perceptual algorithm models if an automatic fitness functions were applied, possibly denigrating the quality of the result. The result of such an algorithmic analysis may also incur a significant efficiency penalty relative to the functionality of any analytical models applied, resulting in an output that may be less qualitative and less efficient than if a human critique were to be used.

An alternative approach to exploring the parameter settings through a process that can adapt its values relative to an environment is artificial neural networks. Through the application of artificial neural networks, a system can learn features of an environment and output results based on the knowledge it has acquired. When considering the complexity of generating an adequate automatic fitness function for *Genesis*, an artificial neural network offers the potential to remove the necessity to require a pre-defined organizational structure of the parameters to modify and the relative analytical outputs that define them, by forming a self-organizing map of the data it has acquired, thereby automatically generating novel outputs without necessary supervision by a human critique.

However, an artificial neural network system requires substantial training in order for it to learn and organise its networks and, when considering the number of possibilities of possible auditory outputs of the granular synthesizers controlled by the onsets of *control* source one and *control* source two, it is currently not possible to form a map suitable enough to incorporate such possibilities. Moreover, forming such a suitable map would require an extensive amount of training, which cannot be successfully completed and organised in real-time. Therefore. they are not acceptably adaptable *while* a system is running, thereby limiting its real-time interactivity. As a result, the implementation of artificial neural networks could be developed in future instances of a *Genesis* system relative to improvement in artificial neural network methodology, but currently, the application of genetic algorithms satisfactorily fulfills the role of permitting real-time exploration of a series of granular synthesizers' parameter settings within *Genesis*.

6.4.2.3 Search Mappings

In a similar vein to the application of genetic algorithms to explore 'novel circumstance'[252], the Call and Response function generates a novel auditory output, relative to the inputs it is provided with. However, unlike the implementation of genetic algorithms in *Genesis* which uses the many parameter settings of the granular synthesizers dictated by the onsets of *control* source one and two to form an output, the Call and Response function applies the sonic features of pitch, tempo and onset of the Call to determine the values of a predefined selection of rhythmic patterns and pitch structures. Therefore, the auditory outputs of the Call and Response function are generated relative to the assigned sonic features of the Call and their application to the predefined formalisms of the Response task. In order to organize the Response based upon the sonic features of the Call, a manageable random search algorithm is applied to match the extracted sonic features of the Call to an array of possibilities, which holds an array of outputs that are randomly selected to form a Response.

In relation to the structure of a Response dictated by the applied sonic features of the Call, the use of predefined arrays, in combination with the auditory signal of the

[252] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49

Response, results in a process in which the structures of a Response's formalisms are limited by the number of allocated predefined arrays; the sonic features of the Response's audio buffer are modified relative to the outputs of the predefined arrays, causing the inherent structures of the recording in the Response's audio buffer to remain in the auditory output of the process, whilst being altered by the results of the Response's selection of the predefined arrays dictated by the sonic features of the Call.

As a result, the structure of a Response's formalisms is wholly reliant on the efficacy of the analytical UGens, which extract the sonic features of pitch, tempo and onset from a Call source. As discussed in chapter 5.5 *Analytical Processes in Genesis*, the Pitch.kr and BeatTrack.kr UGens applied to obtain the pitch and tempo of the Call respectively, feature constraints, causing their outputs to not necessarily match perceived pitch and tempo of the composer. In addition, although the onsets obtained with the Onsets.kr UGen can be adjusted via thresholds in real-time through the relative MultiSliderView in the GUI, the perceived onsets may still not match those of the composer. Therefore, the structure of a Response's formalisms for the Call and Response function is reflective of the interpretation by *Genesis* of the pitch, tempo and onsets of the Call. However, considering the evaluative feedback, the relatedness of the responses of *Genesis* to the inputs of the participants was strong, indicating that such 'anomalous' results were generally accepted by the users as part of the ongoing compositional process.

Considering the random search approach applied for the Call and Response function, the sonic features obtained from the Call can be allocated promptly to select arrays that hold values with transferable musical symbolism. For example, within the Call and Response function, arrays holding duration values are applied, selected relative to the number of onsets within the call. So, if many onsets occur within a Call, this creates a 'busy' audio signal, which can be reflected by the consequent selection of an array that would mirror the 'busy' Call by applying short durations between buffer position changes in the Response. Such a random search function can be completed in real-time, by modifying the interval between the Response task relative to the duration between the buffer position changes. The use of an approach, such as

grammars, also offers the capability to apply arrays with transferable musical symbolism. However, a grammar method requires a highly abstractive, hierarchical syntax and offline method for the representation of such musical symbols, which may unnecessarily complicate the Call and Response process, without improving the efficiency or quality of the Response task.

A random search process is also applied to the selection of arbitrary values of the *slave* sound-object's Warp1.ar UGen, in conjunction with the values of filters, a reverb and panning of the overall auditory output mix, as described in chapter 5.4 *Generative Processes in Genesis*. However, unlike the random search process used within the Call and Response function, all values are selected with no explicit external modification of their values by sonic features or otherwise, other than the bounds and the intervals between the process, relative to the parameter and its minimum/maximum value allocated in the GUI. As a result, the process acts autonomously when selecting its values in real-time from the prescribed bounds of each parameter, thereby requiring no outputs from any of the analytical processes of the real-time input sources, rendering it highly efficient.

However, despite the absence of analytical processes directly influencing the outputs of the process, due to the optional application of the onsets from a *control* source triggering the envelope of the *slave* sound-object's Warp1.ar UGen and the simultaneous resetting of its buffer position relative to the value defined by the random search process, the modification of the *slave* sound-object's Warp1.ar UGen's buffer position parameter by the process *appears* to be timed in sync with the triggering of the *slave* sound-object's Warp1.ar UGen's envelope by the relative onset of a *control* source; the update of the buffer position occurs at the intervals between the process dictated by in the interval value defined GUI, but is only applied if the buffer position is reset by a *control* source onset, thereby syncing the modification with the onset from a *control* source triggering the *slave* sound-object's Warp1.ar UGen's envelope. The result of this process can be heard in audiovisual example *20. Random Search Processes* on the accompanying DVD in the *Audiovisual Examples* folder.

The spectral following process, which maps the MFCC values of the *control* source to the filter frequencies of the granular synthesizers' band-pass filters effectively and efficiently represents an overall spectral character of a *control* source on the *slave* sound-object; the sum of the MFCC values reflects the spectral density of a sound-object, with the respective mapping for each filter frequency to the MFCC sum forming a representation of a *control* source based on spectral density. As a result of the process, spectral modifications within the real-time input source can be applied to the *slave* sound-object, in real-time. However, due to the requirement of efficiency to minimize latency in the real-time interaction between the *control* source and the *slave* sound-object, significant analysis of the MFCC data is limited, resulting in a restricted ability by the process to represent subtle and discrete changes in the MFCC data. Despite this, substantial changes in a *control* source's spectral density are represented, generating a process that successfully characterizes a *control* source's *overall* spectral density, in real-time.

Similar to the spectral following process, the envelope following process also applies spectral density, but instead applies the FFTPower.kr UGen's output to represent the loudness of the *control* signal, with the resulting loudness representation used to sustain the envelope of the *slave* sound-object's Warp1.ar UGen. In addition to the loudness of the *control* signal dictating values of the envelope, the number of onsets over time from the *control* source defines the attack and release times. The principle of using onsets to represent a sound-object's envelope time's results in an output that is based upon the dynamic change in onsets over time at a described threshold. Therefore, an envelope's attack and decay times can be defined relative to the number of onsets present in an auditory signal.

In order to generate consistent and applicable values, relative to the number of onsets, a predefined selection of twelve attack and release times are applied to minimize anomalous values within the envelope following process's task with optional multiplication by a GUI Slider. The use of predefined values does considerably limit the number of possible envelope times, but anomalous values may generate undesirable auditory outputs; instantaneous envelope times, negative envelope times or unnecessarily long envelope times may be generated if no direct bounds are given,

resulting in errors in the auditory output that cannot be modified or removed in real-time. Despite the limitations of the maximum number of possible envelope times by a predefined series of values, selected relative to the number of onsets, the envelope following process efficiently and effectively envelope's the *slave* sound-object's Warp1.ar UGen relative to an *control* sources onsets, while ensuring errors within its auditory output are minimized.

In relation to the mappings within *Genesis*, the interaction they provide between the sonic features of onset, MFCCs, pitch, tempo and loudness, extracted through the analytical processes, successfully and noticeably control the parameters of the generative processes in real-time; the application of a symbolic representation of pitch and tempo, combined with the subsymbolic representation of the timbre, onsets and loudness define a system through which the sonic features of the real-time input sources can be identified in the resulting output of the *slave* sound-object and is reflected in the visualisation provided by the dynamic scoring system.

Considering the research aim of discussing the effect of interaction methodology, in terms of the models of interaction available within *Genesis* and the level of *overall* influence *Genesis* may have on a resulting composition, this can be modified on-the-fly and in real-time, dependent on the desired compositional process and the relative application of the available analytical and generative processes selected through the GUI, as demonstrated in chapter 5.3 *Interactive Processes in Genesis*.

The ability of *Genesis* to apply multiple models of interaction in real-time results in a system that is not bound to one specific model of interaction such as a Conductor Model (Winkler, 2001) combined with the use of MIDI to communicate between input and output sources, apparent in many commercial musical composition applications such as Sibelius. The advantage of the approach applied in *Genesis* offers a composer significant freedom to explore different models of interaction in real-time, relative to a desired compositional technique. For example, if applying a real-time auditory *control* source with an indeterminate conditional structure such as a live audio stream of a train station platform, the system can generate responses relative to the symbolic and subsymbolic representation of its sonic features extracted from its

auditory signal, which can be applied to a desired model of interaction such as an Improvisation Model (Winkler, 2001), through which a human controller can modify the parameters of the *slave* sound-object controlled by the indeterminate sonic features of the incoming audio stream.

In terms of the interaction between the sources providing the Call and the Response generated using the audio of the current overall auditory output mix, this is relative to its application; the source of the Call dictates the model of interaction. For example, if using a live instrumentalist to form a Call, it is possible to form an Improvisational Model (Winkler, 2001), in which the live instrumentalist and *Genesis* interact with each other, explicitly influencing each other's performances through their pitch, tempo and onsets. In addition, a live instrumentalist could apply a notated score to form a Call, with a consequent Response by *Genesis* having no explicit influence on the instrumentalist following a Call, thereby introducing a Conductor Model (Winkler, 2001) of interaction. In contrast, if applying a Sample UGen's auditory output to form a Call, it is not possible currently to modify the output of a Sample UGen without a human controller of *Genesis*. Therefore, only a Conductor Model (Winkler, 2001) can be used if a *Genesis* system is to run autonomously, creating Responses based upon a determined source, similar to the application of a notated score by a live instrumentalist for generating a Call.

6.4.3 SuperCollider, *Genesis* and the GUI

Considering the advantages and disadvantages of using music programming languages for the construction of digital music systems, and the application of SuperCollider for *Genesis*, the implementation of live coding within *Genesis* executed through the post window, or written as strings within the actions of all GUI objects for consequent application of their values via routines for the GUI Live Coding method, reflects the real-time method of interaction permitted by the SuperCollider programming language; through live coding in *Genesis*, novel sound-objects scripted for use as *control*/*slave* sources, GUI objects, *SynthDefs* and modifications of any parameter settings can be generated in real-time. Therefore, as noted previously, although an underlying primary architecture of *Genesis* is prevalent, through live

coding, it is possible to extend the fundamental architecture of *Genesis*, rendering it highly advantageous for users familiar with programming code.

In terms of the GUI objects that control the generative and analytical processes in *Genesis*, many default objects are applied such as Button, Slider and PopUpMenu. The default objects are used purposefully to offer familiarity of the *Genesis* graphical user interface to conventional GUI objects, as opposed to specifically designed abstract GUI controllers through SwingOSC. Considering the application of live coding within *Genesis*, as highlighted in section 4.1 *Interaction with Creative Systems*, a substantial amount of learning and understanding of programming languages is required to implement compositional methods that necessitate computer code. Therefore, for users of *Genesis* not familiar with such approaches, the use of recognizable and distinguishable GUI objects helps to ensure adequate control of the predefined parameters within *Genesis* without deluging the user with unknown methods of interaction.

Moreover, considering the scale of interactive, generative and analytical processes within *Genesis*, simplification of the GUI also helps to guarantee processes are clearly and consistently displayed, hopefully avoiding a misperception by the user of a GUI object and its function. This is demonstrated clearly by the responses given by the participants in the evaluation feedback. Therefore, choosing music programming languages that offer libraries of familiar GUI objects would appear to be beneficial when selecting which music programming language to use when designing an environment when instrumentalists are to use a system.

The GUI live coding method within *Genesis*, through which the GUI interactions can be live coded to a hidden post window, along with a relative clock value and wrapped as a routine, offers the capability to automate a significant number GUI controls in real-time; considering the importance of feasibility when modifying and mapping many parameters, the GUI live coding method generates an efficient and effective method of re-applying real-time GUI interactions in real-time. For example, the values of a series of granular synthesizers' playback rates, adjusted over time by GUI interactions can be re-applied via their consequent real-time allocation to a routine,

with that routine's playback executable in real-time, relative to the value of the adjustable *Genesis* clock, allowing further adjustments to be made to other GUI objects as the newly created routine is executed. As a result of the application of the GUI live coding method, real-time interactions with the GUI that control the predefined parameters of *Genesis* can be saved and applied in real-time, offering greater real-time feasibility to control many parameters simultaneously, as opposed to the requirement by systems such as Logic or Pro Tools to modify parameters offline, for consequent automation in real-time.

Furthermore, the real-time functionality of *Genesis* and the SuperCollider programming language permits the application of the live sampling method, allowing real-time recordings of the overall auditory mix and allocation of the real-time recordings to the *Genesis* Sample UGens for consequent playback and analysis through the *Genesis* system. Therefore, sound-objects can be generated, in real-time, through live coding and/or the overall auditory output mix of *Genesis*, strengthening the notion that any conceivable and attainable auditory source can be applied to the real-time input sources of *Genesis*, not only for the modification, manipulation and arrangement of the *slave* sound-object, but also to form a *control/slave* sound-object itself.

6.4.4 Quantification of *Genesis*

The fundamental application of a sound-object's sonic features for the control of other sound objects by *Genesis* is an important feature of sample-based concatenative synthesis (as introduced in chapter 4.1 *Interaction with Creative Systems*), which is 'an emerging approach to sound generation based on concatenating short audio excerpts (samples) from a database to achieve a desired sonic result given a target description (e.g., a score) or sound (Schwarz, 2000)'[253]. So, within a sample-based concatenative synthesizer, an input source's sonic features can be compared to an existing database of sounds, with the best match to the input source resulting in the

[253] Maestre et al. 2009. Expressive Concatenative Synthesis by Reusing Samples from Real Performance Recordings. *Computer Music Journal* 33(4): 24

synthesizer's auditory output. Schwarz (2006) proposed four applications of concatenative synthesis as listed in the example below[254]:

High-Level instrument synthesis - this method applies the context of a database and a target unit, thereby allowing it to create natural and seamless transitions by using its matched contexts. The result is high-level control of a synthesiser with gaps in the context filled by best-fit in the database.

Resynthesis of audio - when a sound-object is placed in the synthesiser, it is resynthesized with a sequence of best match units, compared and selected by features such as pitch, onset and amplitude.

Texture and ambience synthesis - aims to generate composition from sound libraries or pre-existing ambience recordings through extension of a soundscape for a specified duration. The process regenerates the character and flow of the ongoing composition through high-level control of its sample library.

Free synthesis – offers a composer a variety of sound databases to control by specified perceptual descriptors. As a result, the composer can explore the sound databases, synthesizing relative to high-level features such as 'bright', 'sharp' or 'wooden'.

Within each application of concatenative synthesis, the analysis of the input source and representation of the samples within the sample database of a sample-based concatenative synthesizer 'can be of type categorical (a class membership), static (a constant text or numerical value for a unit), or dynamic (varying over the duration of a unit), and from one of the following classes: category (e.g. instrument), signal, symbolic, score, perceptual, spectral, harmonic, or segment descriptors. Descriptors are usually analysed by automatic methods, but can also be given as external metadata, or supplied by the user, e.g. categorical descriptors or for subjective perceptual descriptors (e.g. a "glassiness" value or "anxiousness" level could be manually attributed units)'[255].

[254] Schwarz, D. 2006. Concatenative Synthesis: The Early Years. *Journal of New Music Research* 35(1): 4
[255] Ibid

Considering the variety of descriptors suitable for a sample-based concatenative synthesizer, there are a wealth of analytical processes that may be may be applied to identify these sonic features (such as those discussed in chapter 3.2 *A Brief Summary Machine Listening*) with the optional use of metadata provided by a user for the adjustment of the analysis or inclusion of subjective descriptions.

So, when defining the *Genesis* system, it could be considered a form of sample-based concatenative synthesis; *Genesis* applies the sonic features, extracted through analytical processes of a real-time input sound source with consequent representation as pre-defined descriptors of pitch, onset, MFCCs, loudness and tempo for the modification, manipulation and arrangement of a real-time sound-object's own sonic features. However, reflecting on Schwarz's proposed applications of concatenative synthesis (2006), there are distinctive differences between the methodology of *Genesis* (as detailed in chapter 5 *The Genesis System*) and the approaches described by Schwarz (2006), highlighted in particular through the approach of *high-level instrument synthesis*. Primarily, Schwarz (2006) describes two cost components: direct matching between source and target, and a continuity factor in resynthesis.

With regard to direct matching between source and target, the pitch and tempo of the *control* source and the *slave* sound-object can be compared optionally within *Genesis* for the consequent application of the *control* source's pitch or tempo to the *slave* sound-object. Therefore, direct matching between source and target is applied for a selected number of sonic features. In contrast, a significant number of sonic feature descriptions are compared for *high-level instrument synthesis* (Schwarz, 2006) such as timbre, loudness and onset with the aim of the result to accurately represent the identified sonic features within the synthesizer's auditory output; in *Genesis* the MFCCs, onsets and loudness are *not* compared between the *control* source and the *slave* sound-object, and instead the MFCCs, onsets and loudness of the *control* source trigger generative processes of the *slave* sound-object, irrespective of the *slave* sound-object's MFCCs, onsets and loudness. Therefore, the resulting auditory output by *Genesis* of a *slave* sound-object dictated by a *control* source's described sonic features represents the sonic features identified within the *control* source *without* extensive comparison to the real-time *slave* sound-object.

In addition, for *high-level instrument synthesis* (Schwarz, 2006), the generation of a sample-based concatenative synthesizer's output sound-objects is resultant of the outputs of the analysis of the input source *over time* and the representation of the samples available within a *database*, with the similarity to the target (the input source) of the database samples bound by the likeness of the samples contained within the database to the input source. Within *Genesis*, there is an absence of a *database* of samples from which to compare and select sound-object's similar to a *control* source, instead applying the currently selected real-time *slave* sound-object for its auditory output regardless of its similarity to the *control* source, reflecting the role of the *control* source for defining the various generative processes within *Genesis*, without extensive comparison to the real-time *slave* sound-object.

Considering that a primary function of *Genesis* is to generate and control auditory outputs in *real-time*, the analysis of sound-objects must also be completed in real-time. Therefore, the extensive analysis and representation of a sound-object's sonic features *prior* to *Genesis* initiation is restricted as *all* analytical processes are executed in real-time, with no application of pre-existing metadata to modify the subjectivity of the results. A significant advantage of analysing sample data *prior* to initiation of a program is that offline analysis is not limited to the constraints of real-time analytical processes, which are significantly bound by their frequency resolution and temporal resolution in combination with being potentially CPU intensive which may possibly cause unacceptable latency in the auditory signal.

As a result, offline analysis can be completed multiple times with relative adjustments to the analysis parameters potentially increasing the accuracy of the result, whereas the results of real-time analysis, unless significant adaptability is applied, produce instantaneous results that are substantially limited in their acute modification once an outcome has been produced. However, the execution of multiple analyses of a particular sonic feature in a system such as *Genesis* is a luxury that cannot be afforded; *all* sources are presented in real-time and therefore cannot be analysed with an offline method, resulting in a reliance of real-time analysis. Due to this, *Genesis*' analytical process is instantaneous, without presenting and analysis over time.

Therefore, the continuity factor required for high-level concatenative synthesis, proposed by Schwarz (2006), is not present in *Genesis*.

Furthermore, the modification of the *slave* sound-object within *Genesis* is completed through a combination of UGen parameter settings such as the filter frequencies of the band-pass filters for each granular synthesiser and the pitch of the PitchShift.ar UGen. In contrast, the principle of concatenative synthesis is to minimise the application of such modifications, instead applying the best-match sample within the database to the target source as its auditory output. Therefore, the more expansive and eclectic the sample database, the higher the potential for better matches to the target source, thereby limiting the amount of temporal or frequency modification to the best match sample. However, the greater a sample database's size, the more complex the organization of the database's sample needs to be, as the relevance of accurate descriptions of a sample's sonic features increases; the more samples contained within a database, the more similarities (and differences) will occur between their sonic features, requiring a highly descriptive and consistent method of sample organization in order to distinguish clearly between samples for possible application to a best-match for a target source.

The complexities of organising a sample database by its sonic features relates directly to the difficulties of conclusively defining perceptual processes, as detailed in chapter 3.2 *A Brief Summary of Machine Listening*, with particular reference to timbral classification; the issues in the construction of a definitive topology of sound-objects, which would allow for a quantitative method of sound-object description, permitting a decisive organizational structure that can be applied to dynamically organize the sample database of a concatenative synthesizer with the possible outcome of producing an increased accuracy of best match results.

In relation to the methods that have been applied to organise the databases of concatenative synthesizers such as Caterpillar (Schwarz, 2000) and SoundSpotter (Casey, 2004), Structured Query Language (SQL) is used to manage the descriptors of the samples within the database, with specific algorithms searching the database for best matches. For example, SoundSpotter (Casey, 2004) 'performs real-time

resynthesis of an audio target from an arbitrary-size database by matching of strings of 8 "sound lexemes", which are basic spectro-temporal constituents of sound'[256]. However, despite the ability to apply database software, the method of description of sound-objects is still not conclusive, resulting in databases that may appear to offer efficient solutions to examine large search spaces, but in fact do little to resolve the issue of sonic feature classification; in SoundSpotter (Casey, 2004), 'by hashing and standard database indexation techniques, highly efficient lookup is possible. Casey (2005) claims that one petabyte or 3000 years of audio can be searched in half a second'[257]. Indeed, such a system is evidently very efficient, yet the issue of selecting *which* sonic features to apply, and at what time relative to an ongoing compositional process, still remains.

The capability of *Genesis* to generate live samples, as described in chapter 5.4 *Generative Processes in Genesis*, offers the potential to create a database of samples in real-time for consequent selection by a concatenative algorithm, which explicitly compares the real-time input sources sonic features to the dynamically changing sample database. However, as previously stated, sufficiently representing and consequently categorizing the sound-objects of such a database is highly complex and inconclusive. In addition, all analyses need to be completed in real-time, restricting the accuracy and performance of the analytical processes.

The limitations of real-time analytical processes are reflected in the limited number of real-time analysis in concatenative synthesizers that expressly apply real-time analytical processes. CataRT (Schwarz, 2005), MoSievius (Lazier and Cook, 2003) and Frelia (Momeni and Mandel, 2005), generate sound-objects based on pre-defined sonic features that can be adjusted relative to pre-defined descriptors presented in graphical user interfaces. Therefore, the descriptors of the target source are not extracted from a real-time auditory source, and are instead defined using values applied through the graphical user interface. Concat (Collins, 2006) is an example of a concatenative synthesiser, which implements real-time analysis of both the source and the target. It allows the control of a target by a source through the weighting of four sonic features (zero crossing rate, log mean square, spectral centroid and spectral tilt),

[256] Schwarz, D. 2006. Concatenative Synthesis: The Early Years. *Journal of New Music Research* 35(1): 15
[257] Ibid: 15

combined with various controls through the UI such as freezing of source material. Although Concat (2006) unique and powerful as a synthesis tool, it is difficult to apply perceptual musical features into the quantifiable sonic features applied. (Concat (Collins, 2006) has a revised version Concat2 (Collins, 2006) which allows user-control of its overall loudness detection[258]).

So, in order to incorporate methods of concatenative synthesis for the organization and representation of samples generated by the live sampling process in *Genesis*, a categorization method must be developed that can form accurate descriptions of the sonic features of the *control* and the live samples that may be used to form a database of *slave* sound-objects. As a suggestion, this could be comprised of neural networks and genetic algorithms for qualitative assessment of the sonic features of the live samples within a database. However, the issue of adequate sonic description still remains, highlighting the requirement of further research in the topic of auditory scene analysis and consequent representation of sound-objects.

As noted previously, *Genesis* applies extensive digital signal processes to modify the auditory output of the *slave* sound-object, which are dictated by generative processes controlled by interactions with the GUI and the sonic features of the real-time *control* sources, contrary to a fundamental method of concatenative synthesis, which is to apply such modifications through a database of samples matching the descriptors defined by the GUI or the target source's sonic features. However, the principle of applying an auditory source's sonic features to another auditory source remains in both *Genesis and* sample-based concatenative synthesis.

Therefore, it must be concluded that despite the absence of a sample database within *Genesis*, and the method of extensive comparison between auditory sources within sample-based concatenative synthesis methods, the principle of applying an auditory source's sonic features for the modification, manipulation and arrangement of another auditory source is certainly present in *Genesis*. As a result, it must be determined that *Genesis* applies a fundamental principle of concatenative synthesis to use the sonic

[258] Collins, N. (n.d.). *Concat2*. [online] Doc.sccode.org. Available at: http://doc.sccode.org/Classes/Concat2.html [Accessed Mar. 2014]

features of auditory sources to control other sound-objects, but that the methodology of applying any modifications in *Genesis* differs considerably to the use of a sample database and extensive *comparison* between auditory sources, as applied in sample-based concatenative synthesis.

Considering *Genesis* does not fit directly into the category of 'concatenative synthesiser', *Genesis* does however fall neatly into the category of imitative synthesis (Grey, 1975; Wessel, 1979; Beauchamp, 1982). As noted in chapter 4.1 *Interaction with Creative Systems*, imitative synthesisers extend the instrumental paradigm through reinterpretation of the perceptual spaces of harmonic instruments via psychoacoustic descriptors. In relation to the previous discussion regarding *Genesis*, concatenative synthesis and its generative/analytical processes, it is evident that *Genesis* matches the criteria of an imitative synthesiser; it reinterprets perceptual sonic features such as pitch, timbre and onset, in real-time, through explicit and implicit mappings to generative algorithmic methodologies.

The real-time application of *Genesis* as an imitative synthesiser that manipulates and arranges the sonic features of other auditory sources creates a unique real-time interactive environment for musical composition. As highlighted in chapter 4.2 *Composition with Real-time Interactive Music Systems*, it is demonstrated that real-time composition can be a method of compositional technique. Considering the interactions between the sonic features of real-time input sources and their influence on generative and analytical processes in *Genesis*, it must be asserted that a real-time compositional process is the predominant compositional technique; a compositional output is generated in real-time, structured by the analysis of a real-time auditory signal's perceived sonic features of pitch, loudness, tempo, pseudo-timbre and onset with consequent application of these sonic features to generative processes that share a commonly understood paradigm.

Due to the application of the commonly understood paradigm within *Genesis* between the sonic features of the real-time input sources and the generative processes, 'the flexibility to build processes which generate new sequences of events every time it is executed, and processes which respond to environmental and human interference

whilst remaining within the boundaries imposed by the programmer'[259] is prevalent. For example, this method of generative compositional process is apparent in the use of fractals, triggered by the onsets of the real-time sources and bound by modifiable constraints relative to their respective parameter. Furthermore, the consideration to minimize latency in *Genesis* helps to ensure the response by a generative process to its assigned environmental sonic feature/s is near instantaneous, maintaining the correlation between real-time sonic events and the results of their interaction to a generative process, reflecting the real-time compositional process achievable through the use of *Genesis*.

Therefore, *Genesis* should be considered a real-time compositional system, applying in real-time the imitative synthesis principle of using sonic features of an auditory source to modify, manipulate and arrange another auditory source, for the principal control of generative processes, with optional adjustment through the GUI of the generative and analytical processes, relative to the desired model of interaction between the compositional output of *Genesis* and the real-time input source/s.

6.5 Evaluation of the *Genesis* System's Compositional Process

6.5.1 An Overview of Creativity with *Genesis*

Considering the compositional process described in chapter 2 *An Introduction to Algorithmic Composition*, due to the different models of interaction the *Genesis* system allows, *Genesis* can model the entire creative process including analysis of the input, model specific stages of the creative process including analysis of the input for application to an external compositional process and generate results based on a creative process without analysing its inputs or outputs. For example, in relation to the description of the entire creative process, if the system is run *unsupervised*, the chosen musical objective of modifying a *slave* sound-object through the chosen sonic features of *control* sound-object is applied, with the indeterminate generative processes dictated by the onsets of the sonic features of a *control* source modelling the subconscious, consequently forming a solution based upon the interplay between the

[259] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49

selected sonic features and the indeterminate generative processes, which are then mapped to their relative parameter for realisation to the dynamic score and auditory output.

With reference to the research aim of discussing the implementation of creativity of with machines, *Genesis* models specific stages of a creative process including analysis of the input for application to an external compositional process such as a predetermined score performed by a live instrumentalist. Perhaps the most significantly modelled stage is the development of ideas by the subconscious; the use of fractals triggered by the onsets of a *control* source's output values relative to their desired parameter through an indeterminate method in effect model the theorised disposition of the subconscious to behave in a random manner, bound by a particular characteristic, which, in the case of *Genesis*, is the buffer position, playback rate, recording rate, and duration. The outputs of the fractals can then be applied to a compositional process occurring externally to *Genesis*, influencing the relative parameter's values in an external compositional process.

The generation of results based on a creative process without analysis of inputs or outputs is present in the random search process which defines the random search process applied to the selection of arbitrary values of the *slave* sound-object's Warp1.ar UGen, in conjunction with the values of filters, a reverb and panning of the overall auditory output mix. As a result, such a method models an entire compositional process without external influence; the musical objective is identified relative to the tasks chosen parameters, with the random processes modelling the assumed role of the subconscious, as the solution is bound by selected minimum and maximum values to be applied and mapped to the relative parameter for realisation of the compositional process. Therefore, considering the above examples, it is certainly apparent that a compositional process or stages of a compositional process can take place within an instance of *Genesis*.

Therefore, in reference to the application of a hybridisation of *adaptive* and *generative* models of creativity, as discussed in chapter 4.1 *Interaction with Creative Systems*, *Genesis* would appear to successfully implement such a method; the level of

adaptive and *generative* creativity is relative to the interaction approach, with the evaluation feedback demonstrating that this can range from highly generative to highly adaptive. As a result, considering Blackwell et al's (2012) aim to emulate human performers convincingly, *Genesis* is perceived to be capable of such ability.

6.5.2 *Genesis* and its role in a compositional process

The purpose of the algorithmic compositional processes within *Genesis* falls in to the two categories proposed by Supper (2000) of 'Modeling new, original compositional procedures, different from those known before'[260] and 'Selecting algorithms from extra-musical disciplines'[261]. For example, the Call and Response process applies original compositional procedures through applying predefined structures that are selected relative to chosen sonic features of pitch, onset and tempo, with the fractal processes applying algorithmic procedures from extra-musical disciplines by using mathematical models to dictate the selected parameters of playback rate, recording rate, buffer position and duration for the granular synthesizers that form the *slave* sound-object's generative auditory output.

The application of the algorithmic compositional processes permits compositional outcomes that are otherwise unfeasible or impossible in a real-time compositional process; the interactive, generative and analytical processes used within *Genesis* control multiple complex mappings between the sonic features of the real-time input sources and the real-time generative processes such as the fractal manipulation of a *slave* sound-object's granular synthesizer's playback rate. Considering the possibility of simultaneously controlling the ten parameters of thirty-nine granular synthesizers though interaction in real-time without algorithmic processes, the concurrent manipulation and adjustment in real-time would be highly challenging in the absence of algorithmic compositional processes; the individual control of selected parameters such as the buffer position, playback rate, recording rate and duration, in the dynamic manner offered by *Genesis* would be substantially restricted without algorithmic compositional processes, thereby warranting the use of extensive algorithmic control of predefined parameters within the *Genesis* system.

---

[260] Supper, M. 2001. A Few Remarks on Algorithmic Composition. *Computer Music Journal* 25(1): 48
[261] Ibid: 48

As noted also in chapter 2 *An Introduction to Algorithmic Composition*, the influence algorithmic processes can have on a compositional process can be categorised in to five categories relating to the level of influence an algorithmic output may have on a compositional process. The *overall* level of influence by the algorithmic processes within *Genesis* is relative to the applied model of interaction and inclusion of a specified algorithmic process. The influence of a particular algorithmic process is also relative to the application of an interaction between a sonic feature of a real-time input source and the algorithmic function, or if the algorithmic process generates results irrespective of any external influence.

So, for example, considering the fractal process defining the buffer positions, although triggered by the onset of a *control* source, the selection of the buffer positions are dictated by the fractal process, with no external modification other than the description of the bounds to complete the process, resulting in a high level of influence by the algorithmic process on the outcome. In contrast, the pitch following process applies a predefined algorithmic process with an output adjusted by the pitch of *control* source one, thereby resulting in a mid level of influence between the algorithmic process and the real-time input source's pitch; the outcome of the algorithmic process is relative to the pitch of the real-time input source, modifying the output of the algorithmic process in terms of a parameter provided externally to the algorithmic process itself.

The use of 'novel circumstance'[262], is highly prevalent in the generative processes of the *Genesis* system as described in section 6.4 *An Evaluation of the Genesis System's Methodology*. Primarily, the application of expressly indeterminate processes, or processes which have an indeterminate disposition, are used to generate unique and individual outcomes, restricted to the parameters and any deliberate bounds of the parameter values. As a result, the *Genesis* system consistently produces 'suggestions' based upon the symbolic and subsymbolic representations of the sonic features of the real-time input sources and their relative parameter mapping within a selected generative process.

[262] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49

As noted in chapter 2.2 *Unpredictability and Randomness in the Creative Process*, random functions are a highly efficient method of generating suggestions, relative to chosen parameters, which must be carefully applied, otherwise the outputs of such a process become arbitrary values, with limited application and validity to a compositional process. Therefore, indeterminate functions such as fractals, which exhibit self-similarity, proposed to form the structures of musical composition (Mandelbrot, 1975) and discussed in chapter 3.1 *An Introduction to Real-time Generative Algorithmic Systems*, are applied in order to efficiently incorporate random functionality within suggested musical structures, thereby rendering the results of the process applicable to a real-time compositional process.

Furthermore, the use of indeterminacy is also ubiquitous in the application of genetic algorithms for the real-time exploration of 'novel circumstance'[263]; the use of mutation functions, which introduce randomly generated values, not correlated to a current population, form indeterminate outcomes with future populations that cannot be conclusively predicted. As a result, the use of genetic algorithms, and the relative level of mutation applied to the evolution of the parameters it dictates forms a process that generates 'suggestions' that are applied in real-time, relative to the symbolic parameters of spectrum, envelope, grain duration, onset threshold, grain pan position and grain playback rate through indeterminate processes that are modifiable through the level of mutation applied.

The application of a 'Devolve' function is used to acknowledge the possible outcome of the modified genetic algorithm may not be regarded as a valid 'suggestion' to the ongoing compositional process; the use of a human critique allows a highly qualitative method of assessment for the outputs of the genetic algorithms. With regards to the breeding process of the Red GA class though which the genetic algorithms are executed, this is highly efficient, generating results almost instantaneously, ready for immediate application in real-time. Therefore, perceived inconsistencies in the compositional process, generated by the genetic algorithms can

[263] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49

be modified quickly and efficiently, minimizing any undesirable consequences to the ongoing compositional process.

Considering the application of sonic features extracted in real-time from real-time input sources, it is necessary to represent the sonic features of the real-time source symbolically and/or subsymbolically with a high degree of efficiency. As noted in chapter 2.5 *Further Considerations of Applying Computational Algorithms within a Compositional Process*, the analysis of acoustic sound sources requires transference from an acoustic signal in to the relevant symbolic or subsymbolic representations of its sonic features. As a result, analytical processes can consequently generate outcomes relative to the symbolic or subsymbolic sonic features it is provided with. The use of live streams necessitates that the extraction of the sonic features must be completed in real-time, with the minimization of latency integral to the correlation of the interaction between a real-time input source's sonic features and the auditory output of any generative processes that apply such sonic features to modify or dictate their outcomes.

With the acknowledgment of latency possibly affecting the compositional process within *Genesis* of applying the real-time sonic features to the real-time generative processes, offline analysis, which may provide complex methods based upon an existing set of data for the assessment of an auditory source's sonic features is not a feasible method of assessment; with real-time functionality, an auditory signal's waveform and any modification of its sonic features occurs in real-time, resulting in a data set that is constantly changing as opposed to a static data set, apparent in an offline recording or analysis file. Therefore, real-time analytical processes for real-time application are constrained to methods, which are highly efficient to minimize latency *and* are able to apply to a dynamic data set through reactive and/or predictive methodologies.

In relation to the different methods of real-time computational analysis available, the use of the FFT offers a highly efficient method of extraction of an input acoustic signal's waveform in the time domain, in to the frequency domain, which can be windowed and assessed as an acoustic signal's frequency components over time. The

frequency components can consequently be represented symbolically and/or subsymbolically, with the representations mapped relative to selected parameters within *Genesis*. Considering the complexity of the analytical processes, which identify particular sonic features within the frequency domain, the real-time analytical UGens applied in *Genesis*, and provided with the SuperCollider programming language, sufficiently represent the sonic features for the principle compositional process of applying the sonic features of a real-time auditory source to real-time generative processes by using their respective reactive and/or predictive processes, without explicit modification of a UGen's structure for the *Genesis* system, forming its interpretation of sonic features.

The role of the GUI within the compositional process dictates the application of the interaction between the real-time input source and the algorithmic generative processes that define the auditory output of the *Genesis* system. In addition, the GUI modifies any predefined parameter relative to a set value, relative to its position in the modifiable GUI objects, as well as the toggling of algorithmic generative processes that are not dictated by the sonic features of the real-time input sources. Therefore, the model of interaction between the real-time auditory sources and their relative generative processes, and as a result, the dictation of the level of influence between computational algorithmic processes and the compositional processes of the human composer/s are defined through GUI interaction.

So, it must be concluded, that due to the capability within *Genesis* to select, via the GUI, various predefined compositional processes, and live code generative processes that may be applied in real-time, *Genesis* forms a multi-functional real-time compositional system, applying an imitative synthesis method to apply the sonic features of an auditory source to dictate another in order to structure the principle compositional process within *Genesis* of real-time time application of a real-time source's sonic features, based upon the interpretation of these sonic features by the analytical processes within the *Genesis* system. Through the system's interpretation of sonic features, the symbolic and subsymbolic representations it generates modify, manipulate and arrange other sound-objects, which can be used with or without predefined external generative processes, and any live coded generative processes the

composer may wish to introduce through the post window, relative to the SuperCollider classes provided in the *Genesis* package.

6.6 Evaluation of the *Genesis* System's Product

6.6.1 Challenges in Evaluation of *Genesis*' Compositional Outcomes

With regards to the different methods of interaction that can be applied to digital music systems, as discussed in chapter 4 *Interactivity in Digital Music Systems*, and the many compositional processes that can be algorithmically controlled within digital systems, it is necessary to contextualize the compositional outputs of the *Genesis* system relative to the compositional outputs of existing generative digital systems. As highlighted previously, the methodology of the *Genesis* system forms an imitative synthesiser, and its indeterminate compositional processes are discussed in the relative generative techniques in chapter 3.1 *An Introduction to Real-time Generative Algorithmic Systems*. However, a comparison of the *Genesis* methodology to other digital systems does not sufficiently address the *product* of a compositional process; although taxonomies have been compiled that attempt to categorise the compositional process of generative algorithms (Boden and Edmonds, 2009), this approach evaluates the *process*, as opposed to the *product*, reflecting a Constructivist analysis of music, also highlighted in chapter 4 *Interactivity in Digital Music Systems*.

A considerable issue in assessing the *product* of a digital system, which applies real-time indeterminate generative processes such as *Genesis*, are the inherent nuance, 'novel circumstance'[264] and individuality of each compositional output; although the same generative processes can be applied to each iteration of a compositional process, the indeterminacy that defines their outcomes generates an output that dynamically changes from one composition to the next. So, due to the explicit variance between compositional outputs, the assessment of a digital system's *product* cannot be conclusively be drawn from one example of a compositional output. In support of this notion, Collins (2008) states 'we could always run a generative music program once only, harvest a single production of five minutes, and claim this to be representative

[264] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49

of the work. Any conventional aural and musicological analysis can then be applied to the fixed product so obtained. Unfortunately, this would be a gross abuse of the reality of generative music systems, which are designed to create multiple productions; we would have learnt nothing of the mechanisms by which such programs operate, of the musical model underlying them, and of the scope of future productions from that program'[265].

Therefore, the analysis of the *product* of such digital systems appears to be bound by their own construct; the *process* itself denigrates the application of conventional musicological analysis, implicating that comparison and evaluation of the *process* is the only valid method of analysing the result of compositional processes by digital systems. However, Collins (2008) proposes a method of analysis for the compositional *product* by viewing the *process* in relation to a spectrogram of the *product*. Therefore, the *process* can be identified and categorised in to functions such as determinate or indeterminate, and visualized within the spectrogram displaying the *product*. As a result, the proposed *character* of a generative process is identified within a *product*, supported by the descriptions of the *process*.

Considering the generative approach within *Genesis*, it is perhaps feasible to apply the method proposed by Collins (2008) to sufficiently analyse the *character* of the *product* generated by the *Genesis* system; if the system is run *unsupervised*, using predetermined auditory sources, an analysis proposed by Collins (2008) would be sufficient, as the conditional behaviours of the real-time inputs do not change between performances, allowing the analysis to compare the *product* in relation to the relative categories of the generative *processes* that may (or not) be applied. Furthermore, if applying an indeterminate *control* source, such as a live stream of a running water mill, the *process* can still be categorised and analysed relative to the *product*, with the acknowledgment within the analysis that an indeterminate source was used to trigger the events of the *product*.

However, although it may be possible to analyse the *product* of a generative algorithm that applies indeterminacy, the analysis is still inevitably tied to its *process*.

[265] Collins, N. 2008. The Analysis of Generative Music Programs. *Organise Sound* 13(3): 241

Therefore, this would imply that in order to describe one's listening experience of a *product* of *Genesis*, or any digital system that applies indeterminate processes, there must be an awareness of the *process*, and more importantly, an understanding of the *process*, thereby allowing the listener to consider the *process's* relevance to the *product*; if a *process* is not understood or made acknowledgeable, then its applicability to an analysis is limited as its role within the *product* may be misrepresented, consequently distorting its relevance.

This establishes a predicament for composers who wish to apply such generative processes; should an audience be made aware of the compositional processes *prior* to a performance, *during* a performance, *after* a performance or never? The answer to this issue centres on three key factors: the situation of the performance, the model of interaction and the intentions of the composer. For example, with regards to the use of *Genesis* in a concert environment applying a *supervised improvisation* model with a live instrumentalist (described in chapter 5.3 *Interactive Processes in Genesis*) and a visual projection of the dynamic score, the visual and auditory cues between a live instrumentalist and the *product* of *Genesis* should present a significantly clear link between the sonic features of generated by the live instrumentalist and the *product* of *Genesis*. Due to the apparent link between the sonic features of the live instrumentalist and the triggering of the generative processes, explicit explanation of the principle compositional process within *Genesis* is perhaps not necessary, with clarification of the consequent generative processes dictated by the real-time input source's sonic features relative to the intentions of the composer.

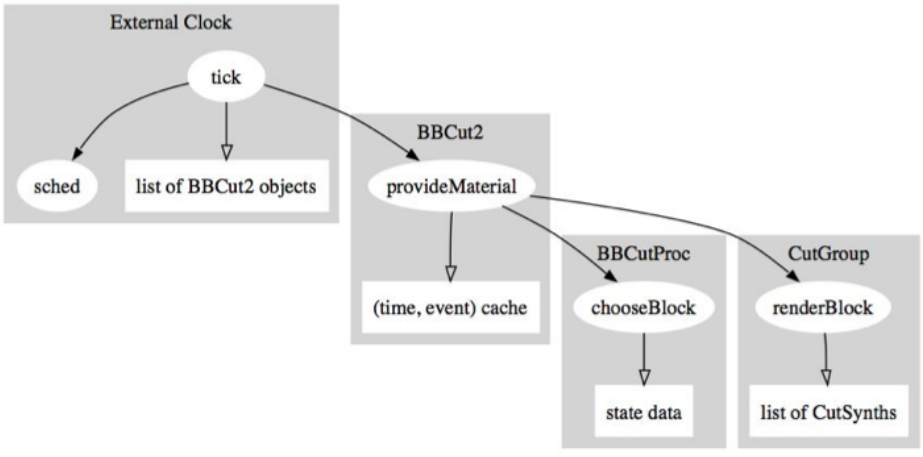
In contrast, if an *unsupervised* model of interaction is applied through *Genesis* in which a series of live streams form each of the *control* sources and the *slave* sound-object, with the composition played via a CD recording, the compositional processes will not be clear; if the *control* sources are not present in the audible output mix, there are no explicit visual or audible cues dictating the *process*, which as a result, would render confusion over the compositional processes and the relevance of their role, perhaps necessitating the requirement by the listener to observe a perspicuous explanation *prior* to the recording, or indeed never, relative to the intentions of the composer. As a result, consequent detailed analysis of the *product* in relation to its

process cannot be conclusively described in such a circumstance unless a significant attempt is made by the composer to inform the listener of the compositional processes applied.

6.6.2 A Proposed Evaluation of *Genesis*' Product

The analysis of the *product* of *Genesis* is highly reliant on the three proposed factors of the situation of the performance, the model of interaction and the intentions of the composer. Indeed, considering *Genesis* is an *interactive* music system, that can function *supervised* or *unsupervised*, with or without and instrumentalist, in circumstances in which a human supervisor is present, more informed evaluation can be made of the *product* and *process*; the evaluative method applied for this thesis, detailed in chapter 6.1 *Evaluation Methodology*, exemplifies and discusses the approaches available to evaluate the *product* and *process* using HCI. Yet, it is shown that this is there no formalised method for such approaches, thereby relying on judgment calls by evaluators on how best to test, assess and obtain feedback. This consequently makes comparison of the *products* of *Genesis* (and its algorithmic components) to other interactive music systems a very challenging prospect.

To exemplify the BBCut2 class (Collins, 2006), which can be applied in SuperCollider for the automated real-time audio splicing of a buffered acoustic signal, has a distinctive model of interaction as illustrated in *Figure 65*[266]:



[266] Collins, N. 2006. *Towards Autonomous Agents for Live Computer Music: Realtime Machine Listening and Interactive Music Systems*. University of Cambridge: 157

Figure 65. Method of Interaction for audio splicing in BBCut2 (Collins, 2006)

The BBCut2 (Collins, 2006) applies an external real-time clock, defined by the user, which dictates the number of 'ticks' per second of a chosen input source. The durations of the generated audio splices are set relative to the duration between the 'ticks' defined by the external clock. A chosen splicing method can then be applied to generate modifications to the audio splice of a buffered acoustic signal, within the duration set by the 'ticks' such as the division of the durations, the buffer position, the playback rate and the amplitude. As a result, a *process* can be identified of applying a set clock to dictate the duration of consequent modification to a buffered audio signal, thereby syncing the adjustment of the buffered audio to the ticks of an external clock.

Now, considering for example the application of the real-time onsets from a *control* source in *Genesis* to trigger the envelope and re-trigger the playback of the Warp1.ar UGen's buffer, relative to its selected buffer frames, an audio splice is generated which syncs to the onset of events within the *control* source. As a result, the duration of each audio splice is determined by the interval between onsets in the *control* source. In addition, the buffer position from where the sound file re-triggers from can be allocated through a random search process, modifying the buffer position to re-trigger from the randomly allocated position, which can be quantized, each time the Warp1.ar UGen is triggered by the onset of a *control* source. The audio splicing process in *Genesis* is illustrated in *Figure 66*:

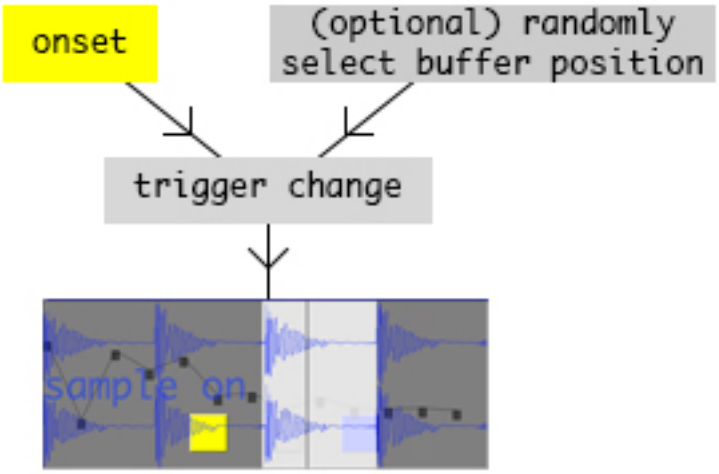


Figure 66. Method of Interaction for audio splicing in Genesis

As a result, there are significant similarities in the *process* of audio splice scheduling in the BBCut2 class and in *Genesis*; the BBCut2 class applies onsets relative to a clock value to define the duration of the audio splices and the *Genesis* system uses the interval between onsets to define the duration of each audio splice and its optional change in buffer position. However, in contrast, the BBCut2 class explicitly applies the formalism of a clock, which attempts to accurately sync a chosen splicing method to the set clock, while *Genesis* does not expressly sync to formalism such as a clock, instead syncing its splicing modifications to the real-time onset of a *control* source.

In *1 - BBCut2* on the accompanying DVD in the *Thesis Recordings* folder, the original sound file is played simultaneously to the generated output of the BBCut2 class, which applies CutBuf1 to assign a splicing method with durations of 0.5, 1 and 2, relative to the clock, set to the tempo of the original sound file.

In *2 - Genesis Quantized* on the accompanying DVD in the *Thesis Recordings* folder, a *control* source, formed of the Sample UGens in *Genesis*, is played simultaneously to the resulting output of the quantized audio splicing process on the *slave* sound-object, which uses the same audio file as the *control* source.

Considering the *products* of *1 - BBCut2* and *2 - Genesis Quantized,* it could be concluded that the fundamental process of applying an external clock or the real-time onset of a *control* source does not intrinsically affect the outcome; it is the indeterminate behaviour of the modifications to the audio buffer that discerns the difference in the *product*, as opposed to the scheduling of the modification. So, if the same clock or real-time onsets were applied to a generated output, the difference in the output is attributed to the use of real-time indeterminate processes to dictate the alterations to the buffered audio. Therefore, despite the inclusion of a digitally accurate clock in BBCut2 to schedule the modifications to digital accuracy, it is difficult to audibly discern between the application of such a process compared to the use of real-time onsets to dictate the *time* of modification within the *product* of the two methods.

However, both *1 - BBCut2* and *2 - Genesis Quantized* apply an audio recording, which is rigidly formulaic, featuring a deterministic 4/4 structure. Deterministic structures are prevalent in BBCut2; the use of a clock implicitly applies deterministic methodology to the scheduling of the audio splicing process by forming a scheduling structure based on the determined constraints of time. In contrast, the audio splicing scheduling method in *Genesis* could be considered both determinate and indeterminate, reacting to onsets of a *control* source regardless of their conditional behaviours, representing the conditional behaviours in the consequent audio splicing output.

In *3 - BBCut 2 Determined* on the accompanying DVD in the *Thesis Recordings* folder, the result of the BBCut2 audio splicing process on a recording of a sustained French horn note is played, which applies CutBuf1 to assign a splicing method with durations of 0.5, 1 and 2, relative to the clock, with an arbitrary value of 95 as no definable tempo can be extracted from the audio recording.

In *4 - Genesis Click* on the accompanying DVD in the *Thesis Recordings* folder, a *control* source, formed of a live stream of structurally determined and non-determined finger clicks, is played simultaneously to the resulting output of the non-quantized audio splicing process on the *slave* sound-object, using a recording of a sustained French horn note.

The *products* of the two processes in *3 - BBCut 2 Determined* and *4 - Genesis Click* can be clearly distinguished, relative to their fundamental method of applying an external clock in BBCut2 and the real-time onset of a *control* source in *Genesis*; the *product* of *3 - BBCut 2 Determined* is discernibly structured by the determinist clock value, with the *product* of *4 - Genesis Click* reflective of the conditional structure of the real-time input source. Therefore, is must be concluded that in fact the *products* of BBCut2 and *Genesis* are intrinsically affected by their respective models of interaction through a composer's application of a particular conditional structure in the scheduling method, which may be chosen in consideration to the situation of a performance.

Furthermore, as noted previously, the behaviour of the modifications to the audio buffer exhibits an influence on the resulting *product*. Considering the generative approaches to audio splicing applied in both BBCut2 and *Genesis*, indeterminate processes are applied, thereby creating an output with inherent 'novel circumstance'[267]. For example, in the BBCut2 class, the CutBuf1 method uses a random function to select between predefined durations, defined by the user, with *Genesis* making use of an optional random search process to dictate the duration and buffer position of the output. As a result, the *products* of both the *processes* within BBCut2 and *Genesis* are numerous, which makes conventional musicological assessment inapplicable, as highlighted earlier, and why evaluation methods grounded in HCI are more suitable.

Therefore, the *product* of the audio splicing method in BBCut2 and *Genesis* must be compared in relation to its *process*, with the implications of the *process* fully understood by the listener to adequately analyse the *character* of the *product*. However, in order to obtain the most fluent and congruent explanation of the *process* and *product*, in chapter 6.1 *Evaluation Methodology*, it is presented that with HCI methodologies, the focus must be on performer-engagement with interactive systems, such as *Genesis*. As a result, the listener must also be the performer, forcing the evaluative feedback to be bound to a very small sample size. Furthermore, in what situation should the interaction be evaluated? As noted by Wanderley and Orio (2002), the volume of situations in which tests could be completed are vast, but it is immensely challenging to define which results are the most valuable.

So, considering the similarities that can occur between the *products*, a comparison between interactive systems is highly dependent on the clarity of the *process* of each system to the listener/performer, the situation of the interaction and the interaction method itself. Considering the remit of this thesis, and the objective of the evaluation is to focus on high-level features in *Genesis* and a single interface trial with the participants of the evaluation, a distinctly broad comparison of the *processes* in BBCut2 and *Genesis* can be made, with the suitability to a desired *product* perhaps relative to such a comparison, suggesting a proposed potential 'success' of a *product*.

[267] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49

Both methods function in real-time, making their application suitable in both live and offline performance scenarios with the major dividing feature being the method of scheduling. Therefore, for real-time audio splicing, the BBCut2 class is proposed to be more 'successful' for formalised and formulaic structuring of a compositional process's schedule, due to its deterministic application of a clock, with *Genesis* proposed to be more 'successful' for indeterminate structuring of compositional process's time scale, due to its impartiality to a specific conditional structure.

However, such proposals are not conclusive. Due to the indeterminate nature of the processes that defines the behaviour of the modification to the audio buffer, it is not possible to decisively state the 'success' of one *process* over another; a matter of 'novel circumstance'[268] within the *process* may cause a *product* to be perceived as more 'successful', contrary to the proposed outlines. It is certainly not conceivable to assess every possible *product* of the audio splicing methods within the BBCut2 class and *Genesis* to form a conclusive analysis of the *products* relative to the *process*. As a result, it must be concluded that detailed comparison and evaluation between the *products* of digital systems cannot currently be resolved, with an existing reliance on the *process* to explain the differences and similarities in the *product*. As demonstrated, this is an undependable method of assessment of a *product*, representing the challenges in quantifiable and qualifiable analysis of extensive algorithmic digital music systems.

So, evaluation of the compositional process in *Genesis* cannot be decided based exclusively on the *product*. As a result, comparison with existing digital systems presents a significant challenge. When considering the *process* within *Genesis*, an assessment of its 'success' must surely be linked to its *product*, but division between the 'success' of the *products* and the *processes* that define them is substantially tied to the situation of the performance, the model of interaction and intentions of a composer, rendering conclusive musicological analysis of *all* possibilities an unmanageable task.

[268] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49

Taking for example the audiovisual examples presented with this chapter, which feature live performances with *Genesis* in various scenarios, the performances demonstrates single instances of the possible outputs of *Genesis*. Due to the inherent indeterminate behaviours of many of the *processes* in *Genesis*, it would seem ill-conceived to generate a finite assessment of the 'success' of *Genesis's product* based upon the accompanying examples of the presented sample size.

Therefore, as Collins (2008) suggests, the *product* can be assessed relative to its *process* to define a *character* of a algorithmic system, which considering the *process* described in chapter 5 *The Genesis System* and section 6.4 *An Evaluation of the Genesis System's Methodology* and the *product* of the performances accompanying this chapter, the *character* of *Genesis* is primarily founded on the conditional structure of its real-time *control* sources, with the application of the generative process dictated by the sonic features of each *control* source respective to their own conditional structures. It must of course be noted that, considering the situation of the performance, the models of interaction, and intentions of the composer, in combination with the 'novel circumstance' generated by the *Genesis* system, such an assessment of the *character* of *Genesis* is not finite, with the potential for numerous *products* to be created by the compositional *process* within *Genesis*, representing various *characters* within the possible compositional outcomes.

6.7 Concluding Remarks

The purpose of this thesis, submitted to accompany the *Genesis* real-time composition system computer music system, is to demonstrate the algorithmic compositional processes applied in *Genesis*, in conjunction with a detailed assessment of the aesthetic considerations of using computational algorithms for a compositional process, which are pertained within the *Genesis* system. With regard to the research aims, described in chapter 1.3 *Aims of the Research*, these are referenced within this section, demonstrating the contribution of this research.

As discussed in chapter 2.1 *Algorithms in the Compositional Process*, it can be argued that indeed *all* compositional processes, whether including *computational* algorithms

or not, apply algorithmic processes. Therefore, the transcription of algorithms to computational algorithmic practices offers composers the capability to explore innovative procedures that are made feasible through the accuracy and efficiency of digital computer systems; complex methods of algorithmic procedures from extra-musical disciplines, along with unique musical approaches can be applied to generate new forms of compositional processes.

In terms of existing algorithmic methods and their relationship with *Genesis*, the application of algorithmic procedures in *Genesis* from extra-musical disciplines is prevalent, as demonstrated in chapter 5 *The Genesis System*; the application of fractal processes for the manipulation of selected parameters of the *slave* object's granular synthesizers is one such example. The use of extra-musical algorithms in *Genesis* not only generates unique compositional processes, but also controls complex and extensive mappings algorithmically, permitting impossible *physical* manipulation of multiple parameter settings to be realized in real-time. As a result, the use of algorithmic procedures from extra-musical disciplines are applied to serve three distinct high-level affordances; introduce contemporary methods of compositional techniques, free the composer from the limitations of the physical manipulation of musical parameters and a low-entry fee.

The constructed limitations of the system, and therefore its *constraints*, are relative to the implementation of the system's musical metaphors of existing paradigms, which are applied to aid in offering a low-entry fee. Therefore, the system is limited in its ability to generate compositions outside of those that are applicable to its musical paradigms. However, in terms of the gestural interpretation and responses created by *Genesis*, the evaluative feedback shows that this is perceived to be successful, implicating that such a method is highly valuable in the design and application of real-time interactive compositional systems. Indeed, considering the suppositions of Overholt (2009), the relatedness of the mappings in *Genesis* is intuitive, perceivable and relative to the source.

With regard to the importance of efficiency in a real-time compositional process, the use of *efficient* computational algorithmic methods which can be applied to many

mappings is of upmost importance; real-time application of physical interactions between analog sources and the digital domain demands near instantaneous response by the computer to ensure the ongoing dialog between the acoustic source and the consequent actions by a digital system is maintained. So, the more efficient an algorithmic process, the more likely a resulting action by a digital system is to be completed near instantaneously, ensuring a resolve between the dialog of a real-time auditory source and its control of another real-time source. With this significant requirement acknowledged, the application of *efficient* algorithms, with complex indeterminate behaviours are used in *Genesis*, in order to efficiently generate 'novel circumstance'[269] within the outputs of the real-time compositional processes.

In addition, the maintenance of the unfolding dialog between the real-time auditory sources is necessary to ensure a common language for the interaction of the acoustic sources and the compositional processes within *Genesis*. Therefore, relatively efficient analytical algorithms are required to extract and consequently represent the sonic features of the real-time auditory sources in a method that allows not only real-time application of the extracted sonic features, but also adequate representations of the specified sonic features in order to guarantee the interactions of the real-time auditory sources and the compositional algorithmic procedures in *Genesis* are commonly understood. Considering the importance of a commonly understood paradigm (Paine, 2002), the use of OSC messaging permits the application of an efficient real-time method of communication, with the messages specified to a user-defined language paradigm such as pitch, tempo, pseudo-timbre and onset between the real-time input sources and the resulting generative processes of the *Genesis* system.

In relation to the influence computational algorithmic procedures may have on a compositional process, the application of indeterminate behaviours in compositional processes offers a composer the ability to provide a digital music system a set of bounds, from which an indeterminate process can select values, relative to the structure of the indeterminate method. For example, the use of genetic algorithms to search a data set provided by the composer to generate novel parameter settings of the

[269] Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 49

granular synthesizers triggered by *control* source three applies indeterminate methodologies by using random functions for the 'mutation' of the current data space, set to the bounds of desired minimum/maximum values of the particular parameter. This results in behaviour by the system that can be predicted in as far as its minimum or maximum settings, but not in regards to the search space in between those values, thereby providing the composer with unique compositional outcomes constrained by the bounds of the search space, generating potentially numerous compositional outcomes from one single data set.

Furthermore, the perceived level of creativity demonstrated by the indeterminate algorithmic processes has been shown to be relative to the user; from the evaluation feedback, this varied from highly artificial to highly human. However, it would appear that the hybridisation of *generative* and *adaptive* creativity applied in *Genesis* has been successful in engaging performers and positively contributing to an ongoing compositional process, an aspect that has made the work on *Genesis* and the accompanying thesis immensely rewarding.

In terms of feature extraction, and the limitations of our understanding of the listening experience, As detailed in chapter 3.2 *A Brief Summary of Machine Listening*, the computational analysis of music and sonic features, and indeed the analysis of music using conventional practices such as Schenkerian analysis, are limited not only by their inherent subjectivity, but also by their predominant application of formalist musical structures and notions, which have limited applicability to the use of contemporary compositional techniques, outside of the pitch/duration paradigm and the application of indeterminate behaviours that cannot conclude finite representations of a compositional process. As a result, the analytical processes applied in *Genesis* generate a representation of the sonic features of the real-time input sources, which are used to form a compositional process, unique to the *Genesis* system.

Considering the concluding remarks, it is proposed that in terms of the epistemic space for musical devices proposed by Magnusson (2010b), *Genesis* is categorised in *Figure 67* below. However, it important to note that this epistemic space should be considered dynamic, relative to the user, the scenario and the interactive

methodologies discussed through this thesis. Therefore, *Figure 67* should be considered a general overview of the epistemic space that *Genesis* covers:

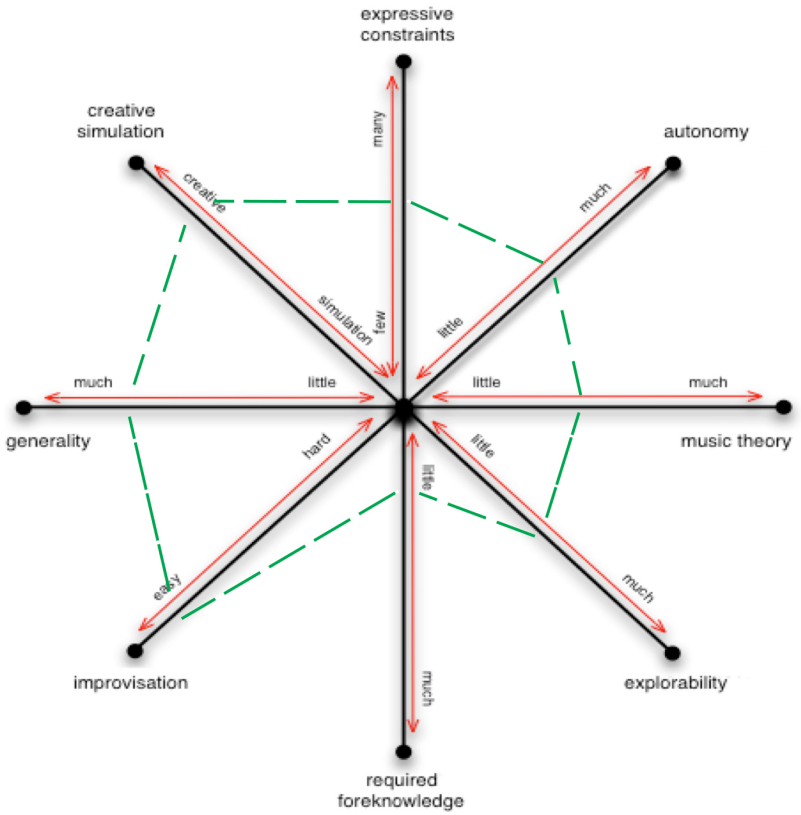


Figure 67. Proposed Epistemic Space of Genesis

In relation to the algorithmic processes applied in *Genesis*, and the representation of the sonic features of the real-time input sources by the analytical algorithms, which dictate selected parameters of the algorithmic processes, it is not possible currently to sufficiently conclude the effect on the *quality* of the compositional outcomes of the algorithmic processes within *Genesis*. It must be argued however, that the acknowledgment to apply *efficient* algorithmic processes serves to render the necessity of such algorithmic procedures for real-time time application, as any loss of interaction in an unfolding dialog must surely denigrate the *quality* of the result; latency between the interactions of one source to another causes a loss of momentary circumstance, resulting in incomprehension between the current state of one source to another, which will substantially affect the consequent actions of the sources, dramatically degrading the intended purpose of real-time implementation, thereby reducing the quality of the output, relative to the real-time principle of such a system.

Therefore, with regards to future research with *Genesis*, the key to its development is further understanding of its compositional outcomes through innovative and novel analysis techniques, extending the evaluation approach presented in chapter 6.1 *Evaluation Methodology*. Through a comprehensive analysis method, not only is it foreseeable that adequate comparison to other generative systems may be a possibility, but the results of such an analysis method may yield the relative importance of particular algorithmic processes within the *Genesis* system, which may signify areas for qualitative improvement of its compositional outputs.

The *Genesis* system certainly demonstrates the applicability of computational algorithmic processes for the extensive real-time modification, manipulation and arrangement of a sound-object controlled by the sonic features of another sound-object. However, despite being formed of a fundamental architecture, through the use of indeterminate processes and live coding practices, the relative success of its compositional outcomes cannot be conclusively defined, such as the *products* demonstrated in the audiovisual examples accompanying this thesis; notwithstanding many of the compositional processes being quantifiable as individual methods of composition, the relativity of the sum of the processes to other compositional approaches cannot be definitively stated. As a result, it is hoped that with technological advancements and further, ongoing research into the analysis of music, future assessments of *Genesis*, and digital systems that apply extensive generative techniques can be considered truly in terms of their *product* as well as their *process*, thereby resolving the fundamental challenge in adequately evaluating and analysing generative music techniques.

Bibliography

Adorno, T and Krenek, E. 1974. *Briefwechsel*. Berlin: Suhrkamp Verlag

Aldwell, E and Schacter, C. 1989 *Harmony and Voice Leading*. New York: Harcourt Brace Jovanovich

American National Standards Institute (ANSI). 1994. *American national standard acoustical terminology*. New York: Acoustical Society of America

Antaki et al. 2004. Discourse Analysis means doing Analysis: A Critique of Six Analytic Shortcomings. *Discourse Analysis Online*. 1(1)

Arfib et al. 2003. Strategies of mapping between gesture data and synthesis model parameters using perceptual spaces. *Organised Sound* 7: 127-144

Battier, M. 2003. A Constructivist Approach to the Analysis of Electronic Music and Audio Art – between instruments and faktura. *Organised Sound* 8(3): 249-255

Bauer, W and Foss, B. 1992. GAMS: an Integrated Media Controller System. *Computer Music Journal*. 16(1): 19-24

Bayle, F. 1993. *Musique acousmatique: propositions – positions*. Paris: INA

BeatTrack.kr UGen Class Help Files. 2012. SuperCollider 3.5.3

Beauchamp, J.W. 1982. Synthesis by Spectral Amplitude and Brightness Matching of Analyzed Musical Instruments Tones. *Journal of the Audio Engineering Society* 30(6): 396-406

Berry et al. (2003). [online]. Available at: http://www.mis.atr.co.jp/~rodney [Accessed Mar.2013]

Berry, R and Dahlstedt, P. 2003 Artificial Life: Why should Musicians Bother?. *Contemporary Music Review* 22(3): 57-67

Berry, R. 1999. Feeping Creatures, *proceeding of the ICMC'99*: 94-96

Berto, F. and Tagliabue, J. (2012). *Cellular Automata*. [online] Plato.stanford.edu. Available at: http://plato.stanford.edu/entries/cellular-automata/#Bib [Accessed 25 Feb. 2015]

Beyls, P. 1980. *Action*, Exhibition Catalogue. Belgium: Kindt Editions

Beyls, P. 1989. The Musical Universe of Cellular Automata. *proceedings of the ICMC'89*: 34-41

Biles, J. 1994. *GenJam –Orginal Paper*. [online]. Igm.rit.edu. Available at: http://igm.rit.edu/~jabics/GenJam94/Paper.html [Accessed Mar. 2013]

Biles, J. 2001. *Autonomous GenJam: Eliminating the Fitness Bottleneck by Eliminating Fitness*. Rochester Institute of Technology. New York

Blackwell et al. 2012. 'Live Algorithms: Towards Autonomous Computer Improvisers' in Computers and Creativity, eds J McCormack & M d'Inverno, Springer, Berlin

Blackwell et al. Live Algorithms: Towards Autonomous Computer Improvisers

Blackwell, T and Young, M. 2004. *Self-Organised Music*. Organised Sound. 9(2): 123

Blackwell, T. 1994. Swarm Music: Improvised Music with Multi-Swarms. *proceedings of the AISB 03 Symposium on Artificial Intelligence and Creativity in Art and Science*: 41-49

Blauret, J. 1972. On the Lag of Lateralization caused by Interaural Time and Intensity Differences. *Audiology* 11: 265-270

Blauret, J. 1997. *Spatial Hearing: The Psychophysics of Human Sound Localization*. Massachusetts: MIT Press

Boden, M. *The Creative Mind: Myths and Mechanisms*. New York: Routledge

Bokesoy, S and Pape, G. 2003. Stochos: Software for Real-Time Synthesis of Stochastic Music. *Computer Music Journal* 27(3): 33-43
Bongers, B. 1999. Exploring Novel Ways of Interaction in Musical Performance. *Proceedings of the 1999 Creativity and Cognition Conference*: 76-81
Bown, O. 2012. 'Generative and Adaptive Creativity: A Unified Approach to Creativity in Nature, Humans and Machines' in Computers and Creativity, eds J McCormack & M d'Inverno, Springer, Berlin
Bown, O. Generative and Adaptive Creativity: A Unified Approach to Creativity in Nature, Humans and Machines: 362
Bregman, A. 1990. *Auditory Scene Analysis*. Massachusetts: MIT Press
Bringsjord et al. 2001. Creativity, the Turing test and the (Better) Lovelace Test. *Mind and Machines* 11: 3-27
Bronkhurst, A and Plomp, R. 1988. The Effect of Head-Induced Interaural Time and Level Differences on Speech Intelligibility in Noise. *Journal of the Acoustical Society of America* 83: 1508-1516
Brown, A and Sorensen, A, 2009. Interacting with Generative Music through Live Coding. *Contemporary Music Review* 28(1): 17-29
Brown, J and Puckette M. 1992. An Efficient Algorithm for the Calculation of a Constant Q Transform. *Journal of the Acoustical Society of America* 92: 2698-2701
Burk, P. 1998. Jsyn - a real-time Synthesis API for Java. *proceedings of the ICMC'98*
Burns, E and Viemeister, N. 1976. Nonspectral Pitch, *Journal of the Acoustical Society of America* 60: 863-869
Burraston, D and Edmonds, E. 2005. *Cellular Automata in Generative Electronic Music and Art: Historical and Technical Review*. University of Technology. Sydney
Burraston, D, Edmonds, E, Livingstone, D and Miranda, E. 2003 . *Cellular Automata in MIDI based Computer Music*. University of Technology, Sydney
Burt, W. 1996. Some parentheses around algorithmic composition. *Organised Sound* 1(3): 167-172
Busse, T and Mansfield, R. 1980. Theories of the Creative Porcess: A Review and a Perspective. *Journal of Creative Behaviour* 26: 91-103
Cadoz, C and Wanderley, M. Gesture-Music. In M.Wanderley and M.Battier, *Trends in Gestural Control of Music*, IRCAM: 1-55
Cage, J. 1961. *Silence: Lectures and Writings*. Connecticut: Wesleyan University Press
Cambouropoulos, C, Dixon, S and Goebl, W. 2001. Beat Extraction from Expressive Musical Performances. *proceedings of the Meeting of the Society for Music Perception and Cognition*
Carbonera, J and Silva, J. 2005. *An Emergent Markovian Model to Stochastic Music Composition*. University of Caxias do Sul
Card et al. 1983. *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates
Carroll, J.M. 2002. "Introduction: Human Computer Interaction, the Past and Present." In J.M Carroll, ed. *Human Computer Interaction in the New Millenium*. New York: ACM Press and Addison Wesley: xxvii - xxxvii
Cascone, K. 2000. The Aesthetics of Failure: "Post-Digital Tendencies in Contemporary Computer Music. *Computer Music Journal* 24(4): 12-18
Chadabe, J. 1997. *Electric Sound: The Past and Promise of Electric Music*. New Jersey. Prentice Hall

Chapman et al. 1996. Self-Similar Grain Distribution: A Fractal Approach to Granular Synthesis. *Proceedings of the ICMC'96*: 212-213
Chareyron, J. 1990. Digital Synthesis of self-modifying waveforms by means of linear automata. *Computer Music Journal* 14(4): 25-41
Chion, M. 1983. *Guide Des Objets Sonores*. Paris. Translation by John Dack and Christine North, 2000
Chowning, J. 1973. The Synthesis of Complex Audio Spectra by Means of Frequency Modulation. *Journal of the Audio Engineering Society*. 21: 526-534
Collins et al. 2003. Live coding in laptop performance. *Organised Sound* 8(3): 322
Collins, N. 2006. *Towards Autonomous Agents for Live Computer Music: Realtime Machine Listening and Interactive Music Systems*. University of Cambridge
Collins, N. 2008. The Analysis of Generative Music Programs. *Organised Sound* 13(3): 237-248
Collins, N. 2012. *Gendy3 Class Help File*. SuperCollider 3.5.3
Collins. N. 2003. Generative Music and Laptop Performance. *Contemporary Music Review* 22(4): 67-79
Collins, N. (2015). *Concat2*. [online] Doc.sccode.org. Available at: http://doc.sccode.org/Classes/Concat2.html [Accessed 25 Jun. 2014].
Cont, A. 2011. On the Creative Use of Score Following and its Impact on Research. *published in SMC 2011: 8th Sound and Music Computing Conference 2011*
Cope, D. 1992. Computer Modeling of Musical Intelligence in EMI. *Computer Music Journal* 16(2): 69-83
Correa, Debora et al. 2008. Neural Network Based System for Computer-Aided Musical Composition: Supervised x Unsupervised Learning. *proceedings of SAC'08*: 1738-1742
Cosi, P, DePoli, G and Lauzzana, G. 1994.Timbre Classifaction by Neural Networks and Auditory Modelling. *proceedings of ICANN*: 925-928
Croft, J. 2007. Theses on liveness. *Organised Sound* 12(1): 59-66
Dahlstedt, P. 2001. A MutaSynth in Parameter Space: Interactive composition through evolution. *proceedings of Music Without Walls, Music Without Instruments Conference 2001*
Dai, H. 2000. On the relative influence of Individual Harmonics on Pitch Judgement. *Journal of the Acoustical Society of America* 107: 953-959
Dannenberg et al. 1990. An Expert System for Teaching Piano to Novices. *proceedings of the ICMC'90*: 20-23
Dannenberg et al. 1997. "A Machine Learning Approach to Musical Style Recognition". *proceedings of the ICMC'97*: 344-347
Dannenberg, R. 1993. Music Representation Issues, Techniques and Systems. *Computer Music Journal* 17(3): 20-30
de Laubier, S. 1998. The Meta-Instrument. *Computer Music Journal* 22(1): 25-29
De Poli, G. 2004. Methodologies for Expressiveness Modelling of and for Music Performance. *Journal of New Music Research* 33(3): 189-202
Desain, P and Honing, H. 1999. Computational Models of Beat Induction: The Rule-Based Approach. *Journal of New Music Research* 28(1): 29-42
Desain, P. 1992, A (De)Composable Theory of Rhythm Perception. *Music Perception* 9(4): 439-454
Di Scipio, A. Systems of Embers, Dust, and Clouds: Observations after Xenakis and Brün. *Computer Music Journal* 26(1): 22-32

Diaz-Jerez, G. (2012). *Overview | Gustavo Díaz-Jerez*. [online] Gustavodiazjerez.com. Available at: http://www.gustavodiazjerez.com/?cat=14 [Accessed Mar. 2013]
Dodge, C. 1988. "Profile": A Musical Fractal. *Computer Music Journal* 12(3): 10-14
Dorin, A. 2001. Generative Processes and the Electronic Arts. *Organised Sound* 6(1): 47-53
Drummond, J. 2009. Understanding Interactive Systems. *Organised Sound*. 14: 124-133
Dunn, J. (2003). *Contents*. [online] Algoart.com. Available at http://www.algoart.com/help/softstep/ [Accessed Mar. 2013]
Ebcioglu, K. 1988. An Expert System for Harmonising Four-Part Chorales. *Computer Music Journal* 12(3): 43-51
Edwards, M. 2007. *Algorithmic Composition: Computational Thinking in Music*. University of Edinburgh
Eigenfeldt, A and Kapur, A. 2008. An Agent-based System for Robotic Musical Performance. *Proceedings of the 2008 International Conference on New Interfaces for Musical Expression*: 144-149
Ericsson, K.A. and Simon, H.A. 1996. *Protocal Analysis: Verbal Reports as Data* (Revised Edition). Cambridge, MA: MIT
Essl, K. 2007. 'Algorithmic composition' in The Cambridge Companion to Electronic Music, eds N Collins & J d'Escrivan, CUP, Cambridge
Fels, S and Hinton, G. 1993. Glove-talk: A Neural Network Interface between a Dataglove and a Speech Synthesizer. *IEEE Transactions on Neural Networkds*. 4 (1) : 2-8
Fiebrink et al. 2009. A Meta-Instrument for Interactive, On-the-fly Machine Learning. *Proceedings of New Interfaces for Musical Expression*, *2009*
Florentine, M and Buus, S. 1981. An Excitation-pattern model for Intensity Discrimination. *Journal of the Acoustical Society of America* 70: 1646-1654
Forte, A. 1962. *Tonal Harmony in Concept and Practice*. New York: Holt, Rinehart and Winston
Fox, J and Carlile, J. 2005. SoniMime: Movement Sonification for Real-Time Timbre Shaping. *Proceedings of NIME' 05*
Francois, JC. 2006. Improvisation Today, Between Orality and Writing. *Contemporary Music Review* 25(5/6): 623-624
Gabor, D. 1947. Acoustical Quanta and the Theory of Hearing. *Nature,* 159: 591-594
Gardner, M and Gardner, R. 1973. Problem of Localization in the Median Plane: Effect of Pinnae Cavity Occlusion. *Journal of the Acoustical Society of America* 53: 400-408
Gardner, M. 1970. The Fantastic Combinations of John Conway's new solitaire game "life". *Scientific American* 223: 120-123
Gartland-Jones, A and Copley P. 2003. The Suitability of Genetic Algorithms for Musical Composition. *Contemporary Music Review* 22(3): 43-56
Gartland-Jones, A, 2003. MusicBlox: A Real-time Algorithmic Composition System incorporating a distributed Interactive Genetic Algorithm. *proceedings of EvoWorkshops/EuroGP2003*: 490-501
Gauldin, R. 1997. *Harmonic Practice in Tonal Music*. New York: W.W Norton
Gerhenson, C. 2003. *Artificial Networks for Beginners*. University of Sussex
Goldberg, D E. 1998. *The Design of Innovation: Lessons from Genetic Algorithms, Lessons for the Real World*. University of Illinois

Goldstein, J. 1973. An Optimum Processor Theory for the Central Formation of the Pitch of Complex Tones. *Journal of the Acoustical Society of America* 54: 1496-1516
Gomez, D. 2006. *Dynamic Musical Thinking; Some analytical approaches to the use of space in instrumental and electronic music composition*. Institute of Sonology, University of Ginebra
Gouyon, F and Dixon, S. 2005. A Review of Automatic Rhythm Description Systems. *Computer Music Journal* 29(1): 34-54
Green, D. 1988. *Profile Analysis*. Oxford: Oxford University Press
Greenhouse, R. 1995. [online] Available at: http://www-ks.rus.uni-stuggart.de/people/schulz/fmusic/wtf/docu.html [Accessed Mar. 2013]
Grey, J. 1975. *An Explanation of Musical Timbre*. Stanford University
Guazzo, M. 2007. *A Cartesian Data Model for Decision Support Systems*.Torino: Codework
Guildford, J.P. 1950. Creativity. *American Psychologist* 5: 444-454
Gumowski, I and Mira, C. 1980. *Recurrences and Discrete Dynamic Systems*. Berlin: Springer
Hegarty, Paul. 2007. *Noise/Music: A History*. London and New York: Continuum
Hermes, D. 1988. Measurement of Pitch by Subharmonic Summation. *Journal of the Acoustical Society of America* 83: 257-264
Hoffmann, P. 2000. The New GENDYN Program. *Computer Music Journal* 24(2): 31-38
Honing, H. 2001. From Time to Time: The Representation of Timing and Tempo. *Computer Music Journal* 25(3): 50-61
Hsu, W and Sosnick, M. Evaluating Interactive Music Systems: An HCI Approach. *NIME 2009*: 26
Hunt, A, Kirk, R and Orton R. 1991. Graphical Control of Granular Synthesis using a Cellular Automata and the Freehand Program. *proceedings of the ICMC'91*: 416-418
Ikeshiro, R. 2011. GENDYN and Merzbow: A Noise Theory Critique of Xenakis' Dynamic Stochastic Synthesis and its Relevance Today. Goldsmiths College, University of London
Järveläinen, H. 2000. *Algorithmic Musical Composition*. University of Technology, Helsinki
Jeffress, L. 1948. A Place Theory of Sound Localization. *Journal of Comparative and Physiological Psychology* 41: 35-39
Jeffress, L. 1972. Binaural Signal Detection: Vector Theory. In J Tobias, *Foundations of Modern Auditory Theory*. New York: Academic Press
Johnson-Laird, P. 2002. How Jazz Musicians Improvise. *Music Perception* 19(3): 415-442
Juslin, P. 1997. Emotional Communication in Music Performance: A Functionalist Perspective and Some Data. *Music Perception* 14(4): 383-418
Kaminsky, I and Materka, A. 1995. Automatic Source Identification of Monophonic Musical Instrument Sounds. *proceedings at the IEEE International Conference, 1995*: 189-194
Kane, B, 2007. L'Objet Sonore Maintenant: Pierre Schaeffer, sound-objects and the phenomenological reduction. *Organised* Sound 12(1): 15-24

Karhunen, K. 1947. Uber lineare methoden in der wahrscheinlich-keitsrechnung, *Ann. Acad. Sci. Fennicea*, Ser. A137. Translated by Selin, I. 1960. in *On Linear Methods in Probability Theory*. Doc. T-131.The RAND Corp., Santa Monica, CA, 1960
Karplus, K, and Strong, A. 1983. Digital Synthesis of Plucked-String and Drum Timbres. *Computer Music Journal* 7(2): 43-55
Kelly, Caleb. 2009. *Cracked Media: the Sound of Malfunction*. Cambridge, MA: MIT Press
Kerman, J. 1980. How we got into Analysis, and how we get out. *Critical Inquiry* 7(2): 311-331
Kiefer, C., Collins, N, and Fitzpatrick, G. 2009. Phalanger: Controlling Music Software With Hand Movement Using a Computer Vision and Machine Learning Approach. *Proceedings of the 2009 International Conference on New Interfaces for Musical Expression*. New York: Association of Computing Machinery, 246-250
Klump, R and Eady, H. 1956. Some Measurements of Interaural Time Difference Thresholds. *Journal of the Acoustical Society of America* 28: 859-860
Knapp, R and Lusted H. 1990. A Bioelectric Controller for Computer Music Applications. *Computer Music Journal*. 14(1): 42-47
Koffka, K. 1935. *Principles of Gestalt Psychology*. New York: Harcourt and Brace
Kramer, G. 1996. Auditory Display: Sonification, Audification, and Auditory Interaces. *Music Perception*. 13(4): 583-591
Krumhansl, C and Kessler, E. 1982. Tracing the Dynamic Changes in Perceived Tonal Organization in a Spatial Representation of Musical Keys. *Psychological Review* 89: 334-368
Kuuskankare, M and Laurson, M. 2006. Expressive Notation Package. *Computer Music Journal* 30(4): 67-79
Laske, O. 1975. *Introduction to a Generative Theory of Music*. Institute of Sonology, Utrecht
Le Groux, S. 2002. *A Neural Network Principal Component Synthesizer for Expressive Control of Musical Sounds*. Paris: IRCAM
Leach, J and Fitch, J. 1995. Nature, Music and Algorithmic Composition. *Computer Music Journal* 19(2): 23-33
Leben, J. 2012. *Artifical Neural Networks in Musical Performance*. Royal Conservatory, The Hague
Lee, M, Freed, A and Wessel, D. 1991. Real-time Neural Network processing of Gestural and Acoustic Signals. *proceedings of the ICMC'91*: 277
Lerdahl, F and Jackendoff, R, 1977. Toward a Formal Theory of Tonal Music. *Journal of Music Theory* 21(1): 111-172
Levitin, D. 2012. Musical rhythm spectra from Bach to Joplin obey a 1/f power law. *proceedings of the National Academy of Sciences* 109(10): 3716
Lewis, G. 2000. Too Many Notes: Complexity and Culture in Voyager. *Leonardo Music Journal*. 10: 33-39
Li, W, Packard, N, and Langton, C. 1990. Transition Phenomena in Cellular Automata Rule Space. *Physica* 45D: 77-94
Licklider, J. 1951. A Duplex Theory of Pitch Perception. *Experientia* 7: 128-133
Licklider, J.1956. Auditory Frequency Analysis. In Cherry,C. 1956. *Information Theory*: 253-268
Lindenmayer, Aristd. 1968. Mathematical Models of Cellular Interaction in Development. *J. Theoret. Biology* 18: 280-314

Longuet-Higgens, H. 1994. Artificial Intelligence and Music Cognition. *Philospohical Transactions of the Royal Society of London* 349: 103-113
Lubart, T. 2000. Models of the Creative Process: Past, Present and Future. *Creativity Research Journal* 13(3/4): 295-308
Luig, J and, Rahimzadeh, A. 2008. *Sinusoidal Modelling and Synthesis*. SE
Luque, S. 2006. *Stochastic Synthesis: Origins and Extensions*. Royal Conservatory, The Hague
Luque, S. 2011. Stochastic Synthesis: An Overview. *proceedings of the Xenakis International Symposium*: 1
Machover, T and Chung, J. Hyperinstruments: Musically Intelligent and Interactive Performance for Creativity Systems. *Proceedings of the ICMC'89*
Maestre et al. 2009. Expressive Concatenative Synthesis by Reusing Samples from Real
Magnusson, T. 2007. The iXiQuarks: Merging Code and GUI in One Creative Space. *Proceedings of the ICMC' 2007*: 332-339
Magnusson, T. 2010a. Designing Constraints: Composing and Performing with Digital Music Systems. *Computer Music Journal*. 34(4): 62-73
Magnusson, T. 2010b. An Epistemic Space for Musical Devices. *Proceedings of NIME' 2010*: 43 - 46
Mandelbrot, B. 1975. *Fractals: Form, Chance and Dimension*. W.H. Freeman and Company
Manoury, P. 1990. *La note et le son*. L'Hamartan
Manousakis, S. 2006. *Musical L-Systems*. The Royal Conservatory, The Hague
Martin, K. 1996a. *A Blackboard System for Automatic Transcription of Simple Polyphonic Music*. MIT
Martin, K. 1996b. *Automatic Transcription of Simple Polyphonic Music: Robust Front End Processing*. MIT
Mathews, M and Schloss, A. 1989. The Radiodrum as a Synthesis Controller. *Proceedings of the ICMC'89*
McCarthy, J. 2007. *What is Artificial Intelligence?* Stanford University
McCartney, J. 2002. Rethinking the Computer Music Language: SuperCollider. *Computer Music Journal* 26(4): 61-68
McCorduck, P. 1990. *AARON's code: meta-art, artificial intelligence, and the work of Harold Cohen*. New York: Freeman
McDowell, H. 1994. Fractal Music Composer II. Documentation accompanying software
Meddis, R and Hewitt, M. 1991. Virtual Pitch and Phase Sensitivity of a Computer Model of the Auditory Periphery. *Journal of the Acoustical Society of America* 89: 2866-2882
Meinhardt, H. 2003. *The Algorithmic Beauty of Sea Shells*. Berlin: Springer
Meyer, L. 1956. *Emotion and Meaning in Music*. Chicago: University of Chicago Press
Meyers, O. 2004. *A Mood-Based Classification and Exploration System*. MIT
MFCC.kr UGen Class Help File. 2012. SuperCollider Version 3.5.3
Millen, D. 1990. Cellular Automata Music. *proceedings of the ICMC'90*: 314-316
Miranda et al. 2000. Categorising Complex Dynamic Sounds. *Organised Sound* 5(2): 95-102
Miranda, E. 2001. *Composing Music With Computers*. Massachusetts: Focal Press
Miranda, E. 2002. *Evolving Cellular Automata Music: From Sound Synthesis to Composition*. Sony Computer Science Laboratory, Paris

Miranda, E. 2009 *Artificial Intelliegence (AI) and Cellular Automata*. Institute of Communications Research, Electronic Music Studio, Berlin
Miranda, E.1993. Cellular Automata Music: An Interdisciplinary Project. *Interface* 22(1): 3-21
Møller, A. 2000. *Hearing: Its Physiology and pathophysiology*. New York: Academic Press
Moore, B, Glasber, B, and Peters, R. 1985. Relative Dominance of Individual Particles in Determining the Pitch of Complex Tones. *Journal of the Acoustical Society of America* 77: 1853-1860
Moore, B, Glasberbg, B and Baer, T. 1997. A Model for the Prediction of Thresholds, Loudness, and Partial Loudness. *Journal of the Acoustical Society of America* 45: 224-240
Moorer, J.A. 1972. Music and Computer Composition. *Communications of the Association for Computing Machinery* 15(2): 10-113
Mozer, M. 1994 . *Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multiscale Processing*. University of Colorado
Munshi, J. 2014. *A Method for Constructing Likert Scales*. Sonama State University
Narmour, E. 1977. *Beyond Schenkerism: The need for alternatives in music analysis*. Chicago: University of Chicago Press
Narmour, E. 1990. *The Analysis and Cognition of Basic Melodic Structures: The Implication-realization Model*. Chicago: University of Chicago Press
Nash, C and Blackwell, A. 2011. Tracking Virtuosity and Flow in Computer Music. *Proceedings of the ICMC'11*: 575-582
Nattiez, JJ. 1975. *Fondements d'une Semiologie de la Musique*, Paris: Union General d'Editons
Nelson, GL. 1999. *Real Time Transformation of Musical Material with Fractal Algorithms*. Conservatory of Music, Oberlin
Olshausen, B. 2000. Aliasing. *PSC* 129 – Sensory Processes: 1-6
Onsets.kr UGen Class Help File. 2012. SuperCollider Version 3.5.3
Overholt et al. 2009. A Multimodal System for Gesture Recognition in Interactive Music Performance. *Computer Music Journal* 33(4): 69-82
Overholt, D. 2009. The Musical Interface Technology Design Space. *Organised Sound*. 14(2): 217-226
Pachet, F and Roy, P. 2000. *Formulating Constraint Satisfaction Problems on Part-Whole Relations: The Case of Automatic Music Harmonisation*. Sony Computer Science Laboratory, Paris
Pachet, F. 1992. *Representing Knowledge Used by Jazz Musicians*. University of Paris
Paddison, M. 1993. *Adorno's Aesthetics of Music*. Cambridge: Cambridge University Press
Paine, G. 2002. Interactivity: Where to from Here?. *Organised Sound* 7(3): 295-304
Paine, G. 2009. Towards Unified Design Guidelines for New Interfaces of Musical Expression. *Organised Sound* 14(2): 142-155
Papadopoulos, G and Wiggins, G. 1999. *AI Methods for Algorithmic Composition: A Survey, A Critical View and Future Prospects*. University of Edinburgh
Performance Recordings. *Computer Music Journal* 33(4): 23-42
Peters, M. 2010. From Strange to Impossible: Interactive Attractor Music. *Contemporary Music Review* 29(4): 395-404
Piston, W. 1948. *Harmony*. New York: W.W. Norton

Plack, C. 2005. *The Sense of Hearing*. New Jersey: LEA
Povel, D and Essens, P. 1985. Perception of Temporal Patterns. *Music Perception* 2: 411-440
Pulkki, V. 2000. Musical Presentations of Fractals. unpublished manuscript for ICMC, 2000
Reynolds, R. 1965. Indeterminacy: Some Considerations. *Perspectives of New Music*. 4(1): 136-140
Risset, JC. 1966. *Computer Study of Trumpet Tones (with sound examples on tape)*. Bell Laboratories
Risset, JC. 1999. Composing in Real-time?, *Contemporary Music Review* 19(3): 31-39
Roads, C. 1977. Composing Grammars. revised, *presented at ICMC'97*
Roads, C. 1979 . Grammars as Representations for Music. *Computer Music Journal* 3(1): 48-55
Roads, C. 2004. *Microsound*. Massachusetts: MIT Press
Rohrmeier, M. 2007. A Generative Grammar Approach to Diatonic Harmonic Structure. *proceedings of SMC'07*
Rosen, C. 1971. *The Classical Style: Haydn, Mozart, Beethoven*. London: Faber and Faber
Ross, B. 1995. *A Process Algebra for Stochastic Music Composition*. Brock University, California
Rowe, R. 1993. *Interactive Music Systems: Machine Listening and Composing*, Cambridge, MA: MIT Press
Rowe, R. 2009. Split Levels: Symbolic to Sub-Symbolic Interactive Music Systems. *Contemporary Music Review* 28(1): 31-42
Rubine, D and McAvinney, P. 1990. Programmable Finger Tracking Instrument Controllers. *Computer Music Journal*. 14(1): 26-41
Ruwet, N. 1972. *Language, Musique, Poesie*. Paris: Seuil
Sachs, M, and Young, E. 1980. Effects of nonlinearities on Speech Encoding in the Auditory Nerve. *Journal of the Acoustical Society of America* 68: 858-875
Sapir, S. 2002. Gestural Control of Digital Audio Environments. *Journal of New Music Research*. 31(2): 119-129
Sarkar, M. 2007. *TablaNet: A Real-Time Online Musical Collaboration System for Indian Percussion*. MIT
Schaeffer, P. 1952. *A la recherche d'une musique concrete*. Paris: Seuil
Schaeffer, P. 1966. *Traite des Objets Musicaux*. Paris: Seuil
Scheirer, E. 2000. *Music-Listening Systems*. MIT
Schenker, H. 1979. *Free Composition (Der freie Satz)*. London: Longman
Schlauch, R, DiGiovanni, J and Reis, D. 1998. Basilar Membrane nonlinearity and Loudness. *Journal of the Acoustical Society of America* 103: 2010-2020
Schouten, J. 1940. The residue and the Mechanism of Hearing. *Proc. Kon. Akad. Wetenschap* 43: 991-999
Schouten, J. 1970. "The residue revisited", in R.Plomp and G.Smoorenburg, *Frequency analysis and periodicity detection in hearing*: 41-54
Schwarz, D. 2006. Concatenative Synthesis: The Early Years. *Journal of New Music Research* 35(1): 3-22
Serra, M H. 1993. Stochastic Composition and Stochastic Timbre: GENDY3 by Iannis Xenakis. *Perspectives of New Music* 31(1): 236-257

Slaney, M and Lyon, R. 1990. A Perceptual Pitch Detector. *proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 1990: 357-360
Smoliar, S. 1977. SCHENKER: A Computer Aid for Analysing Tonal Music. *SIGLASH Newsletter* 10(1/2): 30-61
Steedman, M. 1984. A Generative Grammar for Jazz Chord Sequences. *Music Perception* 2(1): 52-77
Stevens, S. 1957. On the Psychophysical Law. *Psychology Review* 64: 153-181
Stevens, S. 1972. Perceived Level of Noise by Mark VII and decibels. *Journal of the Acoustical Society of America* 51: 575-601
Stowell et al. 2009. Evaluation of live human-computer music-making: quantitative and qualitative approaches. *International Journal of Human-Computer Studies*. 67(11): 960-975
Supper, M. 2001. A Few Remarks on Algorithmic Composition. *Computer Music Journal* 25(1): 48-53
Szumski, A. 2011. Finding the Interference. Karhunen-Loève Transform as an Instrument to Detect Weak RF Signals. *Inside GNSS*. May/June
Terasawa, H, Slaney, M and Berger, J. 2005. Perceptual Distance in Timbre Space. draft for *ICAD'05*,
Terhardt, E, 1974. Pitch, Consonance, and harmony. *Journal of the Acoustical Society of America* 55: 1061-1069
Therrien, C. 1989. *Decision, Estimation and Classification: An Introduction to Pattern Recognition and Related Topics*. New Jersey: Wiley
Thurlow, W, Mangels, J and Runge, P. 1967. Head Movements during Sound Localization. *Journal of the Acousitcal Society of America* 42: 489-493
Todd, P. 1989. A Sequential Network Design for Musical Applications. *proceedings of the Connectionist Models Summer School*, *1989*: 76-84
Toivainen, P and Krumhansl, C. 2003. Measuring and Modelling Real-time Responses to Music: the Dynamics of Tonality Induction. *Perception* 32(6): 741-766
Toiviainen, P. 2000. *Symbolic AI Versus Connectionism in Music Research*. University of Jyvaskyla
Truax, B. 1987. *Real-Time Granulation of Sampled Sound with DMX-1000*. Simon Fraser University
Truax, B. For Otto Laske: A Communicational Approach to Computer Sound Programs. *Journal of Music Theory*. 20 (2): 227-300
Tsang, CP and Aitken, M. 1991. *Harmonising Music As A Discipline of Constraint Logic Programming*. University of Western Australia
Turing, A.M. 1950. Computing Machinery and Intelligence. Mind 59: 433-460
Vaidhyanathan, S, Minai, A, and Helmuth, M. 1999. ca: A System for Granular Processing of sound using Cellular Automata. *proceedings of the 2nd COST g-6 Workshop on Digital Audio,* Trondheim, *1999*
van Noorden, L. 1983. Two-channel Pitch Perception. in M.Clynes *Music, Mind and Brain: The Neuropsychology of Music*: 251-269
Vanhanen, J. 2003. Virtual Sound: Examining Glitch and Production. *Contemporary Music Review* 22(4): 45-52
Vercoe, B, and Scheirer, E. 1999. SAOL: The MPEG-4 Structured Audio Orchestra Language. *Computer Music Journal* 23(2): 31-51
Verfaille, V and Arfib D. 2001. A-DAFx: Adaptive Digital Audio Effects. *Proceedings of the DAFx01 Conference*

Vickery, L. 2012. The Evolution of Notational Innovations from the Mobile Score to the Screen Score. *Organised Sound* 17(2): 128-136
Voss, R, and Clarke, J. 1975. 1/*f*noise in Music and Speech. *Nature* 258: 317-318
Wanderley, M and Orio, N. 2002. Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI. *Computer Music Journa*l. 26(3): 62-76
Wanderley, M. 2001. *Gestural Control of Music*. IRCAM, Paris: 2
Weinberg, G and Gan, S. 2001. The Squeezables: Toward an Expressive and Interdepent Multiplayer Musical Instrument. *Computer Music Journal*. 25: 27-45
Wessel, D and Wright, M. 2002. Problems and Prospects for Intimate Music Control of Computers. *Computer Music Journal*. 26(3): 11-22
Wessel, D. 1974. *Report to C.M.E*. University of California
Wessel, D. 1979. Timbre Space as a Musical Control Structure. *Computer Music Journal* 3(2): 45-52
Whitman, B and Ellis, D. 2004. *Automatic Record Reviews*. MIT
Wight, M and Freed, A. 1997. Open Sound Control: A New Protocol for Communicating with Sound Synthesizers. *proceedings of the ICMC'97*: 101-104
Wightman, F and Kistler, D. 1992. The Dominant Role of Low-frequency Interaural Time Differences in Sound Localization. *Journal of the Acoustical Society of America* 91: 1648-1661
Wightman, F. 1973. The Pattern-transformation Model of Pitch. *Journal of the Acoustical Society of America* 54: 407-416
Wigner, E. 1932. On the Quantum Correction for Thermodynamic Equilibrium. *Physical Review* 40: 749-759
Wilson-Bokowiec, J, and Bokowiec, M. 2006. Kinaesonics: The Intertwining Relationship of Body and Sound. *Contemporary Music Review* 25(1/2): 47-57
Winkler, T. 2001. *Composing Interactive Music: Techniques and Ideas Using Max*. Massachusetts: MIT Press
Winograd, T. 1968. Linguistics and Computer Analysis of Tonal Harmony. *Journal of Music Theory* 12: 2-49
Winograd, T. 1983. *Language as a Cognitive Process*. Boston: Addison-Wesley
Wishart, T. 1994. *Audible Design*. York: Orpheus the Pantomime Ltd
Wishart, T. 1996. *On Sonic Art*. London: Routledge
Witten, I and Frank, E. 2005. Data Mining: Practical Machine Learning Tools and Techniques. San Francisco: Morgan Kaufmann
Wolfram, S. 1983. Statistical Mechanics of Cellular Automata. *Review of Modern Physics* 55(3): 601-644
Wolfram, S. 1984. Universality and Complexity in Cellular Automata. *Physica* 10D: 1-35
Woolf, S and Yee-King, M. 2003. Virtual and Physical Interfaces for Collaborative Evolution of Sound. *Contemporary Music Review*. 22(3): 31-41
Wright, M. 1998. Implementation and Performance Issues with Open Sound Control. *proceedings of the ICMC'98*: 224-227
Xenakis, I. 1960. *Grundlagen einer stochastischen Musik*. Berlin: Ars-Viva-Verlag
Xenakis, I. 1971. *Formalized Music: Thought and Mathematics in Composition*. Bloomington and London: Indiana University Press
Xenakis, I. 1992. *Formalized Music*. revised, New York: Pendragon Press,
Xenakis, I. 1996. Determinacy and Indeterminacy. *Organised Sound* 1(3): 143-155
Zahorik, P. 2002. Assessing Auditory Distance Perception using Virtual Acoustics. *Journal of the Acoustical Society of America* 111: 1832-1846

Zizek, S. 2000. *The Art of the Ridiculous Sublime*. Washington: University of Washington Press
Zwicker, E and Scharf, B. 1965. A Model of Loudness. *Psychology Review* 72: 3-26