Exploring complex rhythmic devices in new music composition through software design

Joe Manton

A thesis submitted in partial fulfilment
of requirements for the degree of
Master of Music (Composition)

Sydney Conservatorium of Music
The University of Sydney
2014

Statement of originality

I declare that the research presented here is my own original work and has not been submitted to any other institution for the award of a degree.

Signed: ....

Date: 31/3/2015

Abstract

This thesis examines the role that complex rhythms perform in my music. I will demonstrate how the software I have created is unique and necessary for this type of rhythmic exploration in music composition and how it differs from existing softwares. I will investigate the practice of hearing one's environment as music and how the development of my software and compositions are integrally linked to this phenomenon and make clear the importance of advanced rhythmic study within this practice. I am particularly interested in extending my own and others' perceptual capabilities to hear more and more complex rhythms accurately and congruently with what they would normally consider 'groove'. To this end, my project involves the development of softwares that: mathematically model the naturally occurring rhythms of specific species of frogs; allow the simultaneous occurrence of ninety-six different tempos; explore Miles Okazaki's *Rhythm Matrix*; enable the creation of new and complex grooves from simple beginnings via performance means; allow for infinitely complex variable mapping of musical parameters and rhythms via simple gestural controls; are completely modular and dynamic in design, thereby freeing the user from normal software design limitations. I will demonstrate the use of these softwares in my music compositions and analyse the compositions from within the context of rhythmic exploration and discovery.

KEYWORDS: composition, computer music, software, polyrhythm, algorithmic, experimental, groove, tempo, beat, poly-tempi, nature, frogs

Acknowledgments

I would like to sincerely thank the following people:

Dr Ivan Zavada for his continued support, guidance, help and efforts throughout my studies and in getting me to the finish line. Professor Matthew Hindson, for his encouragement and advice. Dr Ollie Bown for his incredible help with the ‘Frog Code’. Isaac Hayward for his ‘rhythm combinator’ javascript/html coding. Elliott James for his friendship, inspirational madness, help with testing software, and proof reading. My parents, parents-in-law and brothers for their love and support.

I would especially like to thank my wife Alex, without whom I would probably still be laying bricks... in the sun… ew!

Contents

List of tables/figures/illustrations

List of Abbreviations

ADSR	Attack decay sustain release
BPM	Beats per minute
CC	Continuous controller
CV	Control voltage
DAW	Digital audio workstation
GSA	Groove superimposition alignment
GUI	Graphical user interface
IMACS	Inter Max-App Communication System
IOI	Inter-onset interval
LFO	Low frequency oscillator
MIDI	Musical interface digital instrument
MOM	Miles Okazaki Method
NRPN	Non registered parameter number
PENCIL	Programming Environment of a New Creative Interface/Interactive Language/Logic.
SysEx	System exclusive
TURDS	The Ultimate Rhythm Discovery Station

CHAPTER 1

Introduction

1.1 Background and aims

In 2014, with our understanding of the world around us and our technological abilities, we can quantify, analyse and extrapolate with more precision what we hear around us, and in more musically varied ways. We can record audio and forensically extract information from recordings with precise and technical means that people have never been exposed to before. The point of this discussion is that now the human race has the ability to understand and learn about evolutionary and phenomenological aspects of the world they live in, with increased ability to educate and train themselves due to technological innovation. Within this context, the focus of this thesis is on music, and arguably, one of the most important components of music: rhythm.

I am particularly interested in extending my own and others' perceptual capabilities to hear more and more complex rhythms accurately and congruently with what they would normally consider 'groove'. The aim of this project is to create a suite of computer programs, literature, and audio, in order to help improve personal complex rhythmic understanding. One of the expected outcomes of this project is for the reader to use their new found rhythmic understanding to musically interpret the world around them. That is, for the reader to develop a deeper, intuitive, groove based musical appreciation of their environment (one where everything is in time and can be interpreted musically).

Further motivation behind the software developed for this project comes from my frustrations with the limitations of commercial hardwares/softwares currently available with respect to their rhythmic capabilities. That is, I appreciate the functionality of existing commercial software/hardware for musical use in all respects except for their rhythmic potential, which I find typically lacks ingenuity and originality. I seek to address these current software rhythmic limitations with my own software, so that I may achieve my intended aims.

1.2 Description of thesis and portfolio

This thesis will address my goals and motivations for creating new compositional tools in the form of computer music software. I will discuss each piece of software and how it can be used in creative ways. I will explore the use of some complex rhythmic devices in music composition, namely poly-tempi, frog call rhythms and nature music, the Rhythm Matrix concept created by Miles Okazaki, poly-tempi music, and performance systems that let you generate complex new grooves quickly and easily. Furthermore, I will investigate the use of one-to-many mapping in musical composition with relation to rhythmic and sonic discovery. In chapter three I will describe my softwares in detail and chapter four will address the compositions that have been created utilising the software developed for this project. The software can be found on CD No.1 (see Appendix D). The compositions can be found on CD No.2 (see Appendix E).

1.3 Definitions

Throughout this document, I will be using some terminology that may or may not be familiar with the reader and so I would like to clarify meaning for the following terms.

1.3.1 Rhythm

The following definitions of rhythm in a musical setting are provided as a context for the reader:

> 1. "Rhythm in music is normally felt to embrace everything to do with both time and motion —with the organization of musical events in time, however flexible in metre and tempo, irregular in accent, or free in durational values......For most listeners, nevertheless, the primary quality of rhythm is as an immediate succession of durations and occasional accents, approached and quitted in ways which participate in the shaping processes of the entire musical fabric, and which may even, in certain very 'rhythmical' works, seem to dominate that fabric." (Whittall, 2014, "Rhythm", para. 1).
> 2. Rhythm "(in the full sense of the word) covers everything pertaining to the *time* aspect of mus[ic] as distinct from the aspect of pitch" (Oxford Music Online, 2014, "Rhythm", para. 1).
> 3. "Movement, fluctuation, or variation marked by the regular recurrence or natural flow of related elements." (Merriam-Webster, 2014, "Full Definition of Rhythm", para. 3).

1.3.2 Complex rhythm

Throughout this text I refer to complex rhythms as being:

1. subdivisions of the beat other than the typically used two, three, four or six partials per beat;
2. prolongation/intensification of displacement of a beat for extended periods of time;
3. illusions as to what the tempo or time signature is;
4. poly-tempi, polyrhythmic and/or poly-metric patterns;
5. those that occur naturally by means other than human;
6. combination of any/all of the above in any way.

1.3.3 Poly tempi, polyrhythm and poly metric

Poly-tempi refers to the occurrence of more than one tempo being used at the same time in a musical scenario.

Polyrhythm is made possible by poly-tempi but the difference between the two is as defined by Nash and Blackwell:

> "As regards experiments in musical time, the notion of polytempi is crucially different from the relatively more common concepts of polyrhythm and polymetre, which both rely on simple integer divisions of the bars or beats in the piece. In contrast, the multiple simultaneous tempi of polytempo music leads to situations where the bar lines and beats of each part in the piece are themselves incongruent. The timing relationships between the events in each part can no longer be thought of, or expressed in, simple integer fractions (e.g. 3 in the time of 2, or 3/2 vs. 6/4), but instead become irrational." (Nash and Blackwell, 2008, p. 1)

Poly-metric refers to two or more coincidental time signatures occurring and can, by means of standard accenting within a bar, yield results that sound like polyrhythm (particularly at faster tempos).

1.3.4 Metric/tempo superimposition and metric/tempo modulation

Metric/tempo superimposition is the implying of a new time signature or tempo but where the existing or old time signature or tempo actually remains constant. This could also be considered to be ‘rhythmic illusion’. In contrast, metric/tempo modulation refers to where the tempo actually does shift to the new tempo and is not just implied temporarily.

1.3.5 Rhythmic displacement

Rhythmic displacement is the shifting of a rhythm from its current position in relation to a tempo, to that of either side of its current position, by any incremental amount of any subdivision of the tempo. A simple example would be a rhythm of four crotchet notes originally sounded as occurring on each beat of a bar of 4/4 being shifted by one semiquaver after the beat. The important consideration of rhythmic displacement is that it occurs at a fixed ratio of the tempo, and if the tempo of the music changes then the displaced rhythm will also adjust tempo such that the ratio remains constant.

1.3.6 Sync and Near-sync

Sync and near-sync effectively mean the same thing in this text (from the perspective of groove) but they have different initial conditions hence the use of both words. I use sync to mean the exact synchronous occurrence of events in time. Near-sync refers to events that happen within a tiny fraction of time from each other (say, less than 30 milliseconds apart), close enough for the observer to consider them as synchronised within the context of their understanding of the groove. The boundary of what is near-sync and what is not synchronised at all is subjective and non-definite, relying completely on the discretion of the listener.

1.3.7 Groove

As defined in The New Grove Dictionary of Jazz (Kernfeld 2014) the following definition of groove in a musical setting is provided as a context for the reader:

> “an unspecifiable but ordered sense of something that is sustained in a distinctive, regular and attractive way” (Feld, as cited in Kernfeld, 2014, “Groove”, para. 1).

The experience of groove is possible with any level of rhythmic complexity, and relies on the listener’s ability to maintain that sense of groove. That is, regardless of any apparent randomness of the rhythmic content, the listener can superimpose this “ordered sense” according to their own interpretation of the implied groove: this is what I call groove superimposition alignment (GSA). As seen in Figure 1.1 the more ordered the rhythmic material is, the easier it is for the

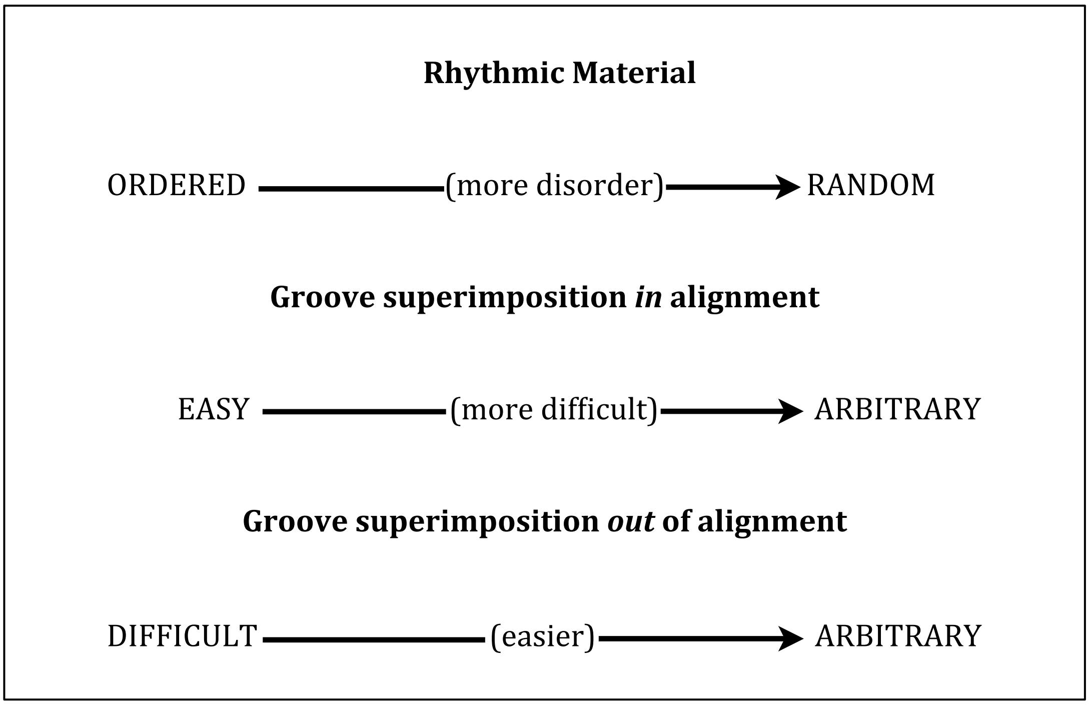


FIGURE 1.1 Maintaining a sense of groove on a spectrum of rhythmic order.

listener to superimpose a groove *in* alignment with that order and the more difficult it is to break away from the implied groove. As the rhythmic material becomes more disordered, the listener's GSA becomes more defined by the listener and less by the rhythmic material itself, therefore making it more difficult to sustain alignment with the material, but easier to sustain an *out* of alignment GSA. As though there is a ratio of GSA strength depending on the material's strength of order (i.e. the placement of strong versus weak components of the rhythm). Further, if the rhythmic material is completely random then the GSA becomes arbitrary. The sense of groove, therefore, is entirely within the subjective experience and superimposed by the listener onto *any* rhythmic material.

1.3.8 [WordsInsideBrackets]

This project has been designed in the multimedia software Max/Msp[1] and the nomenclature used to denote the name of a Max/Msp object involves the use of brackets around the name of the object. All Max/Msp objects discussed in this thesis will be indicated by using these brackets. For example: [thispatcher], [bpatcher], [metro], [button] and so on.

[1] For more information on Max/Msp, see www.cycling74.com

CHAPTER 2

Music - background, nature, rhythm matrix, metronomes

2.1 Context and general musical aesthetic; stylistic considerations

I have been interested in and sought out music of increasing rhythmical complexity since I was a teenager. Consequently, the music I compose is largely rhythmically complex in many ways. I tend to gravitate toward rhythmic emphasis in the composition, rather than placing importance on pitch, or timbral concerns. This often results in simple pitch material that is used to exhibit complex rhythmical relationships between and within phrases throughout my music. This type of compositional approach most formally comes under the heading of totalism (Gann 1997).

There are, however, problems that arise when writing rhythmically complex music in western notation. Particularly when using computer software to do so. Further, I have found that musicians (even highly trained ones) are typically inaccurate when it comes to the performance of complex rhythm and guess work is often used. This is a big problem for a composer who wishes to use these rhythms a great deal in their work and who wants to have their compositions actually performed (let alone performed accurately). There just is not enough time (money) to enable the required rehearsal to accomplish most of the musical ideas I want to present to the world when using musicians to do so. Moreover, rhythmically complex music such as poly-tempi music depends on accuracy of performance for the complex and often subtle relationships between the tempos to be clearly appreciated by the listener. Therefore, if a performer does not accurately perform any of the notated rhythms they are then likely to destroy the intent of the composer. Naturally then, over the years, I have gravitated toward realisation of my musical ideas via other methods. Namely, computers.

2.1.1 Music notation concerns and performance considerations

Commercial music notation software such as Sibelius[2] is very powerful and altogether an exceptional tool for any composer who notates music for people to perform. The capabilities of Sibelius for notating complex nested tuplets are very good and time signatures can be as varied as one wishes. However, writing music with complex rhythm is not without its pitfalls. For example,

[2] http://www.sibelius.com/home/index_flash.html (accessed 7/3/2014)

traditional western music notation does not have a note that equals five smaller units. There is a note that equals one, two, three, four, six, seven, and eight smaller units, but not five. This has been debilitating to my creativity when notating my music as, for example, the concept of playing in the time signature of 20/16, where the beat is every fifth semiquaver (thereby giving four beats to a bar), might often require a note that equals five semiquavers to be used. One workaround is to fill the score with ties between notes to add up to five semiquavers, and another would be to place the piece into 4/4 and add quintuplets throughout to generate the five notes per beat required. However, both of these solutions are cluttered, time consuming, and not ideal.

Further, when notating rhythm, there are standard ways to group beats and beams of quavers (and other shorter duration notes) together. This can make it difficult at times to find a suitable balance when writing a score with advanced rhythmic concepts, as for example, where two parts have notably different rhythmic material at the same time within a score, grouping the rhythm in one way might make it easy for one musician to read but hard for the other musician to read (based on the content). This may cause the composer to abolish consistency within their score and allow for different grouping practices for each part. Unfortunately this solution can lead to further concerns and problems for the composer such as difficult, convoluted and time consuming processes in order to create the score in this way (due to software limitations) and also difficulty for a conductor.

Yet another concern in traditional western music notation of rhythm is to do with groupings that suggests the way a passage is to be ‘felt’. In grouping a passage in the way it is to be felt, it may make the passage much more difficult to read by the musician, yet if grouped ‘correctly’, the composer’s intention for how the passage should be felt rhythmically is destroyed and the musician will not engage the rhythm in the way the composer intends.

These concerns and frustrations with western notation of rhythm, coupled with the typical lack of accuracy in performance by human musicians, have led me to pursue composition via non notation based systems, utilising the computer to create and to perform my rhythmically complex ideas in composition.

2.1.2 Totalism

According to Kyle Gann, a prominent thinker and writer about the musical genres of our time, totalism is music that has *"*the formal and textural clarity of minimalism, the energy of rock, the dissonance of modernism, and the rhythmic intricacy of Asian musics or even Cowell or Nancarrow*"* (Gann, 2001, "Totalism", para. 4). And "music that appeals to audiences on a sensuous and visceral level, and yet which still contains enough complexity and intricate musical devices to attract the more sophisticated aficionado" claiming that it is music with the "ability to yield more and more information on further hearings..." with "...inherent complexity, especially rhythmic complexity (...) Totalism can generally be characterized as having a steady, articulated beat, often flavored by rock or world music. That beat becomes a background grid for polyrhythms of great complexity (...) For totalists, being able to hear and calculate the complexity is essential (...) Totalist harmony can be either consonant, dissonant, or both - the distinction having ceased to be very important - but it is fairly static, concentrating on harmonic or melodic images that are easily memorable even when quite complex." (Gann, 1997, p. 355).

I find these descriptions of totalism to be surprisingly fitting for much of my own algorithmic compositional aesthetic, particularly when this genre was entirely unknown to me prior to my research for this thesis. I spent a long time working on musical ideas prior to discovering that they could fit under Gann's description of totalism, which encompasses many of the ways that I approach my own algorithmic, generative compositional processes. The concept of focussing on complex rhythm, often poly-tempi in nature, that still has a recognisable groove, and generating pitch/harmonic material in simple ways to make sure that the rhythm is the important feature of the music is very important to me as a composer.

2.1.3 Generative Art

My compositional output for this project, being largely created by means of software that is designed to generate complex rhythm that is parsed through user definable pitch content, is a form of generative art. Philip Galanter provides this definition of generative art:

> "Generative art refers to any art practice where the artist uses a system, such as a set of natural language rules, a computer program, a machine, or other procedural invention, which is set into motion with some degree of autonomy contributing to or resulting in a completed work of art." (Galanter, 2003, p. 4)

2.1.4 Progressive rock/jazz/metal/fusion/math

In recent years, there has been a growing trend in certain rock, metal and jazz/rock fusion music around the world towards more complexity in rhythmic content. Bands like Meshuggah, Planet X, and Mats Morgan are a few notable examples. The music of Swedish heavy metal band Meshuggah often uses long scale metric superimposition and other complex rhythmic devices (Pieslak, 2007) while maintaining simple pitch content and a consistent, relatively unchanging timbral aesthetic in a fashion somewhat congruent with the genre of totalism. Similarly, the music of Planet X, a band whose main compositional contributor is the drummer of the group (Australian musician Virgil Donati), contains many examples of complex rhythmic devices such as polyrhythm, poly-meter, rhythmic displacement, metric superimposition, metric modulation and more. So much so, that I have no hesitation considering it a main feature of his/their compositional style from composition to composition rather than something that the band does every so often. Moreover, the compositions of Virgil Donati in his own bands (The Virgil Donati Band and On The Virg), are almost exclusively built on those same rhythmic devices. Donati's rhythmic ideas have been instrumental in shaping my own musical ideas over the years that I have been composing. Figure 2.1 provides an example of rhythmic displacement used in the composition *Native Metal* (track one off the album *Serious Young Insects* by the band On The Virg, 1999). In this example, you can see the hihat part shift by one semiquaver (bar 107) and then before returning to a non-displaced state, proceed to be displaced by a semiquaver sextuplet (bar 108). This type of extended displacement is one of many occurrences of rhythmic complexity found in Donati's music.

2.2 Naturally occurring rhythmic systems and chaos

Utilising dynamic systems in music composition provide a way to include order and control but with the benefit of automatic variation with self similar tendencies. The inherent nature of the system is that it will apply similar patterns of behavior at both the micro and macro level (Harley, 1995). Further, chaotic systems can be used to model the evolvement of naturally occurring phenomenon in time (Steinitz, 1996) and for reasons of my rhythmic interest, they are likely to provide a large playground through which to explore new grooves and generative rhythm as will be discussed forthwith.

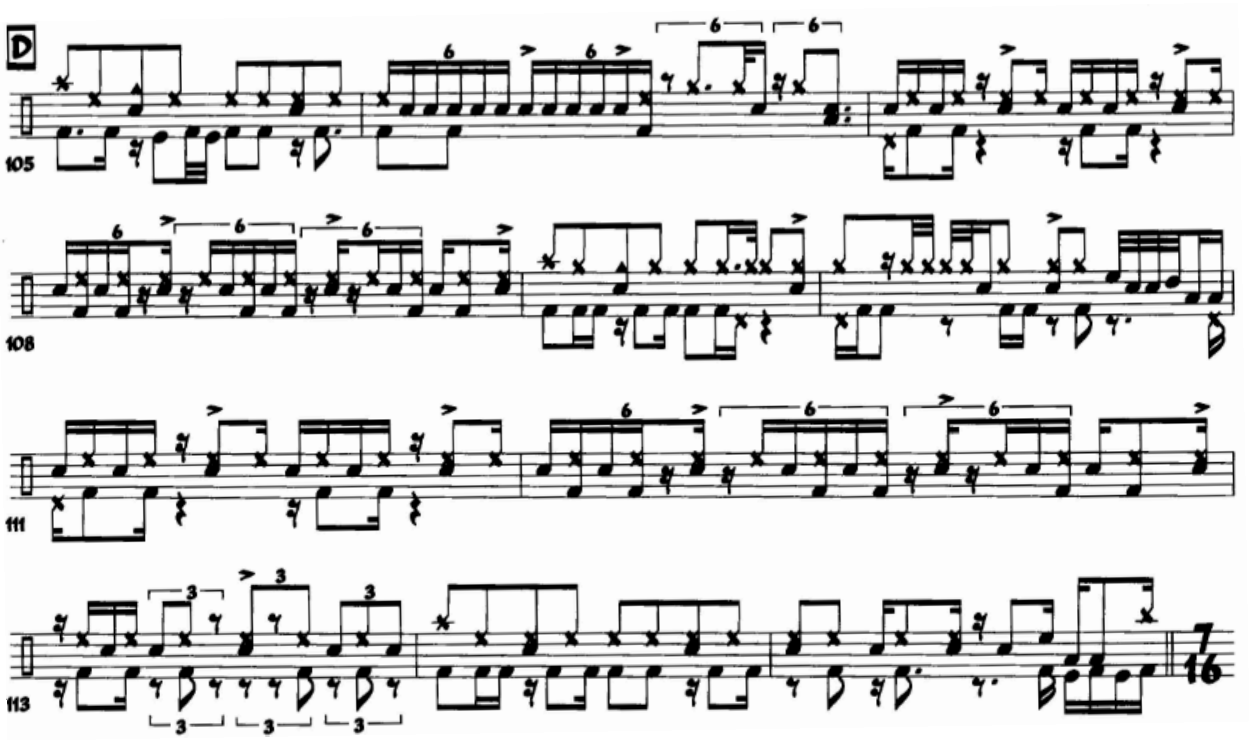


FIGURE 2.1: Excerpt of drum notation from *Native Metal* (Donati, 2005)

2.2.1 Frog call behavior

It is often observed in the calling behavior of certain species of frog, that male frogs call with a mostly consistent periodicity and yet interact with each other somewhat systematically so as to better attract female frogs (Greenfield et al., 1997; Greenfield, 1994; Aihara et al., 2006). The calling is not only non-random, but also follows criterion of a mathematically quantifiable nature.

Aihara et al. have indeed set out to mathematically model the calling behavior of the Japanese tree frog *Hyla japonica* for the potential benefits to robotics and/or artificial life (Aihara et al., 2006; Aihara et al., 2008; Aihara et al., 2011). Their modeling makes use of nonlinear dynamics and a representation of the frogs as coupled phase oscillators which, most importantly for my application, allows for the hocket-like nature of these frogs' calling patterns. Their research incorporates only the phase differences and the mathematical attractors to the frogs' "asymptotically stable equilibrium point." (Aihara, 2008, p. 30) (The point where the frogs call with relatively strict periodicity in almost perfect antiphase to each other). They leave out variation in frequency of calling, as well as equations that provide control over the build up time to the convergence of the anti-phase calling structure, both of which I will need to consider for future revision to the algorithms I have created in my software. And while they have limited their equations to a chorus of three frogs, my software allows for but is limited to six frogs currently. However, while not included in the user interface, it does have the capability of running any number of frogs in the chorus at any time.

While Greenfield et al. approach the problem of frog calling from the perspective of evolutionary biology, their research is also beneficial to the application of my software design in its mathematical modeling of frog calling behavior. For my application, the most valuable element of the model presented by Greenfield is the "central nervous system oscillator that may be inhibited and reset by an acoustic stimulus such as a neighbour's call" (Greenfield, 1997, p. 1355), (he calls this a 'resettable oscillator'). This is a similar feature to that espoused by Aihara but approaches the problem using a different mathematical approach (Monte Carlo simulations) and takes into consideration a more realistic setting of the calling scenario. For example, Greenfield addresses the masking effect of frogs calling at the same time, that is, that a frog who calls at the same time as another frog may not hear the other's call resulting in neither frog resetting its call timing accordingly. This process is important for a realistic model of frog calling and is lacking in Aihara's model.

Another element Greenfield examines and Aihara neglects is the application of the distance between frogs and the variation of volumes and time values resulting from this distance. In a computer model of calling frogs (as I endeavor to create), all of these elements (and probably more), need to be considered. Implemented currently in my software is Aihara's method but in future, Greenfield's methods (and a combination of the two) should be investigated.

My personal inspiration for using the rhythms of frog calling behavior in musical composition comes largely from the fact that in my backyard I have frequently observed a chorus of approximately six frogs whose rhythmic interaction have provided hours of inspired and stimulated listening. These frogs have been identified by Australian frog specialist David Stewart (personal communication, October 14, 2013) as *Crinia signifera* - Common Eastern Froglet. I have observed that these frogs call (in their small chorus) in almost identical ways to the Japanese Tree Frog as modeled by Aihira. As Aihara's modeling algorithm is relatively accurate to frogs found in my aural environment, it provides an intuitive foundation for translating my observations of nature into music.

2.2.2 Eco-structuralism

Developing out of eco-composition, eco-structuralism (as defined by Opie) is the process by which nature "...recordings are analyzed and the resulting data is mapped onto resynthesis processes to create new audio material" (Opie, 2006, p. 10). Opie's data mining techniques include analyses of amplitude, frequency, timbre and spatial structure. His method is to re-map the mined data onto new parameters to show the characteristics of the chosen data in a new musical way. Opie refers to these abstracted data sets as 'structures'. The primary rule of eco-structuralism is that these structures must be derived from nature recordings and must remain in series (chronological order) no matter how it is transformed (for example transformations such as elongation, compression and inversion among others are discussed), that is, the data can not be segmented and mixed up. Opie defines these rules and considers practical applications. (Opie and Brown 2006). Strangely, in Opie's available writing on eco-structuralism to date, the strict use of periodicity and rhythmic data extracted from nature recordings, seems to be missing from his thoughts on the subject.

The use of periodicity and rhythmic data from recorded material is however explored by Sandred in his *Interpretation of everyday gestures - composing with rules* (Sandred, 2004) where he considers, in layman terms, a process for quantifying rhythm of a non-musical source into notate-able music to expand a composer's music generating toolset. Unfortunately, Sandred does not offer a mathematical model, nor (disappointingly) does he provide the details of the computer program he created to generate the results discussed in his paper.

The concept of taking existing data and re-synthesising it in the audio domain has many applications including, but not limited to: the monitoring of a mechanical system by an engineer listening to compressed sonification of data (listening for inconsistencies or abnormalities in a large scale system such as a factory manufacturing plant); and improvising electronic musicians and composers creating new music. For my project, taking data from nature (frog call rhythms for example) and then mapping it onto a spectrum of quantisation between naturally occurring rhythms and highly simplified (quantised) rhythms is probably somewhat of a tangent from Opie's eco-structuralism concept. However, I believe that it still follows the primary rules defined by eco-structuralism with the difference that my 'data' is generated algorithmically and is not itself from natural sources, (yet is modeled entirely on natural phenomenon) and remains in series,[3] although through quantisation the structural integrity of the series may be somewhat compromised. Sandred's paper (un-intentionally) provides a link between Opie's research and that of more detailed studies in quantisation that are discussed now.

2.2.3 Quantisation of nature

In my own software, I seek to have the option to quantise the modeled frog call timing into common (or uncommon) musical time values. The quantisation process from naturally occurring rhythms to any other version of them along a quantisation spectrum is a problem in which many considerations need be made. Human rhythmic perception studies will illuminate necessary components to consider. I am seeking to create software that incorporates current knowledge of human rhythmic perception so as to maximise the musical merit that is output by the software. For example, as discussed by Leigh M. Smith of Cowan's research, the subjective present, short auditory stores, and long auditory stores, offer great insight into ways in which computer models might be made. As Smith states, "These limits influence the process of grouping temporal events, establishing limits a computational model should address" (Smith, 2000, p. 16).

As well as promoting his own *multi-resolution wavelet analysis* of rhythm which has applications in rhythmic transcription, quantisation and realtime musical accompaniment by a computer, Smith provides a comprehensive overview of relevant human perception research with regard to rhythm (Smith, 2000). There are currently many varied methods of rhythmic quantisation proposed by researchers in the field. Cemgil et al. provide a model of quantisation called *vector quantisation* which makes use of Bayesian statistics and demonstrates how its application works with greater accuracy than rounding methods (Cemgil et al. 2000). Both Smith's and Cemgil's methods of quantisation seem to be the most applicable and useable as aids in the future revisions of my software.

2.2.4 Bugs and animals = music

In his book *The Great Animal Orchestra*, the sound recordist Bernie Krause provides insight to a lifetime of experience of being in nature, simply listening. His work as a field recordist has

[3] Maybe my project could be labelled as a sub-category of eco-structuralism under the title of *eco-rhythm* (or similar).

taken him to all parts of the world and his exposure to many and varied natural environments should be highly respected due to its immense scope. He suggests that the evolution of music may have preceded the development of language and that music has evolved directly from our experience in the natural world (Krause, 2013). That is, from nature itself, came music. While this may seem to be unscientific conjecture (to which I concur), as an artist, I find myself looking deeply at where my inspirations come from and to this end, nature has been a true source of musical inspiration for me as a composer. Similarly, David Rothenberg in his book *Bug Music* provides multiple accounts of listening to nature (particularly bugs in this case) and interpreting it as music. He places great importance on rhythm and noise throughout the book and oft discusses the precedence of rhythm to all other possible musical qualities (through natural evolution of insects and similar creatures) by millions of years (Rothenberg, 2013).

When I listen to certain music, it very often invokes landscape. Not just a painting, but a moving landscape. This internal sense of traveling through terrain is very strong in my inward imaginings while listening to many types of music. Certain improvised drum solos by Australian drummer Virgil Donati, invoke in me a sense of traveling down a calm stream in a canoe at a fixed speed with the drums being played in time with the visual placement of objects that lie on the shore (and beyond) perpendicular to the movement of the canoe. So the canoe is a timeline and the banks of the stream provide the musical score.

Similarly, much of my own through-composed instrumental music invokes the sensation of traveling through a landscape of some sort. Often futuristic and often on a fixed medium like a train on rails. This to me implies the sense of continuous time, a fixed momentum, an unwavering movement forward where one becomes a passenger to the flow of sound coming from the instruments or speakers. It is interesting to note that in order for me to best hear nature as music, I need to be completely still while time moves everything onwards. I would stand outside, next to a small body of water (usually a dam or pond), and after some time become entrained by the calling rhythms of the frogs and insects nearby. I would start to lose my sense of self and begin hearing everything happening around me as one ‘thing’ playing music. This is the most succinct way I can explain the phenomenon. During these experiences, my complete awareness of reality was entirely focussed on the sound world around me as a musical instrument, playing itself out to the score of cause and effect (albeit chaotic). During these experiences, it became completely apparent that the most central element linking everything together was (when things were occurring in time): rhythm.

The perspective of the speed of this movement of time, this groove, is completely arbitrary. The unimportance of the ‘tempo’ of the progression of time is one of the most enlightening things to consider when listening to nature as music. However, the act of applying a non-fixed, preferential weight to one tempo over another at any point in the moment to moment experience (based on the incoming perceptual information or not), *is* important to my experience of hearing music in nature. That is, I perceive the events occurring in time, against an arbitrary and oft changing tempo (sometimes more than one concurrent tempo), thereby creating a groove that is unified by that arbitrary perception of a pulse. It is from giving weight to the *perception* of a pulse (that is entirely subjective), that groove is given the possibility of existing.

This general concept of the perception of groove is entirely personal to each individual listener, even if the pulse is physically presented to the human senses (like a ticking clock for example). It is the listener, the observer, the subjective interpreter of events, the one who places weight or importance on a pulse, that is the creator of the groove. For me, the ability to listen to nature as music follows this train of thought and is repeatable at any time. Why stop at nature? Indeed… What I seek to obtain in my own life (and engender in others) is the ability to listen to *all that exists* as music, not just nature. The difficulties I personally have with obtaining this state of mind are related to the problem of noise pollution as described by Krause (Krause, 2013). That is, I have some difficulty accepting man made sounds within the sphere of a nature setting and they are more often than not a distraction than an additional musical flavour. Perhaps further reading and understanding of John Cage’s work will help me in this regard.[4]

2.4 Rhythm matrix

The American composer and jazz guitarist, Miles Okazaki, published (online) a book called Rhythm Matrix[5]. This book contains a numbered list of 4928 *rhythmic modes* as Okazaki calls them. The rhythmic modes are every single possible permutation of groups of two and three subdivisions of the beat across a four beat measure, from quavers (1/8th notes) through to demisemiquavers (1/32nd notes) with all tuplets in between. Okazaki describes the book in the

[4] That said, Irving Godt would beg to differ, claiming that music must be created by humans with intention, and where those two (and other) criteria are not met (like sounds from nature for example), then it is not music, no matter how much you want it to be so. (Godt, 2005).

[5] http://milesokazaki.com/fourpulse.php (accessed 07/02/2014)

following way: “There are no ‘rhythms’ in the list, only groupings of 2 and 3, which could be used as ‘containers’ or ‘skeletons’ for actual rhythmic material.”[6] He also provides three possible ways of utilising the book in practise with a short description of ‘linear practise’, ‘contrapuntal practise’ and ‘dynamic cycle practise’.

Having worked through Okazaki’s Rhythm Matrix book in numerous of my own ways in practise, I started to develop an even more refined awareness of normal every day sounds occurring as rhythmic events pertaining to a pulse. That is, it became easier to fit the world around me into a musical structure rhythmically, with said structure being a constant pulse. It enhanced my rhythmic perceptual capabilities in listening and also physically in performance.

As I found the method so simple and beautifully architected, I decided to produce my own ‘Rhythm Matrix’ books, built on the same design concept. The design concept is simply this:

1. Choose any amount of beats.
2. Choose any subdivision of the beat.
3. Choose any groupings of the chosen subdivisions. (for example, groups of three and groups of five).
4. Notate all possible permutations of chosen groupings of chosen subdivision across chosen amount of beats ensuring that the end result begins and ends on an even grouping.
5. No mode is listed twice.

Following this procedure, I wrote two of my own Rhythm Matrix books (see Appendix C). The first book I created could be thought of as an addendum to Okazaki’s original book as it is also across four beats. It differs from his book as it contains all possible permutations of groupings of: two and four (triplets, quintuplets and septuplets only); three and four; two and five; three and five; and finally four and five (where his book used groups of two and three only). All of which are again, across quavers through to demisemiquavers. This process yields 1745 rhythmic modes. The second book I created is groupings of three and four across six beats again using all subdivisions between quavers and demisemiquavers. This process yields 5981 rhythmic modes. A diagram showing all of the possible rhythmic pathways (or all of these modes) is shown in Figure 2.2. Both of these books, as well as a javascript based html code (written by my friend Isaac Hayward

[6] http://milesokazaki.com/img/fourpulse/003fourpulse.jpg (accessed 07/02/2014)

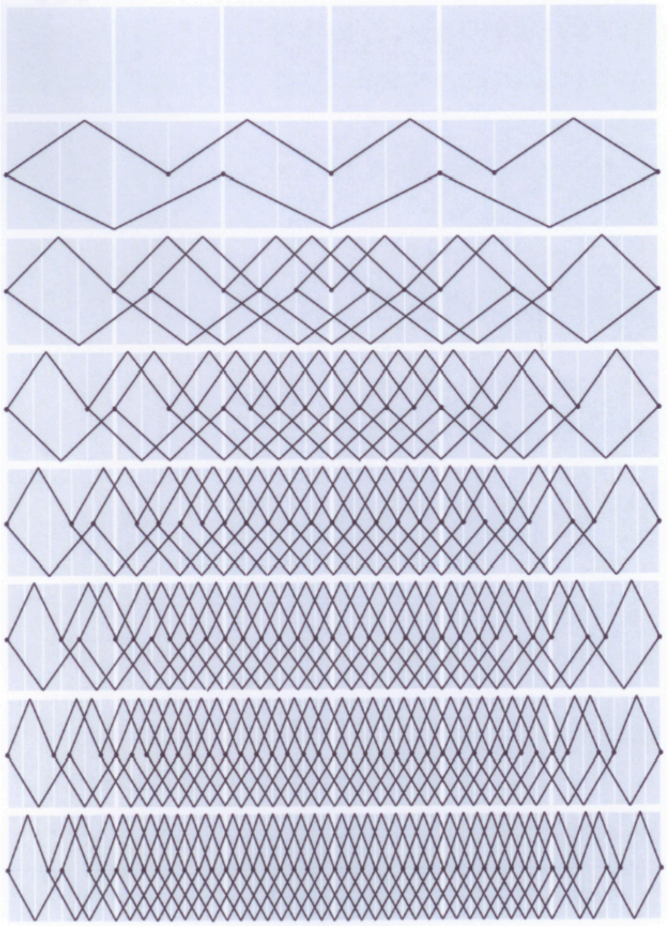


FIGURE 2.2: Rhythm Matrix groupings of 3 and 4 subdivisions across four beats.[7]

specifically for me and used with permission, for the purpose of being able to generate lists of rhythmic modes after defining the overarching parameters) are included in Appendix C. After creating the books, I decided to create software in Max/Msp to be able to perform any of these modes. This software, called The Miles Okazaki Method (MOM) is discussed in Section 3.6.

2.5 Metronomes and poly-tempi music

At some point in my teenage years (now in my thirties) I began experimenting with multiple hardware metronomes running at the same time. To me, hearing the complex yet surprisingly simple relationships between multiple speeds being produced by the simple (and unchanging) sonification of digital metronomes was a joy and very inspiring. It was not until much later that I discovered Gyorgi Ligeti and Conlon Nancarrow as being two notable composers of poly-tempi music,

[7] In the diagram (Figure. 2.2), you can see how close many naturally occurring events in time would be to the quantised positions the rhythmic modes and how one could navigate (in relation to a constant speed) from the naturally occurred rhythm along a quantised rhythmic pathway through any combination of the modes to achieve a perceptually significant match to being perfectly in time (or near-sync).

Nancarrow through his player piano pieces and Ligeti most profoundly through his composition for 100 metronomes. These are discussed forthwith.

2.5.1 Taking control of Ligeti's Metronomes

Ligeti's *Poème Symphonique pour 100 metronomes* explores poly-tempi music heavily yet relies on chance and many non-determinate parameters. In repeated performances of the work, the dynamic and textural structure of the work is about all that can be guaranteed to be similar from one performance to the next. The interest and influence I have taken from this work is its use of multiple, concurrent and unique tempi. This opens a doorway into an endless world of stimulating and interesting rhythmic grooves and patterns. These rhythms are best discovered in *Poème Symphonique* during the second half of the piece, typically after many of the metronomes have stopped ticking, enabling the rhythmic interaction between the remaining devices to be able to be heard more clearly. It is like a kaleidoscope of rhythmic ratios continuously getting simpler as each metronome voice drops out. The listener is also better able to latch onto the rhythms due to repetition and the lessening complexity thereby facilitating better understanding of the grooves and patterns created by the remaining metronomes. I noticed also that the rhythms discoverable in *Poème Symphonique* contain many similarities to naturally occurring sonic phenomena such as frog and insect calling. These similarities are due to the metronomes phasing in and out of sync with each other across multiple complex rhythmic structures and subdivisions concurrently.

2.5.2 Poly-tempi composition

During the twentieth century, there have been a few notable uses of machines to realise compositions of greater rhythmical complexity. The Rhythmicon, an electronic instrument created in 1931 by Leon Theremin for Henry Cowell[8], provides a composer with a tool that can perform up to sixteen different tempos at once (based on the harmonic overtone series). Further, Conlon Nancarrow and James Tenney utilised player pianos to perform many of their compositions that contain poly-tempi material. This use of the machine enables music to be realised by a composer without relying on human performers, and indeed allows the creation and performance of music that would exceed the physical (and mental) capabilities of a human performer. To create my own poly-

[8] http://musicmavericks.publicradio.org/rhythmicon/ (accessed 12/02/2014)

tempi compositions, I too am utilising a machine, the computer, and in particular, I am creating my own software for this creative exploration.

Robert Wannamaker recently published a paper in Music Theory Spectrum that provides a succinct overview of the major contributors in the world of poly-tempi music composition followed by detailed mathematical analysis and descriptions of the rhythmic particulars of a poly-tempi system (Wannamaker, 2012). He utilises Farey sequences to explain the rhythmic patterns that can be found within the context of the poly-tempi phenomenon. Many rhythmic features are described in detail of the divisive polyrhythmic array. This is an excellent paper to review should one wish to understand more about poly-tempi systems. In summary, Wannamaker discusses the following concepts, all of which are applicable to poly-tempi music and useful for analysis of my own poly-tempi compositions contained in this portfolio:

- *Rhythmic glissandi* (also Farey arc, also flanking curve): where the tempos are aligned in such a way that they appear to rise toward or fall from a beat that is rhythmically separated from the other material.
- *The five distinct heirachical levels of perceptible rhythmic groupings*:
    - individual attacks;
    - attack groups residing together on a single flanking curve, and having the character of arpegiando figures or grace notes preceding or following a metrically stronger main attack;
    - a constant periodicity equal to that of the highest voice, which is variously subdivided into ‘tuplets’ by the cyclical patterns of voices presented in the succession of attack groups;
    - the irregularly spaced transitions between these different n-tuple meters, which correspond to transitions between sets of flanking curves about asymptotes of different reduced denominators n;
    - the periodicity of the polyrhythm as a whole, which….corresponds to unity.
- *Buffer zones*: His analysis explains how ‘buffer zones’ exist near strong ratios with low denominators (like 1/2 and 1/3). (Wannamaker, 2012)

2.5.3 The metronome game - rhythmic discovery and learning made fun!

Inspired by Ligeti’s *Poème Symphonique* I purchased over one hundred simple hardware digital metronomes during the course of this research and began experimenting with different combinations of tempos and start time offsets. During this experimentation I noticed many of the rhythmic particulars mentioned in Wannamaker's article.[9] For example, the 'buffer zones', 'flanking curves' and the 'rhythmic arpeggiations traversing multiple rhythmic layers' (Wannamaker, 2012). These rhythmic effects became highly intellectually and musically stimulating and encouraged new thoughts in me about rhythm and time in music. Particularly in my interests in developing skills at being able to listen to nature and hear it as music. For me, the 'way in' to that experience being a rhythmic perception, being capable of discerning complex input and interpreting it against an arbitrary pulse to produce an internal sense of groove (see FIGURE 1.1).

I devised a simple game to play with the metronomes that, in playing, provided much further inspiration and understanding of these rhythmic peculiarities. The game requires:

1. many metronomes (arbitrary amount);
2. preferably each one being the exact same brand and design/model (so that they are perceptually identical aurally);
3. LED lights that visibly flash when the metronome itself 'beeps';
4. a playing surface.

2.5.3.1 The game instructions

1. Choose any amount of metronomes you like and set them all to either a random tempo or incrementally (from any starting point). Example: 20 metronomes set to tempos incrementally increasing by 1BPM, starting from 60BPM.
2. Make all metronomes have the same settings (except for the BPMs)
3. Start all of the metronomes (the start time of each metronome with respect to other metronomes is not important).
4. Place all running metronomes on a flat surface facing upwards so that the flashing LED of each one is visible.
5. Sit and listen to the metronomes all running together and concentrate only on them.
6. After an arbitrary amount of time, begin to notice which metronome stands out as being a reference point from which you are hearing all of the other metronomes in relation to.

[9] Let it be known that I had not read the article until much later, in fact, I built the Spiral Rhythm Clock and was making music with it long before discovering Wannamaker's article.

7. Listen to the rhythms of all the metronomes combined from the perspective of the tempo you have found to focus on for an arbitrary amount of time.
8. Find the metronome that you are using as your reference tempo (by looking for its flashing LED that should correspond with the tempo you are hearing), and then turn it off.
9. Repeat step 6 to 8 until only one final metronome is left on.

Following these steps to play this game you should see that there is no winner and no competition, no time limit, and no reward other than the experience gained of listening to *x* amount of time of poly-tempi beaps from *y* amount of different tempo perspectives.

This game starts out quite difficult, depending on how many metronomes are being used, as it can take some time to 1) find an individual tempo amongst all the metronomes running to hear and isolate as the tempo from which to listen them all from, and 2) physically find that metronome on the playing surface. Especially when there are tempos that are only one BPM apart from each other. You must observe carefully that the tempo you are hearing is actually coming from the metronome you think you have spotted as being the one.

2.5.3.2 Game variations

There are many ways this game can be varied, each with its own benefit. For example, if step 6) as defined above is too difficult for the player then maybe instead they can arbitrarily choose any of the running metronomes physically and purposefully force your rhythmic perspective to use its tempo from which to perceive the resulting rhythm of all of the metronomes combined. (Turning the chosen metronome up louder than the others might be a good method to aid in this process). Further to this variation, the player could turn up (or focus on) more than one metronome at a time and listen to the resulting rhythms from the perspective of the groove created by two (or more) metronomes at different tempos combined.

Another variation might be to include more than one player, each following the exact same steps listed above but both listening to the same group of metronomes. Although completely relying on honesty and fair play, maybe the winner of this variation would be the person who removed the most metronomes by the end. Making the game into a competition like this could be useful in providing the player with incentive to find tempos quicker and as a result, enable them to get better at it in the first place.

Yet another variation might be to impose musical structures like time signatures and phrase lengths (in bar numbers) from which tempo perspectives are maintained before moving on to the next. This variation provides additional and deeper layers of rhythmic perception and musical abstraction from which to understand the resulting rhythmic results of this poly-tempi scenario.

Lastly, a performance component could be introduced to the game, where the player has to play along with (in any way, using any instrument, percussive or not) their chosen tempos in addition to following the normal steps of the game. This variation is particularly effective if the player is playing the game in pairs (or with a group of players). Further variation to this might be that the player must not actually articulate the chosen tempo directly. Meaning, the player can not simply tap out the tempo they have chosen. This final variation causes a more physical interaction with the other metronomes that are running and therefore might cause the player's listening, reactive and predictive skills to be developed or enhanced.

2.5.4 The beginnings of Spiral Rhythm Clock

For the purposes of this discussion, I am seeking to strip musical parameters back to the rhythmic element only and not be concerned about pitch or timbre aside from the following criteria (in accordance with my 'totalism' aesthetic): 1) other musical qualities must not get in the way or distract from the rhythms, nor be a focus, and 2) they must articulate the rhythms sufficiently. That is, the envelope and duration of the sound must be adequate for the listener to fully comprehend the inter-onset-interval (IOI) of each rhythmic iteration and its relationship in time to the other events surrounding it. For this reason, sonification of the tempos with short durations and with envelopes that have a fast attack are preferred and are generally the type of settings that I have used in the compositions contained in the portfolio. (Percussive sounds such as those produced by a wood block being struck by a drum stick are an excellent choice).

2.5.4.1 Perceptual relationships between tempos

When two separate tempos co-exist, the interaction between them follows a structure that is similar no matter what the tempos actually are. The structure of this interaction is akin to Reich's phase music in that the return to and falling away from sync is continuous. If the two tempos are very similar in value, 140 and 141 beats per minute for example, beginning in perfect sync with each other, the faster tempo will precede the slower tempo by increasing amounts until a half way point is reached where they will appear to be in hocket with each other (in this example, at the thirty second mark). After this time the faster tempo will approach the slower tempo again but from the opposite direction in time until they coincide in perfect synchrony at the sixty second mark. Over the course of the whole minute, the rhythmic effect of the two concurrent tempos will often sound as if they perform partials of simple subdivisions of the beat. (In this case, either the faster or slower tempo can be arbitrarily chosen as being 'the beat'). These subdivisions include triplet quavers, semiquavers, quintuplets, sextuplets, septuplets and so on. Whether they achieve a mathematically perfect subdivision partial (sync) or one that is perceptually close enough (near sync) to sound as though it is any particular subdivision partial is not important for this discussion.

Arbitrarily adding a random third tempo to this example, illuminates the perceptually simple subdivision sync/near sync even more so as it enables more continuity. That is, where there are only two metronomes, aside from the quaver hocket sync, all other perceptual subdivision syncing only contains two of the subdivision partials (the other partials not possible to sound with only two metronome sources). So with three metronomes, it is perfectly reasonable to observe a triplet with all three partials of the triplet sounding during the phasing of the metronomes at some point. Adding a fourth enables the possibility of all four semiquaver partials being sounded in sync or near sync from the perspective of any of the tempos used and so forth.

As you add more tempos to the mix, these syncs or near syncs are perceivable from multiple tempos often, even at the same time, creating many more occurring sync events from different tempo perspectives throughout the poly-tempi phasing system. This is where some highly sophisticated and most interesting grooves can be found. When the listener is able to change which tempo they perceive as the beat from which subdivisions are perceived at any given moment during the phasing, they are then participating in a rhythmic 'choose your own adventure' kaleidoscopic experience.

In a poly-tempi system that has fine control over all parameters such as Spiral Rhythm Clock, it is also possible for a composer to coerce the listener into perceiving specific tempos or subdivisions at any given time by altering usage of pitch, timbre, duration and dynamics of the sounds used. Typically and within certain limits, I have found that faster, higher pitched sounds tend to attract my attention as being the beat from which to perceive all other activity.

2.5.4.2 Meta-beats within poly-tempi systems

The poly-tempi system described so far is not limited to listening to only one tempo at a time as the main tempo from which all the other tempos are considered as relating to. As discovered by playing the game described in section 2.5.3, it is also possible to construct ‘meta-beats’ out of two or more tempos from which all the other tempos occurring become the phasing subdivision partials. This can be done either intentionally or arbitrarily in a moment to moment subjective way. The meta-beat, even while it is itself phasing and non-permanent, can become the perceptual structure that is ‘fixed’ while the other tempos are the ones perceptually phasing within the context of that meta-beat. This becomes much easier to perceive when the meta-beat is derived from tempos in their sync or near-sync phasing within the context. For example, if 140, 141 and 109 beats per minute (BPM) are running concurrently with 140 BPM and 141 BPM started in sync and 109 BPM started arbitrarily, then at the thirty second mark it is possible that the listener can perceive 140 BPM and 141 BPM as being in near sync quaver hocket (or tempo doubling) for a period of a few seconds, during which time the 109 BPM tempo can relate to this hocket/tempo doubling combination in a variety of ways rhythmically. This can be extrapolated out indefinitely, particularly with large amount of metronomes being used in the context. This is why I find the poly-tempi system to be an immensely rich and fertile playground of possibilities for enhancing rhythmic awareness and discovering new grooves. A person can use this system to train themselves to hear and understand rhythms of increasing complexity from a musical perspective and the possible combinations are effectively inexhaustible.

2.5.4.3 Poly-tempi similarities in nature

The rhythms generated in this poly-tempi experience are very often similar to what a person can observe when listening to frogs or insects calling. The phenomenon of gravitation towards sync in the calling behavior of frog and insects (as discussed in section 2.2.1), often produce rhythms akin to any poly-tempi system. But these similarities are only temporary, appearing only in short ‘glimpses’. In my observation, the more tempos that are used in the poly-tempi system, the more frequent these ‘similar to nature’ rhythms are produced due to the increased complexity of adding more tempos to the mix. Adhering to the sonic restrictions previously stated regarding my preferred sonification methods of a poly-tempi system, one can create a rhythmic study or composition that can simulate, or in many ways be similar to, naturally occurring phenomena such as frog and insect calling.

2.5.4.4 Poly-tempi conclusion

Stemming directly from my experimentation with multiple hardware metronomes and the rhythmic particulars of interest that I discovered as listed above, I decided to create a piece of software called the Spiral Rhythm Clock (SPIRAL) (which will be discussed in more detail in section 3.7).

CHAPTER 3

Software Design

3.1 Rhythmic complexity in commercially available music software/hardware

In this time, most commercial computer software and hardware designed for use in creating music is highly limited in its rhythmic capabilities. Almost all commercial music sequencers available, even in the most widely used and highest regarded professional digital audio workstations (DAW) such as Pro Tools, Logic, Ableton Live[10] et al, contain severely crippling rhythmic limitations. Such limitations are predominately based on the lack of options in the quantisation of timed events.

Most commercial software and hardware music sequencers are limited to the following musical rhythmic grid systems: 1/4 notes, 1/8th notes, 1/16th notes, 1/32nd notes, 1/64th notes, the triplet variation of same, the dotted variation of same, and the 'swing' variation of same. While there are exceptions, very few popular DAWs allow for grids of quintuplet, septuplet and other higher order tuplets, and even rarer still are nested tuplets[11] found within these applications. This grid based limitation is also almost always coupled with an embarrassingly small amount of possible time signatures on offer for the composer to use. Many music making programs (in software or hardware) offer the options of only 2, 3, or 4 beats per bar and do not even allow the possibility to change that time signature within the composition. That is, the composition is fixed in that one time signature for its entirety. This limitation can be worked around by placing events on the grid in ways to make it actually sound like it is changing time signature but the user interface experience makes this very difficult and time consuming.

Adding to this conundrum is the incredible amount of hardware devices (almost all of them!) whose user interface has sixteen pads or buttons corresponding to the sixteen grid locked steps in time from which the music that the hardware has been created to create is presumably expected to be written.

[10] http://www.avid.com/US/products/family/Pro-Tools; http://www.apple.com/au/logic-pro/; https://www.ableton.com/ (accessed 07/03/2014)

[11] 'Nested tuplet' simply referring to tuplets within tuplets (within tuplets (within tuplets (etc)))

To be clear, I do understand the refuting arguments one could make about the design and intention of these hardware units, with regard to expected users, using the devices for specific musical genres. While there is an obvious concern in a market system to create tools for the public that will ultimately lead to profitable return for the proprietor, I am discouraged by the lack of creativity in hardware design in this time for the exploration of rhythmic possibilities.

If the user interface grid cannot be changed, I, the user, am most likely going to either resort to sticking within the bounds of that grid system or, spend a great deal of time counting tiny little lines and moving events around manually to a new system that I have to imagine in my head while looking at something entirely different. This is a problem for a composer who does not want to stick to a rhythmic grid, imposed by the design of the software or hardware, yet wishes to use all the other features and functionality of the software. Of course, much music has been, can be, and will continue to be made within these limited grid systems. Further, I am not denying the potential complexity of rhythmic output made possible when only utilising these simple grid systems. There are many ways (work arounds) to achieve complexity using such tools. The problem is that these work arounds are, in most cases, far too convoluted, involved, and difficult to achieve, and so deter the user, me, from bothering in the first place.

While I have been arguing that the current tools available for composers are most often severely limited rhythmically, there of course exist some exceptions to this. Bounce Metronome (by Robert the Inventor)[12] has many poly-rhythmic possibilities but is predominately designed as a practise tool rather than a tool for composing. It is also built for Windows[13] operating system only. However, with its MIDI capabilities, one could record or send the MIDI output and use the rhythms generated by Bounce Metronome from within their DAW, assigning pitches manually according to their needs. However this is a slow and painstaking process, with an inherent lack of efficiency which in itself is a deterrent. In other words, Bounce Metronome is not primarily designed as a composition tool.

[12] http://bouncemetronome.com/ (accessed 29/03/2014)

[13] http://windows.microsoft.com/en-au/windows/home (accessed 17/03/2014)

A plethora of other, *non-commercial* software exists and can be found via internet forums such as Max/Msp and Reaktor[14] that do allow for more sophistication in their rhythmic capability. These applications can often suffer however from the lack of customer support (because they are offered free in a 'use at your own risk' way), as well as a lack of documentation and poorly designed user interfaces.

As a composer who wants the practicality and excellent functionality of using the existing popular commercial software/hardware tools for composing, but does not want to be confined by the rhythmic sequencing limitations of said tools, one needs to take matters into one's own hands. Thus, I have designed five specific applications within the Max/Msp programming environment that break down the walls of the existing grid system (status quo) of contemporary music composition tools. These tools are now at my hands as a composer, and I am free to experiment with them in any musical way I see fit.

3.2 Modularity in software design

The design model of modular synthesis is very interesting from a composers point of view. This design model being that you are presented with a limited amount of modules that can be patched together in pre-defined ways with each parameter being editable via an interface consisting of, among other things, knobs, buttons and sliders. This gives the synthesist a highly advanced playground of sonic exploration from which to create, invent and imagine new sounds and sound morphology. This is the idea that strikes me most powerfully as a composer. The ability to have high level musical data manipulation modules, pre-built with their own functionality and purpose, and to be able to patch them together in any way you can imagine, yields a very versatile possibility of output.

A traditional composer of notes and rhythms writing for traditional acoustic instruments has before them a similar modular experience. One that is also infinite in scope. But that composer has to start from scratch. Ideas must be notated one by one and even with copy and paste functionality of modern notation computer software, the process can be somewhat slow. Contrarily, taking the notation out of the experience and giving the composer a program that has many complex features

[14] http://cycling74.com/newest-topics/; http://www.native-instruments.com/forum/forums/reaktor.30/ (accessed 7/3/2014)

enables a different type of creativity. Even if the end result is still 'notes and rhythms', and even if the end result would have been notate-able using traditional music notation, a composer using such software must engage different creative methods and incorporate a new musical thinking to achieve results that will be pleasing to them.

Many such computer music programs exist today that are modular in design with a high-level user interface for organising and controlling musical ideas/sounds. Reaktor by Native Instruments, the Kyma system, Max/Msp and Pure Data are but a few.[15] Then there are even higher level modular design systems stemming from these existing applications. In particularly, many users of Max/Msp have built their own libraries of abstractions and modules (utilising Max/Msp's [bpatcher] object) that can be incorporated quickly into their Max/Msp patching. Such examples are Beap, 110 Modular, Jamoma, Euromax, telePort, and Vizzie.[16] Though not entirely, most of these existing systems are limited to the processing of audio/video and not data or MIDI information.

Here I would like to directly quote Shawn L. Decker et al, regarding the benefits of a modular software design for use in composition:

> 1. "Synthesis routines and composition tools can exist as independent programs implemented in the manner that is most efficient and appropriate to the task.
> 2. Because synthesis and composition programs are not bundled together in large packages, the repertoire of programs can grow gracefully through the addition of new synthesis algorithms or new composition tools
> 3. Composition programs are independent of the particular synthesis algorithms and also independent of whether the synthesis is performed by hardware or software. Thus it is possible to provided a unified approach to real-time and non-real-time activities.
> 4. Individual composers with unique needs can customize the environment by adding their own programs, but they need not create their composition software entirely from scratch.
> 5. Special categories of users (for instance, beginners in sound synthesis) can be given software environments with user interfaces and synthesis programs that are tailored to their needs and capabilities.
> 6. It is possible to substitute new hardware for synthesis programs or for compositional input routines. This allows new hardware devices to be integrated easily into the software environment." (Decker, Kendall, Schmidt, Ludwig, Freed, 1986, p. 29)

[15] http://www.native-instruments.com/en/; http://www.symbolicsound.com; http://cycling74.com/; http://puredata.info/ (Accessed on 17/02/2014)

[16] https://github.com/stretta/BEAP; http://jamoma.org; http://www.behance.net/gallery/Euromax-for-MaxMSP/782013; http://juvul.com; http://cycling74.com/2010/11/19/introducing-vizzie/ (Accessed on 17/02/02014)

Even though this particular article was written nearly thirty years ago, I find it to be particularly relevant to the concerns of the software design of this project. Note that, whilst Decker et al are discussing synthesis in the article rather than data parsing and data generation, everything mentioned here can be applied to my modular concept of data parsing and data generation instead of synthesis.

3.3 One-to-many mapping of MIDI gestures

Gesturally, there are currently many methods with which a performer/composer can interact with electronic music, be it in the form of computer software or other hardware sound producing/editing units. Some of these include: knobs, faders, sliders, proximity sensors, X/Y joysticks and/or touch pads, bend-wheels, light sensors, bend sensors, pressure sensors, velocity sensing pads, aftertouch, buttons, switches, turntables, touch screens, gyro-meters, accelerometers, and so on. There are also interesting hardware devices (primarily designed for other purposes) that can be used in musical applications as well like the Nintendo Wii-mote, Microsoft Kinect, and many game controllers like an XBox 360 controller and so forth.[17] In most cases, a user has a *one-to-one* relationship with their gesture (using these physical devices) and an editable parameter (within their sound making setup). For example, they might turn a knob (on a MIDI controller) which controls the frequency cut-off of a filter inside a music software plugin[18], this hardware knob turn would match the turning of a user interface knob on the screen of the computer directly. In some software/hardware, one can assign the same knob to control more than one parameter at the same time. This is one-to-many mapping. The same input value of the knob will be sent to all parameters it is linked to. There are some exceptions, but this is largely what a user is limited to when mapping hardware control devices to editable parameters within their music making environment.

In recent years, some software has been developed that allows one-to-many mapping of a single input controller to many parameters within the software itself. Notable examples of this are Tornado by Sugar Bytes, The Finger by Native Instruments, and Stutter Edit by iZotope.[19] However, each of these examples are effect units for processing audio and the one-to-many

[17] http://www.nintendo.com; http://www.microsoft.com; http://www.xbox.com (Accessed on 17/02/2014)

[18] Typically, with standard 7-bit MIDI, the knob will have only 128 steps of resolution (0-127) which will be internally scaled by the hardware/software into a value that makes sense for the particular parameter that is being edited by the knob.

[19] http://www.sugar-bytes.de/; http://www.native-instruments.com/en/; https:www.izotope.com/ (accessed 17/02/2014)

mapping exists solely within the application itself, with no further output of this parsed MIDI information to control other devices.

There are some applications that do forward the input to many and varied MIDI outputs. Numerous of these are iOS applications (for the Apple iPad, iPhone, iTouch devices).[20] Lemur by Liine, Gestrument by Jesper Nordin, Beatsurfer by DRUW, and Konkreet Performer by Konkreet Labs[21] are the most notable iOS applications, with the latter being the most versatile one-to-many application currently available. Konkreet Performer allows single gestures on the touch screen to control up to 64 parameters at once via MIDI. The parameters of Konkreet Performer only change when interacted with on the screen as there are no automation features such as low frequency oscillators (LFO) or similar.

While these applications allow for one-to-many mapping of gesture to parameters, those parameters are still modified in a uni-directional, linear fashion as per the input gesture, with no option for making any individual parameter traverse the 0-127 range in a non-linear and/or multi-directional manner contrary to the original physical gesture. (As differs from my software One Knob To Rule Them All).

These examples of commercially available software for the processing of a MIDI input to many parameters of MIDI output are powerful in their possibilities, but they do not go far enough for me and my compositional ideas. I want to be able to control many parameters with one gesture but I do not want all of those parameters to be limited to a uni-directional, linear output. So I created the software One Knob To Rule Them All (ONE KNOB) which will be discussed in detail in chapter 3.8.

3.4 Inter Max App Communication System (IMACS)

IMACS is a highly re-useable and easy to implement system that enables a user of Max/Msp to create data sending links between their own separate Max/Msp patches in real time. IMACS exists due to my increasing desires to have my main Max/Msp patches able to trigger each other or control each others various parameters without hard coding those connections in. I wanted it to be

[20] http://www.apple.com (accessed 17/02/2014)

[21] https://liine.net/; http://gestrument.se/; http://beatsurfing.net; http://konkreetlabs.com/ (accessed 17/2/2014)

dynamic and non permanent. I wanted it to be free for myself (or the end user) to connect whatever they like, whenever they like and not be limited by a hard coded, built in application design. This dynamic form of using Max/Msp as an end user will change the way I patch in the future. That is to say, all new patches/software that I make in future will include the IMACS system. Any new addition to the system will therefore make any previously made patch even more powerful with new options for creative discovery and design. It is built around Max/Msp's scripting messaging system to the [thispatcher] object combined with the [hover] object. IMACS enables wireless connections to be scripted into existence between and within existing Max/Msp patches. It is not designed to work on its own, instead it is designed to work in conjunction with other Max/Msp apps.

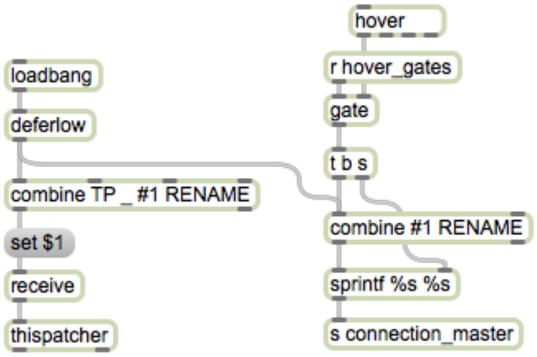
For any newly created patch to obtain complete access to the IMACS system, the user need only apply the following steps:

1. paste *the contents* of one small piece of Max/Msp code, that is, a small patch (located on Appendix D Software Portfolio CD (CD 1) labelled "D.1.4 pasteMeIntoPatchForImacsUsage.maxpat") into patch desired to be included in the system and then replace all text: "RENAME" with a unique name for that particular patch (See Figure 3.1).
2. either insert an [autopattr] object OR give varnames (scripting names) to all objects that are desired to be included in the system (within the patch that the previous step was made).
3. insert the IMACS patcher (included on Appendix D Software Portfolio CD (CD 1), labelled "D.1.5 IMACS.maxpat") into the main patch window via a [bpatcher] object OR simply run it as a separate patch on its own (see Figure 3.2).

IMACS began as a simple communication system between max patches but as I started using the system more, it became clear that it would be very beneficial to have some data parsing modules to place in between the connections I was making. Thus PENCIL was born...

3.5 Programming Environment of a New Creative Interface/Interactive Language/ Logic (PENCIL)

PENCIL is an application designed specifically for open ended possibilities. It is mainly an environment for parsing rhythmic and/or MIDI information in many and varied ways. It is a



```
loadbang
deferlow
combine TP _ #1 RENAME
set $1
receive
thispatcher

hover
r hover_gates
gate
t b s
combine #1 RENAME
sprintf %s %s
s connection_master
```

FIGURE 3.1: Contents of “PasteMeIntoPatchForImacsUsage.maxpat”

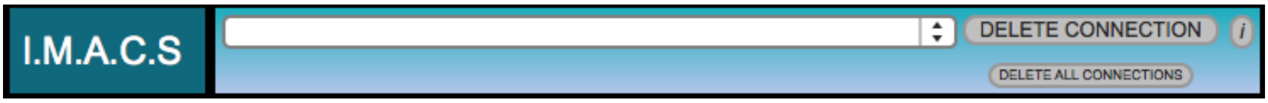


FIGURE 3.2: IMACS user interface.

modular programming environment built within a modular programming environment (Max/Msp). One of the most powerful features of it is that you can connect any piece of the interface to any other piece of the interface by means of a specially designed ‘send and receive’ system (IMACS). This connectivity is not limited in its scope and there are even parts of the interface that you can create connections to (or from) that are capable of completely destroying the functionality of the program. PENCIL was designed this way to give the user complete creative control of their MIDI setup and to provide an interface by which the creative possibilities for composing new music are limitless in scope.

The PENCIL environment differs from Max/Msp by allowing me to compose using my software more quickly and efficiently with a high amount of versatility and creativity. The main reason for this is that using the pre built modules save on patching time. Essentially, it is simply less work for me as a composer while working on ideas and it is more convenient and efficient than the process of normal patching within Max/Msp. It is higher level programming than Max/Msp. PENCIL is designed with easy to comprehend, ready to use modules that can be placed anywhere within the user interface environment and can be connected to any other part of the environment (or any other IMACS ready patch outside of PENCIL too, for example The Ultimate Rhythm Discovery Station discussed in section 3.10).

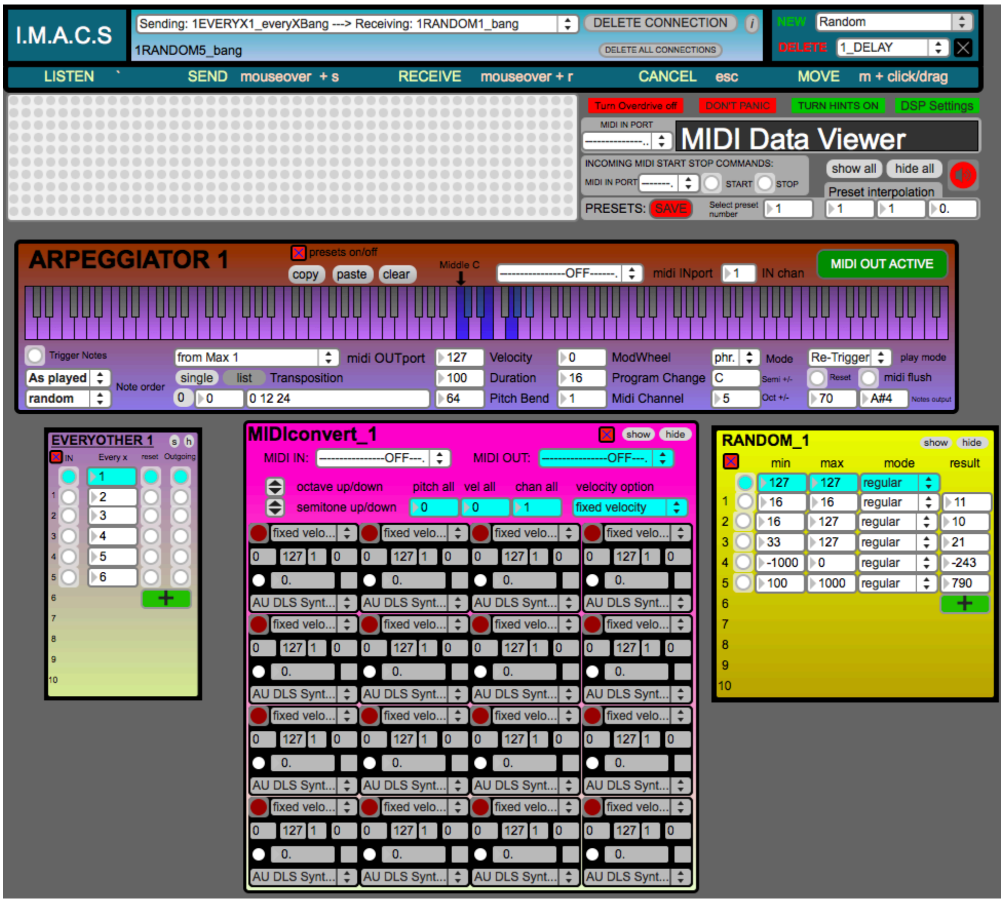


FIGURE 3.3: Example of PENCIL interface with four modules loaded.

There are currently 40 modules (listed in Appendix A.1 and A.2) that can be loaded into the PENCIL environment window (see Figure 3.3 for a simple example of PENCIL in action), some of them are basically just a user interface for existing Max/Msp objects (like the [scale] object for example) and other modules are more sophisticated devices (like the arpeggiator module). All of these modules can be moved around the interface and positioned wherever the user wants them by simply clicking and dragging them with the mouse (after pressing a keyboard shortcut 'm'). The modules are used to create an event based system that can control and manipulate existing max patches, themselves and/or hardware/software.

This modular based system that I have created differs to the aforementioned modular [bpatcher] based systems in chapter 3.2, in that it requires no Max/Msp 'patching' at all. A composer using PENCIL will never need to open the Max patch in 'patching mode' and go behind the scenes, connecting objects, sub-patchers, abstractions and [bpatcher]s with patch cords. All 'programming' is done from 'presentation view'. And yet the software is still modular and dynamic in design. It starts with a blank canvas from which the user can select modules to create, and can then move them around the user interface just like the parent application Max/Msp. Each of the modules I have created are accessed via a menu and are scripted into existence when the user selects them. Ultimately, they are simply pre-saved Max/Msp patches, dynamically loaded into a freshly created [bpatcher] object that is given: arguments for its window size/location; an incrementally indexed reference number; and other variables pertinent to the module created. The modularity does not end there however as many of the modules have slots for creating more of the same type of module again within its very own window. Thereby creating a modular programming environment (dynamically scriptable and modular [bpatcher]s), within a modular programming environment (PENCIL), within a modular programming environment (Max/Msp).

Of course, the results of the programming effort on my part to create such a system is up to the composer/user at the time of use to decide, just like the output of a modular synthesiser is up to the hands and mind of the synthesist commandeering it, except that the modules I have created to use within PENCIL are tailored mainly for the exploration and parsing of MIDI and/or rhythmic information. Meaning, the resulting output engineered by the user will mostly gravitate naturally toward that of rhythmic creativity or discovery within any MIDI environment of their preference. Although, PENCIL could quite equally be used in other ways as well.

I am constantly thinking of new ways of using PENCIL. I am often struck with thoughts of "what if I were to link *x* to *y* and have *x* controlled by *z* which is going through *n* and *l* which in turn is controlling the offset of *y* which is also controlling the range of *n* and the speed of *l*…." and so on. The fact that this is at all possible and that it will generate rhythmic (or sonic) results unheard of before that *I can actually imagine* in my mind prior to patching it all together is an absolute dream for a composer. It is an absolute joy to use this system. Even if the usage is more experimental and less planned from the outset. My mind races with the possibilities and is excited with the results on a daily basis. And yet, after using the system for a few months, I know very well that I have not even scratched the surface of the possibilities with the software. Not to mention that not only is it open ended in how it can be patched, but it is also open ended in that new modules can be created at any time in the future that will expand its potential exponentially. This system will no doubt keep me as a composer, who wishes to use new and deliberately rhythmically interesting software, busy, entertained, and fulfilled for years to come.

3.5.1 The available modules of PENCIL

The available modules I have created for use within the PENCIL environment are listed in Appendix A.1 and A.2, each with short descriptions. Note that none of these modules are for processing audio. Instead, they are all built to generate timed events and/or parse data.[22]

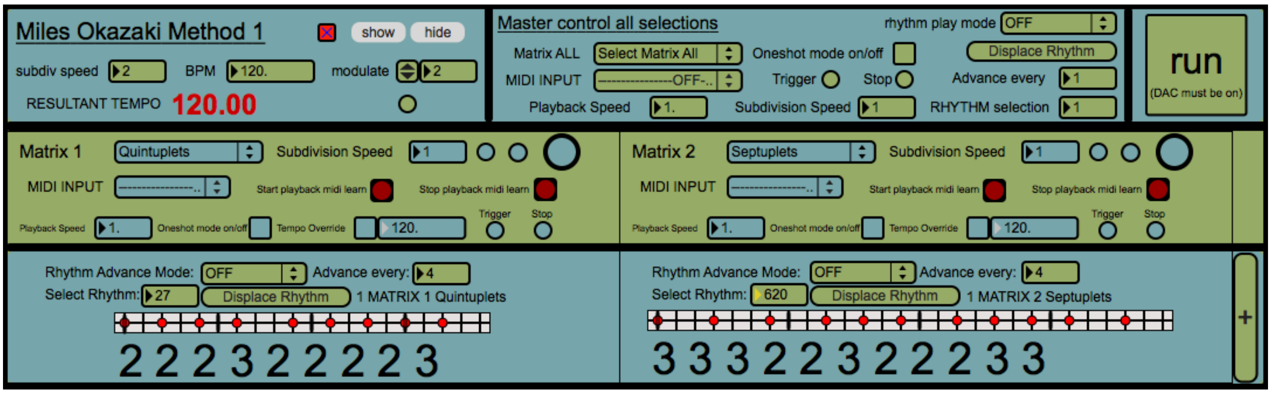
3.6 Miles Okazaki Method (MOM)

The PENCIL module 'MOM' is entirely built around Miles Okazaki's *Rhythm Matrix* book. MOM allows the performance of: any of the rhythmic modes from *Rhythm Matrix*, at any subdivision speed and at any tempo (based on a percentage ratio of a master tempo) against: any other rhythmic mode with the same variables. Within each module are many options like: rhythm selection (from within pool of currently selected subdivision), displacement of the rhythm by one subdivision partial step, randomise or increment rhythm (from within pool of currently selected subdivision) every x amount of repetitions.

It is also possible to override the tempo settings for any individual module so that it can run completely independent of the parent module tempo yet still be controlled by other of the parent module's global settings. Rhythms can be triggered and stopped and can be looped or not. Each module contains a user interface object ([matrixctrl]) that graphically shows the currently selected rhythm and also allows the customisation of new rhythms at the user's wishes (See Figure 3.4). This essentially allows MOM to work as a traditional rhythmic step sequencer but with all of the above extra options for adding rhythmic complexity. Up to thirteen of these modules can be dynamically loaded into any individual parent MOM module, thereby allowing up to thirteen rhythmic modes to be coincidently performed (per parent MOM module). There is no limit within PENCIL as to how many parent MOM modules you load (excepting the computational load that running many modules at once would incur).

MOM is fully IMACS ready, meaning any component of it can be controlled by, or can be used to control, any part itself and/or any other pieces of software in this portfolio. This means it

[22] Note: contained within a few of the available modules are third party Max/Msp objects from the Modal Object Library created by V.J. Manzo vjmanzo.com/cv. Copyright.



Miles Okazaki Method 1 ☒ show hide
subdiv speed 2 BPM 120. modulate 2
RESULTANT TEMPO 120.00 ○

Master control all selections
rhythm play mode OFF
Matrix ALL Select Matrix All Oneshot mode on/off □ Displace Rhythm
MIDI INPUT ---------------OFF-.. Trigger ○ Stop ○ Advance every 1
Playback Speed 1. Subdivision Speed 1 RHYTHM selection 1

run
(DAC must be on)

Matrix 1 Quintuplets Subdivision Speed 1 ○ ○ ○
MIDI INPUT ---------------.. Start playback midi learn ● Stop playback midi learn ●
Playback Speed 1. Oneshot mode on/off □ Tempo Override □ 120. Trigger ○ Stop ○

Matrix 2 Septuplets Subdivision Speed 1 ○ ○ ○
MIDI INPUT ---------------.. Start playback midi learn ● Stop playback midi learn ●
Playback Speed 1. Oneshot mode on/off □ Tempo Override □ 120. Trigger ○ Stop ○

Rhythm Advance Mode: OFF Advance every: 4
Select Rhythm: 27 Displace Rhythm 1 MATRIX 1 Quintuplets
2 2 2 3 2 2 2 2 3

Rhythm Advance Mode: OFF Advance every: 4
Select Rhythm: 620 Displace Rhythm 1 MATRIX 2 Septuplets
3 3 3 2 2 3 2 2 2 3 3

FIGURE 3.4: Miles Okazaki Method user interface

could, for example, be used to control the structure of a composition, rhythmic material, rhythms within rhythms (nested tuplet rhythms), tempo changes, pitch control, and any other musical parameters the user can imagine. This effectively provides many rewarding and deeply creative possibilities with the exploration and discovery of new grooves and rhythmic patterns being the primary intention of the author..

3.7 The Spiral Rhythm Clock (SPIRAL)

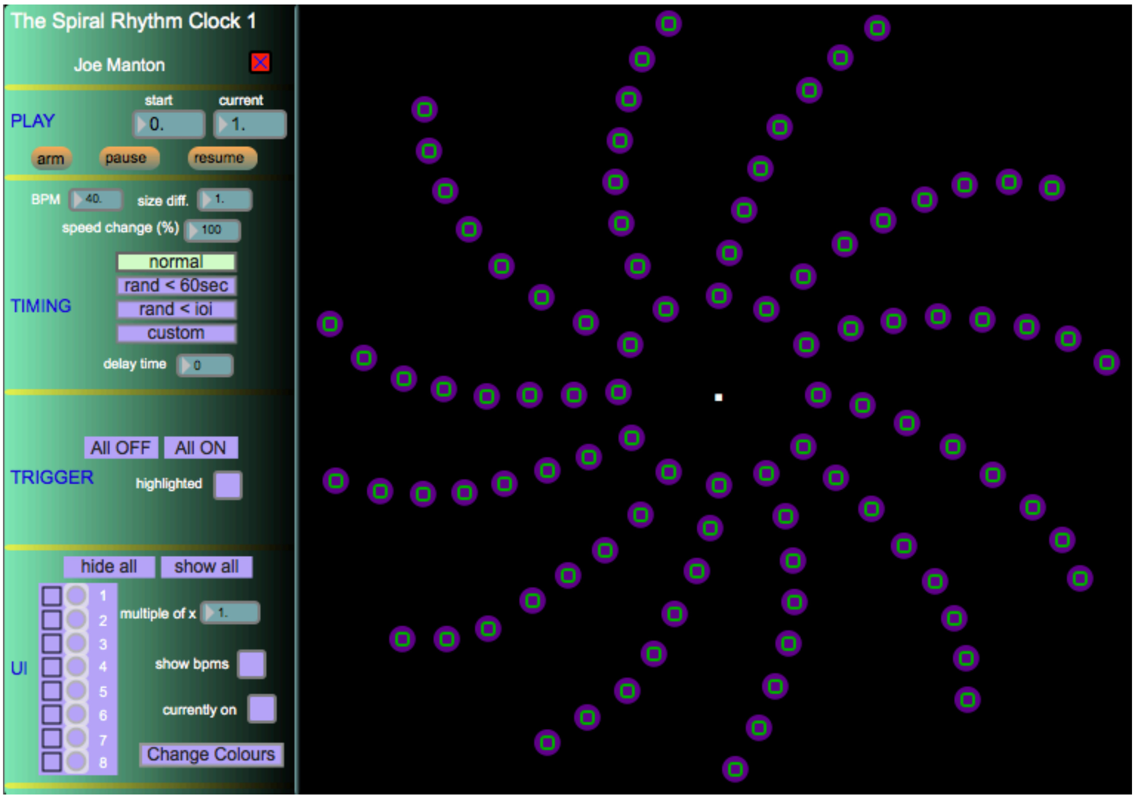
The non-determinate parameters of Ligeti's *Poème Symphonique* are precisely those that I wanted to have control over for determinate, repeatable and creative composing in a poly-tempi system. Developing SPIRAL allowed me to investigate the compositional and educative arena of poly-tempi music. It consists of up to ninety-six concurrent and unique tempi whose output can trigger anything the user likes within the IMACS system (including but not limited to MIDI). This enables the user to create repeatable, sophisticated rhythmic structures previously only imaginable or highly time consuming to create manually via traditional computer notation methods. SPIRAL, in conjunction with IMACS and the other PENCIL modules, enables the user to create a timing system of impressive and sophisticated complexity.

The rhythms created by SPIRAL will repeat identically on the minute, every minute, if the tempi used are all whole numbers. They have subtlety and complexity that require repeated listening to fully comprehend. Therefore the one minute structure of repetition is in fact, most helpful for the appreciation and understanding of the rhythms by the listener. SPIRAL can also utilise tempos that are not whole numbers and where this is the case, the cycles of repetition can quickly stretch from one minute out to practically infinite length in comparison. Is a cycle worth exploring that will not repeat itself for 4000 years? Absolutely yes. It is just as interesting as any other. As with any other element of music or previously constructed system, it comes down to user preference and the choices of an individual creator/composer/practitioner.

The metronomes in SPIRAL are laid out in 8 circles with twelve metronomes per circle set out as though on the face of a clock. Each arm is slightly curved in a clockwise direction for aesthetic purposes (See Figure 3.5). A visual phenomenon created by this interface configuration is that the flashing of the metronomes while it is running will spin in a multitude of different spiraling patterns back and forth over the course of the phasing. There are many 'rhythmic nodes' during a complete phasing cycle (that is, one minute, where whole numbers are used for the tempos) where metronomes who have a common denominator will flash together at prescribed times. This phenomenon also allows for a unification of rhythms at many points during the phasing. For example, all metronomes divisible by three will sound/flash together again every twenty seconds (providing they are all started at the same time). All metronomes divisible by four will sound together again every fifteen seconds and so on.

Each group of metronomes with a common denominator also has its own collection of 'nodes' where tempos within that particular group sound together at various times in the phasing cycle. These rhythmic nodes enable rhythmic cycles within rhythmic cycles (meta-cycles) to occur across the greater phasing cycle. They each also contain the near sync or sync activity of multiple subdivisions from each of the tempo's perspectives throughout the node cycle. In other words, there is a large amount of rhythmic 'harmonicity' in effect when using a system of such complexity as ninety-six separate tempos running coincidently.

There are a number of options to allow for a diminishment of the harmonicity in SPIRAL. In each of the options, the metronomes once started will continue to click at their normal tempo. The first option is the normal and default behavior of the application, all metronomes begin at once. The second option causes each metronome to be started at a random point within the period of one minute. The third option is a randomised time for a delayed start of each metronome within the period of its own IOI. Finally there is an option to allow the start time of each metronome to be delayed by a defined amount from the previous metronome (in ascending order of tempos) and this delay time can be defined in milliseconds by the user.



start 0.
current 1.
BPM 40.
size diff. 1.
speed change (%) 100
delay time 0
multiple of x 1.

FIGURE 3.5: Spiral Rhythm Clock user interface.

There are many other important options for controlling the behavior of the SPIRAL application. The user has the ability to change the speed of the entire system while maintaining all of the original ratios between the tempi. This enables very highly complex systems of large numbers of the metronomes (up to ninety-six) to be played back at much slower speeds (as slow as 1% of the original tempi) illuminating rhythms and grooves that would have been too fast and overwhelming at full speed to be appreciated. The user can change the size difference between each metronome speed (from the default of one) to any other number including zero and negative numbers. Setting the size difference between each metronome's tempo to zero lets the user effectively 'pause' the system in its current state with all of the metronome's tempos changing to the speed of the slowest tempo while remaining in the same ratio or position from each other as they were at the time of 'pausing' and thereby creating the arbitrary rhythm at that moment to repeat identically at that slowest metronome speed.

SPIRAL could also be used to help make highly complex rhythms actually easier to perceive. For example, if each tempo is given two different pitched notes that it triggers one after the other repetitively, the user can better grasp that particular tempo in the mix of all the other tempos as they can lock onto a very simple pattern of 'up down up down up down' or 'tick tock tick tock' whilst the other complexity surrounds and interacts with the simple pattern perceptually. Further enhancement can be made by automating the volumes of each pair of notes via other PENCIL modules (for example ONE KNOB).

Finally, in addition to many other user controllable options in the application, all parameters of SPIRAL can be automated or controlled remotely using IMACS (and also therefore MIDI hardware/software and all modules from the PENCIL environment). This automation feature makes SPIRAL a generative rhythmic composition tool of unprecedented complexity and control, and a highly unique environment, allowing the composer to explore infinite rhythmic possibilities.

3.8 One Knob To Rule Them All (ONE KNOB)

ONE KNOB is a gestural control application that gives the user many ways to control musical parameters via a dynamic interface that grows according to user needs. The user provides an input and the input is parsed through ONE KNOB, creating a new output. The amount of ways that the input can be parsed and changed to a new output in ONE KNOB are practically infinite. As a composer who works with music technology in both hardware and software form, I am constantly trying to create new sounds and ways of interacting with and controlling these tools. In my experience and in my searching for controller systems, I found that there were gaps to fill within the softwares available. I could not find the tools that I imagined would exist. Hence, I built ONE KNOB to fill one such gap.

Although not limited to this functionality, a typical input of ONE KNOB would be an integer between 0 and 127 (like standard MIDI data of a continuous controller or a note on message for example). This input is then fed into a [function] object which converts the input into a new number based on the graphical position of the line inside the [function] object. Figure 3.6 shows an example of the user interface of ONE KNOB. Observe that the "[function] 1" (the lower left box) that has two points (the start and end points of the line) generating a straight line, starts at position 0 with a value of 127 and ends at position 127 with a value of 0 (where the first number is an index point for the incoming integer and the second number is the output value for the specified index point). This [function] will generate an output that is completely inverse to the input. That is, while the input travels from 0, 1, 2, 3, 4….124, 125, 126, 127 the output will conversely travel from 127, 126, 125, 124….4, 3, 2, 1, 0.

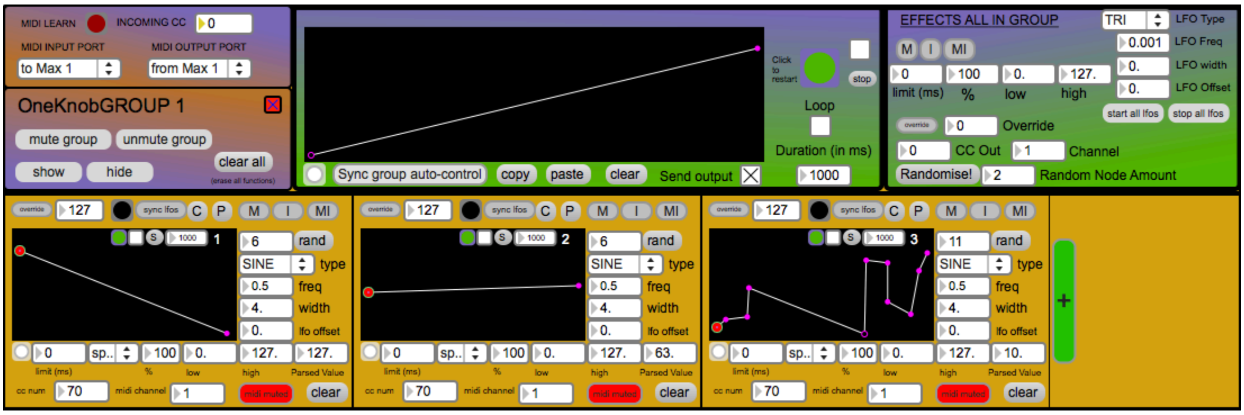


FIGURE 3.6: ONE KNOB user interface.

'[function] 2' in Figure 3.6 again uses a simple straight line with only a start and end point but has a smaller range: The line inside '[function] 2' starts at position 0 64 and ends at position 127 74. In this example, the input value of 0 will return an output value of 64 and as the input value increases to 127, the output will be scaled and increase from 64 up to 74 only.

The input does not have to be linear, it can jump around randomly, or move backwards and forwards in any way the user wants. The input value refers to an index point on the [function] line and returns whatever value the [function] line is at that index point. Similarly, the [function] line does not have to be limited to a straight line with only a start and end point, it can have any number of points and these points can be placed anywhere within the bounds of the [function] object enabling lines that go up and down with any degree of slope, as demonstrated by '[function] 3' in Figure 3.6).

The user interface of ONE KNOB allows the creation of multiple [function]s dynamically where the input value is parsed through each [function] separately at the same time. This enables the user to use one knob (of a MIDI controller for example) to control various parameters at once, with the line inside each [function] being capable of different shapes (an advanced form of one-to-many mapping). An example usage of this functionality might be: the user wants to control multiple parameters of their synthesiser with one knob turn. So the output of '[function] 1' could control filter cutoff frequency; the output of '[function] 2' could control an LFO (Low Frequency Oscillator) speed; the output of '[function] 3' could control the pitch bend, with all of these [function]s parsing the input differently as per Figure 3.6.

ONE KNOB includes many features to enable further parsing of the output generated by the line inside the [function] object. Some of these features include: the scaling of the output to any floating point low and high fixed output; the limiting of the speed of the output to any amount by measurement of milliseconds; the probability of the output occurring or not by means of a gate that can be controlled via percentage of probability; the ability to impose an LFO on the position of the input with various user definable parameters of control; the ability to synchronise all LFOs; and more.

The generated line can be created using a computer mouse to click and draw points and change the shape of the line and the line can also be randomised. The input value can come from a midi controller, or it can be automated. This automation can be sent from another digital audio workstation sending its own automation data via MIDI. The automation can also be done from within the ONE KNOB module window itself as there is yet another [function] object in place purely for this purpose. The lines within the [function]s can be mirrored, inverted (or both at the same time) by user interface buttons and they can also be copied and pasted to any other [function]. Any of the above mentioned features can be automated themselves as all parameters of ONE KNOB are included in the IMACS system. Each [function] module within each ONE KNOB group is capable of ignoring the incoming input for the group it resides in and being overridden by a different source. This overriding input includes yet further automation options. Alternatively, ONE KNOB can be used as a type of gestural sequencer as well. The output can be sent to the Noteout module in PENCIL for example, and the speedlim and gate functionality within ONE KNOB can be automated for rhythmic interest (even by other ONE KNOB modules).

As a composer, the possibilities with the ONE KNOB are so infinitely varied that it becomes highly daunting to try and list even a few of the things that can be done with the software. I will however describe how I have used it in my own compositional output: controlling of synth/effect parameters via MIDI continuous controller messages, control voltage (CV) inputs to analog synthesisers and effect units, editing of System Exclusive strings and Non Registered Parameter Numbers; mapping output of [function]s to the creation of midi notes, thereby triggering both software and hardware samplers, synthesisers, drum machines and more.

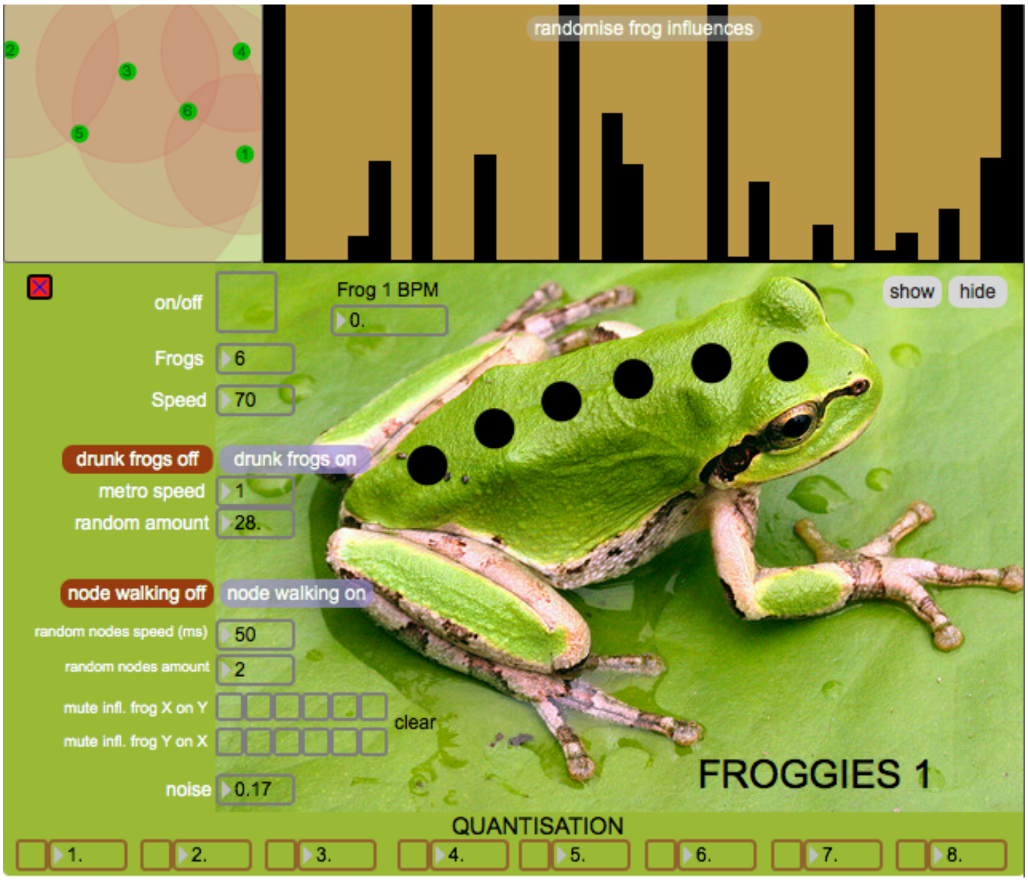
As ONE KNOB can be mapped in any way the user can think of with the available modules of PENCIL, there are many ways in which it can be used to generate, control, influence, design, and refine rhythmic content to create rhythmic complexity and/or create new sounds.

3.9 FROGGIES

FROGGIES is a rhythmic generator that simulates frog call behavior with six frogs interacting with each other in real time via a non-linear dynamic equation matrix. The rhythmic beauty of the output of FROGGIES resides in its inherent striving to achieve sync between the calling frogs at all times whilst relying on a choatic system that is fully dependent on the initial state of the system. Within the algorithm, each ‘frog’ within FROGGIES ‘listens’ to the calling of the other frogs and adjusts the timing of its call to try to achieve a hocket-like sync between the call time of the other frogs. Often this process results in the frogs calling in a distinct and strict hocket sync, but once deviating from this, the frogs’ call times can briefly resemble other distinct rhythmic tuplets like triplets, semiquavers, quintuplets and so on, as well as all variance of in between rhythmic placement of calls.

FROGGIES is a module available within the PENCIL environment. I used a non-linear dynamic equation provided by Aihira to generate the interactive rhythmic calling of each frog in FROGGIES (Aihira et al, 2008). I enlisted the help of Dr Oliver Bown from the Architecture Department of Sydney University (Design Lab) to port Aihira’s equation via java programming and the [mxj] object into Max/Msp and with the kind permission of Dr Bown, was able to use this code within FROGGIES (see Appendix B.1 for the frog code).

The interface for FROGGIES includes two graphical ways to change the influence that each frog has on each other (See Figure 3.7). The first of which is a circular, proximity and ‘sphere of influence’ based approach whereby the location of each (numbered) frog and the strength/loudness (sphere of influence) of its calling can be edited via the use of a [nodes] object. This GUI object controls the frog’s relationship to all the others all at once. That is, the user is able to edit each individual frogs’ submission/dominance over the other frogs influencing their call times accordingly. The second graphical method of changing the influence of the frogs on each other is via a bar graph [multislider] object which provides the same functionality as the [nodes] object



on/off
Frog 1 BPM 0.
Frogs 6
Speed 70
metro speed 1
random amount 28.
random nodes speed (ms) 50
random nodes amount 2
mute infl. frog X on Y □□□□□□
mute infl. frog Y on X □□□□□□
noise 0.17
□ 1. □ 2. □ 3. □ 4. □ 5. □ 6. □ 7. □ 8.

FIGURE 3.7: FROGGIES user interface.[23]

previously mentioned, except modification is only possible on an individual frog to frog influence basis. You could also use this [multislider] to draw new relationships quickly with a computer mouse. Further, there is a randomise button which will automatically provide new influence amounts for all frogs against each other.

FROGGIES has the ability to slow down/speed up the calling speed and also has controls for the random automation of inter-frog influence amounts. It can even be run so slow as to be useful for the generation of, for example, the structure of a composition, or at medium speeds for the generation of the structure of individual musical phrases. Further, it is completely IMACS ready and so can be automated/controlled by and can automate/control any other element within the IMACS system. This inter-connectivity provides the means to make audible the rhythms generated by FROGGIES in any way the user desires.

[23] Photo of frog used in FROGGIES user interface is of the Japanese Tree Frog *Hyla japonica* and was retrieved on August 13, 2013 from http://farm1.staticflickr.com/81/232646672_713d8673c8_z.jpg

As with all other modules within the PENCIL environment, all settings within the FROGGIES module can be saved as presets and these presets can be changed or interpolated between manually or via automation. This allows for the generation of self similar states/grooves, but with all of the variation one would expect from a chaotic system that is inherently reliant on initial system states. For example, one could save a few different frog-to-frog influence states and then periodically (or rhythmically) change between these saved states. Then, each time the preset is changed back to one already heard, the chaotic system may repeat itself almost identically to the last time the preset was heard, or provide any amount of variation from mild to extreme.

Included in FROGGIES is the ability to mute any individual frog being influenced by the timing of the other frogs calling. As well as the ability to turn off the ability of any individual frog being able to influence the timing of any other frog calling. This enables, among other things, the ability to create a 'fixed' tempo frog that will continue to call at a steady rate while the other frogs interact and change around it. Also included is a 'noise' element. This provides the user with a method of applying a continuous, variable amount of randomness to the influence each frog has on each other for an even more natural-like 'frog-scape'.

Quantisation of frog call timing is made possible by means of eight, user definable values of quantisation that are completely linked to the call time of frog number one. Each of these definable values can be switched on or off individually at any time and any number of them can be running at once. The method used to quantise is as follows: frog one's tempo in beats per minute is sent to a master metronome that is stopped and restarted every single time (and at the precise moment) that frog one actually calls. The eight outputs of the master metronome are sent to eight [onebang][24] objects. The output of all frog calls are sent to the right inlets of these [onebang] objects and therefore will only 'call' after the metronome/s that is/are switched on 'ticks'. As stated, the master metronome controlling the quantisation runs at the speed of frog one however the eight outputs of this metronome can be set to any floating point number above the value of 1.0 which provide a division of the BPM set by frog one. This means the user can quantise the call timing of all the frogs to any degree of complexity or simplicity that they like.

[24] The [onebang] object only allows an event received in its left inlet through after receiving an event in its right inlet, after which time its internal gate closes again until receiving another event in the right inlet again.

Composing new music with a chaotic, dynamic, rhythmic generator such as FROGGIES within the IMACS system places the composer in a unique position whereby they can automate new musical systems that emulate nature, or by which they can take the given modeling of nature and bend it into new, extreme settings, far removed from the original 'frog calling behavior' source.

3.10 The Ultimate Rhythm Discovery Station (TURDS)

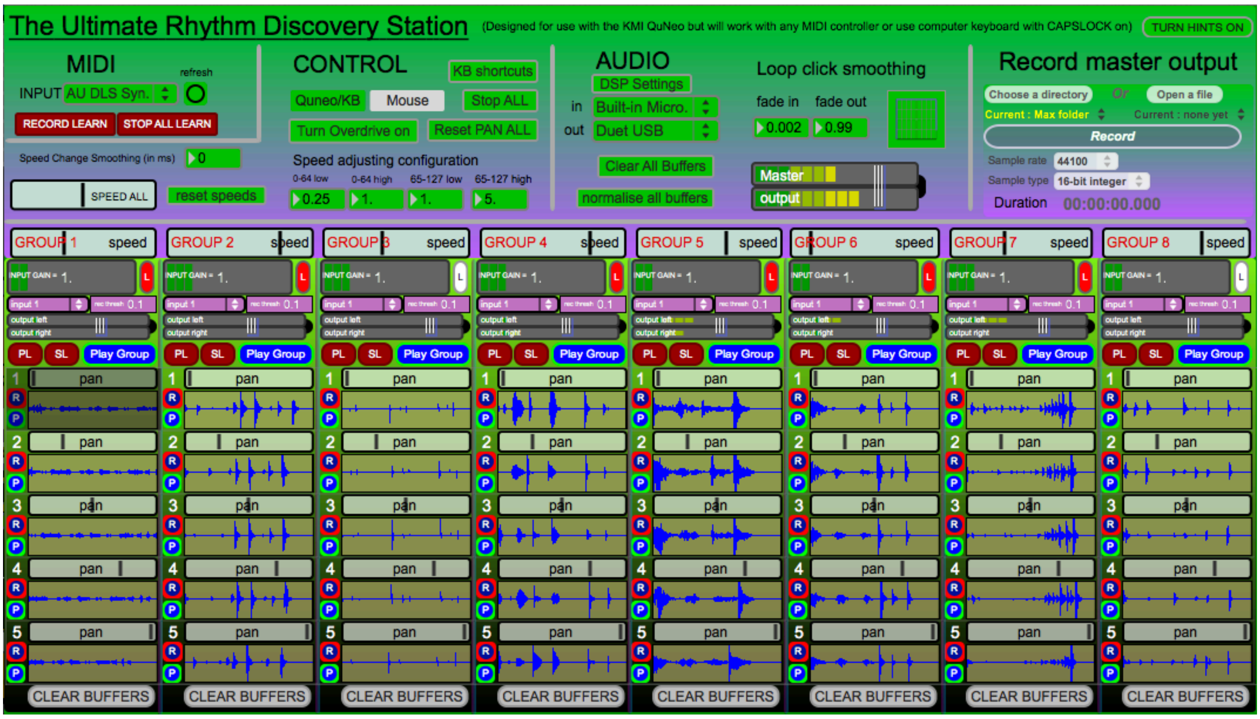
The Ultimate Rhythm Discovery Station (TURDS) (see Figure 3.8) is an application designed for the sole purpose of creating and discovering new rhythms and grooves via performance means.[25] Some of the other applications in this portfolio (specifically MOM, ONE KNOB, and the Motion Recorder module from PENCIL) can be used in the same way as I intended TURDS to be used but they are limited to data/MIDI processing only, not audio. Though by no means limited to this usage, the intended use for this application is for the user to record themselves performing eight unique rhythms, five times each (overdubbed), and then create entirely new rhythms out of these recordings by altering their speeds and triggering them in real time via a MIDI controller.[26]

When five separate performances of the one rhythm are recorded individually and played back at once, panned evenly across the stereo sound stage, the result can sound somewhat like a group of performers performing the rhythm, albeit a bit out of time with each other. This loose feel in the timing can create or add realism to the perceptual interpretation of the playback as being actually performed by a group of performers rather than one performer recording themselves five times. This results in an environment where the performer feels as though they are in control of a group of performers and can make those performers play ever more complex rhythms and grooves. The looseness of the timing helps to add a sense of realism to this phenomenon. This is the general sonic aesthetic of TURDS.

TURDS enables the layering of up to five samples or recordings of duration up to one minute into eight separate groups. Each group's speed can be controlled as a whole from any

[25] Although it is also possible to use it in other ways for other purposes.

[26] I designed the application for use with the QuNeo MIDI controller (by Keith McMillan Instruments) however any other MIDI controller could be used instead.



INPUT AU DLS Syn.

Speed Change Smoothing (in ms) 0

Speed adjusting configuration
0-64 low 0.25
0-64 high 1.
65-127 low 1.
65-127 high 5.

in Built-in Micro.
out Duet USB

fade in 0.002
fade out 0.99

Current : Max folder
Current : none yet
Sample rate 44100
Sample type 16-bit integer
Duration 00:00:00.000

INPUT GAIN = 1.
input 1
rec thresh 0.1
output left
output right

FIGURE 3.8: The Ultimate Rhythm Discover Station user interface

negative to any positive multiplication of the original speed.[27] This means recordings can also be played back in reverse (even while recording them!). After a recording has been made, the beginning of the [buffer] that has been recorded into will be automatically cropped to the first sound that is larger than a user definable volume threshold. This is so that synchronised playback of each recording slot is automatically created for the convenience of the user. This is particularly useful when using TURDS manually (without the automation of IMACS) and when using TURDS for rhythmic purposes (the original and intended use of the application).

To use TURDS, the performer would record themselves performing the same rhythm into the five available slots of group one and then proceed to fill the other groups in a similar fashion to group one, each group with their own rhythm. The rhythms for each group need not be related in any way to each other. It is then up to the performer to use a MIDI controller (or the computer keyboard) to trigger the playback of the recorded groups to create new and interesting rhythmic grooves. For example, while triggering different groups and setting up a rhythmic tempo/groove, the performer might change the speed of a single group while maintaining the existing rhythmic groove causing a new rhythmic challenge for the performer. This forces the performer to hear the

[27] The default speed control is scaled so that an input control (for example, MIDI) of 0-64 equals 20% to 100% of the original speed and 65-127 equals 100% to 500% of the original speed. These settings can be configured to other amounts if required in the user interface including negative speeds of any multiplication.

new timing of the rhythm (due to its speed being changed) in relation to the existing groove they have set up. Hence the speed control option exists. This is one of many possible ways to use the application.

Grooves that are performed can be resampled (recorded) into new groups, which can then be factored into the next performance which can then be resampled into new groups and so on. There is definitely a lot of interesting music to be discovered in the recursive nature of the software. Particularly when playing with the speed controls offered. TURDS can also be used as more of a compositional tool by recording longer, musical ideas, playing groups back at different speeds and resampling them into new slots. For example, if you record the sound of a music box into the different buffer recording slots offered by TURDS (using a microphone), then vary the speeds of the groups and resample into new groups, the results can be very harmonically (and rhythmically) interesting from very simple source material. That is, you can transform a single, simple recording into an elaborate and intricately detailed soundscape within a very short amount of time by means of resampling the performance of playback triggering and speed control of playback whilst resampling the output and triggering the play back of the newly resampled material ad infinitum. There are 40 recording slots (or sample buffers) offered by TURDS to allow for this.

TURDS is also completely IMACS ready and so every part of it can be controlled by my other Max/Msp applications. Connectivity options include playback and recording of groups and individual cells, and also speed changes for group playback. This connectivity therefore allows for automated recording and/or playback, opening up many more creative avenues of exploration for the use of the application. The automation of TURDS can yield fascinating results; however, my own interests with this application are in its real time performance capabilities. In other words, TURDS allows the ability to physically create new complex rhythms/grooves from arbitrary, potentially very simple, and potentially non-related source material. It allows this in real time and in an improvisatory manner. As a result, I have found that my own sense of time and groove has been challenged and developed/improved by using TURDS. This makes me more capable as a composer to now develop more sophistication in my rhythmic explorations and enables me as a listener (and performer) to have a greater awareness of groove (and sense of time) in potentially or seemingly unrelated events (like those discoverable in nature).

CHAPTER 4

Composition Portfolio

The following subsections contain the analysis of the studies that form the basis of my composition portfolio. These studies were constructed primarily within the context of the rhythmic devices (software) discussed in this thesis.

4.1 Study No. 1

This study utilised SPIRAL and PENCIL (in particularly the arpeggiator module). An arbitrary subset of metronomes from within SPIRAL were selected and used to trigger four separate arpeggiator modules. These arpeggiator modules were each configured with settings that were similar but slightly different from each other. The end result is a harmonically rich progression of consonant pitches moving through a short series of modulations (causing numerous dissonances that are resolved shortly after) via the arpeggiator playback options.

This study features the mirrored qualities of SPIRAL's rhythmic behavior as the rhythms that have occurred up to the halfway point are then completely retrograded in playback to the end of the study. Many instances of 'rhythmic *glissandi*' (Wannamaker, 2012) are present throughout.

4.2 Study No. 2

This study consists of two smaller parts, both of which were created with the following modules within PENCIL: ONE KNOB, NoteOut, Speed Limiter, Maths, Arpeggiator, Scaled Value, and Random. I also made use of the preset system within PENCIL.

Firstly, to create this piece I relied on certain characteristics of the synthesiser I wanted to use, namely, the Yamaha TX816. This synthesiser has eight identical Yamaha DX7 synths within it. Each of these units are called a TF1 and all of the TF1 modules are accessible at once or individually via MIDI. I wanted to exploit the common texture of loading the same patch (sound settings) into each TF1 module and then triggering them slightly differently with PENCIL and also with each TF1 module positioned individually across the stereo image from left to right, with slightly different tuning to each other. I chose a bell-like sound for this composition as I am particularly fond of the richness and clarity that the Yamaha DX7 achieves with its Phase Modulation synthesis (commonly referred to as Frequency Modulation or FM synthesis) to create bell-like timbres.

I used ONE KNOB to generate pitch gestures and also to control speed of playback of such gestures via random, maths and speed limiting modules. I used the preset system to store particular gestures that ONE KNOB had created (after being parsed through the other modules listed above, whose settings were stored within each preset also). The pitch gestures were parsed through Vince Manzo's Modal Object Library[28] and the different modes were triggered in realtime by myself during the recording. I used the arpeggiator module to control the preset system. Each step of the arpeggiator was triggered by the loop start point of the master ONE KNOB group automation control. During the recording of this study, I interacted with the (mostly) automated patch by causing disruptions to the rhythmic output of each module so that the timing of each module oscillated in various ways (via ONE KNOB's LFO system) around a common speed (as set by the varying gestures of ONE KNOB). The result is a stunning interaction of rhythms, sonic timbres, stereo imaging and harmonic movement with a great deal of unity built into the composition via the careful use of the software.

4.3 Study No. 3

This study demonstrates and illuminates rather well some of the inherent patterns and underlying rhythmic qualities of running ninety-six metronomes at once. The audio track consists of ninety-six different BPMs (40-135 set incrementally) slowed down to 5% of the speed (resultant BPMs: 2-6.75), all started at the same time, and triggering different pitches (descending 5ths) on the Yamaha TX816 synthesiser in groups of 12 BPMs (8 groups). It plays through the first five seconds of the retarded minute and then is stopped.

One of the interesting rhythmic phenomena of these settings of SPIRAL is that clearly audible simple rhythmic relationships are heard from the beginning. That of the polyrhythms 2:3 (decelerating), then 3:4 (decelerating), then 4:5 (decelerating) and so forth, with overlapping of these later in the track to eventually produce 3:4:5:6:7:8:9:10 (see Figure 4.1).

[28] Modal Object Library, created by V.J. Manzo vjmanzo.com/cv. Copyright.

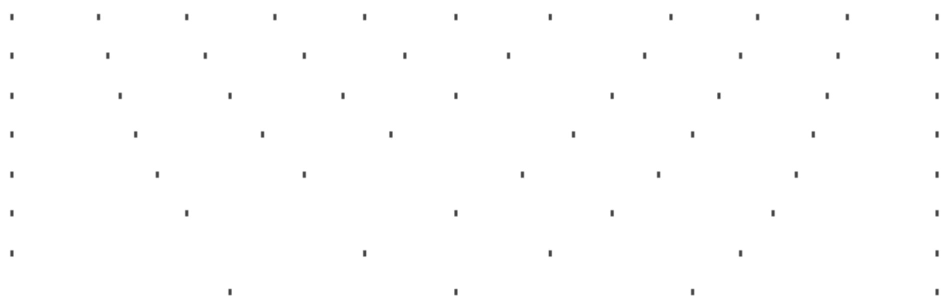


FIGURE 4.1: Excerpt of MIDI data output by Spiral in the recording of Study No. 3. Visually demonstrating Wannamaker's "flanking curves".

4.4 Study No. 4

This study uses exactly the same rhythmic material as Study No. 3. That is, the Spiral Rhythm Clock is run using the same rhythmic settings as before but with different pitches being triggered by each group of the eight groups of twelve metronomes (See Figure 4.2). The main difference with this extra study is that in addition to the TX816 synthesiser being triggered, I have also sent the MIDI note data from SPIRAL to a drum machine (Roland TR505). The reason for this extra study is to demonstrate that new levels of rhythm can be perceived by the inclusion of the drum machine. The drums provide a new layer of temporal information even though they are being triggered at the same time as the TX816 synth. This is due to the kick, snare and open/closed hihat drum sounds indulging our preconceived notions of their use within commonly heard drum beats in popular music. I assigned the pitches of the lydian dominant mode to the rhythms. The choice of pitch, however, is arbitrary.

4.5 Study No. 5

As one of the aims of this thesis is obtaining a deeper understanding of musical time, exploring multiple tempo layers is a focus of this study. The piece begins with a single metronomic pulse and is structured so that the individual (and separate) tempos are introduced gradually. The work is entirely automated and created using SPIRAL. Midi was sent to the TX816 synthesiser as well as the Yamaha FS1R (an eight operator FM synthesiser). Only eight metronomes of the ninety-six possible metronomes within SPIRAL were used, and they were set to the BPMs: 60, 73, 86, 99,

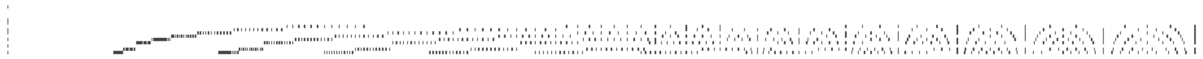


FIGURE 4.2: MIDI data recorded from SPIRAL for Study no. 4 (complete).

112, 125, 138 and 139 (chosen arbitrarily). Each metronome's entry was delayed by varying amounts based on its position within the interface of the SPIRAL application (that is, which arm and circle it belongs to in order from 12 o'clock of the inner most circle being the first, to 11 o'clock of the outer most circle being the last metronome to be triggered via the delay mechanism built into SPIRAL mentioned previously). This composition features SPIRAL running at 100% of the speed. It features only eight pitches with each metronome being assigned its own pitch that does not change throughout the entire work. This piece fits very well into the genre of totalism as defined by Gann (1997) as its main focus is rhythm, indeed, multiple tempos coinciding with little (to no) variation in pitch or timbre aside from the structure of the piece as it unfolds. The choice of pitches used was arbitrary. The main sonic consideration for the piece was that each note should be of short duration with a fast enough attack to make the rhythms produced clearly audible, to emphasise the rhythmic character of the work.

4.6 Composition No. 6

This is another strictly totalist composition similar to Study No. 5 in its static harmonic/melodic content and use of SPIRAL, however it differs in the settings used. Firstly, all ninety-six metronomes are used instead of only eight, each one set to its own BPM. The time delay between the starting point of each metronome is 1200ms. The entire system is running at 5% of the whole number BPM speeds of 50-145 (resultant BPMs: 2.5-7.25). While both this study and Study No. 5 begin with a solitary metronome voice and grow in complexity from simple beginnings, this particular piece maintains a more sophisticated evolution of complexity as one would expect from using ninety-six different BPMs over only eight. Where Study No. 5 repeats itself every minute (once all eight metronomes are running), this study takes longer to repeat itself, hence its longer duration (however, it is cut short well before it repeats itself).

4.7 Composition No. 7

This piece uses MOM exclusively with no other software or hardware being used. MOM was originally designed as a standalone application before it was integrated into PENCIL and while it was standalone, it had its own audio synthesis capabilities (these were removed when it was integrated into PENCIL). The audio is actually created by the rhythmic output of the individual matrices triggering a very fast and loud transient that is sent through a band-pass filter with extreme resonance settings employed. This creates the pitched material heard within. Each MOM module was able to automate (via a [line] object) the change in frequency of the band-pass filter cutoff. These changes in frequency of the filter were controlled by a random algorithm in the patch (that included the ramp time to the new pitch) and triggering of this system was performed in real time during the recording of the study.

Only four matrices were used in MOM with the audio output of each sitting in a fixed panned location in the stereo field. This helps to facilitate the listening of each subdivision being used against the others.

4.8 Study No. 8

A commonly used compositional device (or system) is that of repetition with change. Repetition and the self similarity of iterated functions (see Figure 4.3) are of interest to me as a composer, as are fractals in general. In this short study, I explore these concepts within a rhythmic context by utilising MOM and numerous arpeggiator modules within PENCIL to create music whose output resembles said systems. There are multiple time scales employed (limited to simple duple rhythms) and the pitched and timbral content does not vary much throughout. I chose short duration notes of equal length and with a fast attack to exemplify the rhythmic content within. The presentation of this particular usage of MOM and the arpeggiator modules could be developed through additional interaction and programming within PENCIL to enable more variation (from subtle to extreme) yet this study demonstrates that MOM can be used to sound similar to standard, currently available step sequencers if one wants it to, albeit with MOM being much faster to set up and generate such material.

4.9 Study No. 9

The timbral quality of insect sounds and other related sounds from nature is explored in this study. As discussed by David Rothenberg in his book *Bug music: how insects gave us rhythm and noise* (2013), synthesisers are excellent tools for the emulation of insect sounds and soundscapes.

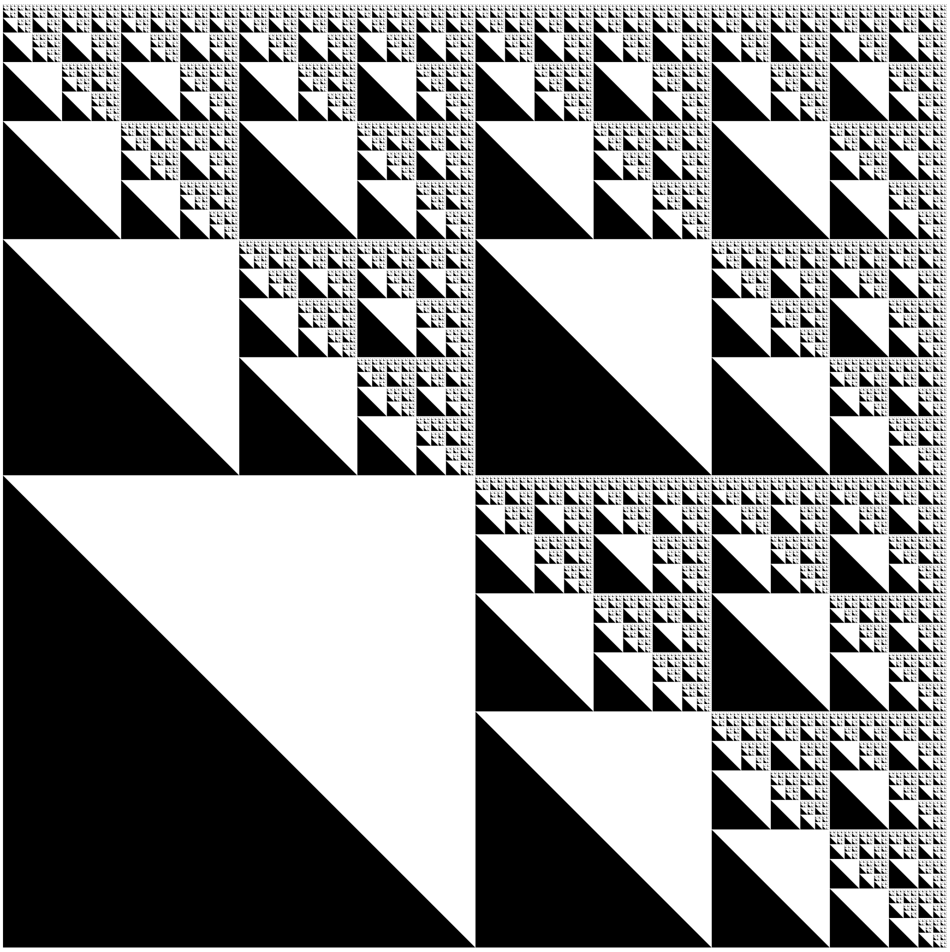


FIGURE 4.3 Example image of an iterated function system fractal[29]

Using synthesisers in this way is of great interest to me and this short composition provides an example of a possible outcome of the process. It features the Elektron Machinedrum (Mk2 UW) being manipulated by ONE KNOB. ONE KNOB is used to send MIDI CC messages to the Machinedrum, creating the grooving 'insect-scape' heard in the example. The Machinedrum, being primarily a drum machine, allowed for the insect emulation to be placed strictly within a fixed and rigid timing system. The internal sequencer of the Machinedrum was used to trigger the audio events (in the time signature of 11/8). Many of the synthesis parameters of the Maschinedrum were changed in realtime by ONE KNOB (via the preset system and automation within PENCIL and ONE KNOB). Syncing between ONE KNOB and the Machinedrum was achieved through MIDI communication and was necessary for the creation of the sounds heard throughout.

[29] "Really large triange fractal" image retrieved on March 18th, 2015 from https://ssodelta.files.wordpress.com/2014/03/really_large_triangle_fractal.png

4.10 Study No. 10

This study demonstrates a simple application of ONE KNOB which is used to generate and trigger pitch on the Yamaha FS1R synthesiser and also used to send control voltage (CV) to a Moog Moogerfooger MF-104m analog delay pedal via a MIDI to CV converter (the Encore Expressionist). The settings on the delay pedal, being controlled by CV that is being controlled by ONE KNOB were automated by ONE KNOB itself. As were the pitches being sent to the FS1R. The patch used on the FS1R was designed by myself prior to the recording. (No presets were harmed…). This composition contains subtle and ever changing rhythmic events also controlled and triggered by ONE KNOB. The complexity of the timing is 'buried' by the use of a slower attack in the amplitude envelope of each note being triggered. A further layer of rhythmic complexity is engineered by the manipulation of the delay time on the analog delay pedal.

4.11 Study No. 11

The use of quintuplet grooves (where each beat is subdivided into five even pulses) in musical pieces is often neglected by composers. This is possibly due to the quintuplet not being part of standard practise routines of musicians (aside from percussionists) thereby making it harder for them to perform accurately. However, just like semiquaver grooves, or triplet shuffle feels can be easily danced to and are rhythmically aesthetically pleasing, quintuplets also provide a very interesting rhythmic feel that can be easily interpreted due to the regularity of the actual beat being constant. This regularity enables easier synchronisation with personal GSA (groove superimposition alignment) as discussed in Section 1.3.7. In exploration of this concept, Study No. 11 is for a six FROGGIES ensemble triggering Native Instruments Maschine[30] software. I made use of quantisation from FROGGIES and used ONE KNOB to control pitch and quantisation amount. Quantisation was centered on a quintuplet groove throughout. The influence of the frogs on each other was in constant variation via automation of the nodes object in FROGGIES except for frog one which was switched off from being influenced by the other frogs thereby enabling a more consistent and solid tempo. This allowed for the constant, driving beat throughout the study and further enhanced the quantisation accuracy (as quantisation in FROGGIES is reset every time frog one 'calls').

[30] www.native-instruments.com/en/products/maschine/production-systems/maschine/ (accessed 30/03/2013)

4.12 Study No. 12

This particular study consists of FROGGIES triggering the Yamaha TX816 synthesiser in this slow and gradually evolving composition. The sounds generated by the synthesiser are full of constant high pitched material to maintain an insect chorus-like aesthetic. Anyone familiar with the loudness and ear piercing droning produced by cicadas in the Australian summer will recognise the sonic aesthetic I set out to achieve (made possible by very long sustain envelope sounds in the DX7 synth architecture). FROGGIES was run at a very slow speed for this study with frog call influences continuously randomised by the 'drunk frogs' settings within the application. This randomisation only affects the *strength of the influences* of the frogs on each other (as discussed in Section 3.9), thereby allowing the output of the dynamic equation at the heart of FROGGIES to morph over time with more variation. Without changing the influence that each frog has on each other, this phase-coupled oscillator system can easily fall into a simply hocket pattern that never changes, hence the randomisation of the influences.

4.13 Study No. 13

This study is presented in four parts, played one after the other and is an example of a possible exploration of rhythm and groove via the TURDS application. Each part was performed in real time without rehearsal and with randomised settings within TURDS. That is, the speeds of each track were randomised prior to (and during) the performance via the Quneo MIDI controller that TURDS was initially designed around. Then I performed x rhythms on the Quneo, triggering y rhythms from TURDS. Performed in real time, means the interaction of the performer with the software is crucial for a successful performance as it would be with performance on any musical instrument. Simple finger clicking and hand tapping on a desk were the source of the audio that was recorded into TURDS initially for each of these examples. Recording was done using the low quality microphone built in to the iMac computer that was being used at the time. This low quality recording is further exaggerated when making speed adjustments to the playback of recorded audio, particularly when slowing down the playback. It is important to note that the audio quality is of no concern for these studies however. They are presented as (quick) demonstrations of TURDS as a practise in groove creation tool. Or literally, as demonstrations of why TURDS has its name: The Ultimate Rhythm Discovery Station. The rhythms created in these studies are entirely different and non-related to the original rhythms that were recorded into each group of the TURDS software.

That is the point, the discovery, maintained balance and procuring of new grooves/rhythms from arbitrary source material.

4.14 Study No. 14

Study No. 14 demonstrates TURDS being used in a very different way. While being created as a ‘rhythm discovery’ program, it is also a powerful system for creating soundscapes and evolving sonic textures as discoverable in this study. The original source material for this study is a short vocal sample recorded using the inbuilt microphone on an iMac computer. This simple original audio sample was transformed in realtime by interacting with and controlling TURDS with the QuNeo MIDI controller. Most notably, I utilised the realtime speed control of the sample playback of each group, whilst also continuously resampling into new recording slots.

CHAPTER 5

Conclusion

5.1 Future

Continuing with the modular design system created in Max/Msp for this project, I seek to create more software modules for the PENCIL environment. These could include new rhythmic sequencers based on systems other than those explored in this project, as well as more simple and practical data parsing modules. Further enhancement could be made to the IMACS system to allow for more than one IMACS system to run at the same time without any cross-talk. Also, the ability for the user to create save-able ‘container modules’ inside PENCIL which would allow the user to quickly load pre-configured setups that they find themselves often building each time they use PENCIL.

I am also eager to get working more with robotic percussion instruments that are controllable via my software. Existing arduino/solonoid based robotic systems such as Robbie Avenaim’s SARPS are perfect examples of the mechanics I would utilise in developing my own system.[31]

5.2 Concluding remarks

With software design tools like Max/Msp at the hands of a composer, the limitations (as seen by the composer/user) of existing softwares and hardwares can be overcome by the creation of new tools. This thesis provided many examples of how software design was used to explore elements of rhythmic complexity in music composition. Firstly, a background was explained, detailing influences and existing similar projects. Then each piece of software was described in short detail including example ways that a user may like to use the software. Finally, the composition studies created with the software and included in the portfolio were explained also in short detail. In all, the thesis attempts to provide the reader with an understanding of the reasons why and processes employed by myself as a composer, to create new software to explore my creative desires, in this case, complex rhythms and grooves.

[31] http://www.robbieavenaim.com/robbieavenaim.com/S.A.R.P.S.html (accessed 07/03/2014)

Bibliography

Aihara, I., Kitahata, H., Yoshikawa, K., & Aihara, K. (2008). Mathematical modeling of frogs' calling behavior and its possible application to artificial life and robotics. *Artificial Life and Robotics,* 12(1), 29-32.

Aihara, I., Kitahata, H., Aihara, K., & Yoshikawa, K. (2006). Periodic rhythm and anti-phase synchronization in calling behaviors of Japanese rain frogs. *METR,* 35, 1-10.

Aihara, I., Takeda, R., Mizumoto, T., Otsuka, T., Takahashi, T., Okuno, H. G., & Aihara, K. (2011). Complex and transitive synchronization in a frustrated system of calling frogs. *Physical Review E, 83*(3), 031913.

Cemgil, A. T., Desain, P., & Kappen, B. (2000). Rhythm quantisation for transcription. *Computer Music Journal, 24*(2), 60-76.

Decker, S., Kendall, G., Schmidt, B., Ludwig, D., & Freed, D. (1986). A modular environment for sound synthesis and composition. *Computer Music Journal*, 10(4), 28-41.

Donati, V. (2005). Virgil Donati Ultimate Play-Along. *Alfred Publishing*, USA. (26)

Gann, K. (1997). American Music in the Twentieth Century. Shirmer Books, New York (355)

Gann, K. (2001). "Minimal Music, Maximum Impact - Totalism" retrieved February 9, 2014 from, http://www.newmusicbox.org/articles/minimal-music-maximal-impact/7/

Galanter, P. (2003). What is Generative Art? Complexity Theory as a Context for Art Theory. In *In GA2003-6th Generative Art Conference* (2003).

Godt, I. (2005). Music: a practical definition. *The Musical Times*, 146(1890) 83-88.

Greenfield, M. D., Tourtellot, M. K., & Snedden, W. A. (1997). Precedence effects and the evolution of chorusing. *Proceedings: Biological Sciences, 264*(1386), 1355-1361.

Greenfield, M. D. (1994). Synchronous and alternating choruses in insects and anurans - Common mechanisms and diverse functions. [Article; Proceedings Paper]. *American Zoologist, 34*(6), 605-615.

Harley, J. (1995). Generative processes in algorithmic compostion: chaos and music. *Leonardo*, 28(3), 221-224

Kernfeld, B. (2014). Groove (i). *The New Grove Dictionary of Jazz*, 2nd ed.. *Grove Music Online.Oxford Music Online*. Oxford University Press, retrieved December 14th, 2014, from http://www.oxfordmusiconline.com/

Krause, B. (2012). The great animal orchestra: finding the origins of music in the world's wild places [Kindle Edition]. Retrieved from amazon.com

Merriam-Webster (2014). *Rhythm*. Retrieved March 8, 2014, from http://www.merriam-webster.com/dictionary/rhythm

Nash, C. & Blackwell, A.F. (2008). Realtime representation and gestural control of musical polytempi. In A. Camurri, S. Serafin and G. Volpe (Eds), Proc. 8th Int Conf on New Interfaces for Musical Expression (NIME'08). Genova Italy June 4-8, 28-33.

Opie, T., & Brown, A. (2006). An introduction to eco-structuralism. In *International Computer Music Conference,* New Orleans, ICMA, 9-12.

Opie, T., & Brown, A. (2010). Aesthetic implications of the eco-structuralist process. In *Proceedings of the Australasian Computer Music Conference 2010*, Australian National University, Canberra, 43-50.

Oxford Music Online (2014). *Rhythm*. Retrieved March 6, 2014, from http://www.oxfordmusiconline.com

Pieslak, J. (2007). Re-casting metal: rhythm and meter in the music of Meshuggah. *Music Theory Spectrum*, 29(2) 219-246

Rothenberg, D. (2013). Bug music: how insects gave us rhythm and noise [Kindle Edition]. Retrieved from amazon.com

Sandred, O. (2004). Interpretation of everyday gestures - composing with rules. Preceeding to the *Music and Music Science symposium*, Stockholm, 1-6.

Smith, L. M. (2000). *A Multiresolution Time-Frequency Analysis and Interpretation of Musical Rhythm.* (Doctor of Philosophy PhD), The University of Western Australia.

Steinitz, R. (1996). Music, maths and chaos. *The Musical Times*, 137(1837), 14-20.

Wannamaker, R. (2012). Rhythmic relationships, farey sequences, and james tenney's spectal canon for conlon nancarrow. *Music Theory Spectrum*, 34(2), 48-70.

Whittall, A. (2014). Rhythm. *The Oxford Companion to Music*. *Oxford Music Online*. Oxford University Press, retrieved March 6, 2014, from http://www.oxfordmusiconline.com

Appendix A

A.1 Sequencer Modules of PENCIL

FROGGIES	A non-linear dynamic rhythmic system modelling frog call behavior (up to six 'frogs' per module). (See chapter 3.9 FROGGIES)
MOM	Allows simultaneous playback of up to thirteen rhythmic modes (per module) from Rhythm Matrix. Can also be used as a traditional step sequencer. (See chapter 3.6 Miles Okazaki Method (MOM)
SPIRAL	Allows simultaneous playback of up to ninety-six metronomes (per module). (See chapter 3.7 The Spiral Rhythm Clock (SPIRAL)

A.2 Data parsing (and other) modules of PENCIL

Arpeggiator	Allows many ways of controlling the numeric order of its output upon receiving an input trigger (reset-able). Also allows output of arpeggiator to be parsed through Vince Manzo's Modal Object Library. It also allows output to be sent to any MIDI channel/port with control of note velocity and duration, and control over behavior of repeated notes. Pitch bend and program change messages can also be sent from this module. The user is also able to use a MIDI keyboard controller to play the notes or chords that the arpeggiator will trigger when it fires. This enables the composer to 'dip their hands' into the computer's algorithmically generated rhythmic output providing it with realtime pitched material to work with.
BangThisValue	Allows a value to be set and banged out at a later time.
CCin	Allows MIDI continuous controller messages to be received into the PENCIL environment
CCout	Allows MIDI continuous controller messages to be sent out from the PENCIL environment
Comment	A simple text box allowing the user to comment out there user interface
Counter	Counts input with control of output direction and loop max/min (reset-able).
Delay	Delays input trigger bang, float or int by x milliseconds (stop-able).
EveryOther	Triggers output bang for every x input bangs (reset-able).
Gate	Allows input to be let through a gate or not based on probability
HotHand	Provides user interface for incoming CC data from the HotHand wireless midi controller (incoming pitch and roll plus the inverse of same).
InThisOrder	Allows incoming trigger to be output to x amount of outputs (up to 20) in sequential order (reset-able)
Line	Generates ramp to a set-able destination over a set-able duration.

Maths	Allows simple maths expressions (addition, subtraction, multiplication and division)
Metronome	Provides a simple metronomic output trigger at any speed definable in milliseconds or beats per minute
MIDIConvert	Allows an typical 16 pad MIDI controller to be remapped in many ways.
MidiFighter	A user interface for the incoming data of the MidiFighter Classic MIDI controller.
MidiIn	Allows MIDI pressure, aftertouch and pitchbend messages to be received into the PENCIL environment.
MidiRouter	Allows the routing of the input of any MIDI input port directly to the output of any MIDI output port on the computer running PENCIL.
Modulo	Finds the remainder of division of one number by another. Triggers bang when remainder equals zero.
MotionRecorder	Allows any incoming MIDI CC (or any value within PENCIL between 0 and 127) to be recorded and played back identically upon request, looped or not. Speed of playback is set-able via percentage of initial speed. Editing (or indeed creation) of data stored is possible via separate user interface window.
NoteIn	Allows incoming MIDI note on and note off information to be received into PENCIL
NoteOut	Allows MIDI note on and note off messages (including duration and velocity), to be sent from the PENCIL environment on any MIDI channel/ port. Allows, incoming notes to be parsed through Vince Manzo's Modal Object Library. Allows sending of pitchbend data and program change messages.
ONE KNOB	Allows parsing of incoming data in many ways. (See chapter 3.8 One Knob To Rule Them All (ONE KNOB).
PitchBendOut	Allows MIDI pitchbend data to be sent out of the PENCIL environment on any MIDI channel/port.

ProgramChange	Allows MIDI program change messages to be sent out of the PENCIL environment on any MIDI channel/port.
QWERTY	Provides user interface for allowing any of the computer keyboard keys to be used within PENCIL as controllers/triggers of events.
Random	Allows random numbers to be created within a definable minimum and maximum range. Also provides option for non-repeating random until all numbers within range have been randomly created whereupon the module is reset and capable of producing any number within the defined range once again.
ScaledValue	Allows an input integer or float to be scaled to a minimum and maximum value and output.
SelectNumber	Allows input number to be checked against variable with positive and negative results triggering separate outputs.
Speedlim	Allows incoming triggers or numbers to be output at regular intervals defined in milliseconds.
Switch	Allows input to be sent to up to 8 different outputs (one at a time).
SysexOut	Allows MIDI system exclusive messages (strings written in decimal format) to be sent from the PENCIL environment.
TapTempo	Allows input trigger to be converted into a tempo of beats per minute with control over how often the calculation is made (average over x amount of input triggers)
Threshold	Allows incoming integer or float to be compared to definable number as being: equal to, equal to or greater than, equal to or less than, greater than, less than, not equal to.
TriggerThisOrder	Allows the precise ordering of trigger events within the PENCIL environment of up to 20 outgoing events per incoming trigger.
Wiimote	Allows the incoming data of a Nintendo Wiimote to be used within the PENCIL environment. Third party software called Osculator must be running in background for Wiimote data to be received by this module.

Appendix B

B.1 The Frog Code (java code) by Dr Ollie Bown

```
import com.cycling74.max.MaxObject;


public class Kuramoto extends MaxObject {

        int N;
        double omega;
        double timeStep;
        double[][] K;  //N squared connections, K[i][j] is the effect of frog j on frog i
        double[] theta;
        double[] thetaDot;
        double[][] diff;
        double noise;

        public Kuramoto(int N) {
                this.N = N;
                declareIO(1,3);
                basicSetup();
        }

        public void basicSetup() {
                omega = 1;
                timeStep = 0.1;
                K = new double[N][N];
                theta = new double[N];
                thetaDot = new double[N];
                diff = new double[N][N];
                for(int i = 0; i < N; i++) {
                        theta[i] = Math.random() * 2 * Math.PI;
                        for(int j = 0; j < N; j++) {
                                K[i][j] = 1;
                        }
                }
        }

        public void randomise() {
                for(int i = 0; i < N; i++) {
                        theta[i] = Math.random() * 2 * Math.PI;
                }
        }

        public void bang() {
                for(int i = 0; i < N; i++) {
```

```
            thetaDot[i] = omega + noise * (Math.random() * 2. - 1.);
            for(int j = 0; j < N; j++) {
                if(i != j) {
                    //accumulate the influence of frog j on frog i
                    thetaDot[i] -= K[i][j] * Math.sin(theta[j] - theta[i]);
                }
            }
        }
        for(int i = 0; i < N; i++) {
            theta[i] += thetaDot[i] * timeStep;
        }
        for(int i = 0; i < N; i++) {
            for(int j = 0; j < N; j++) {
                diff[i][j] = (theta[i] - theta[j]) / Math.PI;
            }
        }

        double[] tempTheta = new double[N];

        for(int i = 0; i < N; i++) {
            tempTheta[i] = theta[i] % (2 * Math.PI);
            while(tempTheta[i] < 0) tempTheta[i] += 2 * Math.PI;
        }

        outlet(0, tempTheta);
        outlet(1, thetaDot);
        double[] neighbourDiffs = new double[N];
        neighbourDiffs[0] = diff[0][N-1];
        for(int i = 1; i < N; i++) {
            neighbourDiffs[i] = diff[i][i-1];
        }
        outlet(2, neighbourDiffs);
    }

    public void printDiffs() {
        for(int i = 0; i < N; i++) {
            for(int j = 0; j < N; j++) {
                System.out.print(diff[i][j] + " ");
            }
            System.out.println();
        }
    }

    public void printStatus() {
        post("N = " + N);
    }

    public void setN(int n) {
```

```
    N = n;
    double[][] newK = new double[N][N];
    theta = new double[N];
    for(int i = 0; i < N; i++) {
        theta[i] = Math.random() * 2 * Math.PI;
        for(int j = 0; j < N; j++) {
            newK[i][j] = 0;
            if(i < K.length && j < K[i].length) {
                newK[i][j] = K[i][j];
            }
        }
    }

    K = newK;
    thetaDot = new double[N];
    diff = new double[N][N];
}

public void setOmega(double omega) {
    this.omega = omega;
}

public void setTimeStep(double timeStep) {
    this.timeStep = timeStep;
}

public void setK(double[][] k) {
    K = k;
}

public void setNoise(double noise) {
    this.noise = noise;
}

public void setK(double[] k) {
    int index = 0;
    for(int i = 0; i < N; i++) {
        for(int j = 0; j < N; j++) {
            K[i][j] = k[index++];
        }
    }
}

public void setSpecificK(int i, int j, double k) {
    K[i][j] = k;
}

public void setAllK(double k) {
        for(int i = 0; i < N; i++) {
            for(int j = 0; j < N; j++) {
                K[i][j] = k;
            }
        }
    }

    public void setKNeighboursOnly(double k) {
        for(int i = 0; i < N; i++) {
            for(int j = 0; j < N; j++) {
                if((i == j - 1) || (j == i - 1)) {
                    K[i][j] = k;
                } else {
                    K[i][j] = 0;
                }
            }
        }
    }

    public void setTheta(double[] theta) {
        this.theta = theta;
    }

    public void setTheta(double theta) {
        for(int i = 0; i < N; i++) {
            this.theta[i] = theta;
        }
    }

    public void setThetaDot(double[] thetaDot) {
        this.thetaDot = thetaDot;
    }

    public void setDiff(double[][] diff) {
        this.diff = diff;
    }



}
```

Appendix C

C.1 Rhythm Matrix

C.1.1 How the patterns are derived

Please see Chapter 2.4 Rhythm Matrix for a description of how the rhythms are derived.

C.1.2 How to interpret the books

1. A crotchet is used to symbolise a group of 2 subdivisions.
2. A dotted crotchet is used to indicate a group of 3 subdivisions.
3. A minim is used to indicate a group of 4 subdivisions.
4. A minim with a dot in the middle is used to indicate a group of 5 subdivisions.
5. The changing stem direction is only used to make the two different rhythmic groupings clearer.
6. Notes that are highlighted are notes that land on a beat.
7. The number at the top of each page is used to remind what the current subdivision is.

C.2 Rhythm Matrix 6 beats

Can be found in PDF format on CD 2 enclosed in the folder named “Appendix C”.

C.3 Rhythm Matrix 4 beats addendum

Can be found in PDF format on CD 2 enclosed in the folder named “Appendix C”.

C.4 Rhythm Matrix Combinator by Isaac Hayward

Can be found in html format on CD 2 enclosed in the folder named “Appendix C”.

Rhythm Combinator HTML script:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Rhythm Combinations & Permutations</title>
</head>

<body style="font:12px Arial, Helvetica, sans-serif;">
<table width="581" border="0">
<tr>
<td colspan=4>
<h1>THE RHYTHM COMBINATOR</h1>
```

```
<div align="right">Isaac Hayward 2012
</div>
<hr style="border:1px #000000 solid;" noshade /></td>
</tr>


<form name="gui">


<tr>
    <td width="116"><label>Beats in bar:</label></td>
    <td width="155"><input type="text" name="fBeats" value="16" size="3" /></td>
    <td width="160"><label>Number of note values:</label></td>
    <td width="122"><label>
      <select name="fBagSize" id="fBagSize" onchange="updateOptions();">
            <option value="1">1</option>
        <option value="2" selected="selected">2</option>
        <option value="3">3</option>
        <option value="4">4</option>
        <option value="5">5</option>
        <option value="6">6</option>
        <option value="7">7</option>
        <option value="8">8</option>
        <option value="9">9</option>
        <option value="10">10</option>
        <option value="11">11</option>
      </select>
    </label></td>
  </tr>
  <tr>

  </tr>
  <tr>
    <td>Note values:</td>
    <td colspan="3">
        <input name="dur0" type="text" id="dur0" value="2" size="2" maxlength="2" />
    <input name="dur1" type="text" id="dur1" value="3" size="2" maxlength="2"/>
    <input name="dur2" type="text" id="dur2" value="5" size="2" maxlength="2" />
    <input name="dur3" type="text" id="dur3" value="7" size="2" maxlength="2" />
    <input name="dur4" type="text" id="dur4" value="11" size="2" maxlength="2" />
    <input name="dur5" type="text" id="dur5" value="13" size="2" maxlength="2" />
    <input name="dur6" type="text" id="dur6" value="17" size="2" maxlength="2" />
    <input name="dur7" type="text" id="dur7" value="19" size="2" maxlength="2" />
    <input name="dur8" type="text" id="dur8" value="23" size="2" maxlength="2" />
    <input name="dur9" type="text" id="dur9" value="31" size="2" maxlength="2" />
    <input name="dur10" type="text" id="dur10" value="37" size="2" maxlength="2" />
</td>

  </tr>

  <tr>
            <td colspan="4">
    <div align="right"><input type="button" value="Calculate" onclick="Calculate();" /></div>
    </td>
  </tr>
```

```
<tr>
<td colspan="4">
<textarea name="output" cols="70" rows="40">
</textarea>
</form>
</td>
</tr>
</table>
<script language="javascript">

var MAX_BAG_SIZE = 11;
var BAG_SIZE = 2 ;
var bag = new Array(MAX_BAG_SIZE);

var BAR_SIZE = 16;

var counter = 0;

var output = document.gui.output;
output.value = "";

var COMBO = new Array();

updateOptions();

function Log(loggables) {
        output.value = output.value + loggables;
}

function AddCombo(digits) {
var result=0;
        for (z = 0; z <= digits; z++) {
                result += parseInt(COMBO[z]);
        }
        return result;
}

function CatCombo(digits) {
var result="";

        for (n = 0; n <= digits; n++) {

                result += COMBO[n].toString();
        }
        return result;
}

function Recursor(digit) {

var i;
        for (i = 0; i < BAG_SIZE; i++) {
                // set digit to be the one we've taken from the bag
                COMBO[digit] = bag[i];

                // now test to see if it's over or equal to
                if (AddCombo(digit) == BAR_SIZE) {
                        //Tell us of the great news
                        Log(CatCombo(digit)+"\n");
                        counter++
                        continue;
                } else if (AddCombo(digit) >BAR_SIZE) {
                        continue;
                }


                Recursor(digit+1);
                continue;
        }

        return;
}

function Calculate() {

        output.value = "";



        BAR_SIZE = document.gui.fBeats.value;
        if (BAR_SIZE == 0) {
                alert("Please enter number of beats in bar.");
                return;
        }

        BAG_SIZE = document.gui.fBagSize.value;

        for (n = 0; n < BAG_SIZE; n++) {

                eval("bag[n] = document.gui.dur" + n + ".value;");

                if (bag[n] == 0) {
                        alert("Please enter note value #" + (n+1) + ", or change the number of note values.");
                        return;
                }

        }

        counter = 0;


        Recursor(0);

        Log("\nFound " + counter + " combination(s).");

}

function updateOptions() {

        for (n = 0; n < MAX_BAG_SIZE; n++) {
                if (n < document.gui.fBagSize.value) {
                        eval("document.gui.dur" + n + ".disabled = false;");
                } else {
                        eval("document.gui.dur" + n + ".disabled = true;");
                }
        }
}

//Recursor(0);

</script>

</body>
</html>
```

Appendix D

D.1 Software CD - CD No. 1

D.1.1 PENCIL

D.1.2 TURDS

D.1.3 Max616Runtime_131209.dmg

D.1.4 pasteMeIntoPatchForImacsUsage.maxpat

D.1.5 IMACS.maxpat

D.1.6 README_SOFTWARE_INSTALLATION_INSTRUCTIONS.rtf

D.1.7 README_PENCIL_IMACS_HELP_TUTORIAL_FILE.rtf

D.1.8 README_TURDS_HELP_TUTORIAL_FILE.rtf

Installation instructions are found in the README_SOFTWARE_INSTALLATION_INSTRUCTIONS.rtf document located on the Software CD No. 1.

Appendix E

E.1 Audio CD (CD 2)

E.1.1 Fourteen composition studies

Track 1: Study No. 1	2:31	min
Track 2: Study No. 2	7:49	min
Track 3: Study No. 3	1:43	min
Track 4: Study No. 4	1:48	min
Track 5: Study No. 5	3:48	min
Track 6: Study No. 6	11:14	min
Track 7: Study No. 7	4:31	min
Track 8: Study No. 8	1:00	min
Track 9: Study No. 9	1:27	min
Track 10: Study No. 10	5:00	min
Track 11: Study No. 11	4:03	min
Track 12: Study No. 12	18:22	min
Track 13: Study No. 13	3:50	min
Track 14: Study No. 14	7:06	min

Total playing time: 74 minutes and 12 seconds