



# University of HUDDERSFIELD

## University of Huddersfield Repository

Freeman, Samuel David

Exploring visual representation of sound in computer music software through programming and composition

### Original Citation

Freeman, Samuel David (2013) Exploring visual representation of sound in computer music software through programming and composition. Doctoral thesis, University of Huddersfield.

This version is available at <http://eprints.hud.ac.uk/id/eprint/23318/>

The University Repository is a digital collection of the research output of the University, available on Open Access. Copyright and Moral Rights for the items on this site are retained by the individual author and/or other copyright owners. Users may access full items free of charge; copies of full text items generally can be reproduced, displayed or performed and given to third parties in any format or medium for personal research or study, educational or not-for-profit purposes without prior permission or charge, provided:

- The authors, title and full bibliographic details is credited in any copy;
- A hyperlink and/or URL is included for the original metadata page; and
- The content is not changed in any way.

For more information, including our policy and submission procedure, please contact the Repository Team at: [E.mailbox@hud.ac.uk](mailto:E.mailbox@hud.ac.uk).

<http://eprints.hud.ac.uk/>

# Exploring visual representation of sound in computer music software through programming and composition

Samuel David Freeman

A thesis submitted with a portfolio of works to the University of  
Huddersfield in partial fulfilment of the requirements for the degree of  
Doctor of Philosophy

December 2013

Minor amendments and corrections April–June 2014

# Summary Table of Contents

Abstract	3
Overview of the Portfolio Directories	5
1: Introduction	6
1.1 Motivation	6
1.2 Initial context	7
1.3 Thesis structure	11
2: Looking at sound	13
2.1 Looking at physical aspects	13
2.2 Looking at psychological aspects	22
2.3 Looking at looking at sound	27
3: In software, on screen	28
3.1 Pixels	29
3.2 Visyn	38
3.3 Sub synth amp map	42
3.4 Conceptual model of a gramophone	63
3.5 Gramophone_005g	76
3.6 saw~onepole~noise~click~	101
3.7 Software as substance	111
4: Spiroid	128
4.1 Spiroid	128
4.2 Some similar spirals	137
4.3 Spiroid display of partials as arcs on an lcd in MaxMSP	144
4.4 Spiroid as an interface for input	154
4.5 Assessment of the spiroid	168
5: CirSeq	172
5.1 CirSeq and thisis	173
5.2 CirSeq inception	175
5.3 CirSeq Zero	182
5.4 Aesthetic analogies in the exploration of CirSeq patterns	208
5.5 Critique and chapter conclusion	212
6: Sdfsys	218
6.1 Introducing sdfsys	218
6.2 Alpha	221
6.3 Beta	241
6.4 Composing with sdfsys	261
6.5 Evaluating sdfsys	274
7: Conclusion	276
7.1 Framing the project	276
7.2 Contextual grounding	277
7.3 Techno-aesthetic development	278
7.4 Tildegraphing	279
7.5 Music via visual representations	279
7.6 The sdfsys environment	281
Detailed Table of Contents	283
Table of Figures	288
Bibliography	291

# Abstract

Presented through contextualisation of the portfolio works are developments of a practice in which the acts of programming and composition are intrinsically connected. This practice-based research (conducted 2009–2013) explores visual representation of sound in computer music software.

Towards greater understanding of composing with the software medium, initial questions are taken as stimulus to explore the subject through artistic practice and critical thinking. The project begins by asking: How might the ways in which sound is visually represented influence the choices that are made while those representations are being manipulated and organised as music? Which aspects of sound are represented visually, and how are those aspects shown?

Recognising sound as a psychophysical phenomenon, the physical and psychological aspects of aesthetic interest to my work are identified. Technological factors of mediating these aspects for the interactive visual-domain of software are considered, and a techno-aesthetic understanding developed.

Through compositional studies of different approaches to the problem of looking at sound in software, on screen, a number of conceptual themes emerge in this work: the idea of software as substance, both as a malleable material (such as in live coding), and in terms of outcome artefacts; the direct mapping between audio data and screen pixels; the use of colour that maintains awareness of its discrete (as opposed to continuous) basis; the need for integrated display of parameter controls with their target data; and the tildegraph concept that began as a conceptual model of a gramophone and which is a spatio-visual sound synthesis technique related to wave terrain synthesis. The spiroid-frequency-space representation is introduced, contextualised, and combined both with those themes and a bespoke geometrical drawing system (named thisis), to create a new modular computer music software environment named sdfsys.

## Acknowledgements

This project was supported by the Arts and Humanities Research Council (AHRC) and supervised by Professors Clarke and Adkins (JMC and Monty) to whom I am eternally grateful, as I am also to family, to friends, and to Huddersfield.

## Copyright Statement

- i. The author of this thesis (including any appendices and/or schedules to this thesis) owns any copyright in it (the “Copyright”) and s/he has given The University of Huddersfield the right to use such Copyright for any administrative, promotional, educational and/or teaching purposes.
- ii. Copies of this thesis, either in full or in extracts, may be made only in accordance with the regulations of the University Library. Details of these regulations may be obtained from the Librarian. This page must form part of any such copies made.
- iii. The ownership of any patents, designs, trademarks and any and all other intellectual property rights except for the Copyright (the “Intellectual Property Rights”) and any reproductions of copyright works, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property Rights and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property Rights and/or Reproductions.

## Overview of the Portfolio Directories

An overview of the portfolio directories – found on the attached DVD, and also available online at

<http://sdfphd.net/portfolio> – is given here, showing only folder names, and only to a depth of two:

- sdfphd\_portfolio\_by\_chapter
  - chapter3\_inSoftware\_onScreen
    - model\_of\_a\_gramophone
    - MoEpixels
    - saw~onpole~noise~click~
    - sub\_synth\_amp\_map
    - visyn
  - chapter4\_Spiroid
    - BensonSymmetryEx3
    - nRadii
    - spiralArc300
    - spiroidArc+timespace
  - chapter5\_CirSeq
    - \_alpha\_preview\_and\_thisis
    - CirSeq\_Zero
  - chapter6\_sdfsys
    - \_alpha
    - \_beta
    - \_composing\_with\_sdfsys

# 1: Introduction

Sound is a fleeting and transitory phenomenon; existent as itself only in the moment. To make sound tangible, so that we can see it, hold it, or freeze it in time, various forms of symbolic representation have been invented. In the composition of music it is representations of sound that are worked with, whether that be the dots and lines of common music notation, a sound-recording medium, or computer software. Different forms of representation emphasise different aspects of sound in the visual domain. This project began by asking:

- How might the ways in which sound is visually represented influence the choices that are made while those representations are being manipulated and organised as music?

## 1.1 Motivation

The motivation for that line of questioning came, in part, from the experience of having elements of my compositional practice examined in research by Jean-Baptiste Thiebaut on the use of sketching by contemporary composers (Healey & Thiebaut, 2007; Jean-Baptiste Thiebaut, 2010). Thiebaut's case study featured a series of my compositional sketches from my 2006 eight-channel work *Octorgan*, and include some early development of what I now refer to as spiroid-frequency-space representations of the frequency-domain. Observation of the spiral structure emergent from several octaves of a scale of notes arranged in a circle, however, for me began in the compositional process of an earlier piece, *six notes of a whistle flute* (2005). Pen and pencil on paper have always featured extensively in my musicking, but it was with those two works that a consciously meditative approach to geometrical drawing in the exploration of musical ideas took seed.

In realising musical ideas which are borne through the free association of geometrical forms with conceptual facets of sound, the challenge was encountered that available software did not share

the same associative values. Ideas had to be re-thought, and thus re-shaped, in order to manifest as sound through the technology at hand. This project thus set out to close the gap between ways of thinking derived from intuitive spatio-visual exploration of ideas to the ways of working permitted by existent software.

Projects such as those by Thiebaut and, to give another example, *InkSplorer* by Garcia *et al.* (2011) seek to develop systems that enhance the paper based activities which are common in the composition of music, particularly in the early stages of a work. Garcia *et al.* and Thiebaut have each followed a model of research, which might be characterised as belonging to the field of music technology in the sciences, in order to develop systems for general use by a target demographic. Rather than proposing general purpose solutions, the present thesis – situated in the field of music technology in the arts – documents and contextualises the search for, development towards, and use of, idiosyncratic compositional responses to questions about representing aspects of sound visually in interactive systems.

The motivation to develop a bespoke software environment in which to compose new music began as a move away from the live laptop performance based work that – alongside computer music pedagogy – had been my focus in the years prior to the start of this project. As a continuation of my earlier compositional practice, this project was initially proposed to explore methods of visual representation specifically designed as circular in appearance. During the progression of the research, in light of new findings and changing perceptions of the work and its context, the scope of this project has been refined, and its aims and priorities have been adapted.

## 1.2 Initial context

### 1.2.1 Representations of sound

It is easy to think of there being sound inside the computer when we work with it, but really we

know that there are only representations of sound – either as recordings or synthetic potential– from which actual acoustic sound can then be produced via transducers attached to outputs of the computer system. Sound recordings exist within the computer as audio-buffers and sound-files in the symbolic form of time-ordered numeric values, but it is possible to think of this simply as being 'digital audio' in the same way that we may think of there being 'analogue audio' on lengths of tape without giving much thought to the physics of magnetic fields and so on. Likewise, the programmatic algorithms describing the synthesis or processing of digital audio in software can be thought of as equivalent to the electronic phenomena in analogue circuitry – whether that is what the software is modelling or not.

Although not limited to do so, software often mimics the audio tape medium, albeit in greatly augmented ways. By accepting digital audio as an extension or continuation of the technological ability to record and reproduce sound, we also accept the ways of thinking about, and working with, sound that the previous technologies provided. Applicable to digital audio are thus, for example, the techniques associated with the *musique concrète* and acousmatic traditions.

The invention of *musique concrète* was possible in the mid-twentieth century because of a practice which took root in the mid-nineteenth century: the art of recording sound as a trace of its vibrational motion. In contrast to music that is made by working with recorded sound, we can look at a form of symbolic representation which has a much longer history in the theory and practice of music: the scored notation of sound parameters. Towards the matter of musical notation, and opening with a statement of aesthetic pertinence, Gareth Loy (2006, p. 11–12) writes:

The realm of personal musical experience lies entirely within each one of us, and we cannot share our inner experiences directly with anyone. However, many world cultures have developed systems for communicating musical experience by representing it in symbolic written and verbal forms. [...] one such system [is] the *Western common music notation system* (CMN). Its prevalence today makes it a good entry point to a broader discussion

Conversational description of this project, and the research context from which it stems, has

often called upon the concept of ‘dots on lines’ (in reference to common music notation) as a starting point because this is a familiar form of visual representation in music. The idea that the act of arranging dot-like symbols on horizontal lines can be described as ‘composing music’ is generally accepted because it is well enough known that such symbolic representations can be interpreted to produce a prescribed sequence of sounding events. The suggestion is then given that, as CMN evolved over time, the visual form of the CMN score presented new opportunities for composers to explore, such as by offering various possible symmetries on the page through which to describe patterns of rhythm and pitch which might not otherwise have been prescribed. The conclusion of this example is then that the way a system of visual representation of sound actually looks to the eye will affect the creative thoughts that one might have about what one might do with those representations, and thus do with the parameters of sound being represented.

Several centuries of CMN use and development have nurtured an approach to music 'which starts from an abstract conception and notation leading to a concrete performance', and it was 'against the “excess of abstraction” of [his] period' that Pierre Schaeffer reacted with the *concrète* approach to music. Schaeffer 'started from the concrete sound material, from heard sound, and then sought to abstract musical values from it' (Chion, 1983, p. 37).

One way of describing the work undertaken in this project would be to say that it seeks to look into the essence of those representations of sound that may commonly be thought of as being the sounds that they represent. Representations of sound in visual form can only ever portray a metaphorical shadow of the thing being represented; only ever can some of the characteristics of sound be shown because sound is not itself a visual phenomenon. In choosing to employ a particular method of representation for the purpose of organising sound as music, one delimits the realm of potential control and thus determines the scope of interaction within that work.

## 1.2.2 Computer music software

A significant characteristic of contemporary culture is the omnipresence of software systems (Manovich, 2008).<sup>1</sup> What possible implications this situation may have to new musics are approached by this research, and the resultant difficulties encountered in defining an artistic 'work' as distinct from the 'tools' used to make it are also explored. Computer music software, as referred to in the title of this project, could be taken as to encompass any and all software which may facilitate musicking in its many and varied forms; from digital audio workstation (DAW) applications to digital media players and contemporary web browsers. This project investigates the interactive processes of composition that transpire between a human composer and compositional materials in their manifest state as visual representations in computer music software. Creation of new software systems (programming) is an integral part of my compositional practice. Whereas programming was, in my works prior to this project, thought of as a means to an end in the pursuit of sound making systems, this research has set as its context the conscious engagement with the software medium.

The original proposal also suggested that multitouch tangible interface technologies would feature in the design of new software works. That direction of research was inspired, at the time, by the Lemur controller from JazzMutant<sup>2</sup> and the reacTable (Jordà *et al.*, 2007) systems. Mobile telecommunication devices with multitouch interfaces were also becoming more common, and the idea of finger-touchable interfaces to software was very appealing. Six months into the project, however, critical reflection at the launch of the Apple iPad (Apple, 2010), lead to a reevaluation of the concept as beyond the scope of this work which chooses instead for its focus to remain on software that is presented on a standard computer screen, or visual display unit (VDU) as was the

---

1 Manovich's digital manuscript for the book *Software Takes Command* is cited; this became unavailable online as it was to be published in July 2013 by Continuum, to 'be part of the International Texts in Critical Media Aesthetics series edited by Francisco J. Ricardo' (Manovich, 2012).

2 An obituary for this pioneering piece of hardware, written in November 2010, reports the closing of JazzMutant who introduced the Lemur in 2005 (Kim, 2010).

descriptive term used in the early days of computer software graphical interface design (Sutcliffe, 1988).

## 1.3 Thesis structure

The following chapter (§2: Looking at sound) describes the techno-aesthetic<sup>3</sup> basis for this practice-based research by examining (first) physical properties of sound, and technologies by which are able to see them, and (second) some of the psychoacoustic aspects of sound upon which portfolio works find context.

Chapter three (§3: In software, on screen) presents six portfolio pieces, each with their own contextualisation; these pieces comprise a developmental trajectory with an increasingly philosophical approach to the methods and materials being worked with. That chapter concludes (§3.7) with a discussion of 'software as substance'.

The spiroid-frequency-space concept, that has already been mentioned (§1.1), is introduced in chapter four (§4: Spiroid). The formalisation of the concept within this project, the wider context of similar structures, and my first software implementations of it are described.

The CirSeq concept is then the subject of chapter five (§5: CirSeq); its inception as a 'time-space' counterpart to the spiroid-frequency-space visual representation is detailed, and the compositional use of a software implementation of the concept is discussed.

Chapter six (§6: sdfsys) describes the work that is a culmination of the preceding works; development of the sdfsys computer music software environment, across its alpha and beta versions are presented. Compositional practice is then explored through new pieces that have been created with sdfsys.

---

<sup>3</sup> The term 'techno-aesthetic' is contextualised, for example, by the (1982) writing of Gilbert Simondon, translation published in Parrhesia Journal (Gilbert Simondon, 2012).

Final conclusions, and proposals for further work are discussed in chapter seven (§7).

## 2: Looking at sound

Sound is a psychophysical phenomenon: there is the physical reality of molecular motion (acoustics), and there is the psychological subjectivity of the human observer (psychoacoustics). To those who work with the medium of sound, awareness of both its physical and psychological properties – as well as of how complex structures of sound behave in both space-time and human perception – are vital. Beyond the depth of awareness that a practitioner may possess is the amount to which that insight is applied in their workings with sound, be that consciously or otherwise.

In seeking to connect with sound itself through interactive visual representations, this project exist within the context of a contemporary occurrence of the periodic tendency described by Nicolas Collins (2012, p. 1):

The acoustical reality of sound, and its quirky interaction with our sense of hearing, periodically drives artists to return to the “year zero” in music—before the codification of melody, rhythm and harmony—and explore fundamental aspects of the physics and perception of sound.

This chapter first examines acoustic properties of sound as the stimulus of visual manifestation and begins to identify aesthetic foundations for the project. Some psychoacoustic aspects of sound are then introduced as developmental foundation for creative works. The themes chosen for discussion here are of those that feature in the portfolio pieces, though the multitude of connections between concepts and works are not all expressed at this stage; the contextualisation provided here is built upon in later chapters.

### 2.1 Looking at physical aspects

#### 2.1.1 Physical patterns

The physical phenomena of sound are prominent, for example, in works by Alvin Lucier. His perhaps most famous work, from 1969, *I am sitting in a room...* (Lucier, 1990), uses the medium of

tape-recording to, in effect, sonify the physical properties of the space in which the piece is performed. In Lucier's *The Queen of the South* sound is directed to give rise to visible patterns on a surface in a way similar those ways associated with the works of Ernst Chladni (1787) and Hans Jenny<sup>4</sup>.

So called Chladni patterns, or figures, are shapes that are 'formed when a sand-covered surface is made to vibrate. The sand collects in the regions of least motion' (Stevenson & Lindberg, 2011a). The surface patterns created by various modes of vibration are geometrical with numerous symmetries, and often similar in appearance to mandala formations. When using an oscillator to stimulate the plate with sinusoidal sound waves, higher frequencies yield more complex patterns than do lower frequencies, with the precise frequency required to produce a particular nodal pattern being dependant on the physical properties of the plate.

---

4 Hans Jenny, 1967, *Kymatik - Wellen und Schwingungen mit ihrer Struktur und Dynamik / Cymatics - The Structure and Dynamics of Waves and Vibrations*



Figure 2.1: Chladni patterns

What Chladni was observing, in the late eighteenth-century, when the edge of a sand-covered metal plate was bowed and a visual manifestation of resonant sound waves was seen to take shape on the surface, was the same thing that Robert Hooke had seen in flour on glass a century before that; and we can today observe the same acoustic principles at play. Contemporary artist Jodina Meehan<sup>5</sup> works with these principles, as illustrated in Figure 2.1 (Frank, 2010).<sup>6</sup> The principles themselves are timeless and unchanging: a statement that, while it may seem obvious, is pertinent to the aesthetics of the project; they can manifest in different ways but the principles remain the same.

---

<sup>5</sup> Editor of *Journal of Cymatics: The Study of Sound Made Visible* (<http://cymatica.com/>)

<sup>6</sup> Other examples of contemporary practitioners working with Cymatic principles include Lewis Sykes (2011), and Meara O'Reilly whose *Chladni Singing* work (O'Reilly, 2010) reminds me of *The Queen of the South*.

With the title of Hans Jenny's 1967 book came the term Cymatics (translated from *Kymatik*) to mean the study of wave phenomena. Jenny invented the tonoscope so that the human voice may act as the stimulus to a surface on which sand is spread. Of an early performance of Lucier's *The Queen of the South*, March 1972, Tom Johnson describes (1989, p. 23):

[...] four singers sit around a square metal plate, about three feet across, with sand sprinkled on it. As they sing into their microphones, the metal plate vibrates, causing the sand to shift into many different patterns.

A year later from that, Johnson writes that the piece has grown in both duration and scope to include multiple surfaces that 'were stimulated by purely electronic sounds' (*ibid*, p. 42):

It may seem odd that a group of people would spend two hours watching minute particles vibrate and listening to the sounds that vibrate them, but there is an odd attraction to this symbolic activity, and most of the audience stayed until the very end. For me, the strongest association is with Navajo sand painting. But instead of a medicine man, the laws of physics are in charge of the mysterious rites. Lucier told me at the end of the evening that his own strongest association is with alchemy and that 'The Queen of the South' is an alchemical term.

He was attracted to the idea because of an appreciation for basic substances and for the mystery of how they interact with one another. Those with scientific backgrounds or with backgrounds in the visual arts would probably have made other associations.

The sounds were quite interesting in their own right, and it seemed like a rare opportunity to be able to watch these sounds as the many beautiful designs took form on the sheets. I felt I was getting a clue to the mysteries of the laws of the cosmos. What more can one ask of a work of art?

Cymatic patterns are visual manifestations of sound in particular conditions. Contemporary Cymatics often employ liquid surfaces and dramatic lighting to capture photographic images of the mesmerising shapes that transcend – from sonic to seeable – domains and stimulate imagination.<sup>7</sup> These spatial manifestations are instantaneous representations of motion; they let us see something of the movement present at a given moment in time, revealing to the human eye dynamic processes of acoustic phenomena. Inspired by the sense of ever-present 'now' that these evanescent patterns portray has come the idea of reversing the process, of controlling visual aspects in the moment and letting sound be derived from the visual patterns being made.

---

<sup>7</sup> See, for example, <http://www.cymatics.org>

## 2.1.2 The temporal trace

A more commonly encountered visual manifestation of the acoustic phenomenon is that which is drawn as a trace of sound pressure variations over time. This describes the typical 'waveform' representation of recorded sound, seen often in computer software. The concept of writing acoustic sound as a graph of amplitude variation along an axis of time is so common place in contemporary culture that one may overlook its impact on the way we conceive of sound as substance in the composition of music.

While the influence of sound recording technology on music in general is the subject of extensive research, this project is more specifically concerned with the visual aspect of recorded sound; the roots of which are found with the phonautograph, now described as 'the first instrument to record airborne sounds capable of being played back' (FirstSounds.ORG, n.d.). Patrick Feaster writes (2008, p. 3):

On 26 January 1857, Édouard-Léon Scott de Martinville of Paris deposited a seven-page handwritten document with the Académie des Sciences. [...] Entitled “Principes de Phonautographie,” this document is the earliest known account of the idea of using a membrane and stylus combination to inscribe atmospheric sound waves.

It was not within the remit of the invention for the phonautograph to be able to reproduce as sound the recordings that it made. Scott described his invention in relation to photography, which was itself a newly emerging technology in the mid nineteenth-century, and suggested that the traces made of sound – the phonautograms – be subject to visual inspection and study. The idea was of 'written sound' to be read by human eyes. It was only in 2008 that the FirstSounds initiative re-synthesised sounds recorded as far back as 1860 as digital sound-files (Hennessey & Giovannoni, 2008). By doing so, they have effectively rewritten the first chapter in the history of sound recording and playback.<sup>8</sup>

---

<sup>8</sup> Previously the phonograph – invented by Thomas Edison in 1877 – had been viewed as the starting point in the history of sound recording; the phonograph was indeed the first machine able to replay as sound the representations of sound that were inscribed on cylinder surfaces.

Research by, and connected to, *First Sounds* has been ongoing during the period of this project. In November 2012, with his book *Pictures of Sound: One Thousand Years of Educated Audio: 980–1980*, Feaster applied and extended the new phonautogram reading techniques (Dust-to-Digital, 2012):

Over the past thousand years, countless images have been created to depict sound in forms that theoretically could be “played” just as though they were modern sound recordings. Now, for the first time in history, this compilation uses innovative digital techniques to convert historic “pictures of sound” dating back as far as the Middle Ages directly into meaningful audio. [...]

In short, this isn’t just another collection of historical audio—it redefines what “historical audio” is.

On a purely aesthetic level this reanimating and rewriting of history can be taken as an example proof of the reality-shifting potential of contemporary technologies. Of course it has always been the case that advancements in technology have brought fresh potential for creative expression and thus an expansion of perceived reality. Since the phonautograph, each newly invented recording medium has introduced new ways of working with recorded representations of sound, and so new possibilities for the composition of music.

### 2.1.3 Harmonic interactions

Another invention of the nineteenth-century that is related to the visualisation of sound as a physical phenomena is the harmonograph. Although grounded in the physics of motion, the harmonograph also connects to subjects that relate more to psychoacoustic aspects of sound, and so the discussion here does extend toward that of §2.2 below. In the context of the preceding discussions, the harmonograph may be thought of as a device for drawing temporal traces of Cymatic interactions, where that term is taken in the broad sense of studying wave phenomena; the wave-like motions in a harmonograph are provided by pendulums. The following description of the device is from Antony Ashton's book *Harmonograph: a visual guide to the mathematics of music* (2003, p. 17):

In the simplest version of the harmonograph two pendulums are suspended through holes in a table, swinging at right angles to one another. Projected above the table, the shaft of one pendulum carries a platform with a piece of paper clipped to it, while the shaft of the other pendulum carries an arm with a

pen.

As the pendulums swing, the pen makes a drawing that is the result of their combined motion.

Showing a two pendulum harmonograph on the left, and a 'unison' trace drawn by a system of this type, Figure 2.2 comprises Fig. 1 and Plate V.(B) from *Harmonic Vibrations and Vibration Figures* (Newton, 1909) which Ashton acknowledges as the basis for his 2003 book.

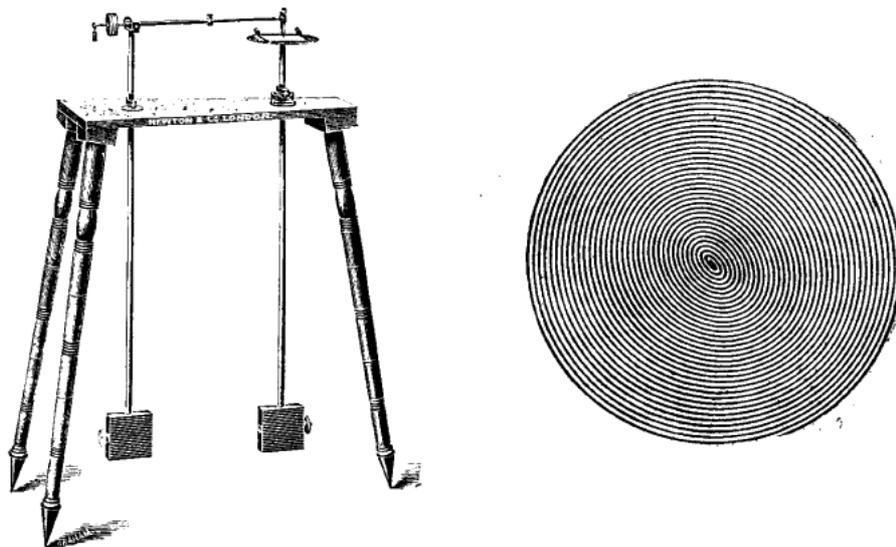


Figure 2.2: Fig. 1 and Plate V.(B) from 'Harmonic Vibrations and Vibration Figures' (Newton, 1909)

Because a pendulum oscillates at a single frequency it is 'a perfect way to represent a musical tone, slowed down by a factor of about a thousand to the level of human visual perception' (Ashton, 2003, p. 16). Changing the position of weights along the length of the pendulum shafts will set how fast they swing.

The simplest harmonograph drawing is produced when both pendulums are the same length [...]. With the pen held off the paper both pendulums are pulled back to their highest points. One is released, followed by the other when the first is at its midpoint. The pen is then lowered onto the paper to produce a circle developing into a single spiral. (*Ibid.*, p. 20)

The spiral is formed in this situation by the decaying amplitude of the pendulum oscillations over time, while the initial circularity can be understood mathematically by the plotting of sine and cosine functions on the two dimensions of a cartesian plane: a reflection of the perpendicular pendulums set in motion with the aforementioned phase difference. If the phase relationship is

shifted, by reducing the time waited before the second pendulum is released, then the circle becomes an oval and if they are 'released together the result will be a straight diagonal line across the paper' (*ibid.*). The traces become more interesting, however, when the lengths of the pendulums are not producing 'unison' frequencies of oscillation; distinct geometric patterns are drawn by the harmonograph when the ratio of the pendulum frequencies is corresponding to that of octave, fifth, or fourth intervals (ratios of 1:2, 2:3, or 3:4) or to that of an overtone (such as 1:3), and so on, whereas non-harmonic intervals yield less orderly patterns (harmonograms).

A source of pleasing variety in harmonograph drawings comes from small departures from perfect harmonies. This seems to involve a principle widespread in nature as well as in the work of many artists. There is a particular charm in the near miss.

An example from music suggests itself here. When two notes are sounded in near unison, the slight difference in their frequencies can often add richness or character to the sound. [...] Eventually [a series of harmonograph drawings in which the length of one pendulum is gradually shortened] fades into a scribble that is a fair representation of discord [...].

For most people this fading of visual harmony occurs at about the same point as the audible harmonies fade. (*Ibid.*, p. 22)

Charles E. Benham, in Newton (1909, p. 37), is careful to point out that 'the analogy of eye and ear must not be pushed too far, and it is rather misleading to call the harmonograph figures "music made visible," as they have been styled sometimes'. Benham uses the ratio of 5:7 as an example of a harmony which is 'pleasing to the eye' but has 'no musical equivalent chord' (*ibid.*, p.36). Whilst Benham suggests caution in connecting the principles demonstrated by the harmonograph to aspects of music itself, I find that the premise for that caution to be flawed. Although Benham's reasoning is rational, it seems – from a contemporary perspective – rather overstated: not least that the given examples of interval equivalence (see Figure 2.3) neglect the fact that equal temperament (ET) is not true to the harmonic proportions being described. The interval of a fifth in ET (an 'equal fifth') is measured as 700 cents<sup>9</sup> whereas – reading from the Intervals appendix of (Benson, 2007, p. 370) – the 3:2 interval ratio, which is the Just and Pythagorean

<sup>9</sup> "We now explain the system of *cents*, first introduced by Alexander Ellis around 1875, for measuring frequency ratios. This is the system most often employed in the modern literature. This is a logarithmic scale in which there are 1200 cents to the octave. [...] on the modern equal tempered scale [...] each semitone is 100 cents." (Benson, 2007, p. 166)

(perfect) fifth, is measured as 701.955 cents.

Ratio of period.		Chord in music.		Component musical notes.
Equal	...	Unison	...	C C
5 to 6	...	Minor third	...	C E $\flat$
4 to 5	...	Major third	...	C E
3 to 4	...	Major fourth	...	C F
5 to 7	...	No equivalent in the diatonic scale		
2 to 3	...	Major fifth	...	C G
3 to 5	...	Major sixth	...	C A
1 to 2	...	Octave	...	C c
1 to 3	...	Perfect twelfth	...	C g
1 to 4	...	Double octave	...	C c'

Figure 2.3: Benham's questionable table of harmonic intervals (Newton, 1909, p.36)

As with Chladni figures, the principles at play in the harmonograph are related to the physical nature of reality. Just as the compositional exploration of Cymatic practices in Lucier's *Queen of the South* is able to captivate the imagination of an audience, and suggest associations with 'the mysteries of the laws of the cosmos',<sup>10</sup> so too may the harmonograph system have compositional value. In support of this notion, consider the following account from Ed Mattson. While it is more common for musicking to be encountered as having the affect of entrainment on a group of people – the synchronisation of human body moving to a musical pulse – Mattson describes a similar collective behaviour based on visual engagement with the harmonograph. Having designed and built a harmonograph, in Bourke (2008) Mattson writes:

I demonstrated the harmonograph at two of the annual shows put on by the Guild of Oregon Woodworkers but was not prepared for the response! It drew huge crowds and gasps of wonder as it did its magical thing. [...] It was not just the drawing itself but the harmonious and mesmerising interaction of the parts of the machine that fascinated people as it swung away and drew the curves. The crowds were literally and unconsciously swaying in unison as they stood watching.

<sup>10</sup> as Johnson put it, see quoted in §2.1.1

## 2.2 Looking at psychological aspects

Whereas the physical motions of sounding objects may give rise to visualisation by virtue of mechanical interaction between materials, the psychoacoustic aspects of sound, being as they are constructed within the mind, require abstract systems of symbolic representation to bring them to the visual domain. Once in the visual domain, any representation of auditory domain attributes must also undergo the physiological and psychological processes of visual perception in order to be comprehended by the human observer. Within such discussion one may venture a question toward what extent perceptions, of either domain, are directly resultant of human physiology compared the extent by which cultural conditioning influences interpretation of physical (light or sound) stimuli; the subject of synaesthesia between sound and visual perceptions may also spring to mind, but it is beyond the scope of this project to investigate these factors. It is, nonetheless, useful to include some context with regard to a few of the differences and similarities found between perceptions in these two sensory domains.

### 2.2.1 Morphophoric mediums and (in)dispensability

The following points of reference from Roger Shepard (1999) serve here as a contextualising primer to the exploration of strategies for the representation of musical pitch and organised time in the creation of portfolio works.

Shepard describes that both pitch and time, like visual space, are 'morphophoric mediums'

(p. 154, italics removed):

In their 1971 paper Attneave and Olson made the point that there is something fundamental about pitch, as there is about time. There are many other attributes to the organisation of sound – such as timbre, spatial location, and loudness – but the attributes of pitch and time have special importance. Attneave and Olson called pitch a morphophoric medium, meaning that it is a medium capable of bearing forms.

Visual space is also a morphophoric medium. For example, if a right triangle is presented in space, the triangle can be moved around and still be recognised as the same triangle. The medium of space is therefore capable of bearing a form that preserves its identity under transformation. Similarly, pitch patterns like simple melodies and harmonies can be moved up and down in pitch and still be recognized by musicians as being the same pattern. [...]

Time is a powerful morphophoric medium. Other dimensions used in music (such as loudness timbre and spatial location) are not morphophoric. [...]

Shepard continues with examples that contrast the domains of visual and auditory perception in terms of cognitive attributes that are either dispensable or indispensable (pp. 155–157). This concept of the '(in)dispensability of attributes' can be illustrated by experiment, as is summarised:

In the visual domain a pair of projectors are used to project spots of coloured light that have two main attributes: colour and location.<sup>11</sup> Loudspeakers, taken as the auditory counterparts to the projectors, are used to produce simple (test tone) sound objects, such that the main attributes in the auditory domain are pitch and location.<sup>12</sup>

First, in the visual domain, a red spot and a green spot are projected side by side, and two discrete objects are perceptible by the human observer. If the two projectors are moved so that the spots of light overlap – thus dispensing of the location attribute – then the human will see a single object: a yellow spot of light. Location is indispensable in visual perception because the two spots of light cannot be seen as separate without that attribute. For the auditory counterpart to this first part of the experiment, middle C is sounded in the left speaker while the right speaker plays E above middle C, and the two different tones are heard coming from the two sides. To dispense of the location attribute, the centre speaker is used to play the same two tones together; the human is able to correctly perceive the two pitches in the sound, and so the location attribute can be said to be dispensable in auditory perception.

Next, the two spots of light are again projected side by side, but this time both spots will be yellow, thus to dispense with the attribute of colour. For the auditory domain equivalent in the experiment, the attribute of pitch is dispensed while maintaining that of location by using both left and right speakers to each play a tone of pitch D above middle C. While the human observer

---

11 Different colours to those described by Shepard are used in this re-telling of the situation. The size and shape of the projected spots are to be considered equal regardless of projector angle etc.

12 For this experiment, imagine the front three speakers of a standard 5.1 surround sound configuration.

continues to see the two yellow spots as discrete objects (thus colour is dispensable), only one sound object would be perceived as being located between the two speakers (and thus pitch is indispensable).

One might think that pitch in audition is analogous to color in vision because both are frequency-related phenomena. However, the experiment using tones and speakers shows that the two are not analogous, because pitch is indispensable and color is dispensable. [...] The indispensability of space in vision and of pitch in audition are parallel to both of those attributes being morphophoric media. So the analog to visual space is not auditory space but auditory pitch. (Ibid., §13.7)

## 2.2.2 Paradox and pitch perception

One finds paradox at various juncture in the unpacking of concepts within this research, and this has been a source of both inspiration and frustration to the creative process. Continuing the theme of contrasting visual and auditory perceptions, and moving towards the discussion of pitch- and frequency-space representations – while also maintaining a connection to the works of Roger Shepard – here are the words that open David Benson's section on 'musical paradoxes' in *Music: A Mathematical Offering* (2007, p. 158):

One of the most famous paradoxes of musical perception was discovered by R. N. Shepard, and goes under the name of the Shepard scale. Listening to the Shepard scale, one has the impression of an ever-ascending scale where the end joins up with the beginning

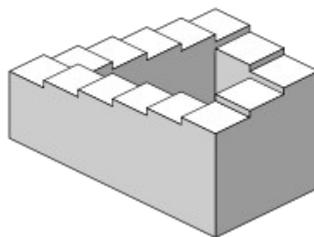


Figure 2.4: Penrose stairs  
optical illusion

The Shepard scale illusion is a 'demonstration of pitch circularity'.<sup>13</sup> 'Pitch is the subjective

<sup>13</sup> ASA present Auditory Demonstrations with links to sound-files; 'The first is a discrete scale of Roger N. Shepard, the second is a continuous scale of Jean-Claude Risset.' (Anon, 1995)

[psychoacoustic] variable corresponding most closely to the objective [physical] variable frequency' (Loy, 2006, p. 158), and the Shepard scale auditory illusion is achieved by playing on the human perception of octaves within the pitch attribute of sound.

While Benson uses the visual analogy of an ever-ascending staircase (Penrose stairs, see Figure 2.4<sup>14</sup>), Roger Shepard himself (in Cook, 1999, p. 158) compares the effect to the upward spiralling of a barber pole in connection to the visualisation of pitch values on a helix; Shepard explains the significance of a helical visualisation as a psychologically founded representation of pitch (*ibid.*, p. 157):

By examining the abilities of subjects to perceive and to produce pitch patterns under transformation, an appropriate representation of pitch can be found. [It has been] found that this representation corresponds to the log frequency scale, which is nice in that it preserves [...] any interval under transformation, such as any arbitrary microtonal relationship. The log scale does not, however, represent the fact that certain intervals are special, such as octaves and perfect fifths. Ethnomusicological studies have found that octaves and perfect fifths seem to be culturally universal; that is, although musical systems and scales of all cultures differ in many respects, they almost always contain an octave relationship, and often the perfect fifth as well. [...]

The German physicist Moritz Drobisch (1855) proposed that tones be represented on a helix. This helix, on the surface of a cylinder, places octaves immediately above and below each other.

As a three-dimensional model the Drobisch type of helix readily manifests the pitch-class and octave-level aspects of perceived pitch in the visual domain.<sup>15</sup> Shepard goes on to describe how extensions to the concept, employing double helix, toroid, and helical cylinder forms, may provide visualisation of the special relationships between both octave and fifth intervals. However, given the two-dimensionality of both paper and screen – which are where my compositional practices transpire – the three-, four-, and five-dimensional constructs described by Shepard seem impracticable. Of course it is perfectly commonplace for three-dimensional objects to be represented on flat surfaces, but, nevertheless, it was decided early on in this project that the visual representations to be explored would all be two-dimensional constructs that may suggest, but do not require, higher dimensions. By working only on the plane, conflicts of perception and problems

14 Public domain image, online at [http://commons.wikimedia.org/wiki/File:Impossible\\_staircase.svg](http://commons.wikimedia.org/wiki/File:Impossible_staircase.svg) (accessed 20130310)

15 Aside it is noted that the date of origin for the Drobisch type of helix has been cited as mid-nineteenth-century.

related to perspective and judgement of adjacency between points of a viewed object – the very things upon which the paradoxical illusion of Penrose stairs predicate – can be avoided.

Returning to Shepard describing the auditory demonstration of circularity in pitch perception (*ibid.*, p. 158):

The inspiration for generating the circular Shepard tones came from the fact that Max Mathews at Bell Labs had created the first program for generating sounds using the computer. This aroused great excitement, and the author did an experiment using a subdivision of the octave into 10 equal steps instead of the traditional 12. A note's position within the octave is called its *Chroma*, and the *chroma circle* is the base of the helix [described above]. The tones were formed by using a symmetrical structure, so that after an octave had been traversed, the ending tone was identical to the beginning tone.

It is, perhaps, worthy of note that computer music software, which was a new thing at that time, played an important role in the creative practice of Shepard's research. A second observation here is that many of my own works have, both prior to and since reading the above quoted, utilised equal division of the octave in to a non-duodecimal number of steps.<sup>16</sup> There is, however, a more pressing motivation for the inclusion of the above quotation, and that is to question this use of the word 'chroma':

Utilization of that word in this context is standard (see, for example, Mauch et al., 2009; Bertin-Mahieux et al., 2010; Sumi et al., 2012), and it has been attributed to Geza Révész<sup>17</sup> (Loy, 2006, p. 163):

Révész (1954) developed a two-component theory of tone, suggesting that there are at least two principal interlocking structures in pitch [perception] which he called tone height [and] chroma

Loy also writes that in developing the demonstration of pitch circularity, 'Shepard (1964) wanted to test Révész's theory' (*ibid.*, p. 167). It seems, nevertheless, that using a colour-rooted word to label an attribute of pitch within a spatial representation is – if not paradoxical, then at least – contradictory to the consideration of the (in)dispensability of colour compared to pitch (as outlined in §2.2.1). To remove the association of pitch perception to conceptions of colour, the term

<sup>16</sup> See for example the nRadii work (§4.4.1), and in some of my 2010 web audio work: <http://sdfhtml5.blogspot.co.uk/>

<sup>17</sup> Geza Révész is author of *Introduction to the Psychology of Music* which was published 1954 as an English translation of *Einührung in die Musikpsychologie* published 1946.

'pitch-class' is preferred for describing position within the octave; thus, pitch-class-circle is written instead of 'chroma circle'; the way that this appears more cumbersome to write is, then, accepted as a paradoxical quirk. The pitch-class-circle is addressed as an aspect of the spiroid-frequency-space (§6).

## 2.3 Looking at looking at sound

This chapter has identified ways in which sound may manifest visual patterns on the surface of a physical system. In Chladni/Jenny type Cymatics the acoustic interactions of physical matter give rise to formations and shapes that favour resonance within the system. To the human eye the manifestation of these shapes is perceived as an instantaneous reaction to the sound that is happening 'now'; oscillations within this type of system are of frequencies that are perceived as sound and so the eye may only perceive the patterns resultant of their periodicity. In the harmonograph sub-sonic oscillations are traced over relatively long periods of time to reveal, at a rate comprehensible to the eye, the shapes of what are, in principle, acoustic interactions.

Harmonograms and Cymatic patterns can bring only specific harmonic content to the visual domain, providing representations of the motion that has occurred within a two- or three-dimensional space. Although manifest over time, the passage of time is not itself represented in these cases. In the phonautograph and related systems the passage of time is represented as one of the visual dimensions allowing representations of arbitrary sound to be written.

Some psychological aspects of perception have also been introduced. Concepts pertaining both to psychoacoustic and visual perception – as well as to the physical phenomena discussed above – are engaged by this project as inspiration in the creation of, and composition with, computer music software systems.

## 3: In software, on screen

Formative works and study pieces from the early stages of the project are discussed in this chapter; each section describes a different piece and its context, with the final section (§3.7) concluding the contextualisation. Connections to the concepts introduced in the previous chapter are demonstrated, just as later works will be shown to connect also with the concepts introduced, explored, and developed in the works here described.

Having chosen to compose music through software-based systems while questioning the possible implications of the visual aspects with such systems, particular attention has been given to thinking about the 'and how' part of the question: what aspects of sound are being shown in software, and how?

Visual representations of sound appear within software as part of the graphical user interface (GUI) on screen. A computer screen is the medium by which a human may observe the virtual worlds that are taken to exist within the computer. It is on, and within, such virtual worlds that this project is focussed, and this includes the pursuit of creating new worlds within worlds through computer programming (though it is rather more typical to use the term environments when referring to these software worlds).

Most of the programming undertaken by this project has been within the environment of MaxMSP (Cycling '74, n.d.). My coding background includes programming with C and C++, and although these languages were not used during this project, awareness remains of a C way of thinking, and that that is underlying the MaxMSP environment. MaxMSP provides high-level constructs that allow for rapid implementation of ideas in code, especially when those ideas have strongly visual elements. With the additional use of Jitter and JavaScript within that environment, a

slightly lower-level of access to (and control of) data, and programming styles other than that of pure data-flow are incorporated into my works. During the project, some works have also involved writing JavaScript for use with HTML5 (which has emerged during the period of this project) within in the environment of a web browser. Similarly, the Adobe Flash environment has been utilised for its widespread availability to potential audiences of interactive works (Adobe, 2009).

## 3.1 Pixels

For the 2009 Huddersfield Contemporary Music Festival (HCMF) I collaborated with Duncan Chapman as part of *The Music of Electricity* (MoE) learning and participation programme. The project was connected to the *Raw Material* exhibition<sup>18</sup> of recent works by Tim Head. Chapman conducted a series of workshops with A-level students and Year 6 pupils, culminating in a performance<sup>19</sup> during the festival. I was commissioned to produce an 'interactive sound installation' for the project: a software-based system through which sounds recorded during the workshop sessions could be heard online. At first hesitant to take on the work, a decisive factor in accepting the commission came from reading texts by, and about, Tim Head which appealed to me as both a creator and an appreciator of things:

Tim Head's work is about instability and uncertainty: of images, of perception and of the individual's relationship with the wider world. [...] His work might be characterised as a search for visual equivalents for the tension between what we perceive to be the truth and what we know to be the truth. (Tufnell, 2003)

Speculation on the elusive and contrary nature of the digital medium and its unsettled relationship with the physical world and with ourselves forms the basis for recent work. [...]

By operating at the primary one to one scale of the medium's smallest visual element (the pixel or inkjet dot) and by treating each element as a separate individual entity the medium's conventional picture making role is bypassed. You are no longer looking at a representation of an imported image in virtual space but are looking directly at the raw grain of the digital medium itself. The medium is no longer transparent but opaque. The slippery weightless world of the virtual is edged out into the physical world. (Head, 2008)

18 At Huddersfield Art Gallery from Saturday 21 November 2009 until Saturday 9 January 2010.

19 26 November 2009, at Huddersfield Art Gallery.

In particular, the idea of 'treating each [pixel] as a separate individual entity' is something that had emerged in my own programming at Masters level, and which has continued since to feature in my approach to the software medium. I eventually gave the name *pixels* to the MoE interactive installation which resides online at <http://www.musicofelectricity.net/pixels>.

### 3.1.1 The sound of electricity

Participants of the workshops were introduced to the so-called twitching-speaker way of making sound that is described in *Handmade electronic music: the art of hardware hacking* (Collins, 2006, p. 20) as:

a beautiful electric instrument, evoking the spirit of nineteenth-century electrical experimentation (think twitching frogs legs and early telephones) out of nothing more than a speaker, some batteries, wire, and scrap metal.

When the positive and negative terminals of a (usually 9 volt) battery are connected to the terminals of a speaker, the electromagnetic interaction within the circuit causes the speaker-cone to be pushed forward (or pulled backward if the polarity is inverted). One of the connections in the simple circuit is left unfixed so that it can be carefully closed by hand to cause the speaker to twitch when the connection is made; a variety of buzzing noises can be achieved by finding positions in which the electrical connection fluctuates, and further modifications to the sound of a twitching-speaker are possible, for example by placing loose objects upon the speaker-cone. Participants recorded and edited a variety of sounds that were made in this way.

Lo-fi electronics such as the twitching-speaker are a feature of my own performance practice. Whereas the decision was made to focus exclusively on software-based systems in my phd composition, I have been interested to bring concepts of that practice into the digital domain.

### 3.1.2 System architecture

From a technical perspective the system that I designed for the MoE *pixels* installation comprises

two main parts: (1) the front end is web based and has an interactive audiovisual element created with ActionScript3 (AS3) in Flash; and (2) at the back end of the system is a maxpat<sup>20</sup> that generates XML entries that are used by the Flash element of the system via drag-and-drop of folders containing sound files. The system was constructed prior to the workshops (using a dummy set of sounds from my own archive). Chapman wanted the workshop participants to categorise the recorded sounds into different types, according to their own interpretation of the material; I suggested the use of six category groups that could then be visually identified using distinct colours on screen. Each pixel of a standard computer screen comprises red, green, and blue elements (RGB), and all available colours are produced in each pixel on screen by setting the levels of these elements. The six colours used in the MoE *pixels* are those made of either two (in equal measure), or just one of the three base elements, as shown in the following table:

<b>Name:</b>	red	yellow	green	aqua	blue	magenta
<b>RGB mask:</b>	1 0 0	1 1 0	0 1 0	0 1 1	0 0 1	1 0 1

Six folders are named to correspond with the six colour names, both on the local machine where the sound-files are edited and categorised, and then on the web hosting which is accessible via FTP connection. The original plan was to use the **jit.uddl** object to handle the uploading of all the sound-files and the generated XML file to the hosting, all from within a maxpat. That aspect of the design was dropped because of issues relating to Jitter objects causing problems within the MaxMSP Runtime on the Windows OS (which is how the back end of the system was to run, while its development was on the Mac OS). The workaround for this problem was to use separate FTP software for the uploads, which meant less automation of the process but more stability.

When the MoE *pixels* webpage is accessed by the public audience, the sound-files are loaded in Flash, one at a time, using information from an XML file which includes the filename and

---

<sup>20</sup> Definition: maxpat (noun) a patch made in MaxMSP and saved with the '.maxpat' file extension.

designated colour of each sound-file. Upon successful completion of each load, a visual representation of the sound-file is created on screen: a random number is multiplied by the appropriate 'RGB mask' (as in the table above); the resultant colour is given to a square that is placed at randomly selected cartesian coordinates which conform to a 20-by-20 grid upon the Flash stage; the square itself is the size of one such grid cell. When the mouse-point passes over one of these coloured squares, the associated sound-file is played and the location of the square is changed, again to a randomly selected cell of the grid. The location selection algorithm is designed so that only an 'unoccupied' cell is given as the new location for a square. After all the sound-files have been loaded, the webpage will look something like is seen in Figure 3.1:

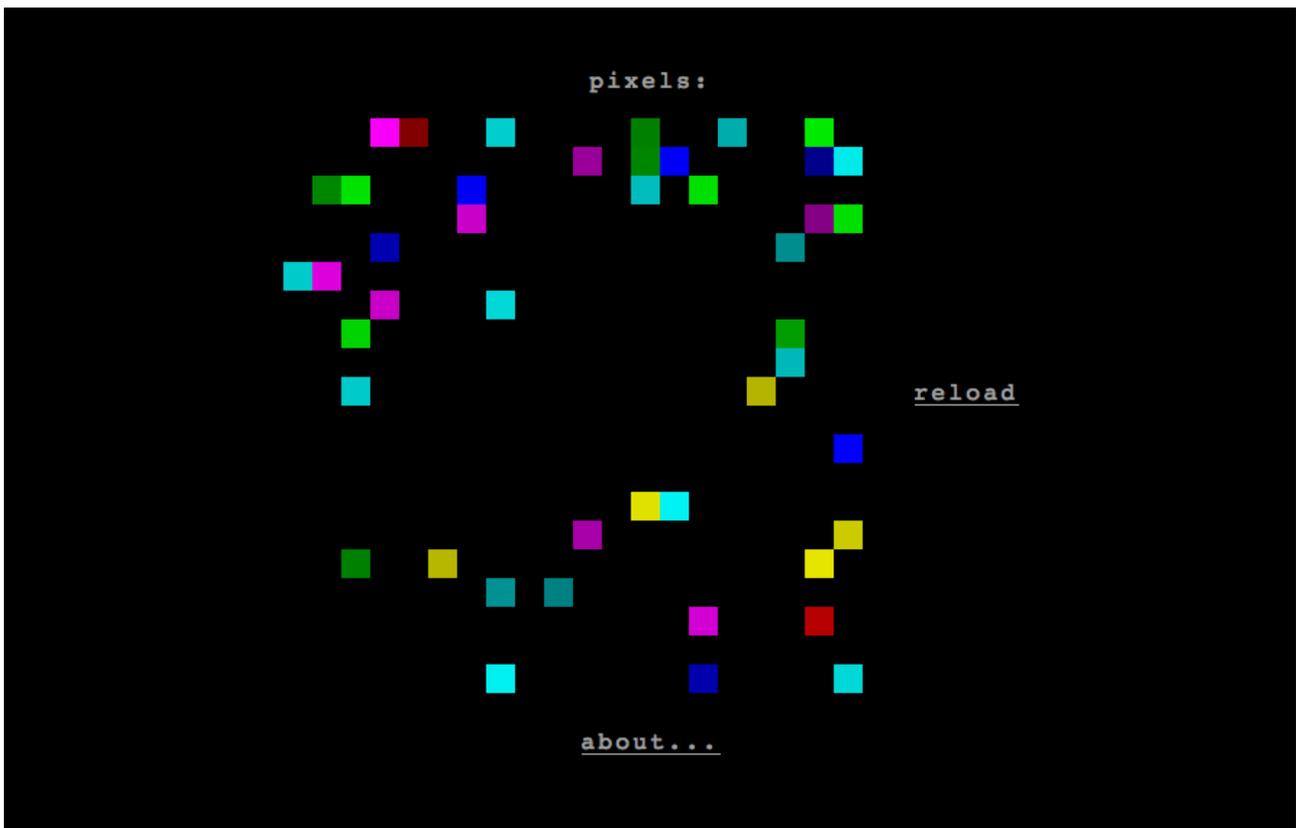


Figure 3.1: MoE Pixels

It was found that in some situations the mouse-point can be moved over a square without the 'play sound and move' behaviour being triggered. Although my initial reaction to this was to

wonder where I'd gone wrong in the AS3 programming, it was soon embraced as feature, rather than a bug, and the 'mouse down' (md) and 'mouse up' (mu) functions were added to the code in order to access the expected movement and sound. There is also another, less often encountered, bug by which a square seems neither able to move nor play its sound; to address this the md function was further modified to also trigger any one randomly selected sound-file from those loaded, thus avoiding a situation in which nothing at all happens in response to the mouse input.

Adapting the code of a program, during its development, to embrace unexpected behaviours as features of the software is a commonly employed strategy in my practice. In many forms of creative art, it is often the 'happy accidents', as Bob Ross put it,<sup>21</sup> that are most compelling and that enable a work to grow beyond its original conception.

### 3.1.3 Audience interaction

None of the above description of the software is provided to the audience of the MoE *pixels* work. When the work is accessed (see Figure 3.1), only three words appear on screen: the title (punctuated with a colon) at the top; a hyperlink to 'reload' the webpage (thus resetting/re-randomising the Flash element) at the side; and, at the bottom, the word 'about...' hyperlinks to general information about the HCMF MoE project (text similar to that given at the start of §3.1). In withholding instruction and description of the work, the aim is to engage the imagination of the audience who are invited (by implication) to explore the screen and discover the sounds. The specific connections that I have made between concepts, sound-files and visual representations may or may not be identified, and it is of no consequence if they are or not. There is no 'correct' way to interact with the system and so no one (who can see and/or hear, and is able to control a mouse-point on screen) ought to feel excluded from participation. Talking to Duncan Chapman and Heidi Johnson (of the HCMF) about the work, during its development, I would often use the term 'Flash toy', but the word toy is in no

---

21 See for example [http://www.youtube.com/watch?v=\\_Tq5vXk0wTk](http://www.youtube.com/watch?v=_Tq5vXk0wTk) [accessed 20130507], or as quoted by Bennett (2005, p.48).

way meant to belittle the work; the most serious and sophisticated electric instruments and softwares can be equally referred to as such, the emphasis being that the playing of/with them is fun.

Correspondence from one visitor to the work included a snapshot of their screen showing 'the fruits of [their] labour' while 'enjoying [the online installation]' – an outcome that must have taken quite some time to achieve, testifying to the success the non-direction approach towards the motivation of individual creativity when interacting with the MoE *pixels* – see Figure 3.2 (Baldwin, personal communication by email, 2009):

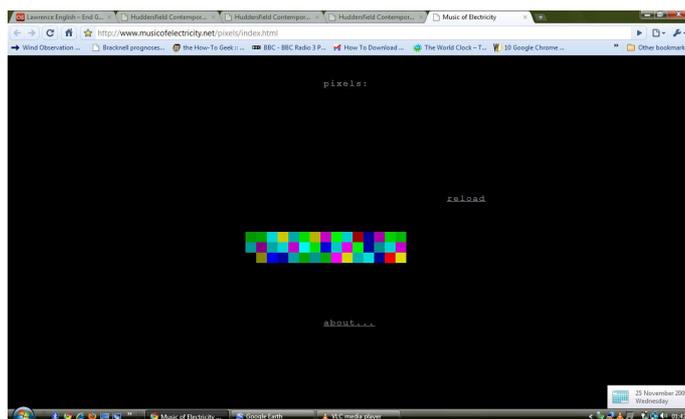


Figure 3.2: MoE Pixels, image from Baldwin (2009)

Another use of the system that was unexpected during its development was instigated by Chapman who suggested adaption of the system as a software instrument for laptop ensemble. A number of non-internet-connected laptops each had a copy of the MoE *pixels* system (both its front and back end parts) running locally, and A-level student workshop participants created their own sets of organised sounds to load in and play via *pixels* during the Gallery performance. Under Chapman's direction the students devised an indeterminate score for the performance which involved dice marked with the six colours.

One of the great appeals of creating software-based works is the way that the software medium lends itself to re-appropriation. It is also of interest to observe that audience experience of software-based artworks tend to involve some sort of participation beyond observation. In the anticipated interaction with *pixels*, the audience are the online visitors who become (perhaps unwitting) participants in an improvised musical experience; free interpretation and self-determined duration and direction exist within the infinite but bounded possibilities of the piece. In the described case of *pixels*-as-instrument, the laptop-performers (by being aware of the underlying software architecture, and interacting also with the backend maxpat element) became a sort of 'super-user' audience to my work; the performance piece that they then created was entirely their own and had its own audience and aesthetics.

### 3.1.4 Inherent paradox

While engaging, to some extent, with the aesthetics of the *Raw Material* exhibition, *pixels* is knowingly abstracted from the true nature of the computer screen medium. Presentation of the work offers the suggestion that each coloured square is a magnified representation of a computer screen pixel (as suggested by the colon after the title above the Flash stage; see Figure 3.1, page 32). Under critical examination, however, the interactive behaviour of entity relocation exhibited in the work betrays that association. A computer screen consists of many pixels which, if one brings one's eye close enough to the screen, can be seen as individual squares that are arranged in the manner of a grid or matrix.<sup>22</sup> The apparent appearance of visual forms on screen is achieved by the different colouration of many pixels across the surface; human perception is generally of the represented form(s), rather than of the picture's elements (pixels) within the medium.

In *pixels*, when a coloured square is seen to disappear from under the mouse-point at the same time as a square of that same colour appears in another location, the perception is

---

<sup>22</sup> Not all pixels are square, but I have yet to work with a computer screen on which they are not.

(potentially/probably/intended to be) of a single entity that has relocated; this identification is supported in the auditory domain by the fact that the then appeared square would go on to trigger the very same sound as before should it be 'moused' again. These squares are, indeed, programmed into the system as 'sprite' entities, so that from the software perspective it really is the case that the same representation of a square is having its position on screen changed. It is thus that the suggested association of those squares as pixels is undermined: in the physical medium of a computer screen the pixels do not move, they only change colour.

One is led to consider that visual representation in software, on screen, is illusionary in every instance. 'There is no spoon' comes a popular refrain from the (Wachowski & Wachowski, 1999) film *The Matrix*, and as noted early in this thesis (§1.2.1) there is no sound on screen: only representations of it. Further, then, one may concede that even the representations of sound that appear on screen are only percepts stimulated in the eye of the beholder by the matrix of pixels that comprise the surface of a screen.

There are different approaches to GUI programming that software may take. Flash employs a paradigm of vector based graphics in which shapes are described in code as shapes with attributes such as size, position, and colour; the programmer need not think about the actual pixels that will be used produce the described shapes on screen, and there are many advantages to this approach. There is, however, an aesthetic appeal – as expressed by Head in the *Raw Material* exhibition – to the direct control of individual pixels, and many of my works incorporate an approach of matrix cell value manipulation.

### 3.1.5 Towards programming with matrices

One seldom thinks of the individual pixels on screen, of how each pixel is changing colour at its fixed position, but rather one accepts as concrete the visual forms that appear upon the display.

Similarly, one seldom thinks in terms of the additive RGB components that create a colour that is

seen on screen. By conceptualising visual representations in software from the ground up, this project has sought to nurture a sense of connection to that aspect of the software medium, as in the type of connection one may expect of a composer to the sounds with which they compose. The influence on this project of Tim Head's aesthetic can perhaps be identified in this sentiment, but the motivation of my exploration in this area is to appreciate the nature of the screen medium as the material basis for visual representations of sound in the software medium.

Data matrices that can correspond directly to the actual pixels on screen are a conceptual starting point from which are programmed software systems that set matrix cell values in order to manifest visual forms on screen. The RGB channels of the GUI domain can be thought of as discrete layers. At a coding level, the three values that describe the colour of a pixel in an image are often accompanied by a fourth value, known as alpha, that is used to describe transparency (or opacity). When working with Jitter matrices in MaxMSP the arrangement of those four layers follows the ARGB format in four planes; the 'char' data type is used because its 8-bit size is sufficient for the numeric values that comprise the four layers of each cell in a grid matrix that is to be displayed as an image on screen. For example, a Jitter matrix that is given the name 'm', and the horizontal and vertical dimensions that match the classic VGA standard,<sup>23</sup> would be declared in the code of a maxpat as **jit.matrix m 4 char 640 480**. 'The planes are numbered from 0 to 3, so the alpha channel is in plane 0, and the RGB channels are in planes 1, 2, and 3.'<sup>24</sup> This project has been mostly interested in working with just the RGB planes from which the GUI is manifest on screen; Awareness of the alpha plane is necessary in programming, but it is considered ancillary to the visual manifestation.

---

23 The VGA standard resolution is commonly known to be 640 (wide) by 480 pixels; see for example (Christensson, n.d.)

24 From 'What is a Matrix?' in *Max 5 Help and Documentation* accessed within MaxMSP, but also online at <http://www.cycling74.com/docs/max5/tutorials/jit-tut/jitterwhatisamatrix.html>

## 3.2 Visyn

### 3.2.1 Introducing visyn

*Visyn* began as an exercise in generating visual patterns on the computer screen; it is an ingenuous exploration of 'visual synthesis' founded on the idea of applying techniques known from audio synthesis and musical composition to a graphical domain. The patterns were synthesised as purely abstract visual manifestations of the underlying processes without the role of representing any other thing. At various times this piece has been disregarded as a dead-end, only to be re-remembered and revisited with fresh perspectives at later stages of the project. The final decision to include this work in the portfolio is based on the aesthetic principles of design exhibited in the piece, which with retrospect, are identified as significant to the project as a whole. While the *pixels* piece (above) brought more clear critical engagement with some of the aesthetics involved, *visyn* – which predates *pixels* by almost a year – shows that my fondness of matrix cell data manipulation and display had already begun to emerge in my practice.

The final version of the work is [visyn\\_10e.maxpat](#) (illustrated in Figure 3.3 below), and opening that maxpat will suffice for the following discussion of this formative study. It may be observed that, in the 'visyn' directory of the portfolio, there are a number of maxpat files that all carry the 'visyn\_' name with appended version numbers; these allow development of the work to be traced.<sup>25</sup> As with all of my works which are version numbered in this sort of way, only one version should be open in Max at any given time; this is, among other possible reasons, so that the commonly named **send**, **receive**, **value** and **jit.matrix** objects etc do not crosstalk between versions (which might lead to undesired results). Looking at the creation dates of the files, the oldest in the visyn directory is found to be [patternSynth.maxpat](#); this first draft name that was superseded illustrates something more of the seed of the idea behind this work.

---

<sup>25</sup> A detailed account of the development through seven incremental stages of the *visyn* system is presented in [sdfphd\\_appendix\\_A\\_visyn.pdf](#), also found within the *visyn* folder of the portfolio directories.

### 3.2.2 Looking at visyn

At the heart of the *visyn* system is an engine which outputs five continuously changing signals; these are produced by sinusoidal oscillators whose frequencies keep changing to randomised values in the range of  $\pm 1$ Hz. Three of the output signals are labeled R, G, and B and have a value range of 0 to 1; the other two, labeled X and Y, have a bipolar  $\pm 1$  value range. These bipolar signals are presented as inputs to the signal routing matrix GUI controllers, of which four can be seen, mid-right of the fullscreen maxpat, in Figure 3.3; each of these GUI controlled mapping units is configured to create a continuous path trace on the surface of the data matrix that is displaying on the left of the screen. When two or more traces are active, they all draw with the same set of RGB colour values that are changing over time.

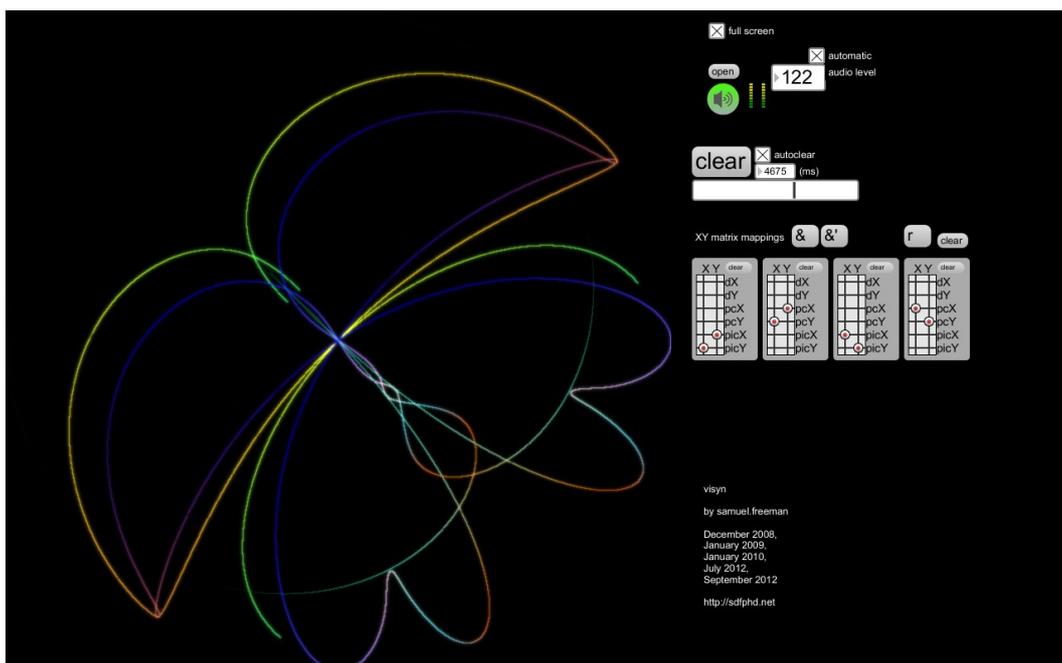


Figure 3.3: visyn\_10e

When the patch is loaded, configurations of the 'XY matrix mappings' default to a preset that can be recalled again by clicking the `&` message on screen; the `&` message triggers a similar mapping that has all four, instead of just two, of the available traces active; the clickable `r` (to randomise) and `clear` messages apply to all four mapping units. The text labels within the GUI

panels of those mapping units can also be clicked to access preset configurations locally.

Continuous generation of the mapped traces causes a buildup of curves on the display,<sup>26</sup> and it is then more difficult to see the changes as they happen and enjoy new path shapes as they form. A large font clickable message to **clear** the displayed data matrix is thus provided in the area above the 'XY matrix mappings' section of the GUI; this message is spatially grouped with 'autoclear' controls allowing the data surface to be reset at regular intervals.

At times, the synthesised pattern will yield a striking image that one may wish to capture. It would be a simple matter to add a button, or key command, by which the data matrix could be exported as an image file, but I have explicitly chosen not to do that in *visyn*. Just as sound is fleeting and transitory, so too the experience of *visyn* is intended to be rooted in the experiential now.

Above the clear/autoclear section of the *visyn* patch interface is an audio control section comprising an **ezdac~** object (that is green when DSP is on), an **open** message (to access DSP options), a large font-size **number** box (labeled 'audio level'; to set the output amplitude, where zero is mute), and a **toggle** to activate automatic control of that audio level value.

### 3.2.3 Visual data to control sound parameters

The audio processing part of the system in this study piece was developed experimentally, and although it represents an important step in the progression of ideas being explored by the project, this particular direction was short lived in my work. The method explored for creating continuous sound from the visually displayed data – as well as containing some avoidable programming errors that were only noticed in review of the piece for this analysis – is somewhat impracticable in terms of computational efficiency in its software implementation. Nevertheless the audio accompaniment

---

<sup>26</sup> More examples of which can be seen within the figures of [sdfphd\\_appendix\\_A\\_visyn.pdf](#), found within the *visyn* folder of the portfolio directories.

to the visual synthesis presented does achieve aesthetic goals.

Data analysis of the visually displayed Jitter matrix provides control values, based on RGB levels in specific areas of the display, for many voices of a basic sound synthesis engine. A 'radius system' for designating frequency values to spatial positions on the plane – conceptually related to the spiroid-frequency-space (§4) – was designed for the *visyn* system. Twelve radii are specified on the surface, each comprising between 9 and 21 sub-matrix sample areas, and each of which corresponding to a voice of the audio synthesis. A frequency value is assigned to each sample area which determines the frequencies of both a sinusoidal oscillator and a resonant filter. For each sample area, the R and the G values from the data analysis control level of sinusoid in the left and the right channels of stereo output, and the B value sets output level (mono/to both channels) for resonant filtered noise. The audible result is a drone that is highly prone to glitches which are oddly welcome as a rhythmic element to the sound.

### 3.2.4 Ever-changing and always the same

*Visyn* exhibits the early (within this project) exploration of working with a data matrix cell-to-pixel-display paradigm, visually tracing algorithmic motion through a two-dimensional space changing over time. As with most of my work, the *visyn* system was developed by incremental steps, with the additive integration of individually simple ideas, each building upon the last.

The visual aspect in this system is not intended as a representation of anything other than itself, but as such it is open to interpretation by the observer. Although the audio is shaped directly by the data that is seen on screen, the synergy is subtle. When the autoclear feature is active the sound swells in waves proportional to the clearing interval; associative description of the filtered noise content and harmonically related sinusoids in the *visyn* sound may call comparison to such things as, for example, ocean waves and Tibetan singing-bowls for a clichéd but applicable appreciation of the meditative character of the piece. The visual patterns themselves, even without

the addition of sound, seem have a captivating way of retaining attention for prolonged periods.

Although there are GUI controls to alter the manifestations of the system, there remains a sameness to the output; the continuous drawing of paths maintains the character of sinusoidal motion that is within a narrow frequency range that limits the speed but allows for reversal of directions. As implied by the inclusion of the `&` and `&'` presets, I tend to think of the output visuals produced by the polar mapping types as being those that define the identity of the piece. Whereas deformed Lissajous-like curves and other square filling paths are possible it is the circularly bounded patterns that I most enjoy. Without interacting with the system one may observe and ever-changing and always the same flow of forms with hints of a metaphorical reference to Cymatics, as if the sound were bringing the shapes into being, though it really is the other way around: the sound being derived from the visualised data.

The lasting influence of *visyn* on my work may be evident in the following sections, and into the later pieces, such as *sdfs*, there are aesthetic and technical themes that were seeded in the *visyn* study.

### 3.3 Sub synth amp map

The questions that initially stimulated the next study piece had a focus on how digital sound might be made visible on screen; in practice, however, the pertinent questions soon become of how that sound is being made in the context of its visual representation on screen.

To introduce *sub synth amp map* there follows a brief discussion contextualising my work in relation to the practice of live coding. The *sub synth amp map* work is then described with further context. As with *visyn*, this piece demonstrates how aesthetic concerns have been explored and developed towards the formation of the more substantial pieces in the portfolio.

### 3.3.1 Live Coding with MaxMSP

Live coding 'has become so prominent as a practice within computer music' that it (almost) 'needs no introduction' (Thor Magnusson, 2011, p. 610).

My first contact with the idea of live coding (LC) occurred in contrast to the concept of live algorithms for music (LAM) in which I had interest during my MA studies (Freeman, 2007).

Whereas LAM are conceived as independent software entities that may interact musically in an improvised performance setting with separate human players (Blackwell & Young, 2006; Bown et al., 2009), the LC paradigm is of humans creating improvised software structures during a performance. After at first viewing LC as a curiosity, the appeal of its aesthetic has gradually infiltrated my perspectives on computer music and software art. In my ongoing activities as a laptop-performer – leading up to and during this project, independently and as an ensemble performer – I have taken to embracing what I see as LC principles to varying degrees.

In practice, LC performances are framed by various *a priori* factors. Questioning some of these factors may help elucidate the unexpected role of LC in this project. The question I ask first is: at what level of abstraction is a coder coding? One may choose to begin by constructing the algorithms of sound synthesis, or (more typically) one may focus on generative algorithms that trigger and alter available sounds. Relatedly it can be asked: what is the programming environment of choice? Choosing to live code in a particular environment determines many aspects of the performance, and

the difficulties, or the near impossibility, of comparatively studying live coding environments due to the markedly different background of the participants and the diverse musical goals of the live coders themselves

have been discussed by Magnusson (2011, p. 615), who also writes that (p. 611):

Artists choose to work in languages that can deliver what they intend, but the inverse is also true, that during the process of learning a programming environment, it begins to condition how the artist thinks. [...]

Learning a new programming language for musical creation involves personal adaptation on various fronts: the environment has an unfamiliar culture around it, the language has unique characteristics, and the assimilation process might change one's musical goals.

Chapter six (§6) will address some of these issues further in relation to the development of the *sdfs* environment within MaxMSP which is my chosen environment for both live and compositional coding.

Common within, but not necessarily universal to, LC performance practice is to mirror of the performer's computer screen to a projector such that the audience may see what is happening in the software. By showing the screen in this way the audience have opportunity to visually engage with what the live coder is doing. The LC aesthetic thus offers an interesting perspective for this project, and its exploration of visual representation of sound. These are, again, matters to which the thesis will later return; in this chapter it is the use of MaxMSP as an LC environment that provides context to the *sub synth amp map* piece.

Video examples of LC in MaxMSP can be found online.<sup>27</sup> As an LC environment, when starting from an empty patch, MaxMSP lends itself to performances in which an 'instrument' of sorts is first created, and is then – while possibly being further added to or modified – played either with human or algorithmic (or both) control of parameters. Emphasis on the soundmaking aspect is in contrast to an event-making focus in LC environments that host ready-made instruments; Impromptu, for example, is an AudioUnit host (Sorensen and Gardner, 2010). The algorithmic control of traditional CMN-like concepts (such as pitch and rhythm patterns) tend to be prioritised in LC performances with Impromptu, although it is possible to write custom DSP algorithms in that environment. Heeding Magnusson's sentiment that comparative study of environments is near impossible, the preceding comparison is intended only to draw attention to where my aesthetic interests are situated: that is with soundmaking more than event-making. I am also attracted to the

---

<sup>27</sup> For example via <http://livecoding.co.uk/doku.php/videos/maxmsp/about>. My own contribution that collection, comprising sessions recorded between 2009 and 2011, is at <http://livecoding.co.uk/doku.php/videos/maxmsp/sdf>

way that expressing musical ideas through code in real-time embraces nowness, and the general sense of transience that I perceive in LC aesthetics. There is the nowness of the moment within the duration of a performance, where the coder creates a musical macrostructure extemporaneously. Also – and in the context of computer programming as a practice that has existed for just a few centuries – there is a sense of nowness at the supra-scale of time of music (Roads, 2004, p. 3): in the history of human musicking, LC is very much a new phenomenon because it is only in recent decades that computers that are portable enough to carry to performance venues, with real-time audiovisual processing capabilities, have come to bear LC praxis.

### 3.3.2 From passing thoughts to pivotal theories

Work on *sub synth amp map* began as a live coding rehearsal exercise: the inception of this short study piece was as a first experiment toward an approach for the live coding of real-time audio synthesis with visualisations also live coded in MaxMSP. The improvised implementation uses similar visualisation techniques to those found in the *visyn* system, but here the system starts with an algorithm for sound synthesis, and then the output of that is taken both to be heard and seen (for further comparison of the two systems see §3.3.8 below).

From one perspective, the patches that comprise this piece are the result of passing thoughts manifest in code, designed and implemented on-the-fly and motivated by swiftness in generating output. However, given that the initial patch was saved – which is not always the case for live-coded practice pieces that are often allowed to perish as fleeting moments – and that it was later returned to for further development, the patches may together be looked upon as rough sketches towards what may have, but did not (in of itself), become a much larger system. From another point of view these pondering prototypes illustrate a number of significant breakthroughs in my own understanding of the aesthetic principles and compositional processes guiding this project.

The creation of *sub synth amp map* was also, in part, a response to the following line of

questioning: Keeping in mind some of the ways that physical aspects of sound have been seen to manifest as visual forms on various types of surface (§2.1) – and how when certain conditions are met within a physical system, visual manifestation of sound waves appears to be spontaneous and inevitable – are there any equivalently natural, or neutral, systems for manifesting 'digital sound' on screen? Rather than modelling, in software, the physical systems that give rise to visualisation of acoustic phenomena, can methods of representation be found that are somehow native to the screen? If so then such natively inevitable algorithms might well be approachable within LC contexts, for surely they would require little coaxing to manifest. Any methodology emergent in this vein is, of course, bound to be dependant upon the programming environment in which one is working. The visual direction of this piece was greatly influenced by the thought patters that *visyn* had previously put into practice.

### 3.3.3 Names and numbers

Files associated with this study piece are located in the 'sub\_synth\_amp\_map' folder of the portfolio directories. Discussion of this work<sup>28</sup> is to be read in view of [sub\\_synth\\_amp\\_map.maxpat](#) which is an amalgamation of various patches that had each taken a slightly different approach to the same material. Within that folder it will also be seen that many of the maxpat and outputted audiovisual files carry in their filename the date on which this study piece was started (20100221). The naming of things by their date of creation is favoured because, to give one reason, it provides real-world meta-context to the content. It is the YYYYMMDD format<sup>29</sup> that is used, and this is in contrast to the DDMMYYYY format employed, for example, by James Saunders in his modular composition *#[unassigned]* (Saunders, 2003).

In the case of this particular piece, edits and outputs that were made when returning to the

---

28 Both here and in [sdfphd\\_appendix\\_B\\_subsynthampmap.pdf](#), found within this work's folder of the portfolio.

29 As described by International Standard ISO 8601, see for example <http://www.cl.cam.ac.uk/~mgk25/iso-time.html> or <http://xkcd.com/1179>

work at a later date have retained the numeric name given to the original patch; although I am presenting this piece here with its four-worded title, the numerical *20100221* is considered an alternative title to the same work. The fact that audiovisual outputs created from this work on later dates are named with the numeric title date could cause confusion, which is why I have chosen to draw attention to the matter here.

### 3.3.4 Looking at sub synth amp map

The title, *sub synth amp map*, reflects the processes at play within the code of this piece, and was assigned to the work when a short video of the audio and matrix output was published online;<sup>30</sup> eight examples of the matrix output, equivalent to still frames of the video, are shown in Figure 3.4 below.<sup>31</sup>

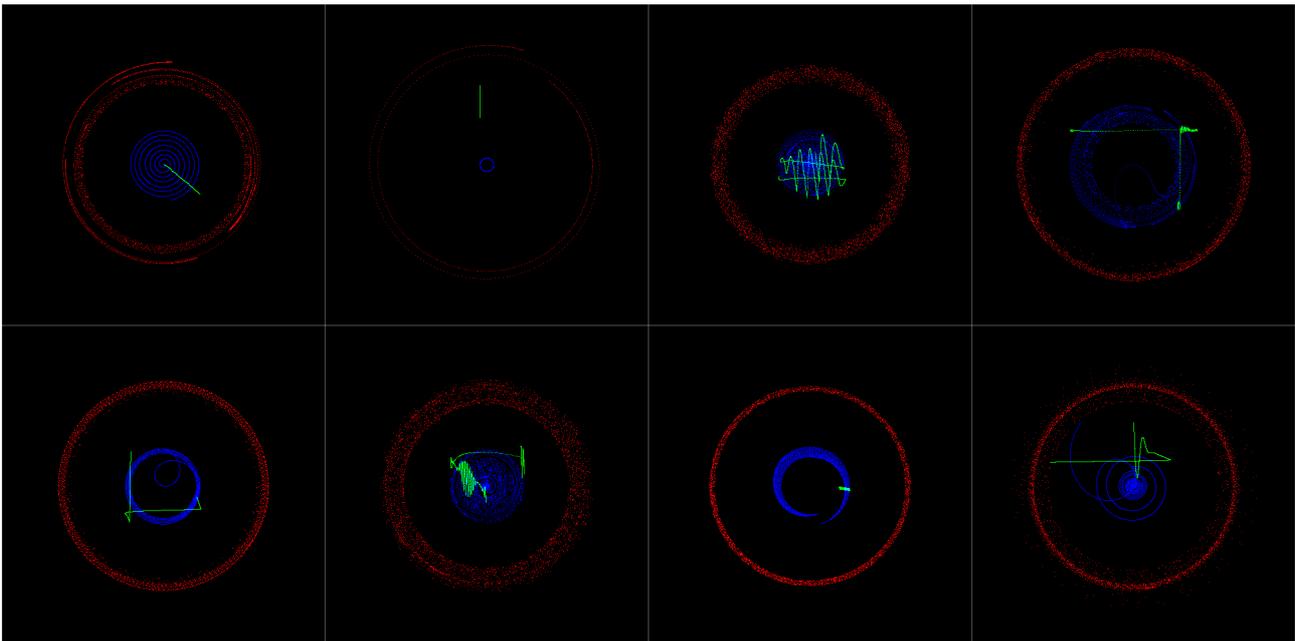


Figure 3.4: subsynthampmap\_matrix\_examples\_4x2b.png

The 'sub synth' part of the title makes multiple references to the content of the piece: it refers to the subtractive synthesis method of soundmaking employed, and to the way that sub-sonic

<sup>30</sup> Video available at <https://vimeo.com/9612972>

<sup>31</sup> Closer inspection of these images is possible in digital copies of this document; another eight examples, showing more variation in the typical output of the piece, are presented in Figure 3.9.

frequencies are used within that synthesis, both for modulation of the filter parameters, and in the audible input part of the synthesis (but the system is not limited to the using frequencies of the sub-sonic range). In the live coded design of the synthesis algorithm, a saw-tooth waveform was chosen as input to a low-pass filter with resonance; that waveform was chosen both because of its visual shape in the time-domain, and because it is harmonically rich in the frequency-domain. The sinusoidally modulated filtering of that input signal provides a time varying timbre that was quick to implement. When the **saw~** oscillator object is set to very low frequencies, its audible output is similar to that of the twitching-speakers described (in §3.1.1) above: the sudden change in amplitude at the hypothetically vertical point in the saw-tooth waveform creates an impulse, or click, while the intermediary ramp period of the waveform is inaudible but visually perceptible at those frequencies. With the fluctuating cut-off and resonance settings of the **lowres~** filter object, the sonic character of such clicks changes over time. Figure 3.5 shows the 'sub synth' algorithm of which there are two copies running in the patch. One copy is connected to the left, and the other to the right channel of the stereo output to the digital-analogue converter (DAC); these two signals are also used within the software as equivalent to those generated signals that were labelled 'X' and 'Y' in *visyn*.

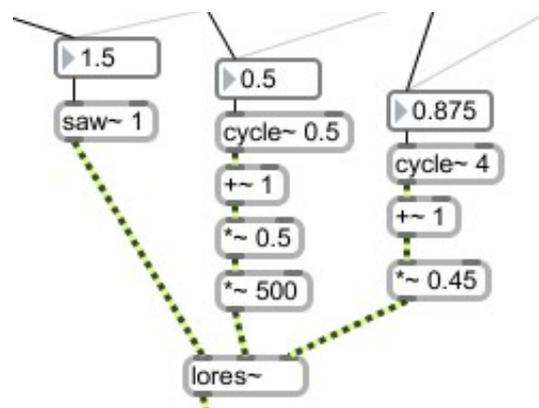


Figure 3.5: The 'sub synth' patch

The 'amp map' part of the title refers to the way that the audio signals are 'amplified' before

being 'mapped' to the dimensions of a Jitter matrix for display on screen; that is in contrast to the polar-cartesian conversions in *visyn* which occurred before value 'amplification'.<sup>32</sup> Another interpretation of the second part of the title is simply that it is the amplitude of the input signals that are being mapped to matrix cell coordinate values. In *sub synth amp map* the angle values used in polar-cartesian stages translate to a great many revolutions of the circumference. Figure 3.6 shows the display mappings that were coded on-the-fly to make use of the RGB layers of **jit.matrix m 4 char 500 500** so that three independent, monochromatic, visual forms can be simultaneously displayed.

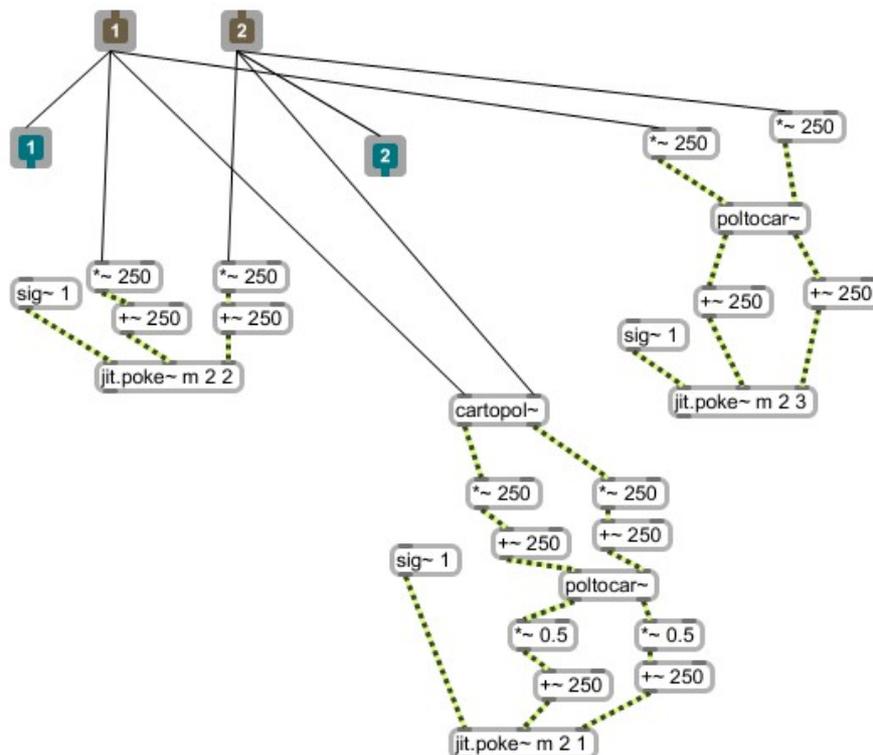


Figure 3.6: The 'amp map' patch

The grids of zero and one data values stored in **jit.matrix m** are visualised by a **jit.pwindow** object that has its size set to match the dimensions of the matrix so that each cell of that matrix will correspond to a pixel on screen. Default behaviour of the patch is for the matrix cell values to be

<sup>32</sup> Also see flow-charts in Figure 3.13 on page 62 for comparison.

cleared immediately after being put on screen. An analogy, for this behaviour in the software, is taken from a conception of Cymatics (the Cymatic experience of seeing sound is of the movement present within a physical system at a given moment in time) by thinking of the input signals as acting to cause movement across the surface of the matrix. As real-time visualisation of sound, the attention of the *sub synth amp map* observer is brought to 'now' because the image of what has just happened is gone from view (almost) as soon as it appears. The rate at which the matrix is put on screen in the present system is determined by the **metro 42** object (see Figure 3.7), which gives approximately 24 frames per second (fps). It is thus that the passage of time is brought into the visual representations of sound in the *sub synth amp map* system, and hence there is no need for time to be mapped to any other dimension of the display. If, as shown in Figure 3.7, the **gate** to the **clear** message is closed, then the traces will accumulate on the display. An example of this trace accumulation is shown in Figure 3.8 which is a matrix export saved to file during the initial coding session for this work.<sup>33</sup>

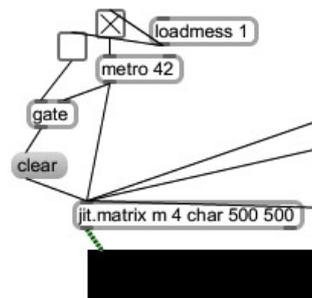


Figure 3.7: The 'matrix clear metro' patch

<sup>33</sup> The file name 'l' being given mindful of this being a first step towards my new way of working in software.

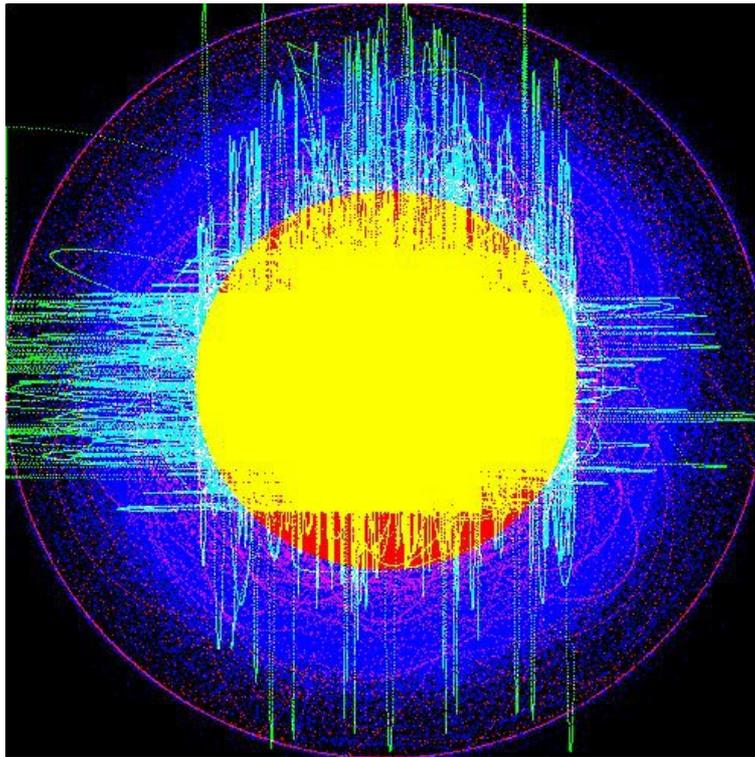


Figure 3.8: 1.jpg

### 3.3.5 Three layers, three mappings

Perhaps the most obvious method of mapping a pair of audio signals to dimensions of the screen pixel destined Jitter matrix is that of associating one signal with the x-axis and the other signal with the y-axis of the area on screen. In recognition of the commonality that such a mapping has in the use of oscilloscopes – which traditionally have a greenish phosphor trace display – plane 2 of the matrix is used as the target for this mapping which is also known as a vector-scope, goniometer, or phase-scope. The implemented algorithm for this mapping, as shown above in Figure 3.6, consists of three stages: (1) the two input signals, which have a range of -1 to 1, are first 'amplified' so that the range numerically matches the size of the matrix; (2) the signals are then offset so that the zero-crossing point of the inputs will appear in the middle of the display; (3) the signals are used to set the axis position values for the writing of a constant value (from `sig~ 1`) into the named matrix using `jitter.poke~ m 2 2` (where the first 2 is the number of 'dim' inputs, and second identifies the

plane of the matrix to poke data to). Because of the way that matrix cells are numbered from the top-left corner of their two-dimensional representation on screen, if the two input audio signals happened to be identical (in-phase) then the display would be of a diagonal line from upper-left to lower-right corner. It may well be expected – from experience of this type of signal display where lower signal values are traditionally manifest as lower on the screen – that the opposite orientation of slope would represent an in-phase signal. The maintained objective of directness and simplicity in the code, however, prohibits the addition of an extra stage to invert the horizontal axis. In a different environment or situation, an algorithm that implements the same three step mapping may automatically manifest with the traditionally expected orientation.

The next mapping to be implemented was for the blue layer of the display in which the input signals, after being 'amplified' as in the green layer mapping, are treated as if they were polar-coordinates and processed by **poltocar~**; the derived cartesian coordinates, once offset to align the zero-crossing of the inputs with the middle of the display, then go to the **jit.poke~ 2 3** object.

Lastly, the inverse translation of coordinate type (from polar to cartesian) is used for the red layer mapping. A process of trial and error lead to the configuration presented in the patch, which is a solution to the problem of polar coordinate values being unsuitable for direct connection to the cartesian-type 'dim' inputs of **jit.poke~**. The input audio signals are connected directly to **cartopol~**, the output of which is 'amplified' and offset in the same way as was the other two mappings. A second translation, back to cartesian, is then applied, and this is followed by a reduction in amplitude (to half the value range at that stage) and yet another offsetting stage before the signals go to **jit.poke~ 2 1**.

Figure 3.9 shows another eight examples of the *sub synth amp map* visual output.

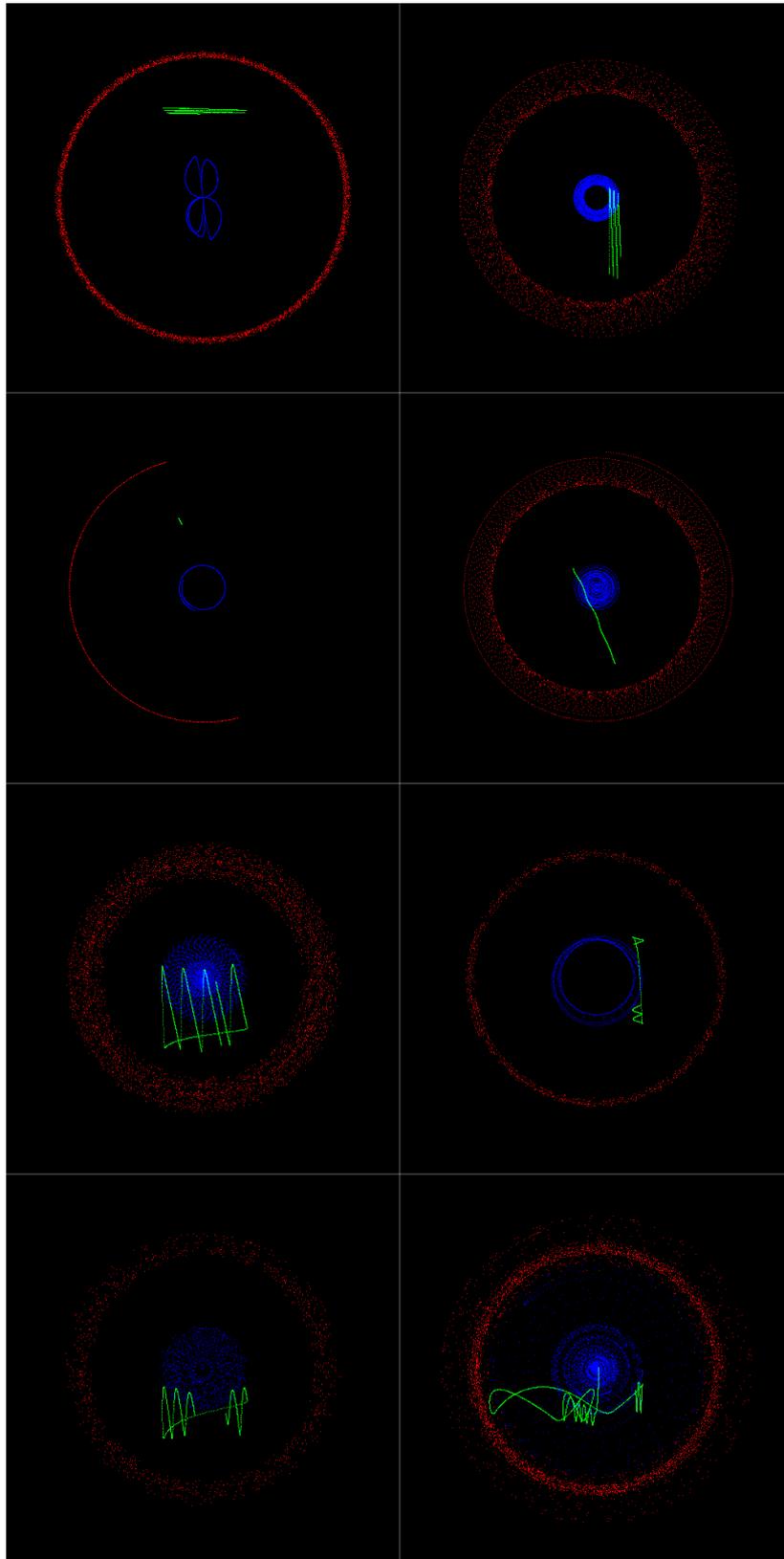


Figure 3.9: subsynthampmap\_matrix\_examples\_2x4.png

### 3.3.6 Input selections

A drop-down menu labelled 'Select source:' provides options for the choosing the input that will go to the display mappings sub-patch of the *sub synth amp map* patch. The last two options in the list allow arbitrary input of a two-channel sound source: either from the output of another active maxpat via **adoutput~** – in which case, to prevent a feedback loop, the audio will not locally be connected to the DAC – or from a two-channel sound-file via **sfplay~ 2** which has a **playbar** object for convenience. The other two source options in the menu select versions of the synthesised sound that was designed alongside the display mappings of the system. One of these, 'saw~lores~etc', connects to the real-time synthesis in the patch, while the other, '20100221\_%.aif' activates playback of sound-files that were recorded of a similar synth structure.

With 'saw~lores~etc' selected as the source, there are three ways to control the sound which is mapped to the on screen display: (1) frequency values can be set using the **number** boxes; (2) these can be randomized within a narrow range by clicking the **r bang** message; or, (3) for a wider range of variety, there are 24 predefined presets which can either be triggered directly via the **preset** object or in a shuffled sequential order by clicks to the nearby **button**. These are the presets that were used in the video published in 2010,<sup>34</sup> and – contrary to the sub-sonic frequencies rationale (§3.3.4) – include many higher frequency values.

When the '20100221\_%.aif' source is selected a continuous audio stream begins that is generatively formed from a set of sound-file recordings. There are 19 one-channel sound-files of duration ranging from a few seconds to a few minutes, and these were recorded during the initial LC rehearsal session as the idea to make more of this work began to set in. The simple algorithm for triggering this material is described by the flow-chart in Figure 3.10. There are two copies of this algorithm in the patch, in order to give two channels of audio. Because those two copies are

---

34 Video link, as given in footnote 18: <https://vimeo.com/9612972>

opening files from the same pool of audio, and because the audio was all recorded using a particular range of parameter values, the 'stereo' sound that is heard and visualised contains a high degree of repetition, but the differing lengths of sound-file duration mean that the exact same combination of two sound-files at a given phase offset is unlikely to ever occur twice. The idea thus occurs again of things that are both ever-changing and always the same.

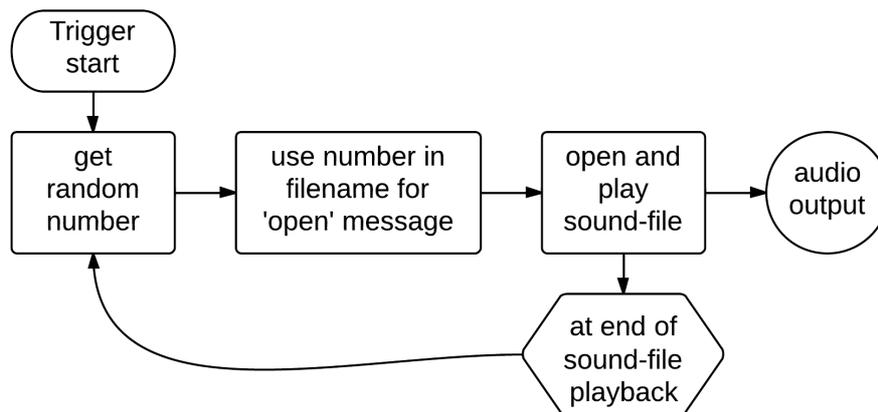


Figure 3.10: 20100221\_sfplay\_random\_flow

### 3.3.7 Lamp-light, Laser beams, and Laptop screens

#### (3.3.7.a) Lissajous figures

Lissajous figures, the well known class of mathematical curves, can be made in the green layer of the *sub synth amp map* display when two sinusoidal signals – of frequencies that are harmonically related (by simple ratio) – are given as input via the 'adoutput~' source selection of the system. The four images in Figure 3.11 show some such examples: clockwise from top-left, these examples are formed of ratios 1:2 (an octave), 1:3, 2:5, and 1:4. These and other examples are discussed further, and with others, in [sdfphd\\_appendix\\_B\\_subsynthampmap.pdf](#).

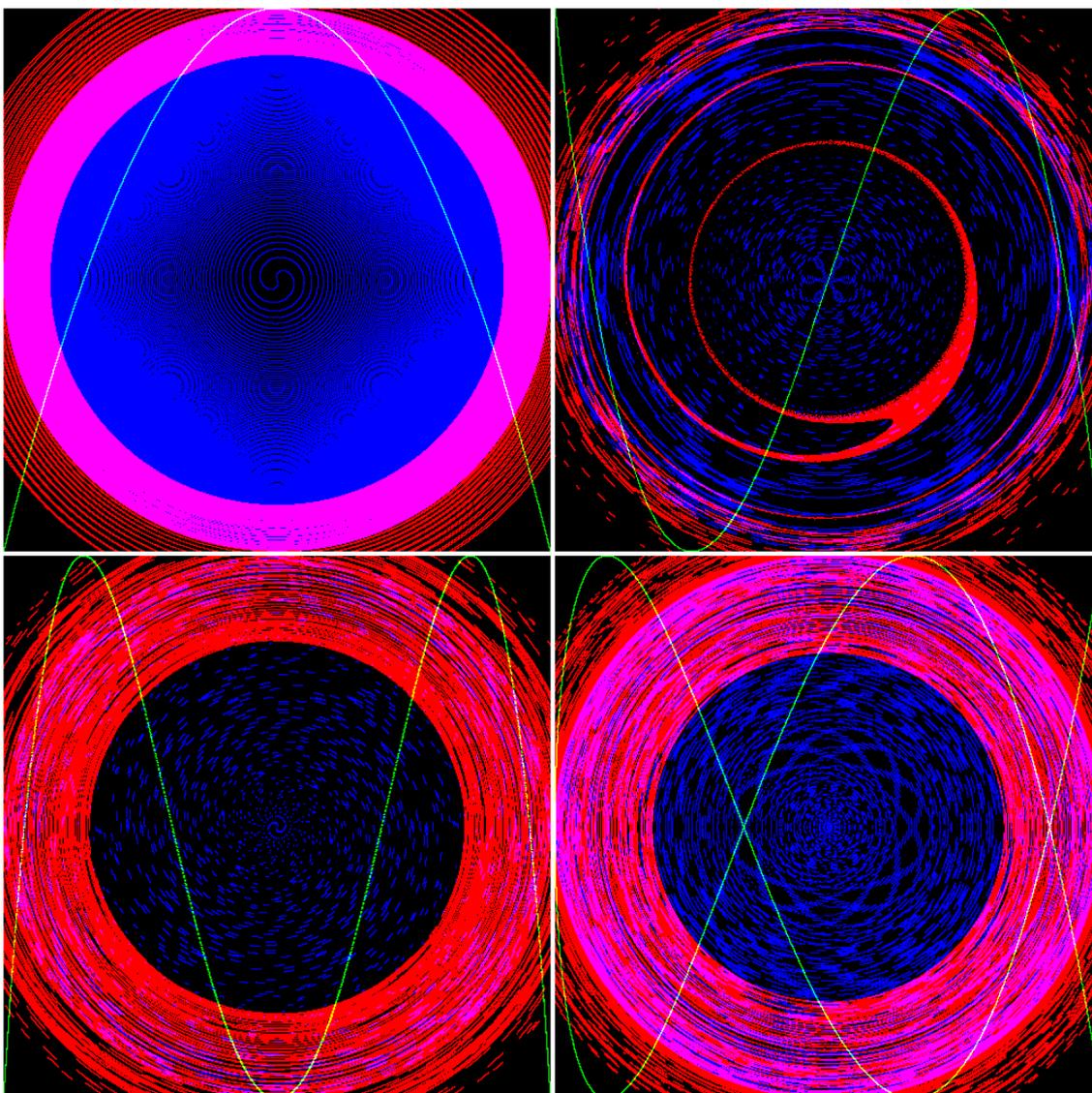


Figure 3.11: Four Lissajous

Lissajous figures are curves similar to those that are produced by the two pendulum harmonograph (Benham, in Newton, 1909, p. 29). The spiralling nature of harmonograms (see §2.1.3) is caused by the decaying amplitude of the oscillating pendulums. In *sub synth amp map*, the spiral formations are caused by the polar-cartesian conversions, and there is no decay of oscillator amplitude over time. A related visual effect of interference patterns, as seen within the blue and black area of figure the octave (top-left) in Figure 3.11 is also discussed by Benham who calls it 'the watered effect' (ibid. 1909, pp. 49–50):<sup>35</sup>

<sup>35</sup> Illustrated in Plate VIII of that publication.

By superimposing two unison figures that have been traced with a needle point on smoked glass, [...] the result is that the watering is shown, while the actual spirals, as such, are not visible. [...] Very beautiful effects may be obtained in this way, and if the two smoked glasses are mounted as lantern projections in one of the holders that allow for the rotation of one glass, a wonderful appearance is thrown on the screen, the watering completely changing form as the rotation proceeds.

### (3.3.7.b) Lamp-light

Projected light is also what was used by Jules Lissajous in 1857: the curves that we know by his name were produced from:

the compounding of the vibrations of two tuning forks held at right angles. Mirrors affixed to the tuning forks received and reflected a beam of light, and the resultant curves were thus projected optically on a screen. (ibid. Newton, 1909, p. 29–30)

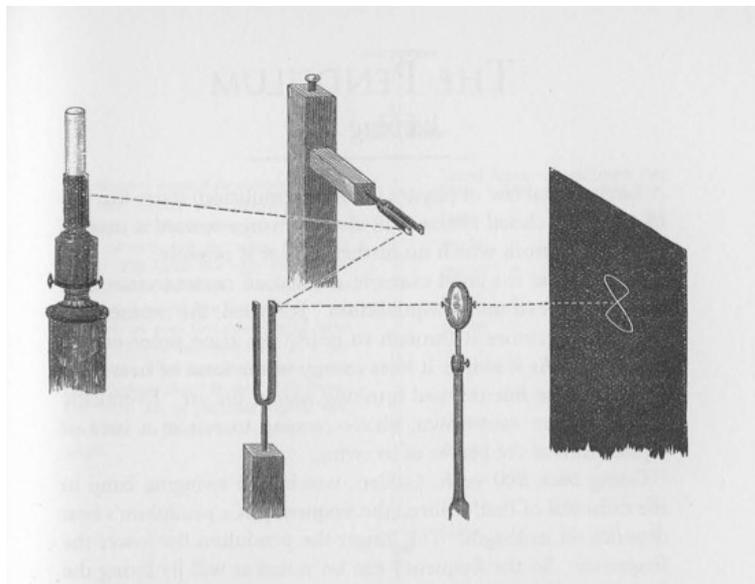


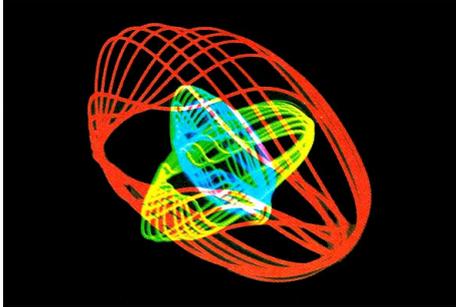
Figure 3.12: Lissajous mirrors

The described configuration of tuning forks, mirrors, a twice-reflected beam of light, and the projected figure manifest by these, are illustrated in Figure 3.12 (Ashton, 2003, p. 15).

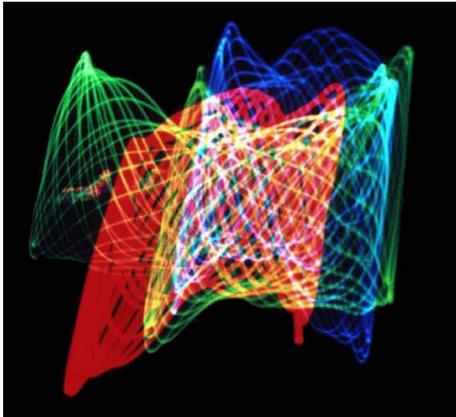
### (3.3.7.c) Laser beams

In 1968 Lowell Cross began a collaboration with Carson D. Jeffries to create a new system that employed those same principles of reflection. Working also with David Tudor, Cross established a method that used a number of laser beams – new technology at the time – each with a pair of mirror galvanometers controlled by amplified audio signals in place of the tuning forks, and 'the very first

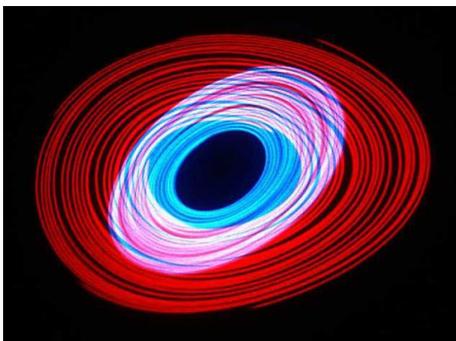
multi-color laser light show with x–y scanning [was presented] on the evening of 9 May 1969' (Cross, 2005). Three images from that *Audio/Video/Laser* collaboration are reproduced below:



*VIDEO/LASER II* in deflection mode, image 1 of 3. Tudor: *Untitled*, © 1969.<sup>36</sup>



*VIDEO/LASER II* in deflection mode, image 2 of 3. Cross: *Video II (L)*. First photograph in deflection mode. © 1969.<sup>37</sup>



*VIDEO/LASER II* in deflection mode, image 3 of 3. Jeffries: *Spirals*. © 1969.<sup>38</sup>

36 Online at <http://www.lowellcross.com/articles/avl/2a.Tudor.html> [accessed 20130619]

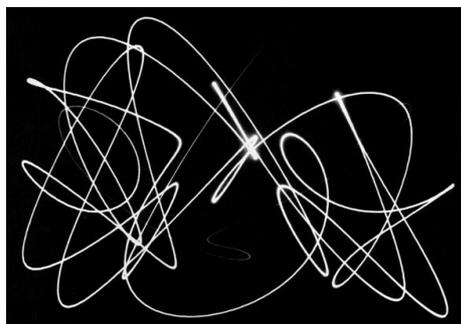
37 Online at <http://www.lowellcross.com/articles/avl/2b.VideoIIL.html> [accessed 20130619]

38 Online at <http://www.lowellcross.com/articles/avl/2c.Spirals.html> [accessed 20130619]

The periodic drive that was described by Collins (2012) – of artists returning to the 'acoustical reality of sound, and its quirky interaction with our sense of hearing' and to 'explore fundamental aspects of the physics and perception of sound'<sup>39</sup> – seems to extend to the realm of visual perception while working with sound. Whether with lamp-light, laser beams, or backlit computer screen pixels, patterns such as those seen in *sub synth amp map* have historical precedence for being involved when human minds push toward new ideas by connecting with fundamental principles.

### (3.3.7.d) Projected oscilloscopes

Before inventing the laser light show, Cross had already used other electrical equipment in a number of video pieces that also had focus on x-y scanning of the visual plane with audio signals as input; for example, here is an image from the 1965 work *Video II (B)*:



Cross: *Video II (B)*, © 1965.<sup>40</sup>

Recent works by Daniel Iglesia have also embraced the x-y 'oscilloscope' technique, but Iglesia has added a z-axis and the use of 3D anaglyph video projections;<sup>41</sup> Iglesia describes his 2010 'work for two piano, two electronic percussion, and live video for 3D', *Cardinality Aleph Two* (Iglesia, 2012)[sic]:

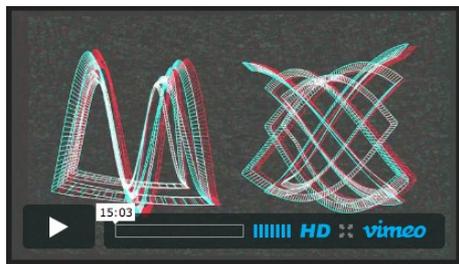
This concert work is built an evolving AV system based on oscilloscopes. Here, each “orb” is created by three oscillators or samplers. The audio output of each becomes drawn coordinates in spacel the first maps

<sup>39</sup> Cited at second paragraph of §2

<sup>40</sup> Online at <http://www.lowellcross.com/articles/avl/0x.VideoIIB.html> [accessed 20130619]

<sup>41</sup> There is also precedence dating to the late nineteenth-century for the 3D presentation of similar curve traces; the cited 1909 publication edited by Newton includes a number of stereoscopic harmonograms (Plate VI and Plate VII).

to the X dimension, second to Y, and third to Z. This is the most literal and low-level possible mapping of sound and video.



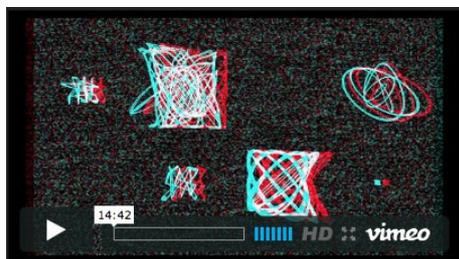
(image: Iglesia, 2010)

### (3.3.7.e) Laptop screens

In 2011 the Princeton Laptop Orchestra (PLOrk) premiered *24 Axes*, by Iglesia, 'for 8 piece laptop ensemble, with live 3D video' (Iglesia, 2012)[sic, but italics added]:

*24 Axes* builds upon the composer's [3D oscilloscope], in which the performer controlled three audio signals [...]. Via the relative frequencies and shapes of the three signals, various forms take shape. These shapes are the literal representations of the sound, nothing more.

For this scored ensemble version, each of the eight performers controls his/her own 3D oscilloscope. They follow a score that is being broadcast in real time to each laptop screen, with symbolic performance data traveling towards the performer in 3D space.



(image: Iglesia, 2011)

Having arrived at *24 Axes*, an ensemble work composed for PLOrk, the discussion of *sub synth amp map* has, in a way, come full circle: my study here began in the context of LC rehearsal, and much of my LC practice has been in the context of ensemble laptop performance, particularly with the Huddersfield Experimental Laptop Orchestra postgraduate group (HELOpg).<sup>42</sup> In these contemporaneous works by Iglesia the audience is presented with 'literal representations of the sound' that is being produced. It seems to me that these works manifest from much the same intention as did the laser light shows by Cross *et al.* some 50 years earlier. Cross writes (1981) that

<sup>42</sup> <http://helopg.co.uk>

his experiments with oscilloscope type display in performance of electronic music, that began in the mid 1960s, were in response to the abstraction of sound production from visual domain which Cross considered a problem in performance situations. When my development of *sub synth amp map* began – prior to reading of the contextualisation that has been given here – it was in mind that an audience may observe the programmatic process of mapping sound signals as well as the visual output of those mappings.

There seems to be an inherent propensity of studied sound to manifest visually as orthogonally projected oscillations. Such mappings can be applied both to analytical ends (as in the studies undertaken by Lissajous, and in the stereo phase-scopes used today by audio engineers), and to aesthetic ends (as in those systems developed by Cross, and by Iglesia). Arising from my own interest in polar coordinate systems, and their possible applications in computer music software, *sub synth amp map* has also integrated polar-cartesian transformations while mapping signals, and the results have been influential towards subsequent works within the project. For this project, however, the main point of interest arising from the study here is as follows: when one composes music in conjunction with a direct mapping of sound signals to visual space, the organisation and control of soundmaking parameters in the music happens according to an aesthetic interaction with those visual representations.

### 3.3.8 Connections and comparisons between study pieces

As stated in §3.3.2 the data matrix techniques exhibited in the *sub synth amp map* study are similar to those found in the *visyn* system; comparison of these two systems is aided by the overview flow-chart representations shown in Figure 3.13: in *visyn* the data that is displayed visually occurs with the flow-chart prior to the audio signals that are heard from the DAC; the *sub synth amp map* system reverses that flow in the way that the audio signals connected to the DAC occur before the displayed data.

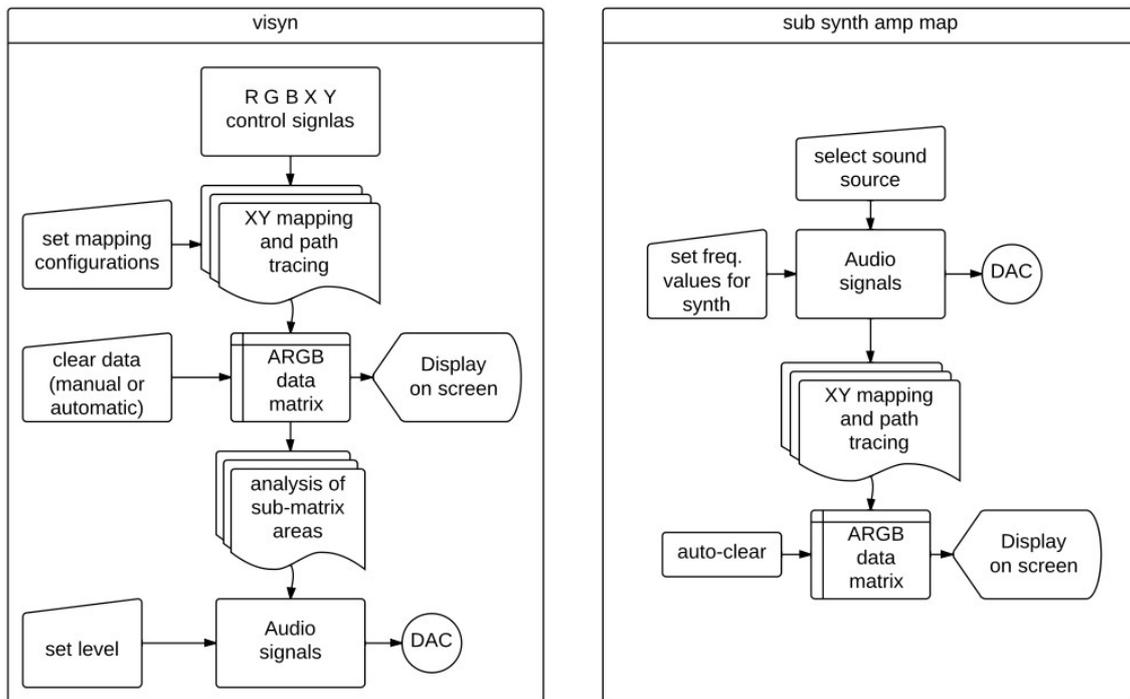


Figure 3.13: Visyn and sub synth amp map flow

My work often progresses by looking for ways in which elements of a system can be reversed. Another example of elements switching roles is shown in a comparison that looks ahead to the piece that is discussed in the next section. In the flow-charts of Figure 3.14, below, data writing poke operations implemented in two works are illustrated; *sub synth amp map* can be typified as system in which audio inputs are mapped for continuous control of to which cell of matrix a constant data value will be written. In contrast, the model of a gramophone has that relationship inverted: audio input continuously changes the data value that is to be written while a consistent algorithm (the model abstraction) controls to which cell that data is written.

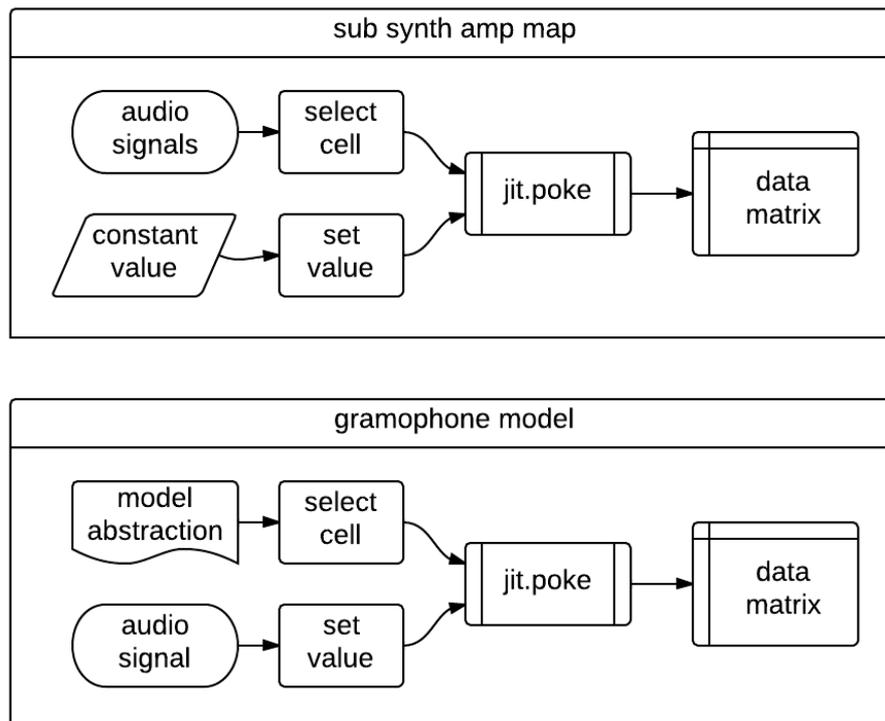


Figure 3.14: Comparative poke flow

In conclusion to this section a different connection is made: the Jitter matrix to which RGB traces are drawn in the *visyn* system is processed by the **jit.wake** object in order to 'soften the edges' of the otherwise 'pixelated' curves. The visual effect being aimed for was that of a glowing luminescence, and the comparison to the images by Cross of *VIDEO/LASER II* in deflection mode (see §(3.3.7.c)) can now be made. The slow fading that the wake effect in *visyn* causes of the visual traces is also manifest (albeit subtly) in the sounding of that system. It is noted now, however, that the visibility of pixel corners in the cell-quantized representations of theoretically continuous curves is something that is now celebrated in the on screen software medium within my work.

## 3.4 Conceptual model of a gramophone

As a composer seeking new methods for organising sound, I have sought to construct ways of understanding the things-with-which-I-work by bringing my attention to the mundane and mostly

taken-for-granted aspects of those things. In the preceding sections of this chapter, that approach has been mostly applied to the computer screen and the use of its pixels; in this section it is the very existence of recorded sound that is questioned. Continuing the theme of rethinking musical technologies that originate in the nineteenth-century, a conceptual model of a gramophone is the subject of the next study piece and which, then, becomes the basis for a software mediated composition (§3.5). Always with my questioning the goal is to inform and inspire both the creation of new systems for soundmaking and the making of music with such systems.

Historically, it was the phonograph that introduced the recording of sound waves to a surface medium via acoustically-coupled stylus, and the phonograph then introduced playback of similarly captured sound, but it was the gramophone that introduced the use of disc surfaces, inscribed with a spiralling trace (groove), as a sound recording medium; the basic format of which is still in use today.

### 3.4.1 Analysis of motion

A brief analysis of the motion found within a gramophone is conducted toward construction of a conceptual model. In the archetypical gramophone playback system, one can think of there being three dimensions of movement: (1) there is the angular rotation of the disc, (2) the radial sweep – from outer edge toward the centre – of the stylus-head transducer, and (3) the motion of the stylus needle itself as it follows the contours of the groove that comprise the sound recording. It is not conceptually relevant to think about how those contour tracing motions of the stylus needle eventually reach human ears as sound: it is enough to think of the stylus as being connected (somehow) to the output of the playback system. Focus remains, for now, with those three dimensions of motion identified. Mechanically, the only active part of the gramophone system described is the turning of the disc for the angular rotation of the disc. The spiral cut of the groove naturally brings the other two dimensions of motion into being as the needle of the stylus head rests

within the groove while the surface rotates. An alternative method for reading a gramophone disc is to let the disc surface be unmoving, and have the stylus head instead be mounted in the chassis of a forwardly active vehicle; thus is the device – pictured in Figure 3.15 (Sullivan, 2005) – that is sold under the name Vinyl Killer (Razy Works, 2008):

Vinyl Killer is a portable record player [...] Instead of spinning your record, it coasts on the surface of the vinyl, gliding the needle over and into the grooves, churning out music from its own built-in speaker.



Figure 3.15: Vinyl Killer

When deciding to build a model of a gramophone it was with the idea that a Jitter matrix would act as the disc surface (albeit as a square, instead of a round, plane), and that that surface would not need to spin (rotation of a Jitter matrix is possible, but unnecessary here); instead the model would include an active stylus head that moves around the on the surface. My model is described as a conceptual one; this study is not an exercise in physical-modelling. With a conceptual model the actual physics of reality need not be of concern, only the idea of the thing in question need be taken as the basis for reconstruction of that thing in another domain.

The work being described here was presented with a similar narrative in June, 2011, at the DorkBotSheffield event.<sup>43</sup> Within that presentation, and here too, the ambiguity that has, by this stage, crept in to the discussion with regard to what exactly is being referred to by a 'gramophone', is not of great concern: the ubiquitous influence of such systems within our culture puts the subject

---

<sup>43</sup> details online at <http://dorkbotsheffield.lurk.org/dorkbotsheffield-1>

matter well within audiences' consciousness.<sup>44</sup> Nevertheless, the model deserves a definite basis, and so the question is posed 'what, in essence, is a gramophone?', and then the answer given (in a form inspired by Senryu poetry) as a 17-syllable stanza:

a gramophone is  
a spiral track of sound-waves  
on a flat surface

Upon that concise abstract the conceptual model is based. For implementation of the model in software, the surface aspect of the gramophone – as mentioned above – can be a Jitter matrix (a plane of float32 data type is used), the sound-waves can be obtained from the computer's analogue-digital converter (ADC), and the spiral track can be defined by a curve drawing algorithm.

### 3.4.2 A spiral on the surface of a plane

To be in keeping with the above analysis of motion in a gramophone system, temporal navigation of the spiral curve that represents the groove should be the consequence of only one 'active drive'. The *theta* value input to the Archimedean spiral equation provides that behaviour: the polar equation to draw the Archimedean spiral is (Weisstein, 2003):

$$r = a * (\theta)$$

To understand that equation in terms of data-flow, it can be read from right-to-left: the angle value *theta* is the input that is processed by multiplication with *a* to give the output *r*. The range of the *theta* input is taken to be from zero to infinity, and the curve, too, is theoretically infinite in its expansion.

Most of us are far more accustomed to working with and thinking in cartesian-type spaces,

---

<sup>44</sup> One may speculate, however, that as some who was born in 1984 I may be of one of the last generations to have grown up with vinyl records as the mainstay of a music collection in the home. The gramophone-like disc format prevails as a medium but it is no longer dominant in the domestic environment.

and although polar-coordinate systems are easy enough to understand, they can be confusing in practice; not least because software implementations of them can differ greatly in terms of what angle appears where on the round, and in what unit of measurement. The visual direction by which angle values will increase can vary too: clockwise motion may represent positive or negative change. It is also common, in some disciplines, for the radius value to be referred to as amplitude, but given the commonly applied usage of that word in relation to sound I prefer to remain with the term radius in the context of polar-coordinates.

Frequent return to polar-coordinate representations in my work is related, for instance, to the perception motivating this project that the rectilinear conception of sound confines musical thinking, and that a more circular conception of things may lead to different compositional decisions being made. There is also the existence of visual 'form constants' which are innate to the human visual cortex (Wees, 1992; Bressloff et al., 2002; Grierson, 2008; Bókkon & Salari, 2012), and are geometrical patterns that can best be described in polar-coordinate terms.

Where an Archimedean spiral represents a gramophone groove, the *theta* value represents the linear time-domain of the recorded sound. The value of *a*, in the Archimedean spiral equation, serves to determine both the direction of the spiralling (rotation is clockwise when  $a < 0$ ) and the size of the spacing occurrent between successive rotations of the line. For every 360 degrees (or  $2\pi$  radians) increase in *theta* the radius will increase by the same specific amount.

More recently invented disc shaped storage media start their tracks (of digital data) toward the centre of the disc and spiral outward, and this is also the way of the Archimedean spiral: increasing the *theta* value is to move along the curve and radially away from the polar origin. In contrast, however, the time-domain groove of a gramophone disc begins from the outer edge and progresses inward. It was therefore part of the implementation of my conceptual model that the appropriate inversion of values be made within the algorithms so that increasing time would provide

a decreasing *theta* value over the required range. In order to reduce complexity for discussion, that aspect of the model is mostly absent in the following descriptions of the work.

### 3.4.3 Stylus abstraction

Before the model can be used to playback sounds, it must first be made to record them. The key to each of these processes is the stylus part of the system. An algorithmic representation of a stylus head has been constructed.

The 'active drive' of the stylus abstraction is a DSP sample-rate counter. When triggered to start, the counter value is processed with three parameter values that have been provided as input to the abstraction; the output is a pair of signals that provide cartesian coordinates for use with objects that access data stored as Jitter matrices. A flow-chart representation of the stylus abstraction is shown in Figure 3.16.

The 'Re-Trigger Mode' and 'Trigger Start' inputs are local to each instantiation of the abstraction. Values for the parameters that are named 'g\_dur', 'g\_mult', and 'g\_a'<sup>45</sup> are received the same to each stylus; this is so that the 'recording head' and the 'playback heads' – each with its own instance of the stylus abstraction – can be aligned to follow the same curve on the data surface.

---

45 The flow chart illustration, and this document in general, use underscore notation for these parameter names (i.e. g\_dur), whereas the maxpat uses a dash (g-dur); this discrepancy is indicative of a change in style within my programming during the project. While the MaxMSP interpretation 'g-dur' is as a single symbol, other languages might see it as “g” minus “dur” and so it would not work there as a variable name.

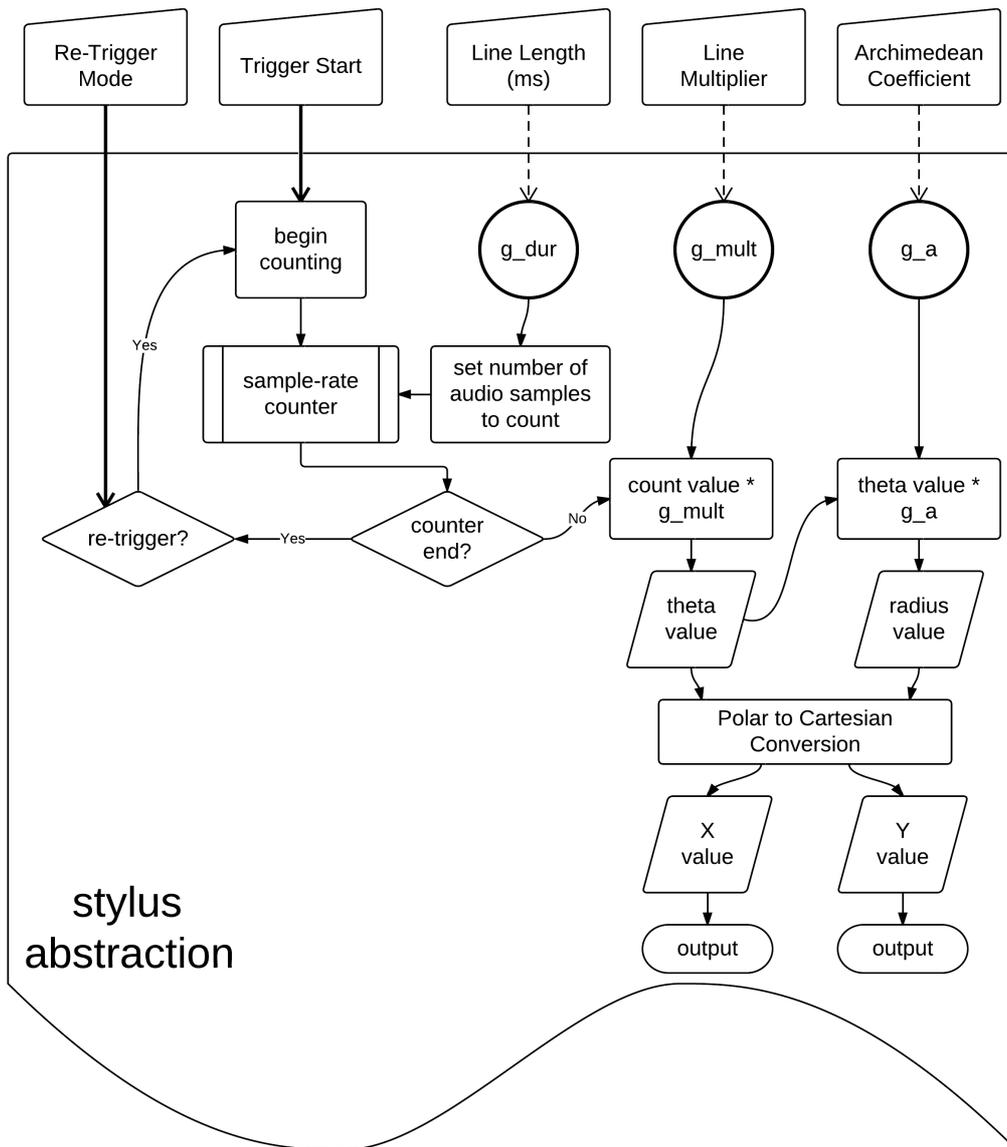


Figure 3.16: Stylus abstraction flow

### 3.4.4 Gramophone\_002\_a

Files pertaining to this study piece are found in the 'model\_of\_a\_gramophone' folder of the portfolio directories, and a neatened-up-for-presentation version of the first working prototype of the conceptual model is [gramophone\\_002\\_a.maxpat](#). Construction of this model was a progression from the work undertaken in *sub synth amp map*, and many of the same principles are found here. There are three styli in the system, and these are represented visually on screen as red, green, and

blue trace points of an RGB matrix. Coordinate signal outputs of the stylus that is represented by blue are used with a **jit.poke~** object to write values from the ADC to the 'gramophone matrix' (data surface); the other two styli, represented as trace points in green and red layers of the 'position matrix', are used with **jit.peek~** objects to read values from the data surface for output to the stereo DAC. Figure 3.17 shows a flow-chart representation of the patch.

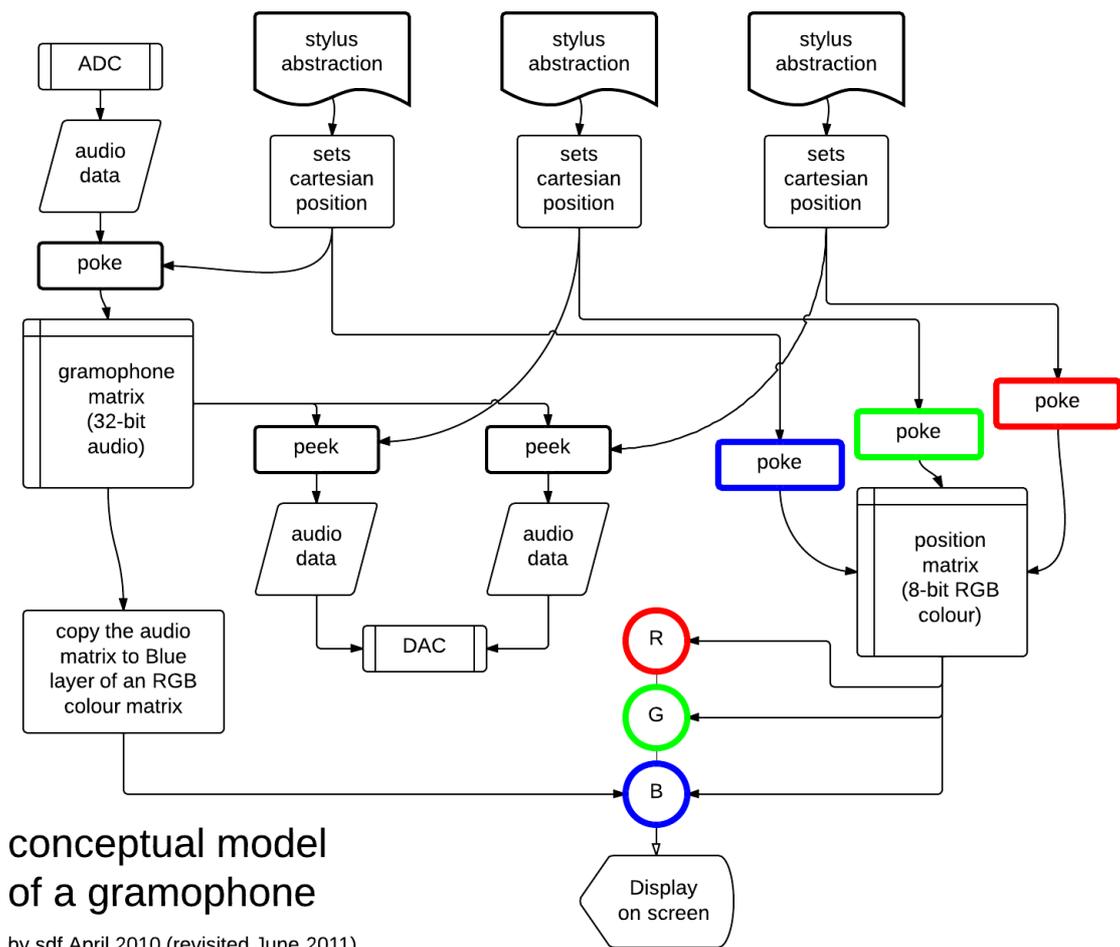


Figure 3.17: gramophone\_002\_a flow

Another version of the same patch, with 'i\_o' appended to the file name, was created so that the system could be used within the context of a larger modular maxpat system within live performance; in that version of the gramophone model the ADC has been replaced by a **receive~** object, and the DAC replaced by two **send~** objects.

Two video demonstrations of the \_002\_a patch have been included in the same portfolio folder; the audio in these screencast recordings is from the output of the maxpat. The first of these examples is [g002a\\_x1.mov](#) (0 min 55 sec); during this video, microphone input is used to record human voice to the data surface using preset 2 of the 'stylus settings', but with the duration parameter value (g\_dur) changed from 2000 to 4000. After input audio data has been written, the two read/playback-head styli are triggered – one, then the other – and the audio output can be heard. The 'in-out-in' re-trigger mode is selected for the (red) stylus that is connected to DAC channel two, at c.39 seconds into the video (as shown in Figure 3.18); it can be heard, thereafter, in the audio output of that stylus that on some of the inward passes of the spiral path, there must be a slight misalignment to the recorded data track because there a roughness to the sound reproduced which one can imagine as being caused by reading cells with zero data interlaced with the expected data. This glitch is caused by that the way the re-trigger modes have been implemented in this proof of concept prototype, but rather than looking upon it as an error or bug to fixed, the effect is embraced as having potential for creative exploitation; precise reproduction and high-fidelity are most certainly not the objectives here.

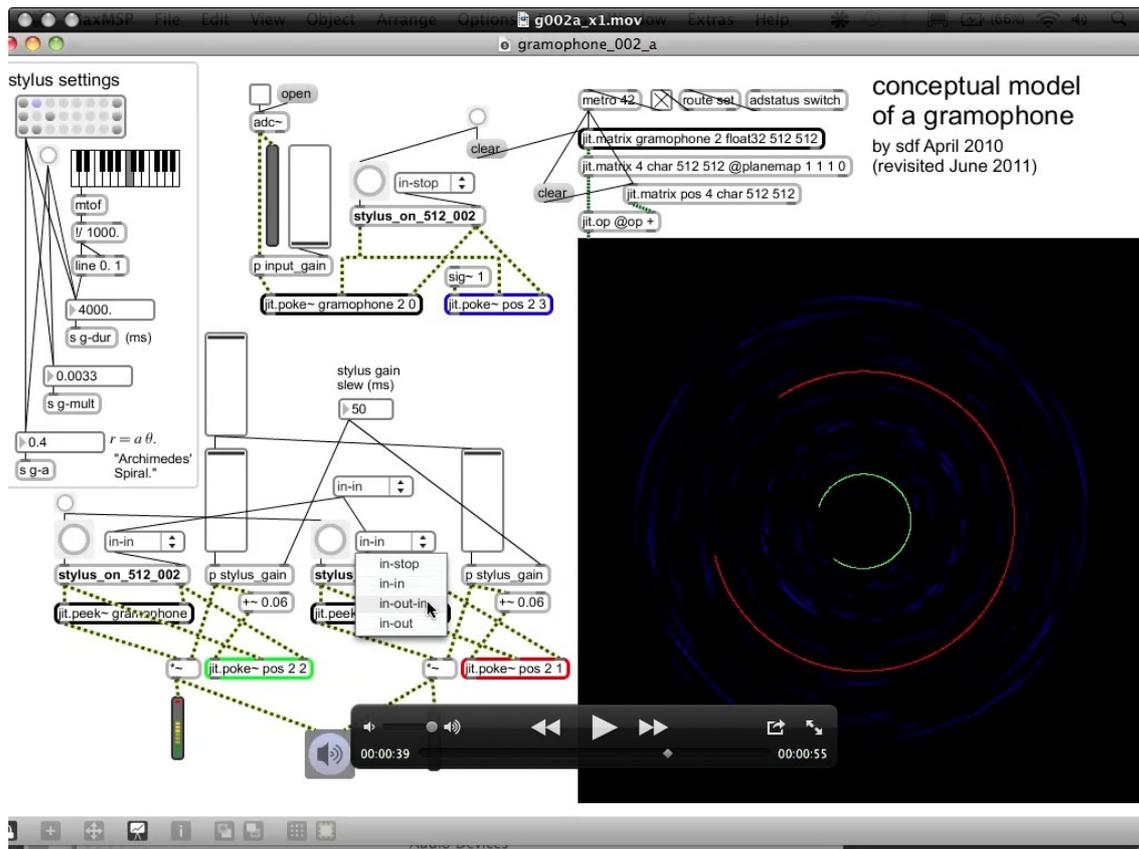


Figure 3.18: g002a\_x1.mov (at 39 sec)

Running in the demonstrated way, the gramophone model can be thought of as a low-fidelity delay-line in which the effective sample-rate is variable across the period of the delay: there are less matrix cells per angular degree near the centre of the data surface than there are at the outer periods of the spiral curve. Whereas the whole of the data matrix is available, only those cells that are accessed by the stylus abstraction during the recording are actually used to store data. Although this may seem an inefficient use of system resources, contemporary computer systems are not short of memory space, and the method developed here has the aesthetic benefit of allowing the audio data to be displayed on screen in a very direct way, with very little intermediary processing: the matrix of float32 type audio data is simply copied to the 'blue' layer of a char type ARGB matrix that is added to the 'position matrix' of the same type (showing traces of the three styli) and is then put on screen (with cell-to-pixel correlation). Disadvantages of that simplicity are discussed below

(§3.4.5).

The second video example, [g002a\\_x2.mov](#) (5 min 43 sec), resumes from where the first had left off, and continues with sound from a soprano melodica at the microphone input to ADC. For this demonstration, the pitch 'A-flat' was chosen to be played on the melodica because it has been found to manifest with a coherent six-armed pattern within some of the preset stylus settings that are used during the screencast video: see Figure 3.19. During the demonstration, the audiovisual output can be observed as audio data is written to the data surface and parameters of the patch are changed on screen.

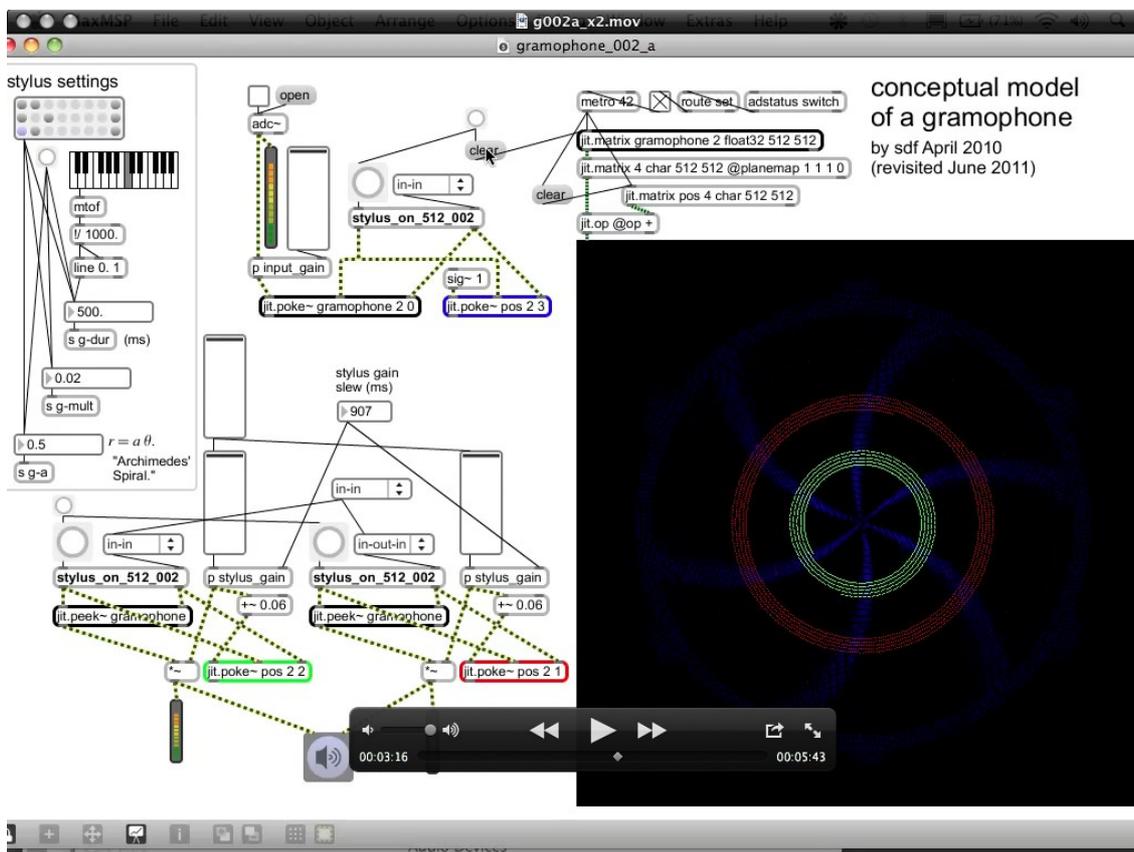


Figure 3.19: [g002a\\_x2.mov](#) (at 3 min 16 sec)

When the stylus settings begin to be changed, at first only the duration value is reduced: playback continues over a smaller radius of the surface than before, but because the  $g\_mult$  and  $g\_a$  values are the same, the alignment of the reading curve is kept upon the written data track. As the

$g_a$  value is increased, so too is the radius of the spiral track and with that the data values being read to form the audio output signals are no longer being read in the same sequence that they were written. As long as the  $g_{mult}$  value is unchanged, however, the audio output retains much of the character of the original input.

The creative potential of the model, as it is implemented here, really begins to show when all three of the stylus settings parameters are altered, especially when taken to extremes. At 3 min 51 sec audio input is once again written to the data surface with the current stylus settings; the  $g_{mult}$  value is then altered and a pitch shift in the output can be heard. Soon thereafter all three parameter values are shifted through ranges that produce some delightful glitch textured sounds in conjunction with aliased and interlaced spiral patterns in the traces of the read-head paths (see, for example, in Figure 3.20).

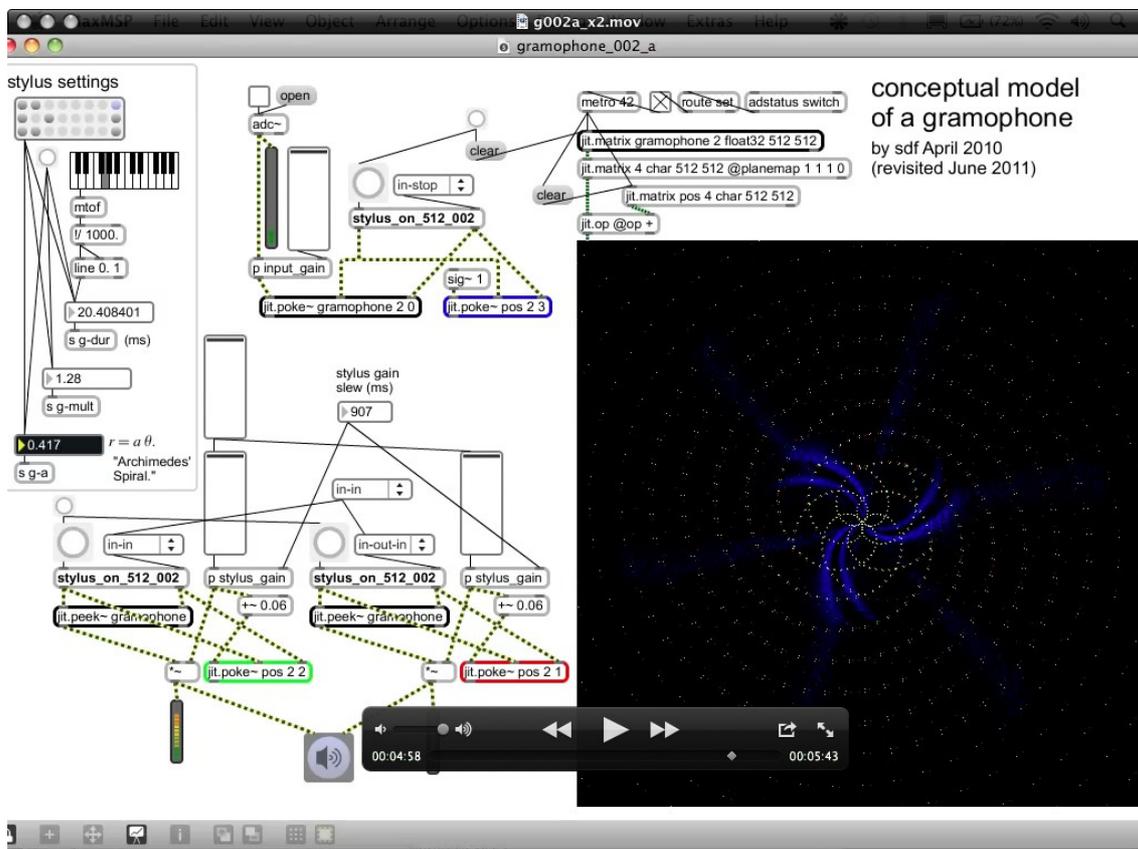


Figure 3.20: g002a\_x2.mov (at 4 min 58 sec)

### 3.4.5 Interacting with \_002\_a: limitations and inspirations

Notice that in the blue layer of the display, the audio data appears as an intermittent trace that oscillates between blue and black. This use of blue on black was perhaps a poor choice for visibility of the data in general, but having green and red as representative of left and right had already been established in other works within the scope of this project.<sup>46</sup> The method by which the recorded audio data is rendered here to the on screen display introduces truncation of the data values such that the visual representation includes only the positive values. Jitter automatically scales float32 values between 0 and 1 to the char data range of 0 to 255 that is used in the ARGB format for screen pixel colouration. The negative values of the sound recording are truncated by that mapping, and are thus represented visually the same as zero values. For a typical audio input recorded to the data surface there ought to be twice as many non-black cells shown on screen. It may also be considered that the non-linear perception of sound signal amplitude in human hearing perhaps ought to be a feature of the processes that convey audio data through colour level in a display. Representation of the bipolar data value range is something that the sdfsys development addresses (see §6.2.5). It was found unnecessary, however, to bring logarithmic scaling or similar into the methods for this aspect of the project.

Simultaneous observation of both the changes (on the RGB matrix display) and their causes is difficult, if not impossible, not only when watching back the example videos, but also when playing the patch itself. Visual manifestation of the three stylus abstraction control parameters as shapes on the data surface display is both spatially and conceptually removed from the interactive elements of the GUI on screen (the number boxes, sliders, menus, and buttons). One's attention is constantly split between two, or more, mental models of the system. My thoughts, at the time of creating this work and exploring its possibilities, thus became of how much better it would be to be able to keep one's eyes upon the shapes and impart control directly upon them within that space on

<sup>46</sup> But in works that are not included in the portfolio; see for example <http://sdfphd.net/looking-at-sound/>

screen; this concept I have come to refer to within my work as the 'integrated display' of audio and control data, and the sdfsys system was developed with that in mind.

Settings to which my explorations of the \_002\_a prototype soon habitually lead were of the short duration, great multiplication value type that produce a particular quality of audio output: as discussed further in §3.5.4, when the stylus abstraction path is set to such that the extremities of the curve are beyond the edges of the data surface fragmentation is introduced to the sound as the output signals oscillate with periods of data/no-data. To understand the clicks that may become buzzes in the resultant sound from these settings it is observed that the **jit.peek~** objects, that are used to read values from the matrix cells, will always return zero when cells coordinates beyond those present in the matrix are requested. The data/no-data description is assuming that there is audio data within the matrix to be read during the periods of those such set paths that do cross the data surface.

Rather than developing the gramophone model patch toward a performance system – which may have included the addition of more read-head styli, more options for variety in their control, perhaps with a more integrated display – the work was instead steered toward a compositional outcome. Various intermediary stages of that development are omitted in favour of a detailed examination of the \_005g version of the conceptual model of a gramophone.

## 3.5 Gramophone\_005g

Opening [gramophone\\_005g.maxpat](#) and clicking on the large, green coloured, button labelled 'click here to start' will start performance of what is considered the first complete composition to be presented in this document – as opposed to the formative studies discussed above. The title *gramophone\_005g* is rather cumbersome, but the work has evaded my efforts to denominate beyond the reality of its inception; the short form *g005g* was adopted in the naming of exported audiovisual

files to exemplify the work, and so that can also be borrowed as an abbreviated title for the piece.

It is advised to ensure that audio output of MaxMSP is working before clicking on the start button. Although the piece is programmed to switch DSP on at its start, it cannot check that an appropriate driver is actively selected, nor that stereo speakers are present after the output of the DAC. There are always a few seconds of silent inactivity after the fullscreen display of the work has been initiated, and before the soundmaking begins; this short pause is intentional within the composition, but it does mean that if there is something wrong with the DSP settings, or similar, then it may take a little longer to identify as one waits for the end of that silent period.

Compositionally this piece is predicated on sequenced control of the three stylus abstraction parameters that have been described above. Subjectively pleasing parameter value combinations, and transitions between such, were identified through experimentation with the model, but rather than score a fixed trajectory between these, the approach adopted was toward a more 'open outcome' (see Jansch, 2011, chapter 4). The composition is an exploration of the unplanned-for audiovisual possibilities found in the conceptual modelling of a gramophone manifest in software; in particular the shapes and sound-forms created when that model is pushed beyond the range of motion expected of the original technology.<sup>47</sup> To some extent the choices made in controlling this software implementation of the conceptual model are arbitrary: the numbers used and their interrelatedness have been crafted according to subjective personal preference and intuition, without theorised connection to the gramophone-ness of the work nor to anything else beyond experience of the in-progress software as an interactive medium. The piece developed through careful tweaking of interrelated aspects in order to provide an open outcome structure that maintains identity as a fixed entity whilst being able to surprise even its author, and thus include an element discovery at

---

<sup>47</sup> I am reminded both of experimentation such as seen in a video featuring a record player retrofitted so that an 'Angle Grinder was used as the drive' (Photonicinduction, 2012), and also of reading about Karlheinz Stockhausen's high-speed rotating speaker system that was controlled from the other side of a 'safety-plated door' at variable speeds 'up to 24 rotations per second' (Brümmer, 2008, p. 15).

each performance. The idea of a thing that is both always the same and every time different continues to be important.

As described below, the control value sequences that play through during the piece are generated when the maxpat is opened. This means that the same control sequence can be performed multiple times, but only until the patch is closed; a new sequence is generated when it is next opened. Duration of *g005g* is just over one minute, and it is recommended that more than one control sequence be generated and presented – the patch (re)opened and run more than once – in order to provide the possibility of appreciating the different-same-ness.

Within the *g005g* system, the control data sequences cannot be conveniently saved/recalled: the option to capture the data from now for use later has not been programmed in to the software, in the same way that an image output feature was deliberately omitted from the *visyn* system. Audiovisual recordings of performances have been made, and while they serve a number of functions, they suffer variously and are but shadows of the work.

### 3.5.1 Sequence generator

Setup for performance of the piece begins when the patch is opened in the MaxMSP environment: **loadbang** objects trigger the algorithmic generation of six lists: for each of the three stylus parameters there is a list of parameter values, and also a list each of transition durations that are used when stepping through the parameter values. After being generated, the values within each of these lists are reordered; the reordering includes randomisation (using **zl scramble**) for parameter value lists. For each of three stylus parameters, the lists of values and durations are then formatted together for storage as a collection of messages in a **coll** object. As an example, Figure 3.21 shows the sub-patch that generates the collection messages used for the *g\_mult* parameter; note that the annotations of 'target range' were developmental notes, superseded by refinement of the piece. The collections of value-duration pairs are stepped through as the piece is performed.

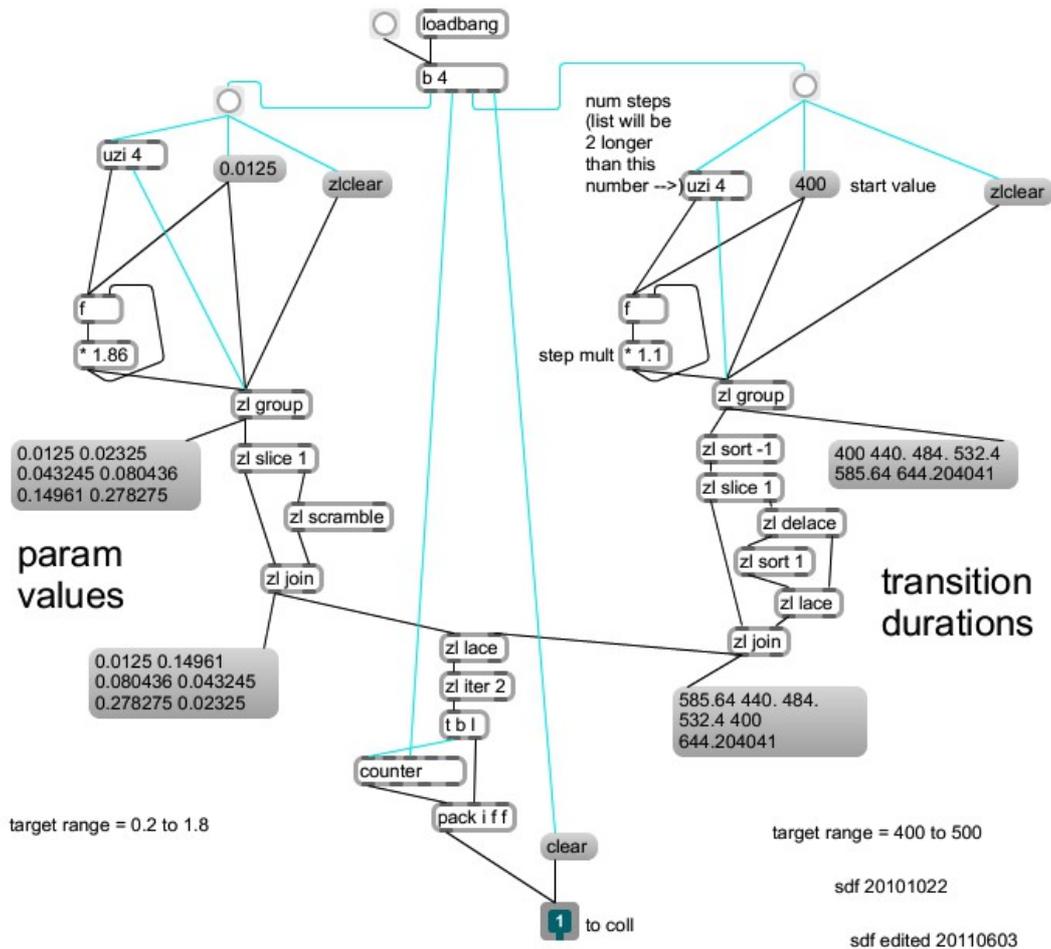


Figure 3.21: set\_g-mult\_seq

### (3.5.1.a) List generation

Generation of each list (before reordering) is determined by a set of three numbers: a start value ( $s$ ), a multiplier ( $m$ ), and a number for the length of the list ( $l$  – which is the same for both of the lists that belong to a stylus parameter). The following table shows the numbers that were fixed upon after a repetitive process of interactive trial-and-improvement to 'tune' the software system:

Stylus Parameter (length of list, $l$ )	Parameter Values		Transition Durations	
	start ( $s$ )	multiplier ( $m$ )	start ( $s$ )	multiplier ( $m$ )
<b>g_mult (6)</b>	0.0125	1.86	400	1.1
<b>g_a (12)</b>	0.1	1.5	320	1.1
<b>g_dur (44)</b>	2	1.25	2	1.2

Although the algorithm to generate the lists was conceived in maxpat form – as illustrated in Figure 3.21 above using **message**, **zl group**, **uzi**, **f** and **\*** objects – it can be expressed more mathematically as the list  $L$  in which:

$$L_n = L_{n-1} * m$$

where  $n$  is in the range of zero to one less than  $l$ , and:

$$L_0 = s$$

It is difficult to put into words the way that I relate to algorithmic concepts when composing, but my thought processes are perhaps somewhere between the two forms of notation demonstrated above; somewhere, that is, between a visual data flow and an abstract sequential algebraic representation of number patterns. It seems significant to consider this because the non-lingual, non-material, abstract number level of the human mind is the realm of much creative thinking.

The algorithm/formula used to generate the lists is quite elementary: indeed most of my work is to do with interconnection of simple ideas. Further, with a more analytical reevaluation of this material an even more simple notation of the same algorithm has been observed, and it is, perhaps, a much more elegant representation of the list generation:

$$L_n = s * m^n$$

With this formulation, the  $n$ th value of the list can be determined without requiring knowledge of the previous value. In contrast, the form in which the list generator was conceived is that of a sequential process that is thought of as to occur in the time-domain, albeit calculated very quickly by the computer. Whereas the  $m^n$  version could just as well be implemented, in MaxMSP or any other environment, the  $L_{n-1}$  version is more aesthetically aligned to the intention behind this part of the system. Generation of the lists is an automated but linear process.

### (3.5.1.b) List value reordering

Reordering of the numbers within each of the six lists is carried out by a number of different **zl** object cascades within the maxpat. There is enough similarity amongst these six different cascades – two of which can be seen in the illustrated sub-patch (Figure 3.21) above – to be able to represent the list item order processing as shown in the two following flow-chart representations. Figure 3.22 represents generation of parameter values lists, and Figure 3.23 is a representation of the transition duration lists. Although these such flow-charts are intended to represent aspects of my work in a non-language-specific way by calling upon general computer programming concepts, in these flow-charts include the specific 'mode' names of the **zl** objects ('slice','scramble', etc).<sup>48</sup>

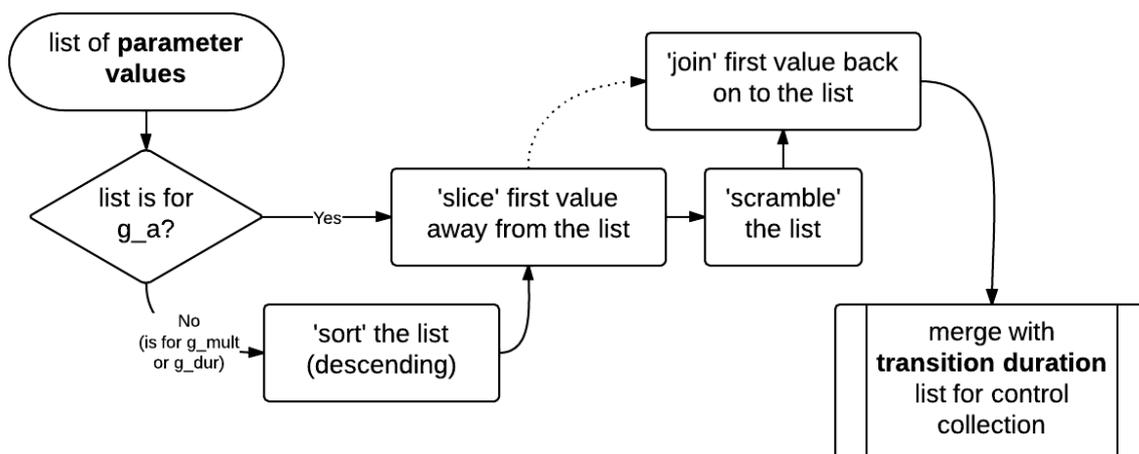


Figure 3.22: g005g\_reordering\_paramVal

48 Description of the modes can be found in the MaxMSP documentation for the object, also online at <http://www.cycling74.com/docs/max5/refpages/max-ref/zl.html>

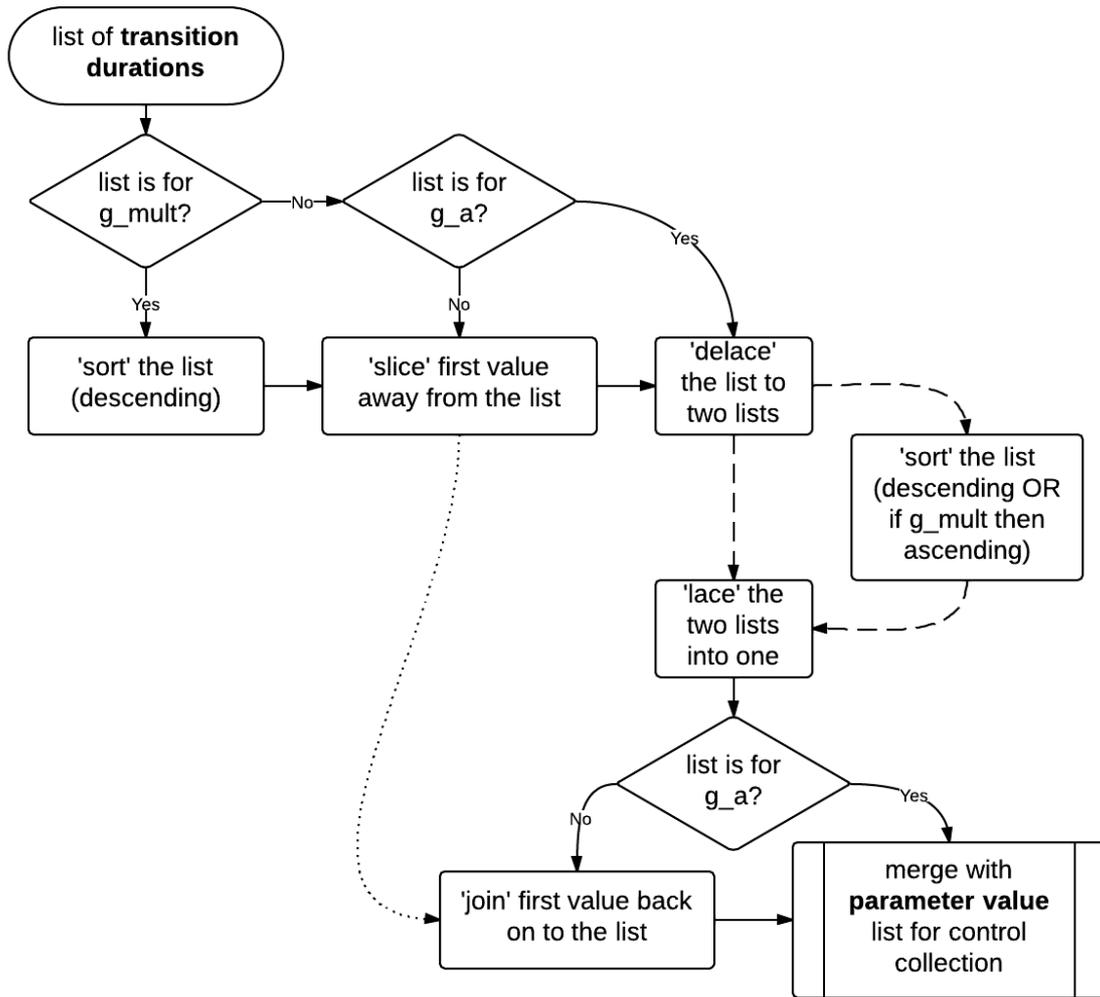


Figure 3.23: g005g\_reordering\_transDur

### (3.5.1.c) Layers of abstraction

As a brief aside to the discussion, an acknowledgment towards the concept of 'layers of abstraction' which has been recurrent as an aesthetic reference throughout this project.

Notice, above, repetition within the words that describe how the *g005g* piece is constructed. Within the stylus abstraction of the gramophone model the *g\_mult* and the *g\_a* parameters are both multipliers; more multipliers are used in generating lists for sequential control of that model. The third parameter of the stylus abstraction, *g\_dur*, is a duration value (expressed in milliseconds); lists of duration values are generated to control the transitions between values during the sequenced

performance. In describing the way that those lists for the sequential control aspect of the composition are generated it has been shown that a sequential algorithm is used. The same concepts are occurring at different layers of abstraction, and the concept of layers of abstraction becomes an important element of the aesthetics of the project in development of later works.

The works, cited above, by Iglesia entered into the research of this project only after the identification of an article, from *Organised Sound* 13(3), as significant to aesthetic underpinning of the project (Iglesia, 2008, p. 218):

At the very start, the elements of computer music were most referential of their own mechanical properties: the most basic element, the square wave, is made by turning one bit on and off. Every advance in digital audio technology has added a distance between our user concept of sound manipulation and the actual operation of raw bits. Ones and zeroes have been abstracted to machine code through the RISC architecture, opcodes and assembly language, which have been abstracted to text programming languages through interpreters and compilers, and onwards to graphical user interfaces, physical interfaces, and so forth. The creation and traversal of these layers of abstraction is itself the history of computer evolution, and, with it, the history of techniques for making and controlling computer sound.

### 3.5.2 Manifestation of g005g

Because each of the parameter control message collections is of a different length, the shorter two collections will be iterated through more than once during a single performance; the ending of the piece – in which the amplitude of the output audio is faded out – is triggered when the last message of the longer `g_dur` collection is reached. Although the `g_a` and `g_mult` parameter values will occur multiple times, each occurrence of a specific value will coincide with different values of the other parameters. The method by which the parameter values are triggered from their collections during the piece is shown as a flow-chart in Figure 3.24.

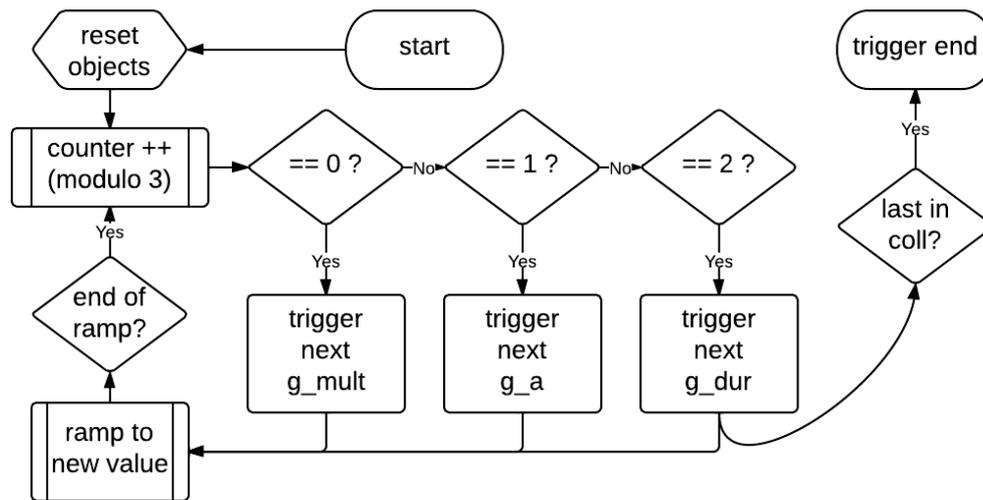


Figure 3.24: g005g\_trigger flow

Often the position of the playback-heads will be set to coordinates that are beyond the bounds of the data matrix, and so the audio output includes many silent periods. Although the durations that control parameter value transitions are the same every time, the values that control where the stylus path sits on the plane will occur in different patterns, and the manifestation of this in time is that those the silent periods are indeterminate and may each be of anything from as short as a few milliseconds – in which case there is a granulation of sound produced – up to as long as several seconds: a six second silence has been measured in the fifth example audio recording of the piece (see §(3.5.4.b) below).

Audio input data recording is not a feature of *g005g*. Cell values of the data surface are instead all set, once, by the output of a **jit.noise** object as modified by **jit.expr**, when the patch is loaded. When composing this piece I was still new to the Jitter realm of the MaxMSP environment, and I was studying to learn about more of its objects. One of the example expressions from the **jit.expr.maxhelp** patch was found to give a pleasingly circular formation when processing a matrix filled by **jit.noise** as illustrated in the middle display within Figure 3.25.<sup>49</sup> The dark area in the

<sup>49</sup> It is beyond the scope of this project to discuss at any depth the optical illusions induced by these processed noise images, but the idea of visual 'form constants' is again brought to mind (mentioned, with citations, in §3.4.2).

centre is caused by lower values in the cells of the matrix there; it was decided to reverse that light-to-dark-inward-ness in the visual form to have, instead, a dark-to-light-inward-ness thus to approach greater values correlating to greater amplitude in the sound as the styli move inward.

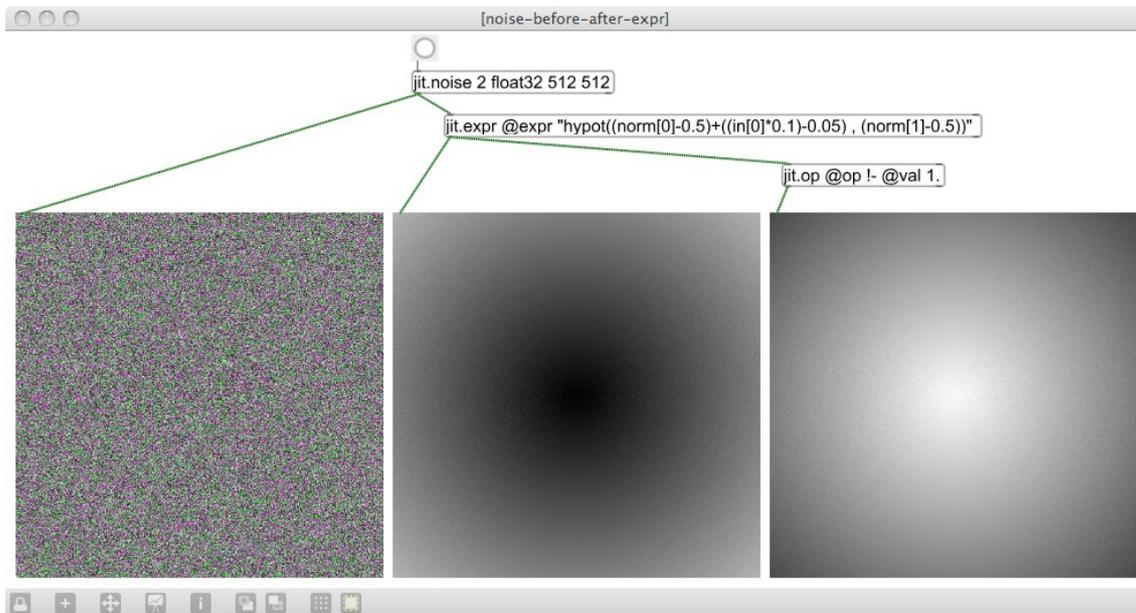


Figure 3.25: jitnoiseexpr

The prototype of the gramophone model demonstrated in version `_002_a` used the blue layer of an RGB display to show the recorded audio data, and the trace of the recording-head while writing that data, but now that a static greyscale data surface is in use the blue layer is conceptually available for somethings else. Perhaps inevitably, an x-y oscilloscope type trace of the audio output channels was established. This addition makes it so that the on-screen display includes visual representation of all three stages in the system: the raw data, the reading process of that data, and a projection of the sounding form of the read data.

By adding the oscilloscope type visualisation it became immediately apparent that the cells of the data surface contain only values between 0 and 1: the blue trace on screen was confined to one quarter of the plane. Audio output, therefore, would only drive each speaker across half of its available range of motion. The twitching-speakers described in §3.1.1, and referenced again in

§3.3.4, are again brought to mind: here it is the unidirectional drive of the battery on the speaker that comes to mind in particular. One option to rectify the situation in the *g005g* system would have been to scale the data matrix to the  $\pm 1$  range, but the solution taken was to follow my first instinct at discovering the issue.<sup>50</sup> As soon as the observation was made, a DSP sub-patch was created on-the-fly with the aim of introducing inverted copies of the output signals with modulated time delays to create a subtle comb-filtering effect.

The DSP section of the *g005g* patch is shown in Figure 3.26. Notice that one of the stylus abstractions (the patch is named `stylus_on_512_002`) is set to re-trigger mode 1, and the other to mode 3 (the second inlet): these manifest as 'in-in' and 'in-out', as is indicated in the pop-up menu objects in `_002_a`, which is not as is annotated within the stylus abstraction patch. The comments within the stylus abstraction patch indicate the behaviours that were being aimed for when programming the re-triggering logic algorithm. Unless there is an aesthetic reason to pursue exactly the self-set goals in the creation of a system, it is standard within my practice to let the unexpected outcomes influence the work. Glitch and surprise are embraced at many levels of abstraction in my work; new directions are often followed when 'errors' occur along the way. Various traces of my thought processes – such as dead-ends in data flow, and misleading comments – are often left intact within the software artefacts, especially in these early works of the project. In later works there has been more attention to the final correctness of software presentation.

---

<sup>50</sup> Evidently I had at first assumed **noise~** like output from **jit.noise**, even though the observed behaviour of **jit.noise** is perfectly logical, especially in view of Jitter's native float32-to-char mapping discussed at §3.4.5 above.

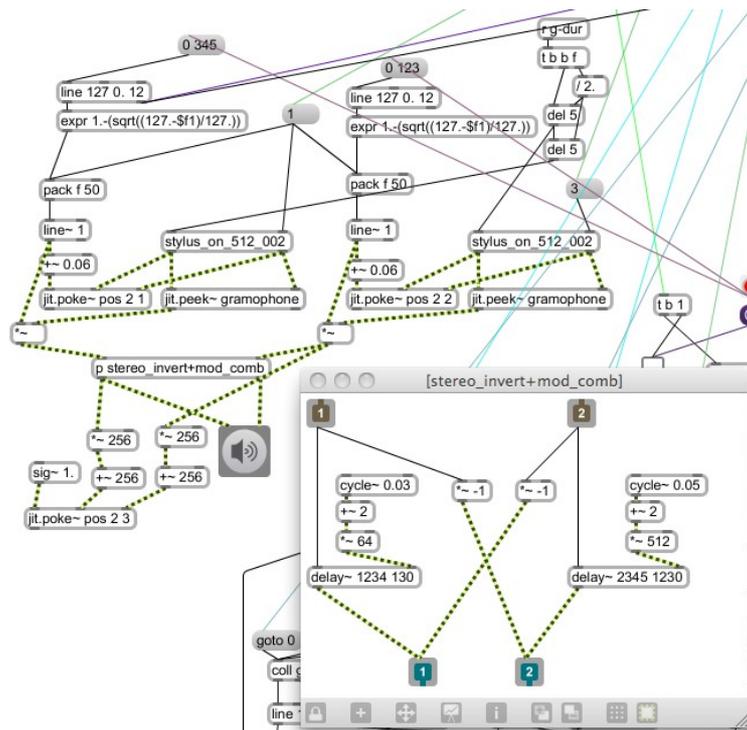


Figure 3.26: g005g\_dsp

Accumulative build up of speculative patching can lead to interesting systems that are in line with original intentions, but with this style of programming there is always a risk of introducing instability. Thorough testing of the *g005g* system, on a variety of Mac OSX computers running MaxMSP 5, was undertaken during the composition of this work. It is full of oddities and workarounds – there are things I would certainly do different if re-coding the system now – but I believe it to be stable and consistent. Only the issues forewarned about in the second paragraph at the start of §3.5 have been found to cause problems in set up for performance.

Unlike the previously described works that had **jit.pwindow** objects to display matrices on screen within the maxpat view, *g005g* uses a floating **jit.window** object that is resized and repositioned adaptively so that the display will fit fullscreen to any resolution (assuming that the x-axis is greater than the y-dimension); the areas either side of the centred square on screen are blacked, and the mouse cursor is hidden while the piece plays. To exit fullscreen and restore the cursor, the escape key can be pressed. By stretching the data surface to the height of the screen the direct cell-

to-pixel concept is no longer present. Just as the MoE pixels piece made suggestion of the screen's smallest elements without being bound to their scale, so too is this approach of stretching cells over multiple pixels on screen. It is hoped that the individual matrix cells will be visible to the human eye: quantization of curvy analogue concepts and the native circularity of their forms to the square cornered cartesian-native-computer-domain is an important element underlying this work.

### 3.5.3 Audiovisual output: open outcome and recorded examples

Video recordings of the output from *g005g* – created either with Jitter, or using the 'Screen Recording' feature of QuickTime (Apple) – have failed to capture that which is found in the native format. Discussion of video codecs, data compression, the reasons for colours appearing wrong (a yellow pixel becoming white for example), and other such technicalities that impact on the quality of a recording are beyond the scope of this project, and (although of consequence) are beside the point of a video's shortcomings. The significant point to emphasise is that video recordings of *g005g* performances – and those of earlier versions of the gramophone model composition – are not meant to represent the work itself: the software-based system, native to the MaxMSP environment, is the artefact that best represents this composition.<sup>51</sup> Video recordings do, however, serve well to document the work, and they are accessible in situations where the software itself cannot run: a video will play, for example, on a tablet or telephone device upon which the MaxMSP RunTime is not available.

Audio recordings of the work are also both useful and redundant in representing the work; the emergence, in this perspective, of a potential paradox raises some important aesthetic questions. One of these questions – already addressed in part – is to do with fixity: performances of *g005g* recorded and saved as sound-files are useful for examination of the audio output (see §3.5.4 below),

---

<sup>51</sup> From the descriptions and diagrams provided in this document there is enough information to reconstruct the composition in another environment.

but as a means of dissemination, any fixed media format completely fails to capture the essence of the work. Adam Jansch explains the problem whilst indicating its context in twentieth-century music, as exemplified in works by John Cage (Jansch, 2011, p. 50–51):

by recording an open outcome work, one particular rendition becomes frozen and exactly repeatable, thereby undermining the whole point of that work being open, as highlighted by Yasanao Tone in his writings on Cage's indeterminacy.<sup>52</sup>

Despite improvements in quality and usability throughout its life, the one characteristic of the record that has remained fixed is its fixity, a facet left untouched by progress since the wax cylinder.<sup>53</sup> To develop the idea of an open outcome record further, therefore, one must challenge this aspect of the record, imagining a new generation of record where the media does not represent the final form of a work, but instead a pool of materials, a process or an algorithm which, when put into action, produce the final form of the work.

Audiovisual recordings of an open outcome work such as *g005g* are only ever documentary evidence of the work that actually exists in another form (as software in this case). There is, in my opinion, a great danger of the recordings being mistaken for the work itself; the likely inferior audiovisual quality of such recordings, and their inherent fixity – allowing multiple viewings of identical data – detract from the aesthetic properties of the piece. While these are important issues that have been considered, they also detract from the focus of this project.

Another question to be addressed, arising from the way that such audio recordings are both useful and redundant (and the potential paradox of this), has to do with the audiovisual-ness of the *g005g* piece itself. Clearly this piece has been created with audiovisual performance in mind: the audience are presented with layered visual representations of the raw data used to make sound, of the soundmaking processes acting up upon that data in real-time, and of the sounding output itself. These visual representations were implemented in a system that was then 'composed for', more than it was 'composed with'.

During the compositional development of this work, all interaction with the audiovisual

---

52 Jansch cites: Tone, Yasanao, 'John Cage and Recording', *Leonardo*, Vol. 13, Groove, Pit and Wave: Recording, Transmission and Music (2003), 13.

53 At this point Jansch provides the following footnote (p. 51): 'Ironically, as wax cylinders were recorded in batches of different performances, and not mass-produced copies of one performance, they can be seen as the least fixed format of such records.' The gramophone disc superseded the wax cylinder, and it seems that my compositional extension to the conceptual model here has inadvertently taken an extra conceptual step back in time.

manifestations was via patching in the MaxMSP environment. Whereas that mode of working is familiar and comfortable, the intention of the project as a whole is to move toward systems of visual representation that were more directly interactive. The directive of the project is to explore the visual representations of sound in software during the composition of new music. The gramophone model, and the *g005g* piece created upon that model, are important steps toward a new way of working with sound on screen during the creative process, but they do not yet embody that goal. As was approached earlier (§3.4.5), an ideal software environment, for me, would have an integrated display GUI comprising visual representations of data, of processes, of system parameters, and so on. In such a system it would not be required for the visual gaze, and cognitive constructs with it, to switch between standard and bespoke environments during the act of composition. A version of that goal has since been realised in the creation of *sdfs*.

Thinking about the compositional process, and the practice of creating custom software environments in which to compose – new software systems within existent environments that restrict possibility down from general purpose programming to a particular set of possible interactions – has lead toward questions pertaining to what acts may be called 'composition' in contrast to those that may be called 'programming'; is it possible to distinguish the two in a practice such as is found in this project? (See §3.7).

To conclude discussion of *g005g*, fixed audio examples recorded of this open outcome work are presented (§3.5.4), and the usefulness of such media demonstrated: waveform and spectrogram visual representations are employed to support above statements made about the piece. The following section (§3.5.5) then contextualises the sonic character of *g005g* by describing how the noises created in exploration of the gramophone model patches ignited a dormant desire to have computer control over particular types of sound known from analogue electronic audio technology.

### 3.5.4 Shapes in time: using rectilinear views of recorded audio output

Sound-file recordings of the audio output of *g005g* were made<sup>54</sup> to facilitate a number of analytical motivations; three types of comparison will be described here using images – created in the GUI of Audacity<sup>55</sup> – of data from various sound-files.

#### (3.5.4.a) Difference (and the samenesses within)

Firstly, in order to confirm that each generated control sequence does indeed produce a different rhythmical structure – in terms of sounding, and silent periods – recordings of four different control sequences were made and viewed together in Audacity: Figure 3.27 shows these four different renditions displayed using the 'Spectrogram log(f)' view mode.

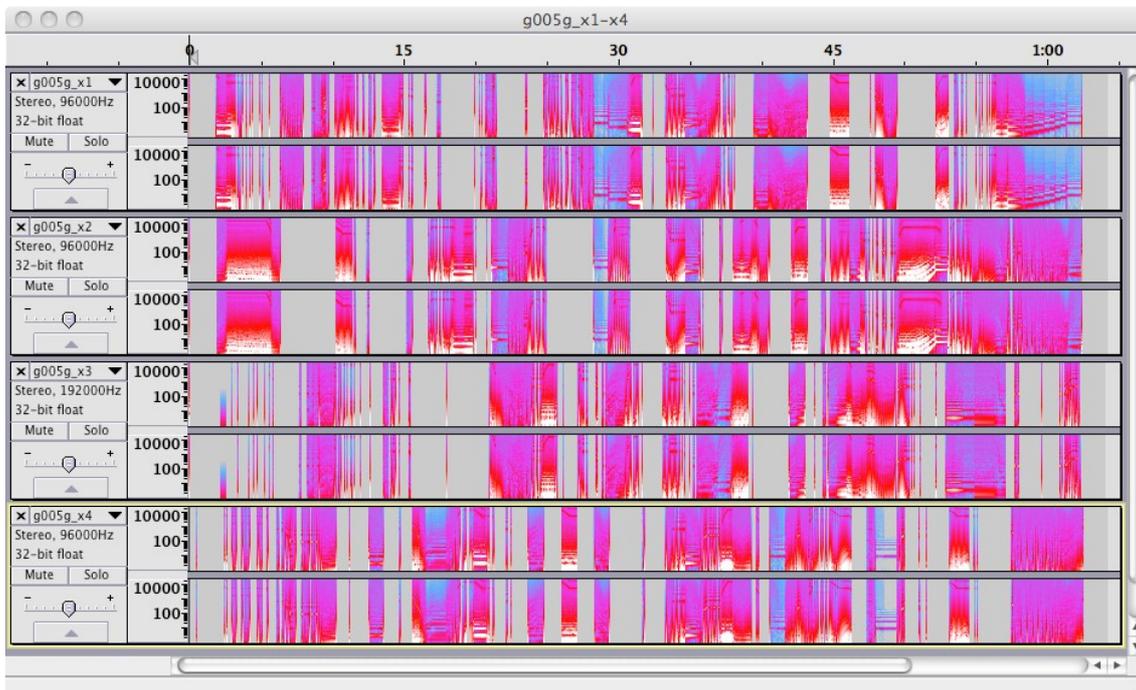


Figure 3.27: g005g\_x1-x4(spectroLog)

The periods of silence are caused by the stylus read-heads being out-of-bounds (§3.5.2), and

54 Saved with the filename component *\_x* to signify that these are exported examples of the piece (e.g. *g005g\_x1.aif*). The files are located in the folder named 'exampleAudio' within the 'model\_of\_a\_gramophone' directory.

55 Audacity is free and can be found online at <http://audacity.sourceforge.net/>

it is the randomisation of the parameters controlling the size and speed of the stylus paths that cause indeterminacy in the rhythmic structure. Timing of the control changes, however, is entirely deterministic (§3.5.1), and closer inspection of these four examples, side-by-side in the illustrated way, can reveal the underlying temporal structure that is common to all renditions. A seven second period from around the middle of the piece is shown in both of the following images: the same seven seconds of data is shown both with the spectrogram view (Figure 3.28), and with the waveform view mode (Figure 3.29). To help identify where those predetermined time-domain boundaries manifest in these recordings some 'label track' annotation is included, and the highlighted selection indicates on such period.<sup>56</sup>

---

<sup>56</sup> To make the duration boundaries more clear in this particular illustration, the four sound-files were offset within  $\approx 50$  milliseconds so that there is better visual alignment at 33 seconds. This adjustment is not so artificial as it may seem when one considers the potential build up of lag between messages in the MaxMSP environment (a 'feature' of the environment that I embrace for the 'fuzzy' conception of temporal precision that it can introduce). Such adjustments were not applied to any of the other examples.

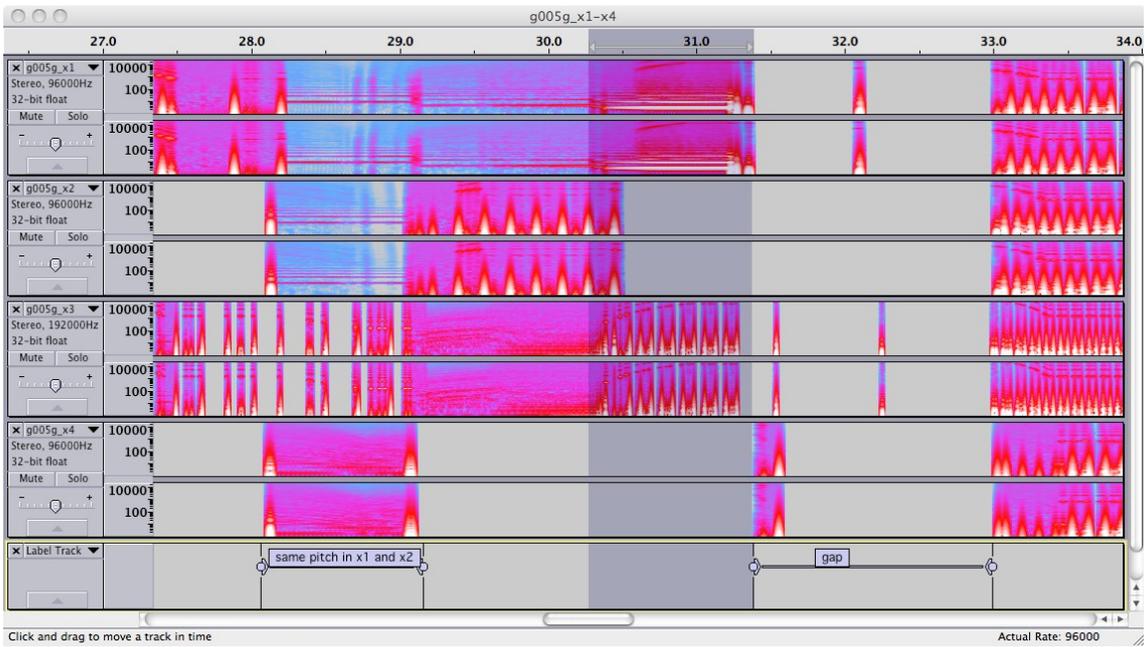


Figure 3.28: g005g\_x1-x4(spectroLog)mid7sec

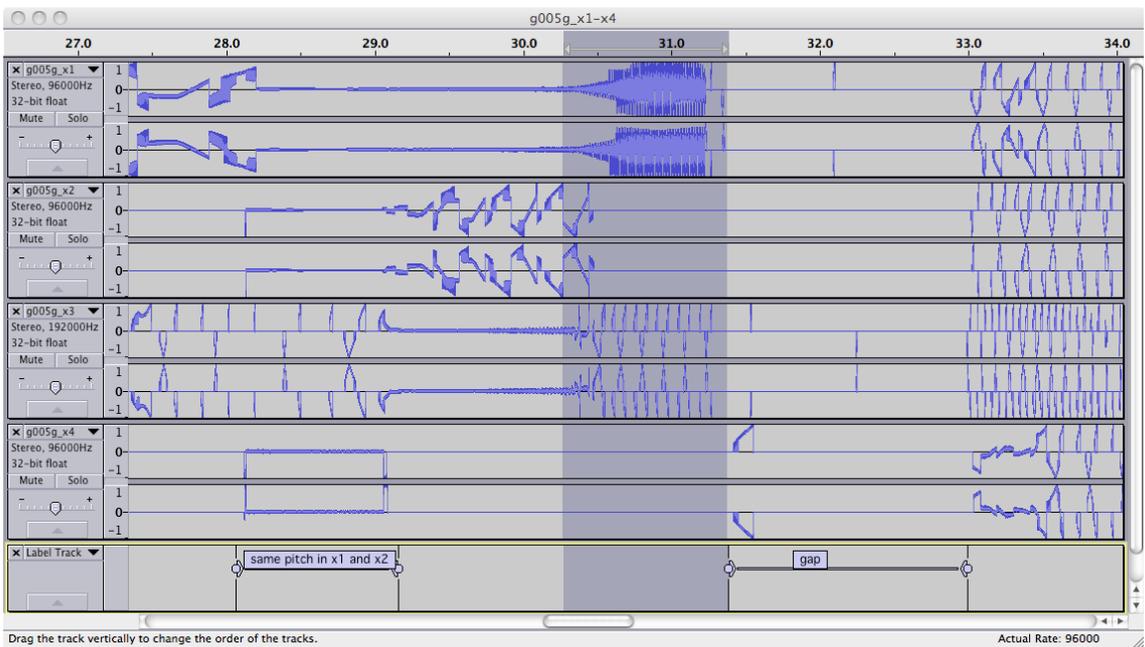


Figure 3.29: g005g\_x1-x4(wave)mid7sec

### (3.5.4.b) Sameness (and the differences within)

Secondly, to complement the above inter- rendition comparison of difference (and the samenesses within), comparisons of sameness (and the differences within) are provided by taking multiple recordings of the same control sequence (by running the performance more than once without closing the patch). Two `_x5` recordings were made, and are illustrated in Figure 3.30.

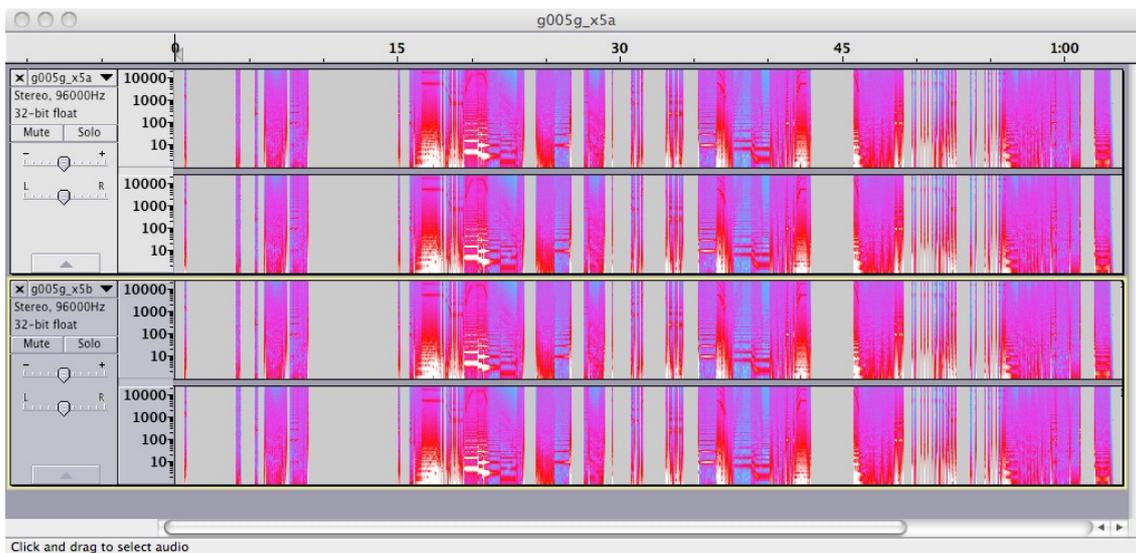


Figure 3.30: g005g\_x5ab(spectroLog)

Four recordings were then made of a sixth control sequence (`_x6`) with the aim that more instances of the same sequence may provide better opportunity to see the small differences occurring between those; in Figure 3.31 the waveform view mode of a four second section is shown, and empty label tracks are used to help visually separate the stereo sound-file tracks. The period illustrated in Figure 3.31 contains a number of slight deviations from samenesses, and although most of these would be inaudible one can know that they are there.

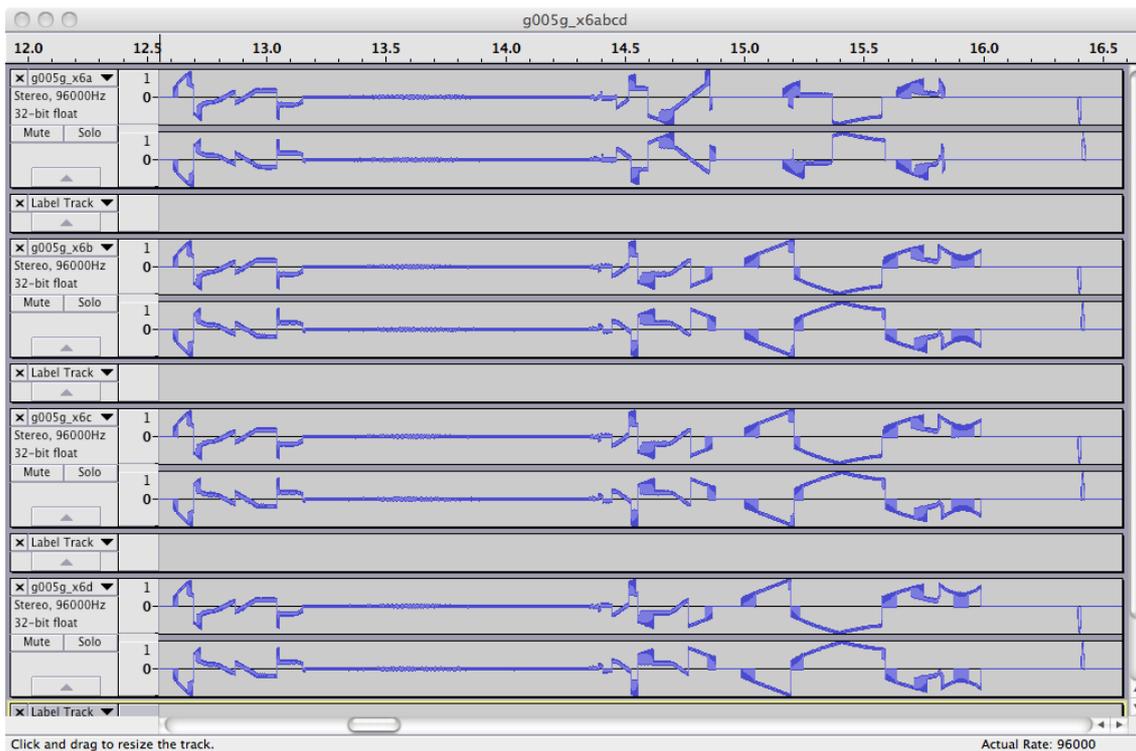


Figure 3.31: g005g\_x6abcd(wave)4sec

### (3.5.4.c) Pre- and post-DSP

The third analytical comparison utilising visual inspection of audio output data from *g005g* is to examine the affect of the DSP that is applied to the raw output of the stylus abstraction read-heads within the system (the DSP patch can be seen in Figure 3.26, at §3.5.2 above). A sound-file comprising four-channels has been recorded: the raw (DSP bypassed, thus unipolar) output of the two read-heads is written to the first two channels, while the normal (bipolar processed version of that data) output is written the other two channels of the sound-file. In Audacity these four channels are then presented as a pair of stereo-tracks, and an empty label track is used between them to provide vertical spacing for better visual separation of the pre- and post-DSP signals.

Another seven second excerpt has been chosen for this comparison: looking first at the spectrogram view (Figure 3.32) the temporal interplay of the two styli, and their differing spectral content can be seen in the upper stereo-track; the frequency-domain content (a summation of the

raw signals) is the same in both spectrograms of the lower stereo-track. Looking at that same seven seconds of data with the waveform view mode clearly (Figure 3.33) shows how the types of wave shape, that have already been shown in the examples above, relate to the raw data from the read-heads. Remember that the first stylus abstraction is set to the 'in-in' re-trigger mode, while the second uses the 'in-out' mode; these explain the vertically-cut-off shape in the uppermost waveform view in comparison to the palindromic counterpart in the waveform view of the second channel.

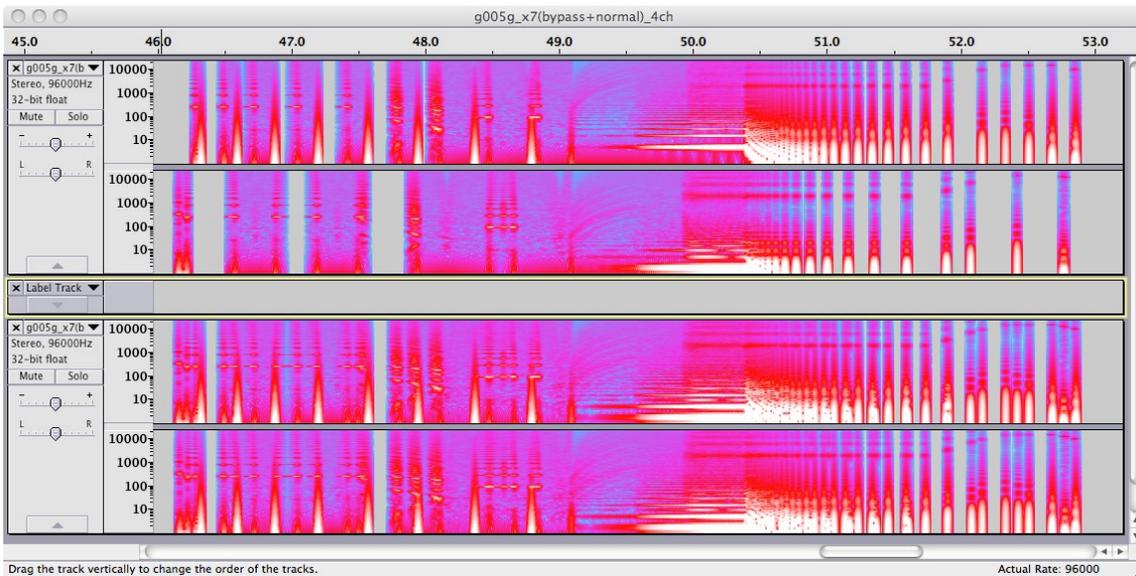


Figure 3.32: g005g\_x7(b+n)46-53(spect)

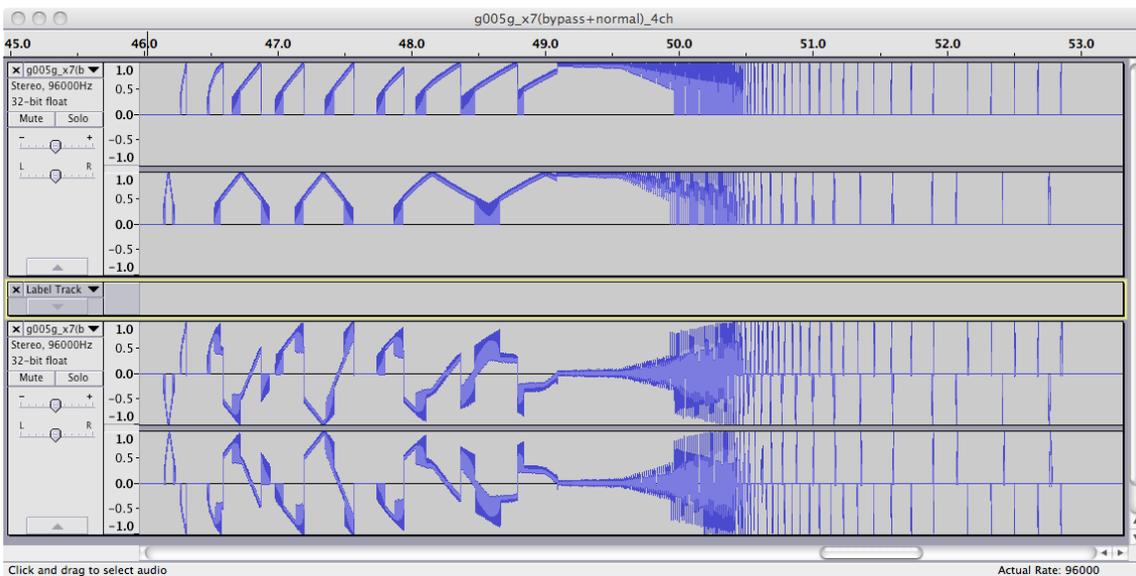


Figure 3.33: g005g\_x7(b+n)46-53(wave)

### 3.5.5 Contextualisation of the sound

The conceptual and practical steps taken to create *g005g* have been discussed in detail with particular attention to the soundmaking processes, but details of the most important factor – the sound itself – have received little mention. Performance of the piece itself can be triggered by the

reader who has access to the software, and recorded audio examples are provided, allowing the reader to hear the sound-world created and to construct their own associations from their perceptions of it. Nonetheless, to contextualise the sound of *g005g*, a fourth type of analytical comparison is formed from audio data included from my earlier electroacoustic. This comparison aims to bring into light what were aural intuitions, at the time of programming/composing *g005g*, in a way similar to those visual inspections presented above.

The joy with which this gramophone-model-based composition was pursued can, to some extent, be attributed to the similarities that I heard of the sounds being produced by the established soundmaking method in the software to the soundscapes that I know from experience with the control of feedback loops within an analogue audio electronics. Contextualisation in Collins (2006, p. 182) connects the so-called 'no input mixing' instrument to David Tudor<sup>57</sup> who, in the early days of electronic music:

used mixer matrices to combine relatively simple circuits into networks that produced sound of surprising richness and variation. Instead of simply mixing a handful of sound sources down to a stereo signal, Tudor interconnected sound modules with multiple feedback paths and output channels. The recent rise of the “no input mixing” school of internal machine feedback has exposed a new generation of musicians and listeners to the possibilities of matrix feedback.

When faced with a system that has both inputs and outputs of the same type (in terms of content or connector, or both), experimentation with feeding those outputs back to the inputs seems inevitable. In addition to the richness of its aural manifestations, circularity in signal flow has aesthetic appeal.

Feedback loops have long been held with high regard in the praxis of my musicking; I have constructed, performed with, and recorded sound from various electroacoustic systems based on signal path loops, often hybridising laptop and analogue technologies. The sounding character of one such system, a particular analogue no-input-mixer configuration without additional processing,

---

<sup>57</sup> The reader may recall (from §3.3.7) that Tudor was also involved in the earliest laser-light visualisations of sound.

is typified by a recording that I made in 2008. That recording was included, with the title *phonicMX881*, on a self-distributed audio-CDR in that year; that collection of works is now available online as a SoundCloud 'set'.<sup>58</sup> Seven short excerpts (durations in range of three to ten seconds) are included in the portfolio directories<sup>59</sup> as .wav files with name-matched .png images showing spectrogram and waveform views together for each example.

The similarity that was perceived between the sounds being made with the gramophone model and the sounds of the 'no input mixing' instrument was met with some amusement, when getting to know the capacities of that software; it seems that regardless of my aesthetic motivations or technological approach, I continue to arrive at a particular place in sound. Through such aurally observed similarities an emotional connection to my new soundmaking process was formed.

The interplay of sound in the two discrete channels of sound was a part of this observation: the differently feedback-looped channels of the stereo-mixer instrument correlating to the raw output of the two read-heads in the gramophone model system.

Conceptually, however, there is little connection from the controlled feedback soundmaking method to the *g005g* composition, and there was no actual study of the similarity before 2012. It ought to be conceded that the perceived similarity of the no-input-mixer to the gramophone model's sonic character must have had an influence on the way that the sound parameters were then controlled in the programming of the composition. The changes of state between sounding textures, for example, that are either sudden or that transition in no more than a second of time, and the combinations of continuous tones and fragmented rhythms, are all general features that are shared by both the *g005g* work and the earlier *phonicmx881* example cited.<sup>60</sup>

---

58 Find the full *phonicMX881* recording online at <https://soundcloud.com/chai/phonicmx881?in=chai/sets/ingle-nook>

59 Folder path: [...]/model\_of\_a\_gramophone/exampleAudio/phonicMX881 excerpts

60 Noted also that, while the waveforms of the *phonicmx881* examples exhibit few 'silences', the extreme high or low frequency content of some periods are, indeed, heard as 'gaps' in the sound.

Visual representation of three of the *phonicmx881* excerpts are included as Figures 3.34, 3.35, and 3.36 below; please refer to the portfolio sub-directories to hear the corresponding audio of these and others.

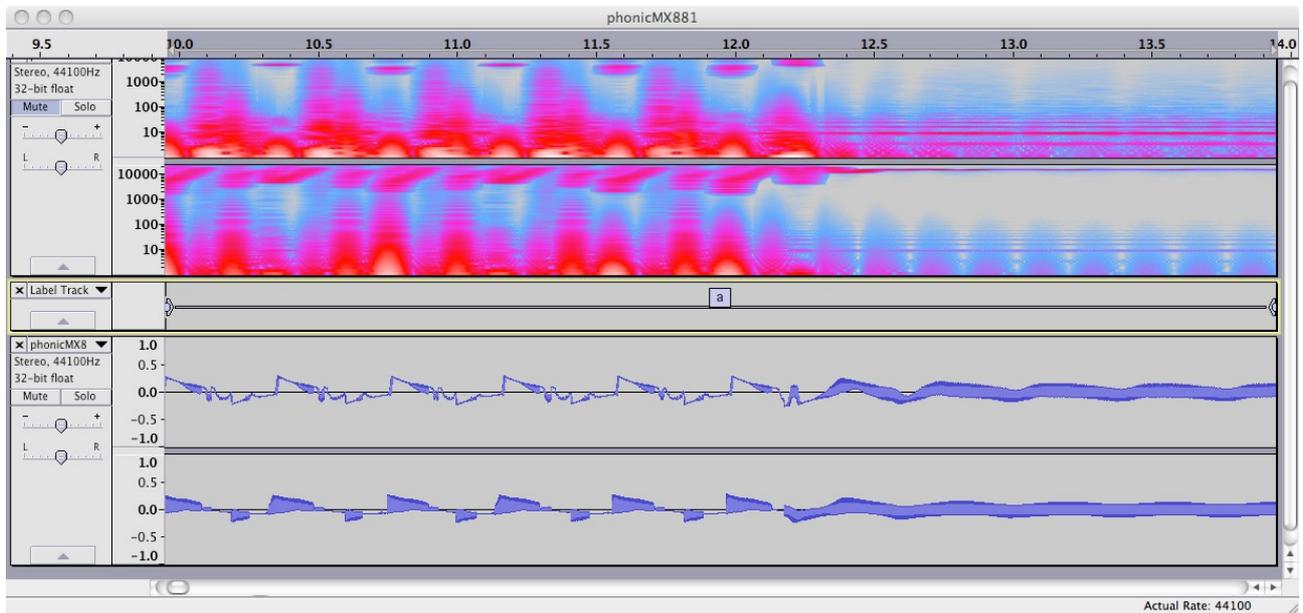


Figure 3.34: phonicMX881\_a

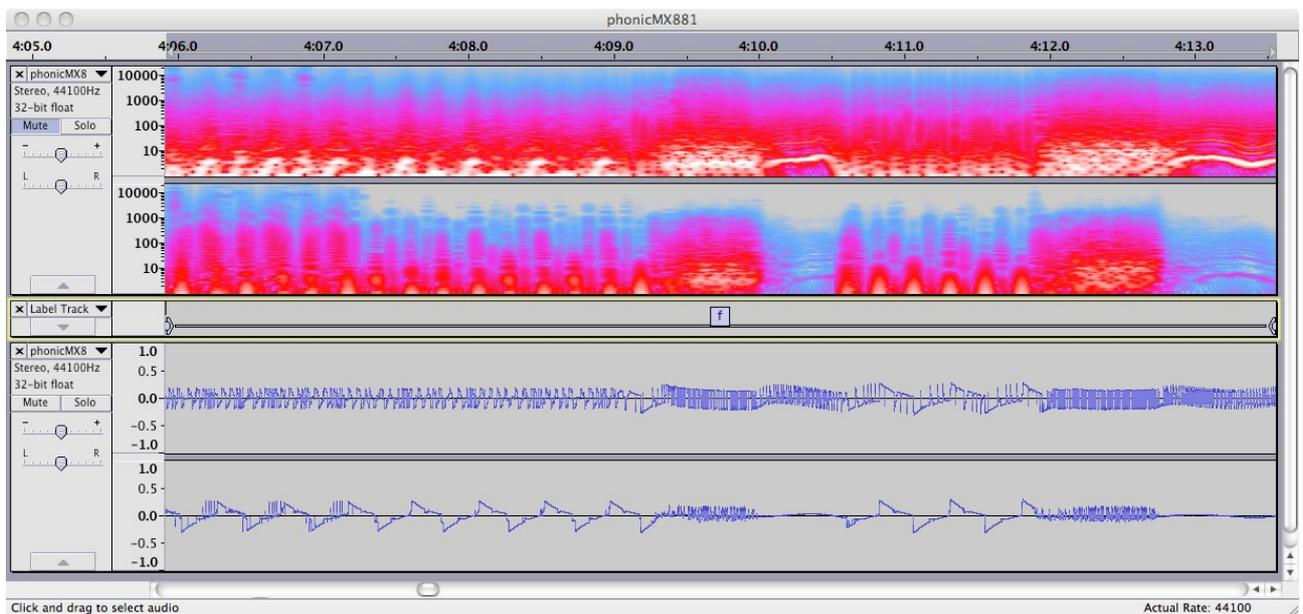


Figure 3.35: phonicMX881\_f

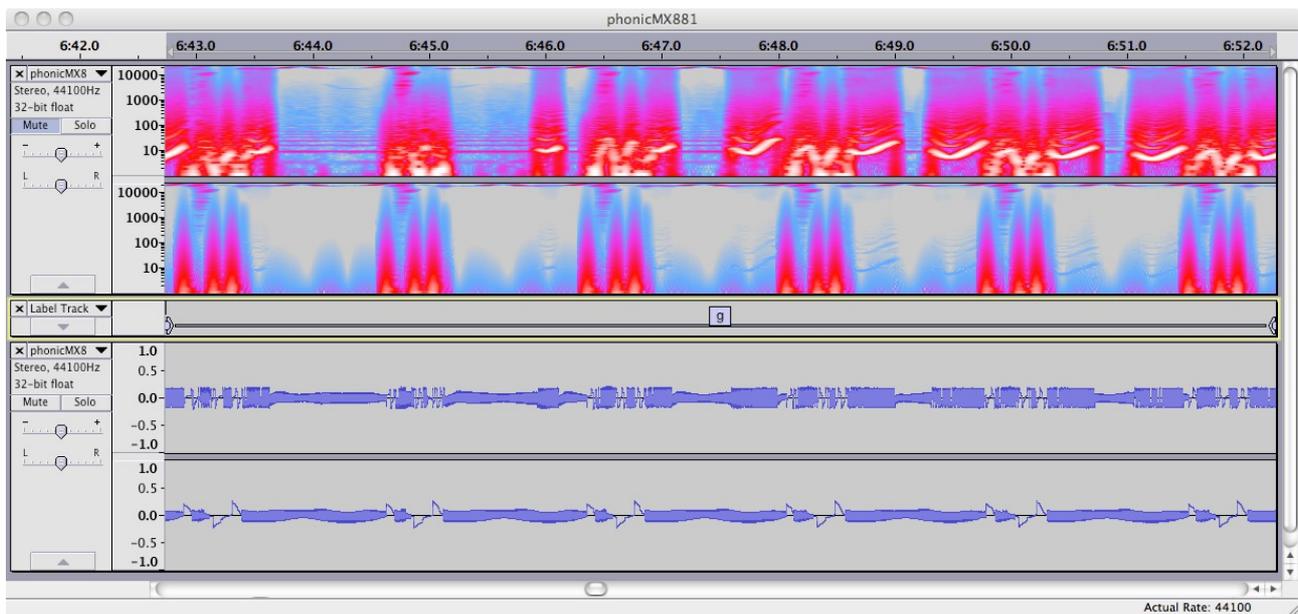


Figure 3.36: phonicMX881\_g

## 3.6 saw~onepole~noise~click~

### 3.6.1 saw~onepole~noise~click~b2.maxpat

Another piece that I composed in 2010, which combines a predetermined rhythmical structure with randomised pitch content, has the title *saw~onepole~noise~click~* (which is easier to say if one omits the tilde from each word). As with *sub synth amp map* (§3.3) the title here describes what the piece is based upon; in this case by taking the names of the MaxMSP objects that are at the heart of the soundmaking in the system. Another similarity of this work to that (§3.3) is the context of a live coding frame of reference during the genesis of the piece, though the patch has then undergone a number of revisions beyond its inception. Whereas a live coding performance version of this piece (if one were to write a text score describing such a thing) would have the structural form of elements being added incrementally, the piece as it stands has its own (open outcome) form derived by its fixed state. The final version of the piece is saved as [saw~onepole~noise~click~b2.maxpat](#).

This piece was actually started shortly before the construction of the model of the

gramophone, but, for the purposes of narrative within this document, the present work is described in the context of the later. In both *g005g* and *saw~onepole~noise~click~* there is found a primary soundmaking process that is augmented by a secondary DSP subsystem. The primary soundmaking process in *g005g* is the conceptual model of a gramophone, and the secondary DSP is the 'corrective' bi-polarising stage (see §(3.5.4.c)); in *saw~onepole~noise~click~* the primary soundmaking process is inspired by the physical modelling of the well known Karplus-Strong (K-S) algorithm (Karplus & Strong, 1983), and the secondary processing, rather than being a constant on the output, is one that takes short recordings of audio data at specific points for later playback within the piece. This is a composition that is similar to *g005g*, both in that it exists as a maxpat containing a green button on which to click to begin the performance, and that it will play differently-but-the-same each time. Each rendition of *saw~onepole~noise~click~* will yield a different set of values that manifest during the piece; soundmaking parameter values are not listed, stored, and stepped through as they are in *g005g*, but are instead chosen at the moment when they are to be acted upon.

Whereas *g005g* incorporates the concept of scored elements – the collections of value-duration pairs that are cycled through – within itself, the *saw~onepole~noise~click~* maxpat is thought of as being the entirety of the 'score' for the piece that is (to some extent) human readable (though not humanly playable in any practical sense). It is a relatively small patch in which all the objects and connections can be seen within a single view (see Figure 3.37 below). A suitably experienced reader may construct a mental projection of how the piece will sound just by 'reading' the visual representation of the work that is the patch, in much the same way that a classically trained instrumentalist may imagine the sounding of notes from the silent reading of a CMN score.

The visual manifestation of the maxpat, as in Figure 3.37, represents the sound-work which is manifest as audio signals (stereo) in a DAC of the computer that is running the software to



## 3.6.2 The domain dichotomy

One of the aesthetic motivations explored by this piece is to do with the dichotomy of time-domain and frequency-domain thinking when it comes to making and processing sound within software. Lines of thought on this are found at different levels of abstraction: in relation to the traditional conception of music comprising the primary elements of rhythm and pitch; that one may contemplate, and play upon, the difference between units of duration and units of frequency (simultaneously using a value,  $x$ , both as  $x$  milliseconds and as  $x$  Hertz); the sound synthesis in this piece combines both impulse-based and sustained-tone-based signals. Further variations of the idea follow in the description below, and the domain dichotomy theme remains as an undercurrent to all subsequent works described by this thesis.

For each of the four MaxMSP objects that are named in the title of this piece (**saw~**, **onepole~**, **noise~**, and **click~**) Figure 3.38 demonstrates some time- and frequency-domain conceptual abstractions :

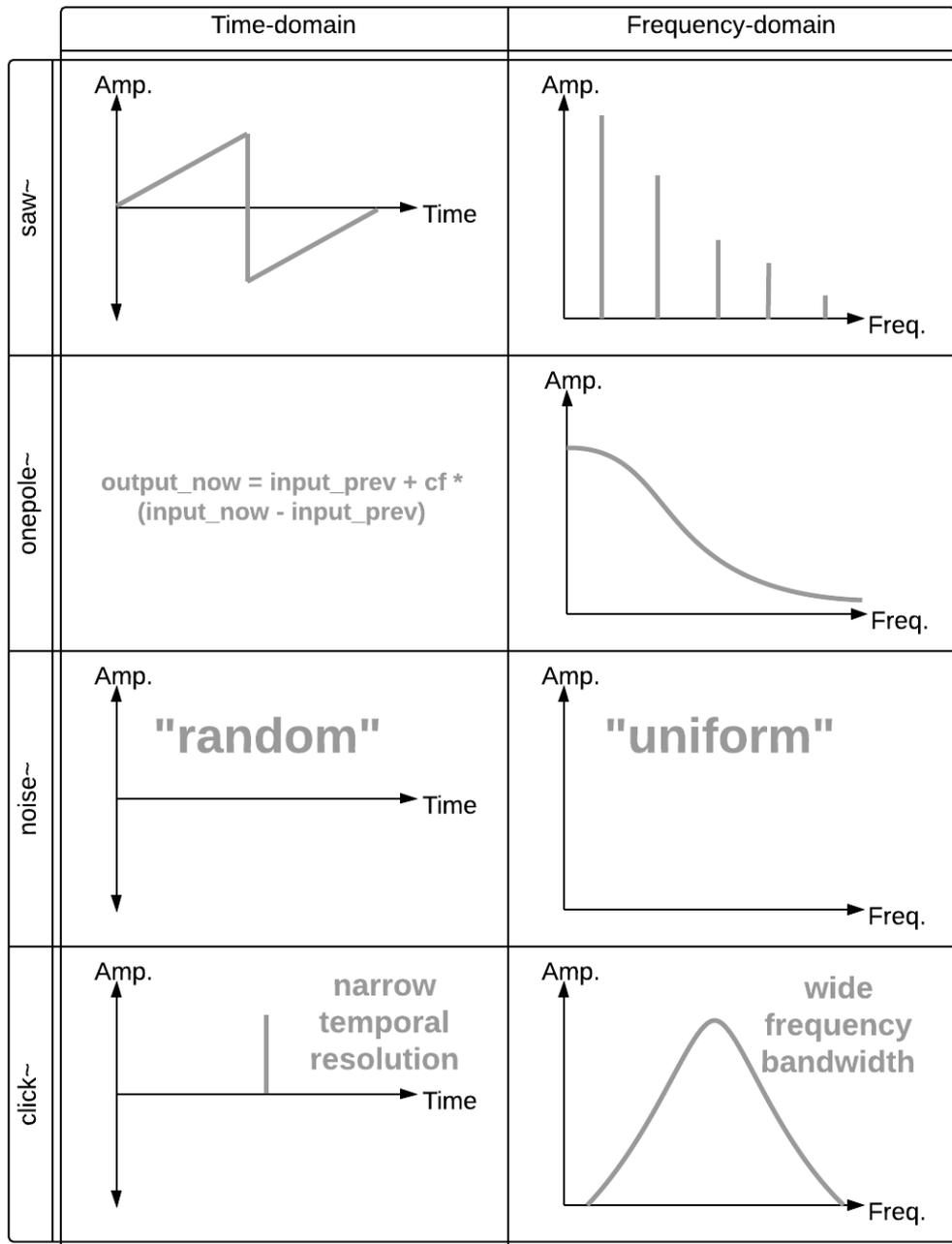


Figure 3.38: saw~onepole~noise~click~(domainConcept)

### 3.6.3 Temporal structure

There are three **metro** objects that cause things to happen within the piece; they may be referred to as A (**metro 1254**), B (**metro 1514**), and C (**metro 125**).<sup>61</sup> The bang messages that are output both by A and by B are used to toggle the on/off state of C. The output of C has four destinations; three comprise the triggering of a sounding event (see §3.6.4), and the fourth is used to determine whether or not more than 600 milliseconds have passed since the previous output of C. If the delay between C bangs is greater than that, then a **counter** (bound to cycle from one to two to three to one and so on) will be triggered to increment.

When the output of the **counter** is one, audio data (taken from a point within the main soundmaking section) begins to be recorded to **buffer~ temp**; when the **counter** is two that recording is stopped and the duration of that recording is stored; on reaching three the **counter** will first output a 'carry count', which is the number of times that the counter has reached its maximum since being reset when the piece was started: if the carry count is five then the 'end of line' **gate** is opened, and in any case the replaying of the recorded audio data is triggered. Playback of the recorded audio data will always be 'backwards' (this is what the duration value stored at counter-two is used for), but may either be at the same rate of play at which it was recorded or at half that speed. If the 'end of line' gate is open at the end of the audio data playback then the piece ends simply by stopping the DSP (which implies the assumption that MaxMSP is not being used for anything other than the performance of this piece).

In the 'saw~onepole~noise~click~' folder of the portfolio directories there is an 'eventTrace' text file that was saved from the Max window after running [saw~onepole~noise~click~b2\(hacked\).maxpat](#) which has had **print** objects added at various points in the patch, including to give a trace of the recorded durations from the buffer DSP section

<sup>61</sup> On the one hand, the time interval values given to these three **metro** objects are arbitrary; on the other hand, the temporal resolutions were specifically selected for the type of rhythmical interaction that I knew they would create.

described above. The first 20 lines of that text file are as follows:

```
START: bang
A: bang
B: bang
A: bang
C: bang
C: bang
C: bang
B: bang
counter: 1
A: bang
C: bang
C: bang
C: bang
C: bang
C: bang
B: bang
counter: 2
recDur: 1502.667099
A: bang
C: bang
```

The starting bang switches both A and B on which means that C is both switched on and then off again within the same 'moment'. C is switched on again by A, 1.254 seconds later, and is able to bang three times before B switches it off. The pause at this point is long enough to trigger the counter, and recording begins. After the next bang from A, C triggers five times before B fires and again the counter ticks; line 18 of the 'eventTrace' text tells us that just over 1.5 seconds of audio data – capturing the output of the last five sounding events – has been recorded. Onward goes the procession, and the timing is always the same. The distribution, and values of, the 'recDur' lines are of interest in terms of the compositional decision to end the piece after 15 ticks of the counter, whereas the first version of the piece stops after 30.

## 3.6.4 Sounding events

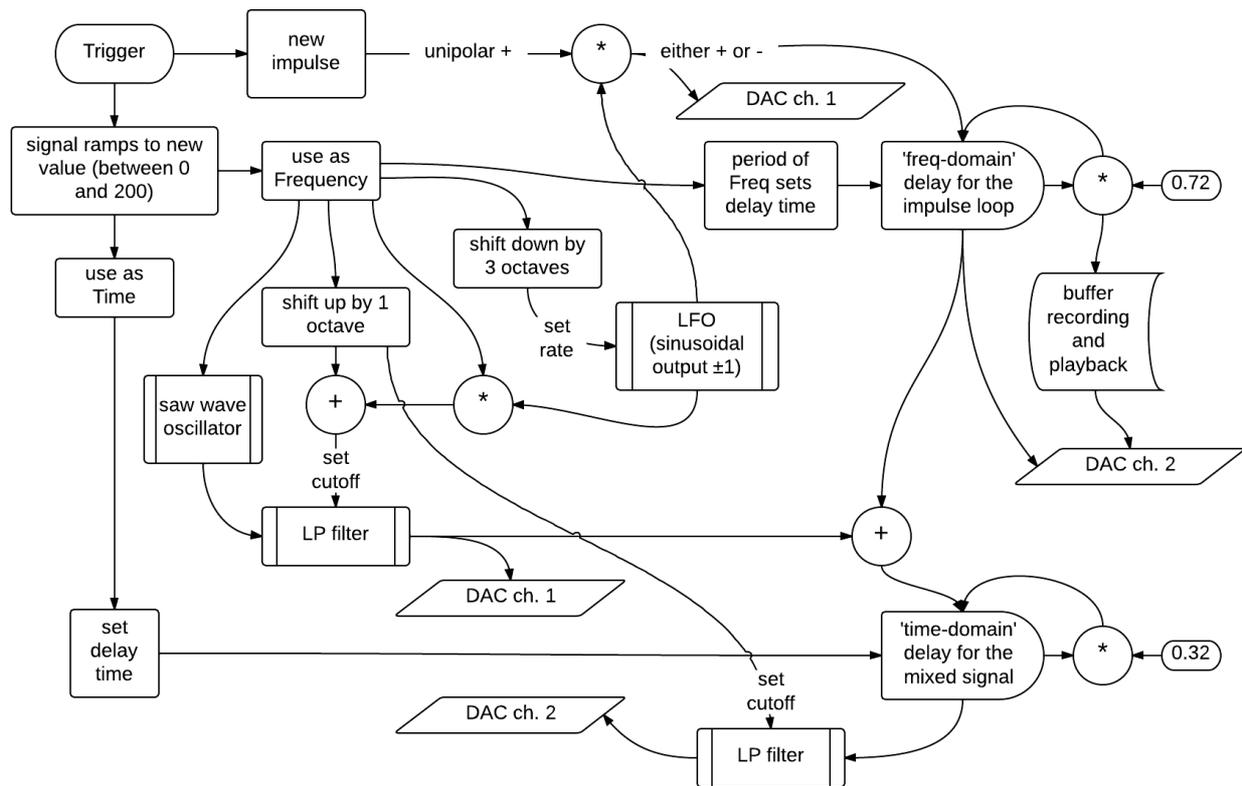
### (3.6.4.a) Triggering sound

The first three destinations of each bang message from metronome C trigger what may, in traditional terms, together be called a new note within the piece; the three parts of these sounding events are: (1) an onset ramp duration in range 10 to 19 milliseconds, selected via **random** and held for use with **line~**; (2) a value in range 0 to 200, selected via **noise~** and ramped to by **line~** over time of the held duration: there are six destinations for the signal output from **line~** which is interpreted both as a frequency and as a duration (see below); finally (3) an impulse of 64 samples in range zero to one, selected via **random**, is used with **click~** to give each sounding event a percussive attack that is extended through a short and pitched decay by a K-S inspired model. The synthesis can be thought of as having two 'inputs': the impulse and the continuous control signal.

### (3.6.4.b) Synthesis flow

Above were introduced the roles of the **noise~** and **click~** objects; the roles of **saw~** and **onepole~** are illustrated, below, in a flow-chart representation of the main soundmaking part of the *saw~onepole~noise~click~* system (Figure 3.39): **saw~** is the 'saw wave oscillator', and **onepole~** is the low-pass (LP) filter, of which there are two. There are also two delay lines within the system; in Figure 3.39 they are described one as being of the frequency-domain, and the other of the time-domain. The former uses the control signal as a frequency value and takes the period of that frequency as the delay time; the later uses the control signal directly to set the delay time.

Due to the interactivity of the elements, the flow-chart representation of the synthesis here appears almost as complicated as the maxpat itself (Figure 3.37, above), but the flow-chart does make more explicit some of the thinking behind work by showing the algorithm at a higher level of abstraction.



### 3.6.5 Scored recordings

It has already been stated that the maxpat constitutes a complete 'score' for this piece of music; as a visual description of the work it conveys precise information for the processes of soundmaking and temporal organisation. Pitch, on the other hand, is only specified as a fundamental frequency bandwidth (in range 0 to 200) that is quantized only to the resolution of the DSP bit-depth.

A copy of the maxpat was modified to give an output of the sounding events approximated as MIDI notes. Three renditions of the piece were then recorded as MIDI data into Cubase (Steinberg), and from which a CMN score was prepared<sup>62</sup> in order to show how those three recordings have identical rhythmic content. An excerpt, from the start of the piece, is shown in Figure 3.40.

<sup>62</sup> Printed to pdf file, see [saw~onepole~noise~click~b.pdf](#) within the portfolio directory.

the start of 'saw~onepole~noise~click~' by samuel.freeman (22 March 2010) recorded as MIDI data three times to show how, although pitches are randomised, the rhythm is determinate.

Figure 3.40: saw~onepole~noise~click~b\_midi\_x

Perhaps it was polemics that motivated the production of a CMN score from this piece; not only was I curious to see how the DAW software would render the information given to it, but also to see how my feelings toward the musical assumptions that underpin the work would be affected. The CMN produced includes only representation of the rhythmical content and the approximate pitches that were played in the three renditions that were recorded as MIDI data. For demonstration of the rhythmical sameness between recordings the CMN serves well; maybe even better than would waveform or spectrogram representations, but none of these were used during the composition of the work.

As a piece of computer music, *saw~onepole~noise~click~* stands as an exploration of a generative rhythmic engine playing a 'software instrument' which has conceptual depths that exceed

those of the programming which expresses them. Within the technical exploration of controlled soundmaking is the aesthetic engagement with the domain dichotomy (§3.6.2), and beyond that the additional dichotomy of score and instrument is brought to attention.

## 3.7 Software as substance

To explore aspects of this thing we call software when it is viewed as the medium of new music, and bringing conclusion to the chapter, this section combines thoughts that were first presented in the 2010 end of year report – particularly the start of §3.7.1 and the model of composition described within §3.7.3 – with perspectives that are held at the end of the project in 2013.

A definition of the word software:

Term used for the instructions or programs executed by a computer, as opposed to the physical hardware that enables the machine to follow them. The comparison of a psychological description of a person to a software description of a machine is exploited in functionalism.  
(The Oxford Dictionary of Philosophy (2 rev. ed.), Blackburn, 2008)

### 3.7.1 One-line programs

To some extent we can often look upon software itself as substantive of an artistic work, as for example with the following one-line BASIC program written by Noah Vawter for the Commodore 64 (C64), which was discussed by Nick Montfort in July 2010, after the release in November 2009 by Keith Fullerton Whitman of an 18 minute recording of the code running on an iPhone emulator of the C64:

```
10 poke 54272+int(rnd(1)*25),int(rnd(1)*256) : goto 10
```

The program code is given as the title of the piece and, by seeding the random number generation, it will (Montfort writes) always produce the same sequence of numbers to 'poke' into the 'memory locations beginning at 54272 [which] are mapped on the Commodore 64 to the registers of the SID (Sound Interface Device)' (Montfort, 2010).

We might think of this one-line program as a type of score to be performed by the C64, or perhaps it is a software-instrument native to the C64 which is then performed by the human user who must type RUN and press Enter to invoke the sound. I have found, while running this program within Power64 (Lieger, 2006),<sup>63</sup> that pressing *Esc* will 'freeze' the audio to give a sustained sonority determined by the contents of the SID at that moment, and that sound will sustain until RUN is entered and the one-line software loop continues. This such behaviour goes little way toward that of a musically expressive instrument; Vawter's one-line BASIC program for the C64 is a limited example of software as substance, but it begins to illustrate some of the ambiguity which can accompany the question of exactly what constitutes a piece of computer music. It also serves to remind that for as long as there has been the widespread availability of programmable computers, those computers have been used for making music; see, for example, *Micro-music for the Commodore 64 and BBC computer* (Herman, 1985). Aesthetic appreciation of this piece by Vawter, for me, includes reflection on the influence of the Commodore 64 on my perception of technologies as a child: I remember using a C64 in the family home, and that it was a thing of some wonder that the computer would 'listen to an audio tape-cassette' in order to load programs.

Another example of the one-line program paradigm is found in the installation *A Show Case for SC Tweets* [sic] (Bartetzki, 2011) exhibited at the 2011 International Computer Music Conference (ICMC), featuring the audio and visual display of one-line programs for SuperCollider:<sup>64</sup> the programme note for that installation suggesting that the SuperCollider tweets – meaning 'pieces of code limited to the short length of 140 characters' and sent via twitter.com – might be thought of as 'mini compositions, smart algorithms, complex rhythms, compact soundscapes, evolving textures, syntactic adventures...', and relates the brevity of the program code

---

63 Power64 is a Commodore 64 emulator that I downloaded (20100729) from <http://www.zzap64.co.uk/c64/c64emulators.html>.

64 SuperCollider was originally developed by James McCartney (1996), 'and is now an open source (GPL) project maintained and developed by various people' (<http://supercollider.sourceforge.net/>, accessed 20130818).

to 'when composers decide to use only limited base material (a sound, a theme) as a starting point but make rich and complex pieces out of them'. Bartetzki's description supports the notion of code as the substance of a composition – of software as the medium of the work; further agreement is found with the idea that composition is about choosing limits, and then working within, and against, and around them.

The pursuit of creating new worlds within worlds through computer programming (as mentioned in the introduction to this chapter), is part of a compositional strategy that seeks to narrow the realm of possibilities that a computer promises. All software-based systems are limited, and not least by the assumptions that underly the forms of representation that are used within them. The six studies described in this chapter have brought question to some of the assumptions that are involved in visual representation of sound in software, on screen, during the processes of music composition; they, and their findings, form the bedrock upon which the later works of the project emerge both technically and aesthetically.

A final example of one-line programs is included here because it is one which also features direct visual representation of the data being output as audio (which is an important idea within this project); *bitop videos* contribute to what Aymeric Mansoux describes as a 'bitop chiptune craze' of the time (Mansoux, 2011):<sup>65</sup>

Unfortunately Youtube was not very happy when I tried to use the shiftop codes as video titles, and it was refused even inside videos description, so here is the detailed playlist with authors:

1. [\(\(t>>1%128\)+20\)\\*3\\*t>>14\\*t>>18](#) (harism)
2. [t\\*\(\(t>>9\)&10\)|\(\(t>>11\)&24\)^\(\(t>>10\)&15&\(t>>15\)\)\)](#) (tangent128)
3. [t\\*5&\(t>>7\)|t\\*3&\(t\\*4>>10\)](#) (miiro)
4. [\(\(t\\*\(t>>8|t>>9\)&46&t>>8\)\)^t&t>>13|t>>6](#) (xpansive)
5. [\(t\\*\(t>>5|t>>8\)\)>>\(t>>16\)](#) (tejeez)
6. [\(\(t\\*\(“36364689”\[t>>13&7\]&15\)\)/12&128\)+\(\(\(\(t>>12\)^t>>12\)-2\)%11\\*t\)/4|t>>13\)&127](#) (ryg) [...]
7. [\(t\\*t/256\)&\(t>>\(\(t/1024\)%16\)\)^t%64\\*\(0xC0D3DE4D69>>\(t>>9&30\)&t%32\)\\*t>>18](#) (ultrageranium)

65 The hyperlinks – linking to the video examples of the works with titles that are in fact the software code of the piece – are preserved in this quotation as they were within the cited source. It is of interest to note the issues, reported by Mansoux, in properly presenting these works through YouTube; but it is beyond the scope of this project to discuss those matters here.

### 3.7.2 Expressivity and technē in programming and composition

Toward answering the questions posed near the end of §3.5.3 – of what acts may be called 'composition' in contrast to those that may be called 'programming' – it seems false, now, to distinguish the two when the software that one is programming is specifically intended to facilitate the composition of a piece music. One may counter that perspective with the conception of a computer music programmer as a digital-luthier of sorts, and there is merit to that view, but the separation that is there suggested – between 'the instrument' and 'the piece of music' involving that instrument – fails to fit with conceptions of the art that have developed during my practice.

Programming itself can be thought of as a form of creative expressivity. In live coding, the real-time manipulation of software, at the program coder rather than end-user level, is a key aspect of the paradigm; the software medium is exposed as a malleable substance that can be shaped extemporaneously. In a different context, Mark Bokowiec (2011, p. 41) has described how, in the composition of a work, programming is one of several different forms<sup>66</sup> of expressivity that

are inter-related and interact with each other in various ways and degrees. An awareness of the interconnectivity of principle forms of expressivity, their interaction and influence on each other, shapes the compositional [process].

To contextualise my own work, awareness is also brought to the interrelatedness of compositional processes, aesthetics, and the technology that is employed in the expression of these as music. While the primary focus of this project is on soundmaking software as the medium through which creativity is expressed, it is recognised that the field of computer music exists within the wider context of electronic/electroacoustic music, and that the historical discourse of these is inseparable from the technological basis of praxis.

---

66 For Bokowiec, composing *V'Oct(Ritual)* with the Bodycoder system, there were four integrated forms of expressivity: 'sonic (electroacoustic), programmed, gestural (kinaesonic) [and] vocal'. (Bokowiec, 2011, p. 41)

The Greek word *technē* is defined as 'the knowledge of how to do things and make things' (Blackburn, 2008), and contextualisation is provided by an article by Peter Manning (2006), titled: 'The significance of *techné* in understanding the art and practice of electroacoustic composition'. Manning cites a 1958 paper by T. Adorno as discussing this concept in a relevant context (Manning, 2006, p. 82–83):

The meaning of the Greek word *techné* from which both 'technique' and 'technology' are derived offers an indication of the unity of this concept with art. If art is the external representation of something internal, the concept of technique embraces everything which pertains to the realisation of that interior substance. In the case of music, not only the realisation of spiritual substance in the score is involved, but the transformation which makes this score accessible to sensory perception as well. In short, both production and reproduction are involved. Musical technique embraces the totality of all musical means: the organisation of the substance itself and its transformation into a physical phenomenon. (Adorno 1958: 11)<sup>67</sup>

Manning (*ibid.*, p. 83) cites both Pierre Boulez (1977)<sup>68</sup> and Agostino Di Scipio (1995)<sup>69</sup> as having 'identified the fundamental dilemma which continually confronts electroacoustic composers' which is a question of choice: either to make use of available technologies in the realisation of compositional ideas, or (in Di Scipio's words) to 'design the tools that are necessary to realise [one's] own idea of composition?'; Boulez is cited as criticising 'the tendency to favour the former approach'. From the outset of this project, my directive has been rooted in the latter of those choices, and the pieces that have been described in this chapter, demonstrate some of the investigative steps taken toward formalising just what my own idea of composition is within this project.

Contemporary projects coming out of IRCAM, such as *InkSplorer* (Garcia et al., 2011), can be seen as being inline with the mission that was set out by Boulez to 'place the technology of the computer centre-stage in the quest [...] for a fully integrated environment' (Manning, 2006, p. 82).

<sup>67</sup> Adorno, T. 1958. Musik und Technik. *Gravesaner Blätter* 4: 11–12.

<sup>68</sup> The essay – cited by Manning as: Boulez, P. 1977. Technology and the composer. Reproduced in S. Emmerson (ed.) *The Language of Electroacoustic Music*, pp. 5–14. London: Macmillan Press, 1986. – is available online as published in *Leonardo* (Boulez, 1978), and that is the version which appears in my bibliography.

<sup>69</sup> Di Scipio, A. 1995. Inseparable models of materials and of musical design in electroacoustic and computer music. *Journal of New Music Research* 24: 34–50.

In 1977, Boulez wrote: 'What is absolutely necessary is that we should move towards global, generalizable solutions' (appears as: Boulez, 1978, p. 62), but that was written at a time technologically removed from the computer as we know it today. Boulez (*ibid.*) also writes that:

one can imagine possible works where material and idea are brought to coincide by the final, instantaneous operation that gives them a true, provisional existence [...] Certainly, the finite categories within which we are still accustomed to evolve will offer less interest when this dizzying prospect opens up: of a stored-up potential creating instant originality.  
Before we reach that point, the effort will either be collective or it will not be all. No individual, however gifted, could produce a solution to all the problems posed by the present evolution of musical expression.

Perhaps then, one may ask if that point has been reached by now? While collective efforts continue to further technological gains for all, the individual now has at their disposal the means by which to go beyond the surface of technological availability and investigate real-world manifestations of their imagination. Solution to 'all the problems' is not my objective; through critical engagement with software as substance, I seek to express something of my own *technē* in the composition of new works: software is conceptualised as the medium through which my computer music is made, and programming thus forms a significant part of what I do in my compositional practice. Whereas Boulez wrote in opposition to 'particular solutions that somehow remain the composer's personal property' (*ibid.*) it is precisely my 'solutions' that I wish to share as contributions to knowledge.

### 3.7.3 Composing as an artist-programmer

It seems appropriate to accept applicability of the term artist-programmer: the artist-programmer is described, by Alex McLean (2011, p. 16), as 'directly engaging and interacting with their code as an integral aspect of their work'. The attention of McLean's thesis is directed (p. 17):

towards the intimate relationship between artist-programmers and their systems of language, understanding programming as a human interaction. The artist is considered for their role as a visionary, in exploring and extending their human experience. [McLean considers] technological support in terms of extending their vision through processes of perception and reflection in bricolage creativity.

### (3.7.3.a) Bricolage programming

Much of the work in this project, in retrospect of its development, can be seen to fit closely with the conception of bricolage creativity presented by McLean. Figure 3.41, below, is from McLean's 'Figure 6.2: The process of action and reaction in bricolage programming', and of which McLean (2011, p. 122) writes that:

[it] characterises bricolage programming as a creative feedback loop encompassing the written algorithm, its interpretation, and the programmer's perception and reaction to its output or behaviour. Creative feedback loops are far from unique to programming, but the addition of the algorithmic component makes an additional inner loop explicit between the programmer and their text. At the beginning, the programmer may have a half-formed concept, which only reaches internal consistency through the process of being expressed as an algorithm. The inner loop is where the programmer elaborates upon their imagination of what might be, and the outer where this trajectory is grounded in the pragmatics of what they have actually made.

There are two points to make here: the first is to compare that cyclical model of bricolage programming to a cyclical model of composition devised during this project, and the second is to pick up on McLeans's choice of word in referring to 'the programmer and their text' (see §3.7.4 below).

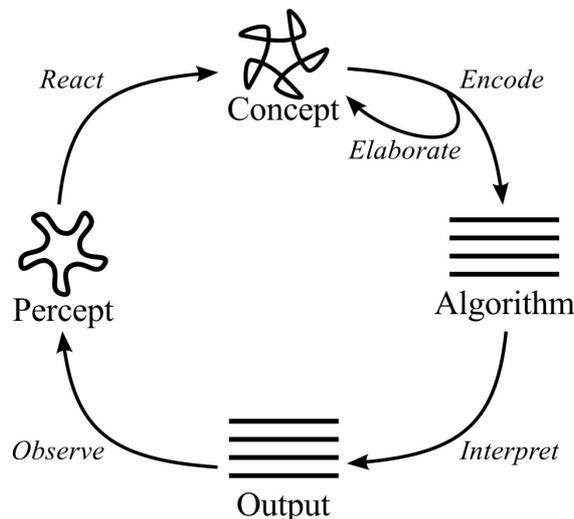


Figure 3.41: McLean\_bricolageprogramming

### (3.7.3.b) Composition model

At the end of the first year of this project, I drafted a diagrammatic representation of a compositional model: it was the combination of (A) an introspective analysis of the ways I had previously been using software in composition, with (B) an idea of how I would like to go on to work in future. In other words, and relating back to the questions of *technē* above, the model was created as an attempt to clarify what my own idea of composition may be. The model has been redrawn as Figure 3.42, below. While the contents of this model had been somewhat informally defined – and were then adhered to even more loosely as the project progressed – it provides significant insight to my thinking at that time, which can now be contextualised as comparable to the more generalised understanding of creative practices provided by McLean.

My model identifies six stages of a cyclical process, beginning with abstract thought and leading to an experience of sound that, in turn, re-triggers the first item of the cycle:

1. abstract thought;
2. specification of concepts;
3. implementation of specifications (as visual representations);
4. control of data via visual representations;
5. render performance of data control;
6. temporal experience of the performance.

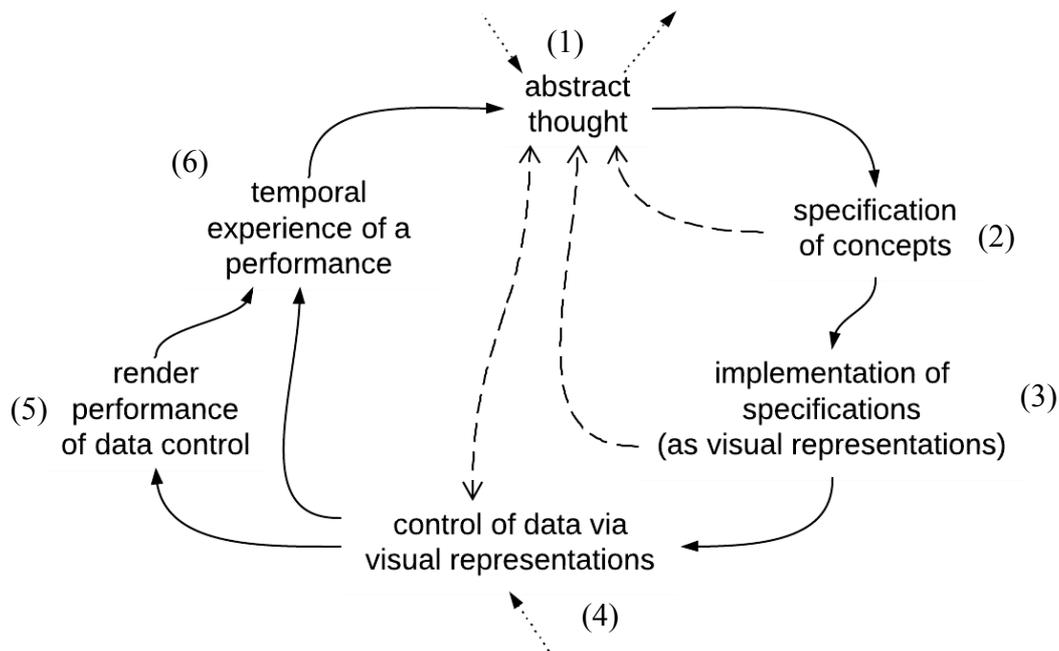


Figure 3.42: CompositionModel

Set as text, this cyclical model might read as follows: Abstract thought (1) can lead to the specification of particular concepts (2); these specifications are implemented in software as visual representations (3); those representations are controlled to manipulate data within the software system (4); a performance of that data control is rendered – though not necessarily in real-time (5); the rendered performance is then experienced as a real-time happening (6), and the experience of that feeds more abstract thought (1). The cycle may continue as a closed loop, but do notice that there are points of external connection (at 1 and 4) through which undefined factors may add to or divert the flow of this process. There are also paths of flow variance within the model that imply processes of internal feedback (via 1) at various stages along the cycle. A bypass of performance rendering (5) is shown because the (1–4–6–1) loop may be seen as the typical cycle of interactivity when experimenting with software, but without editing it, and before making the compositional decisions for control that based on that experience.

### (3.7.3.c) Comparison of cyclical models

The autonomous and introspective origins of my composition model (Figure 3.42) are emphasised while the superimposition of it and the cited 'process of action and reaction in bricolage programming' is demonstrated in Figure 3.43:

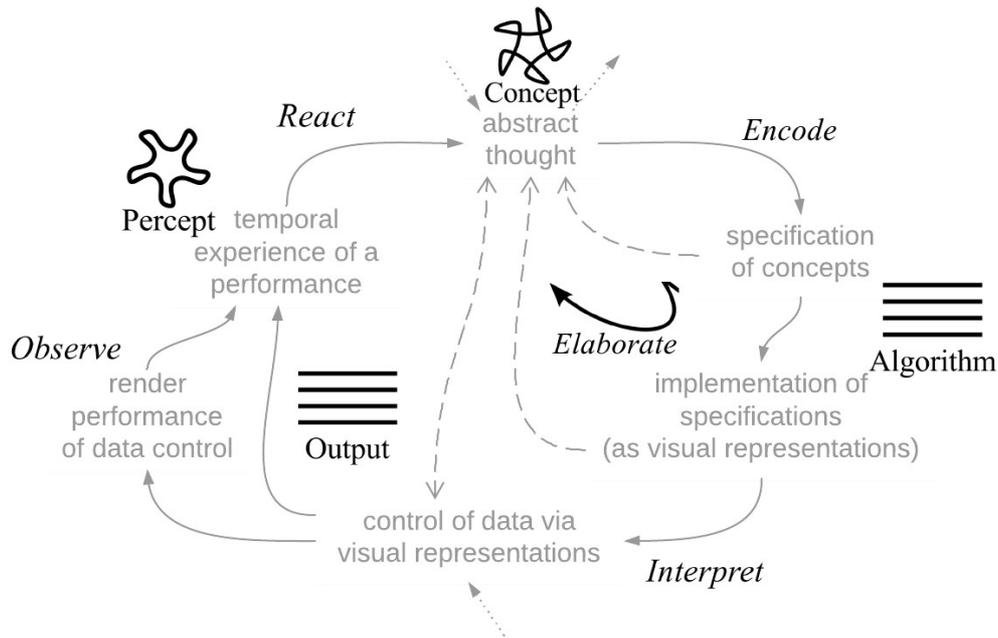


Figure 3.43: model of composition + model of bricolage programming

From my aesthetically focused perspective, the close resemblance in the spatial arrangement of conceptual items in each model is appreciated as somehow reaffirming my practice based research methods in the context of more formally recognised creative processes. A structured analysis of types of creative process that are, apparently, exhibited within my work is provided by McLean with contextualisation that employs the 'Creative Systems Framework' and 'the extended mind hypothesis (Clark, 2008)<sup>70</sup> (McLean, 2011, p. 124–125).

<sup>70</sup> Clark, A. (2008). *Supersizing the Mind: Embodiment, Action, and Cognitive Extension (Philosophy of Mind Series)*. Oxford University Press, USA. [<http://books.google.co.uk/books?id=1cgwUvk3OiIC&hl=en>]

### 3.7.4 On the context of text in programming

Further to the definition of software – cited at the start of §3.7, and which refers to the 'instructions or programs executed by a computer' – is the following written by Ge Wang, which also connects to the significant conceptual themes both of layers of abstraction, and that this is a project concerned with human activity and perception:

*A program* is a sequence of instructions for a computer. *A programming language* is a collection of syntactic and semantic rules for specifying these instructions [...] The programming language acts as a mediator between human intention and the corresponding bits and instructions that make sense to a computer. It is the most general and yet the most precise tool for instructing computers.  
(Wang, 2007, p. 55)

#### (3.7.4.a) Programming with text

Getting now to that second issue arising in the quotation given at §(3.7.3.a): where McLean writes of 'the programmer and their text', the word text is loaded with significance. It is to the text of source code that reference is there being made, and the significance is contributory to the aesthetics of seeing software as substance.

It is important to note that so called 'visual programming' paradigms – such as in the MaxMSP environment – are still based on text, albeit that the text is enclosed in spatially arranged 'boxes' within the patches that comprise the source code; this is as opposed to the (rectilinear) lines of plain text that comprise source code in the more textual programming languages. Any programming language, however, 'can be considered in visual terms, we do after all normally use our eyes to read source code' (McLean, 2011, p. 103). In my quest to explore visual representation of sound in computer music software, the appearance of text, as and when it is used in code toward soundmaking, must also be considered.

The concept of programming and the concept of text are in many ways inseparable: 'computer programs were first conceived in terms of weaving', writes McLean (2011, p. 68) with reference to the Jacquard loom technology of the early-nineteenth-century – see Figure 3.44 (after

Knight, 1858) – which inspired Charles Babbage<sup>71</sup> towards 'the first conception of a programmable universal computer' (McLean, 2011, p. 13):

Although Babbage did not succeed in building the analytical engine, his design includes a similar card input mechanism to the Jacquard head, but with punched patterns describing abstract calculations rather than textile weaves. While the industrial revolution had troubling consequences, it is somewhat comforting to note this shared heritage of computer source code and cloth, which contemporary artists still reflect upon in their work (Carpenter and Laccetti, 2006).<sup>72</sup>

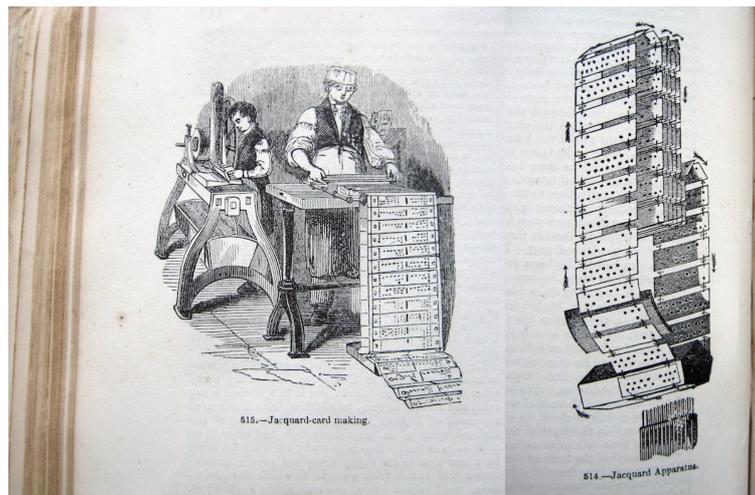


Figure 3.44: JacquardCards

Further to that shared heritage of programming for both computer and cloth, McLean (p. 68) adds that 'perhaps the same is true of writing, as the word text is a dead metaphor for cloth':

An ancient metaphor: thought is a thread, and the raconteur is a spinner of yarns – but the true storyteller, the poet, is a weaver. The scribes made this old and audible abstraction into a new and visible fact. After long practice, their work took on such an even, flexible texture that they called the written page a *textus*, which means cloth. (Bringhurst, 2004, p. 25)<sup>73</sup>

### (3.7.4.b) Source code text as a layer of abstraction

The text of source code in programming today is many layers of abstraction removed from the actual hardware that is physically controlled by it. Harman (2008) has described the layers of abstraction involved for the example of a program written in Java; in Figure 3.45 I have added my

71 By toughing on this aspect of computing heritage it necessary to make mention also of Ada Lovelace who is said to have been 'the first person known to have crossed the intellectual threshold between conceptualizing computing as only for calculation on the one hand, and on the other hand, computing as we know it today: with wider applications made possible by symbolic substitution.' (Fuegi & Francis, 2003, p. 16)

72 Carpenter, E. and Laccetti, J. (2006). Open source embroidery (interview).

73 Bringhurst, R. (2004). *The Elements of Typographic Style*. Hartley & Marks Publishers, third edition.

understanding of the parallel layers of abstraction for the maxpat 'source code' which runs in MaxMSP:

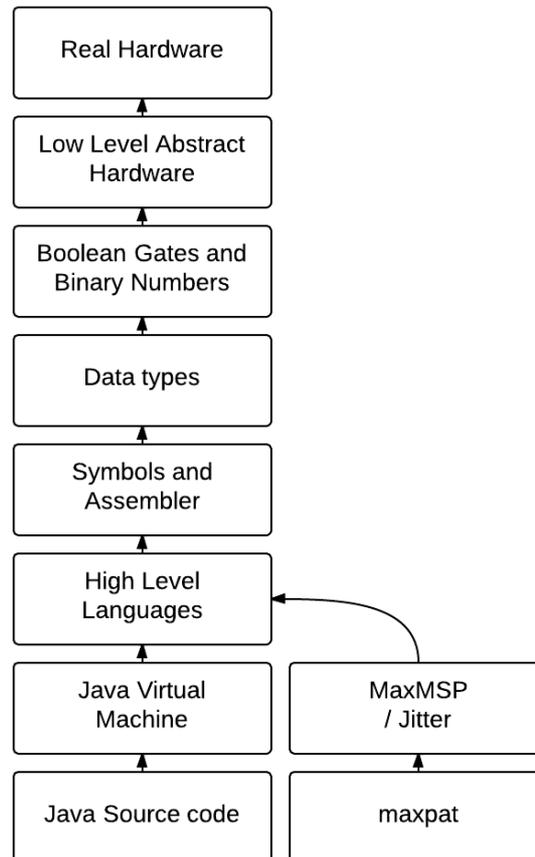


Figure 3.45: layers of abstraction

At its lowest layer of abstraction, the modern computer is a machine based on manipulation of binary changes between states typically thought of in terms of as on and off;<sup>74</sup> programming at that level would be analogous to controlling the presence and absence of holes in a punched card control sequence for a system like that of the Jacquard loom (as the humans depicted in Figure 3.44 are shown to be doing). Multiple layers of abstraction reduce the complexity of the computer to the point where text can be used to describe algorithms in the code that then constitutes software. The

<sup>74</sup> The actual behaviour of the transistors inside real hardware of a computer is 'extremely complex to model accurately. You need complex partial differential equations to even *approximate* the behaviour of each transistor.' (Harman, 2008). For another perspective on how transistors work see <http://youtu.be/IcrBqCFLHIY> (accessed 20130801).

artist-programmer is then at liberty to devise their own, additional, layers of abstraction in order to bring the available methods for control of the computer closer to their own conceptions of (how) reality (could be). In the creation of *sdfs* – more fully described in its own chapter (§6) – a microcosm is conceptualised in which multiple levels of abstraction are represented: there is abstract access to audio hardware; there is a concept of different data types; it operates as a sort of virtual machine that is a modular system which must be programmed, via text, to do things. All of that is extending beyond what, in Figure 3.45, is the *maxpat* layer of abstraction.

### 3.7.5 The medium is the meta-medium is the tool is the message

Given that the artist-programmer is creating new levels of abstraction with which to create, the question again is brought to mind of the distinction – if there indeed be one – between the 'tool' and the 'work'.

The following quotation from Alan Kay can be found in *The Art of Human-Computer Interface Design* (Laurel & Mountford, 1990, p. 225) and is also repeated by Lev Manovich in *Software Takes Command* (2008, p. 83) where it is described as being 'as accurate and inspiring today as it was when Kay wrote it':<sup>75</sup>

It [a computer] is a medium that can dynamically simulate the details of any other medium, including media that cannot exist physically. It is not a tool, though it can act like many tools. It is the first metamedium, and as such it has degrees of freedom for representation and expression never before encountered and as yet barely investigated.

I have been writing of software as substance, and placing it as the medium with which I work, but software – or as for Kay, the computer in general – is often described as a meta-medium. Within the conclusion to his thesis, McLean describes the artist-programmer as being 'engaged closely both with their target medium, and the meta-medium of the source code' (2011, p. 150). Is

---

<sup>75</sup> The source – Kay, Alan (1984) "Computer Software." *Scientific American* 25:3, pp. 52–9 – is cited widely, including (I thought it of interest given the above discussion of text as in textiles) by Cathy Treadaway in *Textile: The Journal of Cloth and Culture* (Treadaway, 2004).

not a meta-medium yet a medium itself? Approaching this subject matter is like peering down a rabbit hole upon the walls of which the words 'the medium is the message' appear in print, paint, light bulbs, laser beams<sup>76</sup> and needlework, and into which one falls at risk of going way beyond the scope of their own domain of practice. To exclude those thought patterns from discussion, however, would be to deny an important aspect of the aesthetic contemplation that has emerged during the project.

Coming from the title of the first chapter in Marshal McLuhan's many times republished book *Understanding media: the extensions of man*, the statement that the medium is the message has attained 'the status of cliché' (Gordon (ed.), McLuhan, 2003, p. 18), but in order to better understand the significance of working with software as substance – of a medium that is also seen as a meta-medium, and as a tool – it seems necessary to look beyond the surface of that meme. The 'message' of the substance is not found in that which it conveys because (*ibid.*, p.31):

the “content” of any medium is always another medium. [...] the “message” of any medium or technology is the change of scale or pace or pattern that it introduces into human affairs.  
[...] “the medium is the message” because it is the medium that shapes and controls the scale and form of human association and action. The content or uses of such media are as diverse as they are ineffectual in shaping the form of human association. Indeed, it is only too typical that the “content” of any medium blinds us to the character of the medium. (McLuhan, 2003, p. 20)  
[...] For the “content” of a medium is like the juicy piece of meat carried by the burglar to distract the watchdog of the mind. The effect of the medium is made strong and intense just because it is given another medium as “content.”  
[...] The effects of technology do not occur at the level of opinions or concepts, but alter sense ratios or patterns of perception steadily and without any resistance. [...] the only person able to encounter technology with impunity [is one who] is an expert aware of the changes in sense perception.

Douglas Rushkoff provides a more contemporary continuation of related themes; in *Program or Be Programmed: Ten Commands for a Digital Age* (Rushkoff, 2010), Rushkoff promotes awareness of the changes in sense perception that the now omnipresent software medium has created whilst proposing remedy to their effects.

Kim Cascone, during a presentation at the University of Huddersfield (in May 2011)

---

<sup>76</sup> Marshall McLuhan's *Understanding media: The extensions of man* was first published in 1964, five years prior to the first audio controlled laser light show (see §3.3.7).

speaking primarily about live performance of electroacoustic music, suggests a twist on the old McLuhan-ism (transcript from my audio recording of the presentation):

My observation over the past five years or so has been that the medium is no longer the message: the tool has become the message. Invariably people will respond by saying that 'the tool is immaterial, that you can make music with anything' and that may be true except that that isn't necessarily the case: any tool that is created for creative use is designed by somebody else – [unless] you're designing your own software – but a lot of the off-the-shelf applications are designed by engineers fighting with marketing departments [... who are] for the most part, not really of that world that they are trying to market to; [...] they may be on the periphery [... but] there's always this kind of not-quite-getting-it that takes place.

So these applications that are made for the public are always designed, and the features that are revealed for the public are thought up by people, its kind of like design-by-committee; again, unless you're using MaxMSP or something and designing your own tools.

So in this way, our attention, our focus, our tools are kind of only allowing us to play in certain areas of the total features that are under the hood, and because they think 'why would anyone want to plug the stove into the refrigerator' [... because] they don't see that people want to really experiment and try weird things, they don't want those areas necessarily played with because something might break, or there's no demand for it, or they just don't see the importance [...]

[The tool] directs us towards a certain type of creation [...]. I really have, unfortunately, come to the conclusion that the tool has really become the message, that if you are working with [off-the-shelf live performance based software] it does sort of transmit its own identity through the constraint or the subset of features and of certain types of sound or plug-ins that it comes with that tend to shape and mould the output in certain ways.

Perhaps Cascone chooses to see a tool where others choose to see a meta-medium, and where I choose just to see yet another medium; in any case, the significance here is to question the influence of the software on the creative practice. Cascone's observation appears to pose a question of whose *technē* is being expressed in a work of computer music: if the 'knowledge of how to do things and make things' – which is the meaning of the word *technē* (§3.7.2) – is built into the software, then unless it was you that built the software that you are using to make music, you may be bound to the forms of expression of someone else's *technē*. Inevitably – and to phrase again what has already been described – the environment in which one works to make one's own tools for working with will similarly influence that creative process by bringing its own set of biases that shape the way that things are made.

### 3.7.6 In software, on screen: programming/composition

The smallest visual element of the screen is the pixel that is native to a cartesian matrix and the RGB colour-space. All visual representation of sound in computer music software is built upon layers of abstraction that extend from these things. Contextualisation of the six portfolio works described in this chapter has touched upon various techno-aesthetic considerations relating to navigation of those layers of abstraction in the pursuit of computer-based musicking. The title of this thesis states that this project is an exploration conducted 'through programming and composition', and from the start of this project I have felt that these things are intrinsically linked, but more and more it seems that they are, or at least can be, the same thing.

# 4: Spiroid

This chapter describes the context and development of the spiroid-frequency-space concept as a visual representation of frequency-domain aspects of sound. As mentioned in §1, my interest in this type of representation has stemmed from a practice of sketching different aspects of sound as abstract properties in geometrical forms. The basic premise of the spiroid-frequency-space is that the pitch-class and octave-level attributes of pitch perception can be mapped to the angle and radius values of a polar coordinate system such as to form a spiral on the plane that represents a frequency-domain continuum. Exploration of the spiroid premise has become an aesthetically motivated endeavour, and was pivotal to the initial inspiration for this project; the concept is present to various degrees in many of the creative works of the portfolio.

The concept of mapping pitch-class and octave-level as polar coordinates is not itself new; it can be connected directly to the ideas outlined in §2.2, and more examples of similar representations are presented within the discussion below. The aim here is to contribute critical evaluation of the spiroid concept through exploration of its uses within my practice, and thereby examine the aesthetic implications and creative potential of this archetypical visual representation.

## 4.1 Spiroid

The word 'spiroid' has been adopted in this work because it seldom appears in everyday use of language; indeed it is absent from many dictionaries<sup>77</sup> and the *Google Ngram Viewer* can be used to see how the word has been used relatively little in published works of the past two centuries.<sup>78</sup> Any

---

<sup>77</sup> No results are found via <http://www.oxfordreference.com/search?q=spiroid> whereas via <http://www.thefreedictionary.com/spiroid> definitions are found from various publications including Collins English Dictionary and Dorland's Medical Dictionary. (Last checked 20130309)

<sup>78</sup> Compare [http://books.google.com/ngrams/graph?content=spiroid%2CSpiroid&year\\_start=1800&year\\_end=2012](http://books.google.com/ngrams/graph?content=spiroid%2CSpiroid&year_start=1800&year_end=2012) to the graph produced by adding, to the comma-separated search box, any one of these: toroid, sinusoid, cardioid, helicoid. In each case the graphed plots representing 'spiroid' and 'Spiroid' are reduced almost to flat by the adjusted scaling of the graph. (Last checked 20130312)

reference within this work to the (or a) spiroid will thus pertain specifically to my own conception of frequency-space representation.

The spiroid can be viewed in the context of similar the models and diagrams that are found in numerous sources (cited variously below), but to introduce it as derivative of those would misrepresent its origins within my practice. So that the reader may observe the autogenous process through which the spiroid concept was founded, an abbreviated account of that incremental progression – extended along the way by additional *a posteriori* contextualisation of the concepts involved – is presented below. Central to this study is the idea of geometrical representations in compositional process; while in my own practice this came first as intuited drawing on paper, the cited contextualisation has both supported and fed into my understandings.

### 4.1.1 Circularity

The first of my sketches that would lead incrementally to the spiroid concept, was a visualisation of a six note scale with the pitch-classes of its notes being arranged, stepwise, on the circumference of a circle. That original drawing – which is reproduced below as Figure 4.1 – was used to contemplate all possible binary pitch pairings within the scale.

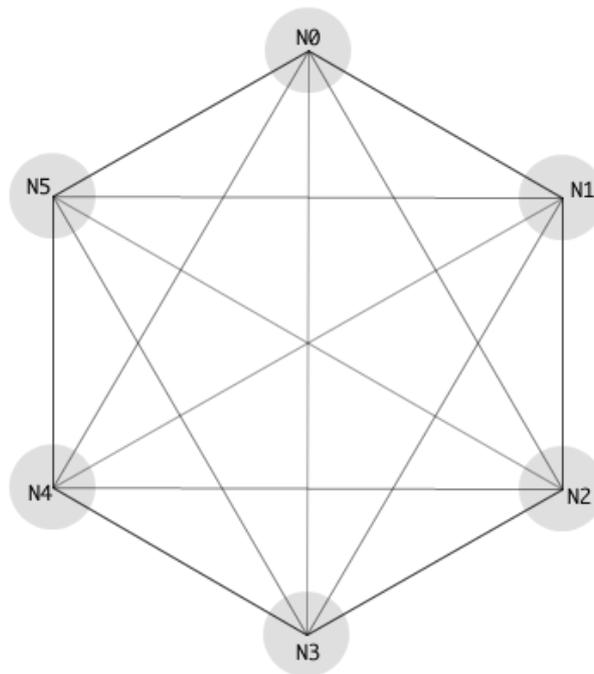


Figure 4.1: Showing six pitch-classes

The six pitch-classes that are labelled N0 to N5 in Figure 4.1 are of unspecified value in terms of their fundamental frequencies or intervallic ratios within this construct. The equidistant spacing within the sketch was not drawn to correspond to equal divisions of the octave, but rather as a purely abstract representation meant to imply equal potential of all constituent notes in the scale. Other than that significant distinction, however, this type of circular pitch-class-space is similar to the 'Tone Clock' diagrams described by Peter Schat (1993).

Composing within the twelve-tone tradition of the mid-to-late-twentieth-century, Peter Schat developed/discovered a system of 'chromatic tonality' as a tool to be utilised by composers (Schat, 2002). The term 'Tone Clock' was first used by Schat to refer to a 'Zodiac of the Twelve Tonalities' (Schat, 1993, p. 73, p. 81) in which 'twelve small clock-faces are contained within a larger clock-face' (see Figure 4.2) (McLeod, 1994, p. 113).

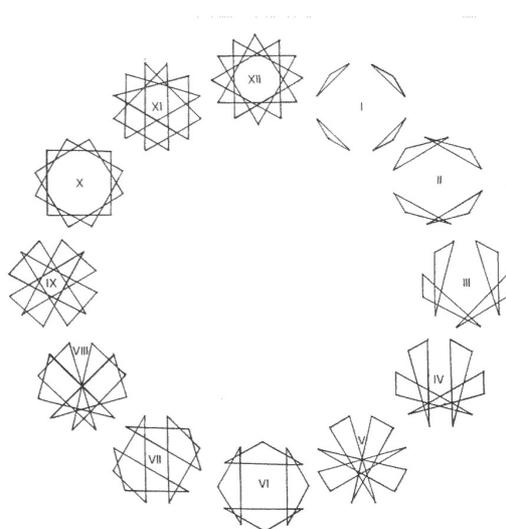


Figure 4.2: tone clock hours

We may refer to the singular clock-face, with its arrangement of the twelve tone equal temperament (12TET) pitch-classes on the circumference of a circle, as a 'tone-clock'. The space inside the tone-clock circle is used to draw lines that indicate relationships between the pitch-classes with disregard to the octave-level attribute; it was because of this visual similarity to the six-pitch-class diagram (Figure 4.1) that I took interest in *The Tone Clock* book.

Drawing triangles within tone-clock figures, Schat describes an exploration – 'in (tonal) space' – of tonality, where tonality is seen as arising from the triad: 'The triad is a (tone-)spatial creation' and there are 'no more than twelve different triads [that] can be formed' within the twelve tone system (Schat, 1993, p. 57).

All other triads conceivable can be reduced to these twelve basic forms. Therefore one can call these the 'Twelve Tonalities'. [...] These triads fit four times into the module of the octave, in such a way that all twelve notes are always involved. (The Tenth Tonality is an exception [...]) [The] descriptions of the different Hours (or Tonalities) is partly exact and partly subjective. Art is no science – there is nothing to prove. (Schat, 1993, p. 57–58)

The tone-clock type construct is also recommended by David Benson in *Music: A Mathematical Offering* (2007) when an exercise on the subject of symmetry in music is set for the reader: Benson writes that – within the excerpt given as common music notation (CMN) – the 'symmetry in the first two bars of Schoenberg's *Klavierstück* Op. 33a is somewhat harder to see' in

comparison to previously given examples; Benson suggests that one 'may find it helpful to draw the chords on a circle' and illustrates the how the first chord of the passage would come out on a tone-clock circle (Benson, 2007, p. 320). To explore this exercise in software with interactive representations on screen, I made a patch<sup>79</sup> in MaxMSP using newly available objects from the *bach: automated composer's helper library* (Agostini & Ghisi, 2012).<sup>80</sup> Two of the GUI type objects from that library are employed in the maxpat: the **bach.score** object for displaying the CMN of the selected chords – selected via tabs of the **tab** objects in the patch – and the **bach.circle** object which brings a tone-clock type visual representation to the Max environment.<sup>81</sup> Figure 4.3 shows the patch with the shapes of the first two chords displayed. A graphical clipping of the relevant text and illustrations from Benson are included in a sub-patch, and there is a basic 'play MIDI' option available.

To add an observation of my own to this exercise, I suggest that if the tone-clock representation were to have at the twelve o'clock position aligned to mid-way between zero and one, rather than at zero, then the visual manifestation of the symmetries that are to be found here would be much more clearly shown. In other words, I would like to rotate, anti-clockwise by 15 degrees, the tone-clock representations that are on screen, but the **bach.circle** objects do not offer this adjustment parameter.

---

79 BensonSymmetryEx3.maxpat which is in the BensonSymmetryEx3 folder in the chapter4 directory of the portfolio.

80 The maxpat was made using *bach... library* 'v0.6.7 alpha' which is copyrighted 2010 by Andrea Agostini and Daniele Ghisi; the latest version and further information can be found online via <http://www.bachproject.net/>

81 Although this object defaults to 'modulo twelve', any positive integer value can be used to set the number of points that appear around the circle, and although it is intended for representation of pitch-classes it could be used to other ends...

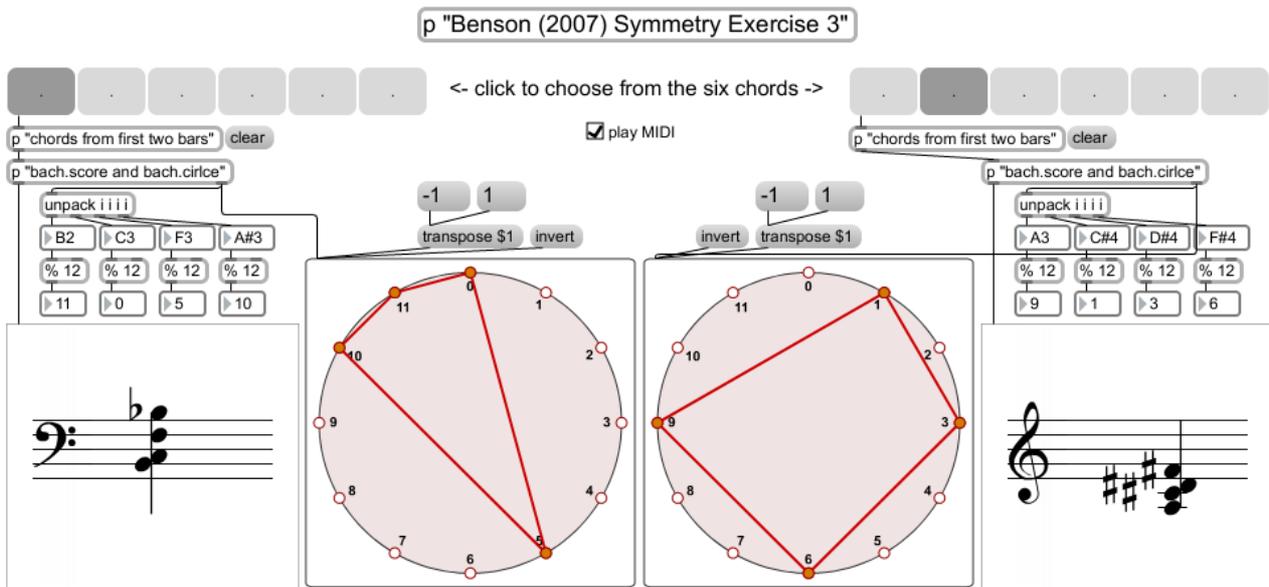


Figure 4.3: BensonSymmetryEx3.maxpat

Further reference to Schoenberg can be found in a contemporary reflection on twelve-tone music that is provided by Vi Hart whose video on the subject (see Figure 4.4) is available both on YouTube (Hart, 2013a) and as a bit-torrent (Hart, 2013b).

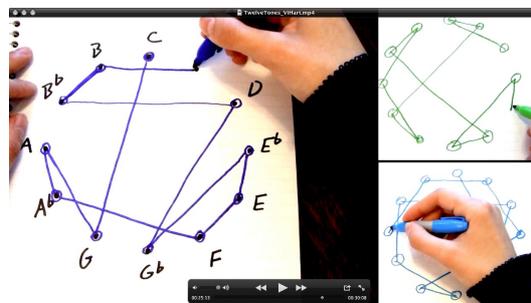


Figure 4.4: TwelveTones\_ViHart

### 4.1.2 Radially extending

The second step, the second sketch, in this simplified version of events describing the how the spiroid emerged in my work, was to extend the abstract pitch-class-space of Figure 4.1 to represent the discrete notes of that scale over multiple octaves, rather than just the pitch-classes. Figure 4.5 shows the initial, naïve, rendition of this radially extending step: by keeping the attribute of pitch-class aligned to that of visual angle on the plane, the octave-level attribute is readily manifest as

proportional to the radius attribute on the plane. At this stage the representation is still highly abstracted from physical reality: although similar to the tone-clock representations described above, the equidistant spacing of its six pitch-classes unrelated to their distribution in 'tonal space'.

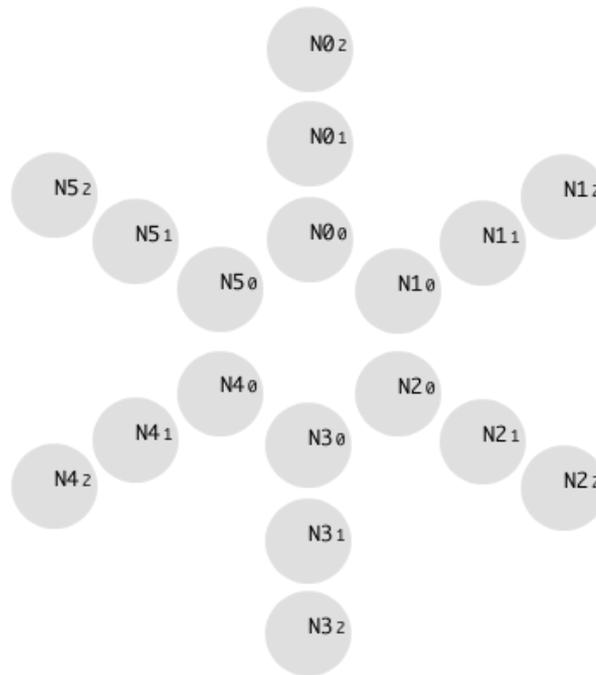


Figure 4.5: Three octaves of six pitch-classes (simple)

The visual rendering of three octaves that is seen in Figure 4.5 fails to represent the way that the N5 pitch of each octave is actually adjacent to the N0 pitch of the octave above. If one were to draw a line to go through each consecutive note of the scale as it is represented in Figure 4.5, then the spiral formed would be somewhat irregular: the visual distance representing the final interval of each octave turn is larger than the preceding intervals, causing a sudden change of angle in the spiral line that passes through them all.

To remedy that discrepancy, and thereby complete the specification of the spiroid archetype, the third step introduces two key elements: (1) the first of these is a conceptual shift, away from the purely abstract representation of pitch-classes from some particular scale, toward the more concrete and mathematical representation of a continuous frequency-space based on the pitch-class-circle of

the tone-clock, as is the base of the Drobisch type helix described by Shepard (§2.2.2); and (2) second is the adoption of the Archimedean spiral equation as the mathematical basis of that representation.

### 4.1.3 On Spirals

The curve that was studied by Archimedes in his treatise *On Spirals* is 'the locus of a point which moves away from a fixed point with constant speed along a line which rotates with constant velocity' (Wells, 1991, p. 8). The polar equation of the Archimedean spiral has been discussed for its use in my conceptual model of a gramophone (§3.4.2). The constant rate of growth in the Archimedean spiral provides a good visual analogy to the constant nature of human perception of pitch where intervallic ratios are heard as being the same no matter where in the frequency-domain they occur. Figure 4.6 shows a computer rendered sketch of the spiroid archetype that highlights the visual equality of octave interval representations.

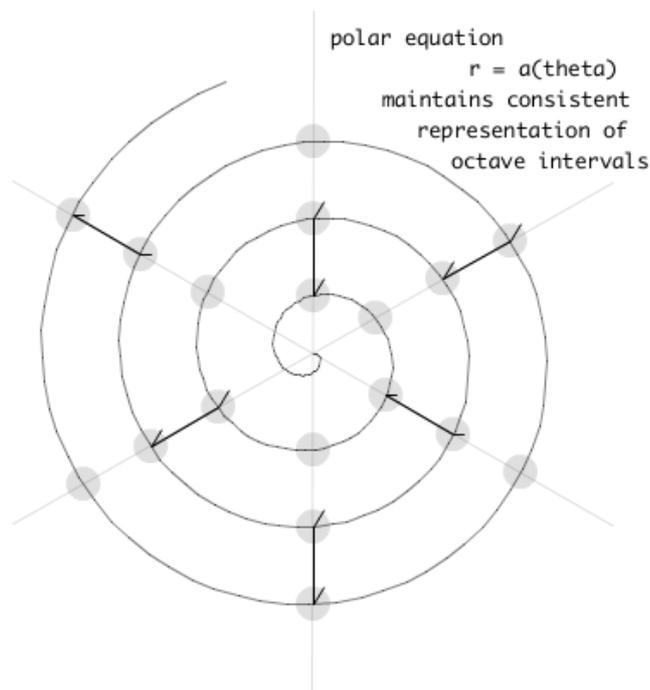


Figure 4.6: Sketch of the spiroid archetype

Each of the six radial lines in Figure 4.6 represents a specific pitch-class, and the fundamental frequency values corresponding to the notes of a scale are found where those radial lines intersect the spiral curve. Unlike the sketches that lead to this stage, recreated above as figures Figure 4.1 and Figure 4.5, what is shown in Figure 4.6 is now a spiroid type of representation wherein angular position is equated to a tone-clock type pitch-class-circle, and the equiangular distribution the six radii in Figure 4.6 thus correspond to what would be a whole-tone scale of 12TET. Unlike tone-clock representations, the spiroid-frequency-space is not limited to representation of quantised pitch-classes.

#### 4.1.4 Curved continuum

The curve that is seen on a spiroid representation, such as in Figure 4.6, is a given as a visual manifestation of (a range of) the frequency-domain continuum. Only a limited range of that theoretically infinite continuum can ever be visually represented by the spiroid in any given manifestation, and that range limitation is determined primarily by the number of turns (number of octaves) that are to be drawn in the available screen space.<sup>82</sup> The actual frequency range depicted is determined by both the number of octaves that are shown and an offset value for the starting frequency of the representation; the setting of these attributes being a matter of choice that will vary to suit different uses of the spiroid-frequency-space.

Frequency values on the spiroid are expressed natively as midi-pitch values, rather than Hertz, because the midi-pitch scale is more closely related to the human experience of the audible frequency-domain. It can be observed that to express an octave in terms of Hertz numeric value a 'doubling of frequency' is the language used; it is a relativistic relationship, and to say there is a change of  $x$  Hertz is almost meaningless without knowing what the value was to start with: a change of  $x$  may be perceived as an octave or as a quarter-tone where it is occurring. By contrast, it can be

---

<sup>82</sup> Software implementations are being assumed, though other manifestations of the spiroid are possible.

seen that there is permanent and stable relationship between human perception and numeric value in the midi-pitch scale; a 12TET quarter-tone is always a change of 0.5 in the midi-pitch scale, and an octave change is always expressed as a change in value by twelve. By extending above and below the 0–127 numerical range of the MIDI specification,<sup>83</sup> and by allowing fractional values to exist between the integers, any required frequency value can be expressed as a midi-pitch value that can then manifest visually on the spiroid-frequency-space continuum curve.

Pitch-class is manifest in the midi-pitch system as a modulo of twelve. Pitch-class is manifest on the spiroid continuum as angle with a modulo of 360 degrees. The 12TET semitone, is thus equivalent to 30 degrees of rotation on a spiroid-frequency-space representation. Values on the spiroid can therefore be converted between midi-pitch and angle, *theta*,<sup>84</sup> by these equations:

$$(\text{midi pitch}) = \left( (\text{theta}) / 30 \right) + (\text{midi pitch offset})$$

$$(\text{theta}) = \left( (\text{midi pitch}) - (\text{midi pitch offset}) \right) * 30$$

## 4.2 Some similar spirals

### 4.2.1 In the inner ear

Connecting the acoustic (§2.1) and the psychoacoustic (§2.2) there is a physiological layer in the process of human perception, and connecting the ossicles of the middle-ear to auditory nerve and the brain is the cochlea. To liken the spiroid to the cochlea is more than conjecture: writing in *Trends in Medical Research*, Kang Cheng *et al.* (2008) describe how angles are equated to frequencies in a cochleaogram. A cochleaogram is visual representation of the cochlea that is used, for example, by the makers of cochlea implants. The model proposed by Cheng *et al.* describes a

<sup>83</sup> The MIDI specification is described in (Roland Corporation U.S. & MIDI Manufacturers Association, 2009).

<sup>84</sup> Remember that *theta* is not bound to a single turn of the circle, but is rather the value that traverses the entire curve of the spiroid.

spiral mapping of frequencies that is very similar to that of the spiroid. (Cheng et al., 2008, p. 37 [sic]):

In this investigation, we originally propose a mathematical model of a logarithmic spiral function with a base 2, in a polar coordinates system, to plot a cochleaogram that covers the whole human auditory range, based on Ohm's 2nd (acoustical) law and published data. Our model mathematical predicts how a cochlea decodes sound information from a frequency domain to a spiral spatial domain. [...] We believe our model is more meaningful and more useful than the previous to hearing-impaired patients and audiology, as well as music therapy and music theory.

Elsewhere, Ariel Moctezuma – in (Moctezuma, 2011, p. 9)<sup>85</sup> – 'shows the frequency-spatial arrangement of audio peak responses in the human cochlea' using a spiral form, but without the use of spiroid-like polar coordinates to align octaves to polar coordinate angles. In both of these cochleaogram examples, the lower frequency values are represented in the centre of the spiral, and this matches the original specification of the spiroid-frequency-space, as expressed in the equations above and also as manifest in my first generation of software implementations (described in §4.3).

## 4.2.2 In print

In my more recent work the inverse radial mapping of the octave-level attribute is used in the spiroid, so that it is higher frequencies that are found toward the centre of the representation, and this is the mapping that was described in *Spiral audio spectrum display system* (US patent) by Allen G. Storaasli (1992). The system described by Storaasli, however, did not employ the Archimedean spiral (see Figure 4.7, below), nor the use of midi-pitch to *theta* transformations, and was described only as a spectrum analysis display whereas the spiroid is also conceived as an interface for frequency value input to software systems. Notwithstanding those significant differences, important remarks which are relevant to the aesthetic motivation of this project are found in the 'Description of the Related Art' section of that document (Storaasli, 1992, p. 1):

Spectrum analyzers sample time-varying signals and display the energy content in x - y format [...] One with a well trained eye might be able to identify certain patterns in a static signal or single frames of a dynamic signal, especially those involving large peaks and valleys. However, it is extremely difficult for the average viewer to recognize pitch and harmonic content in most audio signals, particularly dynamic

85 [https://wiki.engr.illinois.edu/download/attachments/49746529/TEAM1S11\\_Fig9.jpg](https://wiki.engr.illinois.edu/download/attachments/49746529/TEAM1S11_Fig9.jpg) (Accessed 20130326)

signals.

Storaasli describes that the aim of displaying values of a spectral analysis with octave-alignment on a spiral is to 'facilitate the recognition of pitch and harmonic content' of an audio signal in the visual domain (*ibid.*); my own developments of the spiroid have confirmed the utility of such spirals in this regard, and it transpires that I have not been alone in pursuing this principle during recent years (see §4.2.3).

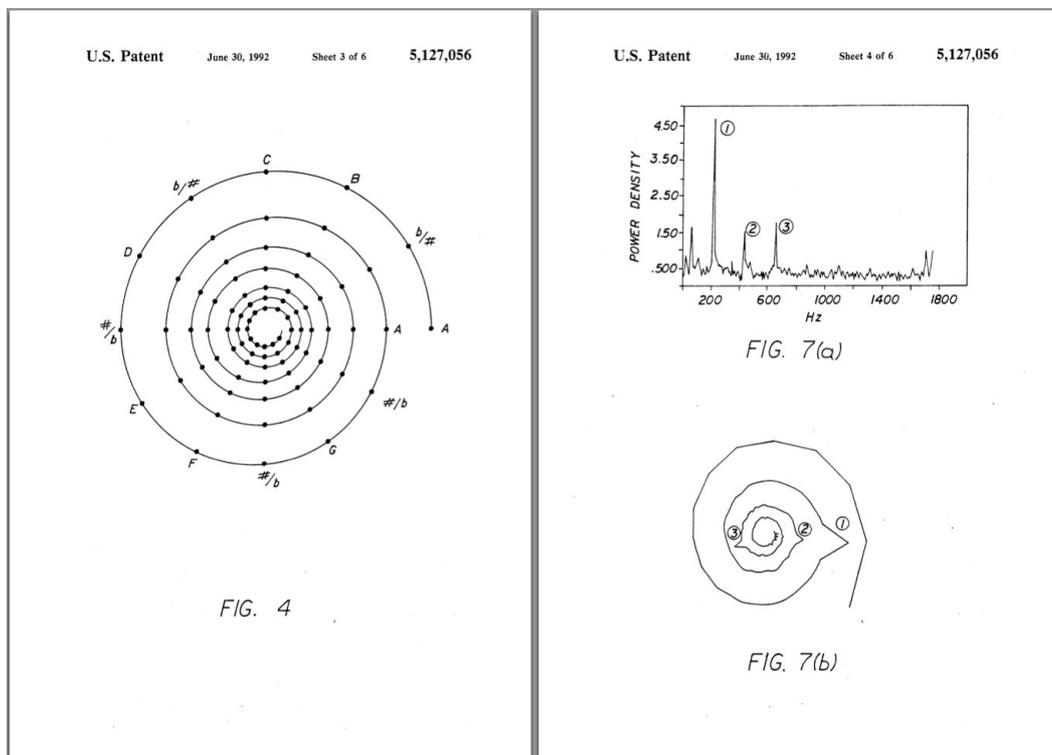


Figure 4.7: Storaasli Pat.5127056

Representation of 'chromatic pitch space' in *Musimathics volume 1* (Loy, 2006, fig. 6.6, p.165) is another example to place lower frequencies at the outer of the spiral illustration (this cited example is included here as Figure 4.8 below), and – also in common with the Storaasli system – has pitch ascending in an anti-clockwise direction. Loy places the pitch-class 'C' at the three o'clock position, which is also something that I later adopted in my practice.

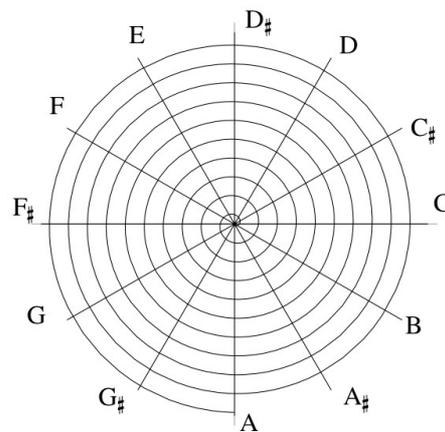


Figure 4.8: chromatic pitch space

It is often in connection to comparative descriptions of various tuning systems that diagrams similar to the spiroid are published. The visual form of a spiroid-like structure is typically employed to illustrate the 'near miss' of the octave alignment after an accumulation of true harmonic ratio intervals; in Figure 4.9, for example, the difference is shown – in the larger, centred spiral – between the frequency that is arrived at after twelve consecutive perfect fifths and the true octave equivalence of a frequency that is transposed up seven octaves from a starting angle at twelve o'clock (Ashton, 2003, p. 40; illustration from p. 41):

This is because  $(3/2)^{12} \approx 129.75$ , whereas  $(2)^7 = 128$ . The difference is known as the Pythagorean comma [which as a ratio is] approximately 74:73.

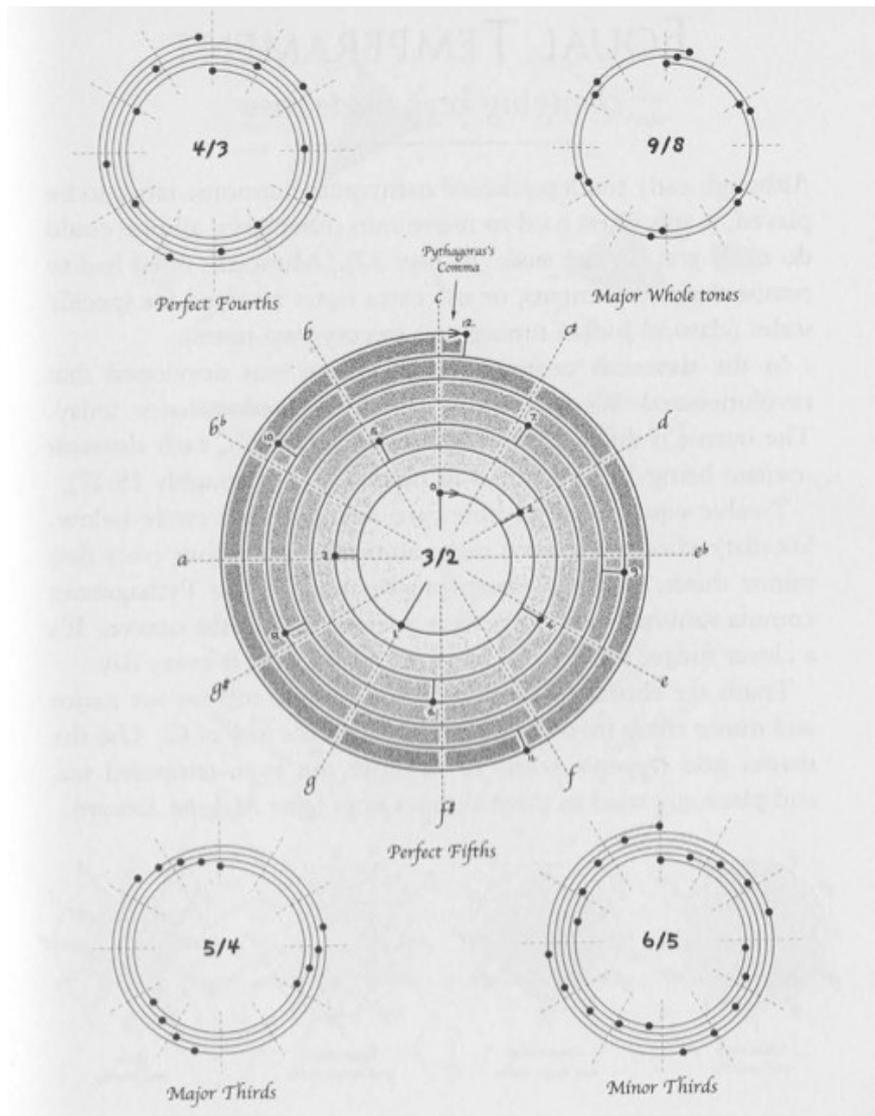


Figure 4.9: Showing the progression of different intervals

Figure 4.9 shows the illustrations of five different ratio interval progressions on pitch-class-circle spirals that accompany Ashton's text. Lower pitches are drawn toward the centre of these spirals, and pitch ascends with clockwise rotation, which is the same as in the original spiroid specification. Printed examples such as these from Ashton – to give another, Benson has similar (2007, Fig. 5.4, p. 165) – aim to elucidate the discrepancy that exists between 12TET and the natural harmonics of musical sound; information that, for me, although intellectually assimilated, only became experientially appreciated after working with real-time interactive software implementations of the spiroid-frequency-space as a spectrum analysis display.

### 4.2.3 Photosounder Spiral

In the first week of August 2013, Computer Music magazine cryptically announced a new plugin to be bundled with the software supplement to the then forthcoming edition of their publication (Computer Music, 2013):

Our next CM Plugin, Spiral CM by @Photosounder! What d'ya reckon it's for? It's out next week, free with CM195. [pic.twitter.com/oZBOQRGGeB](http://pic.twitter.com/oZBOQRGGeB)

The image that is linked to at the end of that tweet is included here as Figure 4.10 below. It clearly shows a spectrum analysis of a sound in which a 'C major' chord is being played, and my first intuition – based on experience with my own spiroid analysis displays – was that the sounds there pictured were from synthesised sawtooth waveforms; experimentation with the *Spiral CM* plugin appears to confirm that reading of this screenshot.

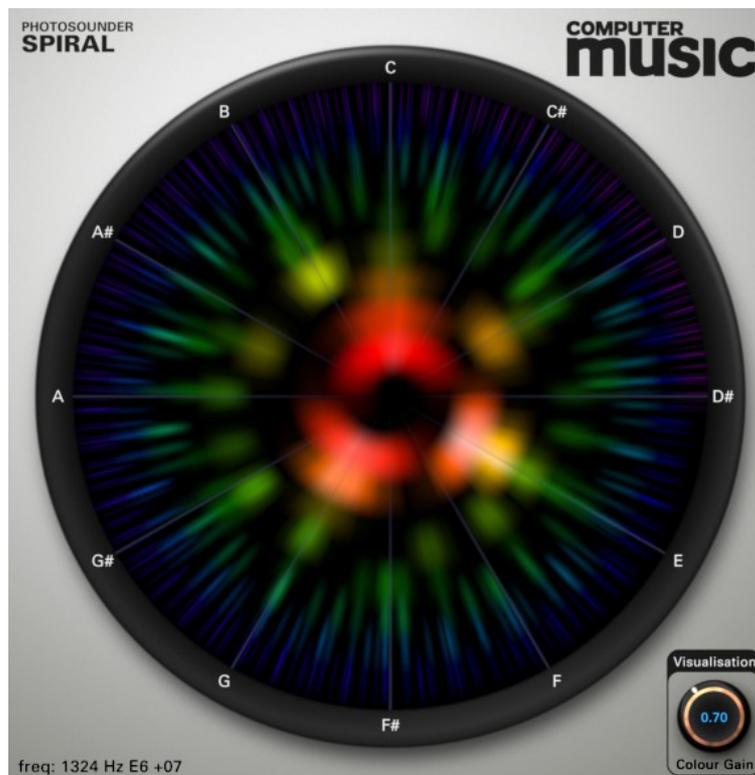


Figure 4.10: Photosounder Spiral CM

The *Spiral CM* version of this software by Photosounder – which can be obtained by

purchase of Computer Music magazine issue 195 or later – has just one parameter to adjust the 'colour gain' of the visualisation. A version, simply called Spiral, with many more controllable parameters is commercially available through the product's web site<sup>86</sup> where Michel Rouzic writes (Rouzic, 2013):

I originally created Spiral in September 2012 both to have something similar to [SplineEQ](#)'s analyser with the ability to provide relevant musical information and to teach myself music by instantaneously analysing the notes of whatever music I would listen to. It turned out to be very efficient and I was able to quickly analyse and memorise musical pieces of varying length and complexity. [...] I also use it to tune my guitar quite fast and very precisely [...] and in general to understand the composition, spectral makeup of various instruments, mastering and stereo mixing of the music I listen to

Through use of my own software versions of the spiroid-frequency-space as a real-time analysis display – dating originally to autumn 2008, and described in §4.3 below – I have found, too, that instantaneous identification of pitch content in real-time audio, and precision tuning for instruments (particularly guitar) are notable benefits that are brought by this type of view into the frequency-domain of sound. In the context of live improvisation within an ensemble, for example, a spiroid display can act as a prosthetic sense of 'perfect pitch' enabling the dominant pitch-classes of the audio input to be effortlessly followed by eye.

The Photosounder implementation of the spiroid-frequency-space concept is visualising a fast Fourier transform (FFT) analysis, and this is the method that was described by Storaasli except that the *Spiral CM* and *Spiral* displays have their lower frequencies closer to the centre of the encircled display which is more like the spiroid initially described in §4.1 above. The linear mathematics of FFT analyses determine that, in terms of pitch-class resolution, there is much more precision at the upper octaves of the frequency spectrum, and this is manifest in the Photosounder plugins by the larger angles of arc that are used toward the centre of the display.

As the start of a series that will be ongoing beyond this project, I have made two videos that demonstrate how the spectra of sinusoidal, triangle, sawtooth, and square shape waveform

---

86 Specifically via <http://photosounder.com/spiral/download.php> (accessed 20130916)

oscillators manifest in Photosounder *Spiral CM* plugin, and explore some of the observations that can be made such experiments (Freeman, 2013).<sup>87</sup> The patches used in those two videos are included in the Photosounder folder in the chapter4 directory of the portfolio.

## 4.3 Spiroid display of partials as arcs on an lcd in MaxMSP

In contrast to the FFT data displays described above (Rouzig, 2013; Storaasli, 1992), my own spiroid-frequency-space displays are based on the (more abstracted) data resultant from pitch tracking analyses; perhaps this choice of approach can be linked to the above described genesis of the spiroid concept – with its initial focus on discrete pitches, rather than continua – but also it is a reflection of my interest in the perceptible cognition of aspects of sound: I chose to work with a smaller number of higher-level data, rather than with the hundreds of data points that an FFT provides. A double meaning is thus observed the description of a 'partial analysis' from pitch tracking, as contrasted to the more complete one of an FFT. The spiroid concept is also set apart from above given the examples of similar spirals by its application to the input of frequency value data (see §4.4), in addition to the output display of analysis.

This section describes my spiroidArc prototype, elements of which have been reused in a number of different works during the project. Construction of the prototype began in the run up to the official start of this project,<sup>88</sup> and its development is a direct continuation of the incremental process described in §4.1.

---

<sup>87</sup> Videos published under the name of Experimental Music Technology.

<sup>88</sup> With the conceptual and practical work progressing at various stages in the two year gap after my MA studies.

### 4.3.1 Analysis and display

Software implementation of the spiroid began with a visualisation of pitch analysis data from the **fiddle~** object in MaxMSP;<sup>89</sup> specifically, using the lists of frequency and amplitude values that describe the sinusoidal components, or partials, of the input audio. The amplitude value of each partial is used to set the (alpha) transparency of the arc representing that partial on screen (the greater the amplitude, the greater the opacity); the frequency values are converted to midi-pitch values from which the pitch-class and octave-level attributes are taken to determine the placement of the arc on the screen. The process is shown as a flowchart in Figure 4.11:

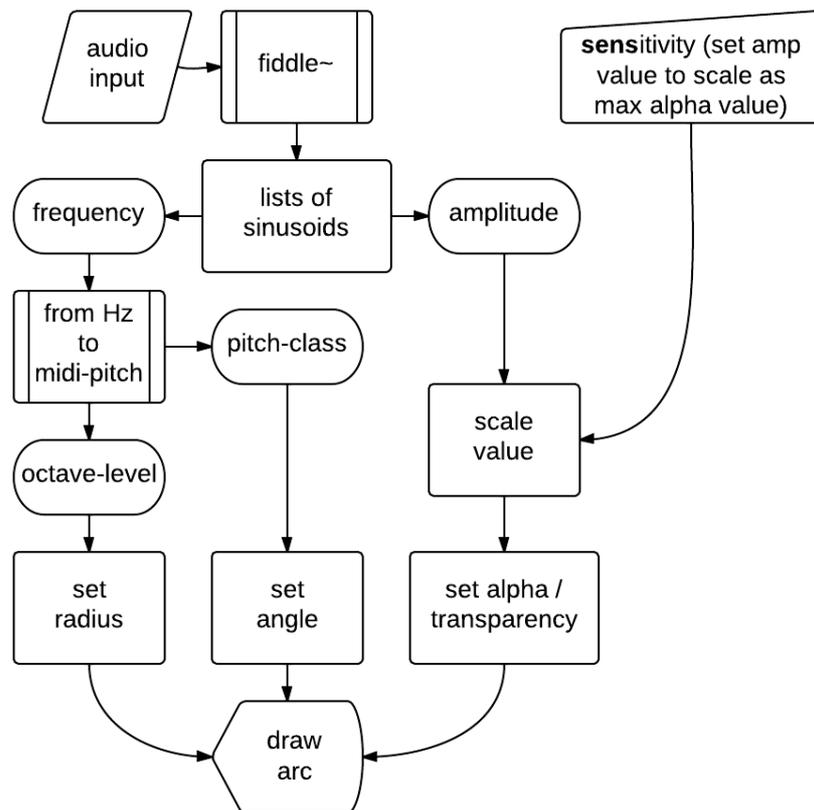


Figure 4.11: spiroid arc display flow

<sup>89</sup> by Miller Puckette, MSP port by Ted Apel and David Zicarelli; available from <http://crca.ucsd.edu/~tapel/software.html> – my use of **fiddle~** had continued despite the newer **sigmund~** object being available.

The **lcd** object has been used for drawing the arcs on screen, and the message format for doing that in a maxpat comprises the keyword 'framearc' followed by six numerical values that define the following three spatio-visual attributes:

- four numbers to define the square/rectangle bounds of a circle/oval (left, top, right, bottom);
- the starting angle of the arc (degrees, with zero at the top, ascending clockwise);
- the size of the arc (degrees).

In the `spiroidArc` patch, numbers for the square bounds of the circle to which an arc belongs are calculated by an abstraction<sup>90</sup> which takes two inputs: a radius value (mapped, in this case, from the octave-level value of each partial), and a cartesian-coordinate for the centre point (fixed for this patch). The pitch-class value of each partial is multiplied by 30 to give the start angle in degrees, and the size of the arc is fixed at eight degrees (which is rounded up from 7.5 degrees which is a quarter of 30 degrees which is the angular size of a semitone).

It will be noted that the equations given in §4.1.4 are not explicitly present in the method described here; indeed the prototype described here was built prior to my formulation of those equations. The result, however, is the same: the radius increases at a constant rate as the frequency value is increased.

## 4.3.2 SpiroidArc

### (4.3.2.a) a\_demo\_of\_SpiroidArc300

The earliest example of my `spiroidArc` display that will be presented is [spiroidArc300.maxpat](#); the number in the file name shows this was 'version three, revision zero' of the `spiroidArc` concept.<sup>91</sup>

The patch is designed as a self-contained unit initially for use – embedded via the **bpatcher** object –

---

<sup>90</sup> The abstraction, originally named `sqr.maxpat`, was later renamed to `ltrb.maxpat` when it was upgraded to include **patcherargs** and **pattr** objects (c. November 2009).

<sup>91</sup> It was created, March 2010, en route to perform at Interlace #36, Goldsmiths (inclusive improv, 2010; Lexer, n.d.).

within my modular laptop-instrument system; it was an expansion of earlier spiroidArc versions that, due to data loss, are no longer available. Version three-zero-zero added a soundmaking subsystem to the spiroidArc patch for modified re-synthesis of data captured from **fiddle~**, but it is only the features of display that are to be discussed here, and in the demonstration patch, described below some of the other features are, by default, hidden from view.

The following explanation is written of [a\\_demo\\_of\\_spiroidArc300.maxpat](#) from the spiroidArc300 folder in the chapter4 directory of the portfolio. There are some 'sound sources' in place for use as input to the spiroidArc patch that is embedded. When the demo patch is opened, it will look something like shown in Figure 4.12, but note that the 'zoom level' of the patch has been increased in this figure. The size of the display was intentionally kept small because it was designed to be on screen at the same time as many other GUI patches within a modular system. When the spiroidArc display is the only patch being used the zoom feature of MaxMSP can be used to increase the size on screen.

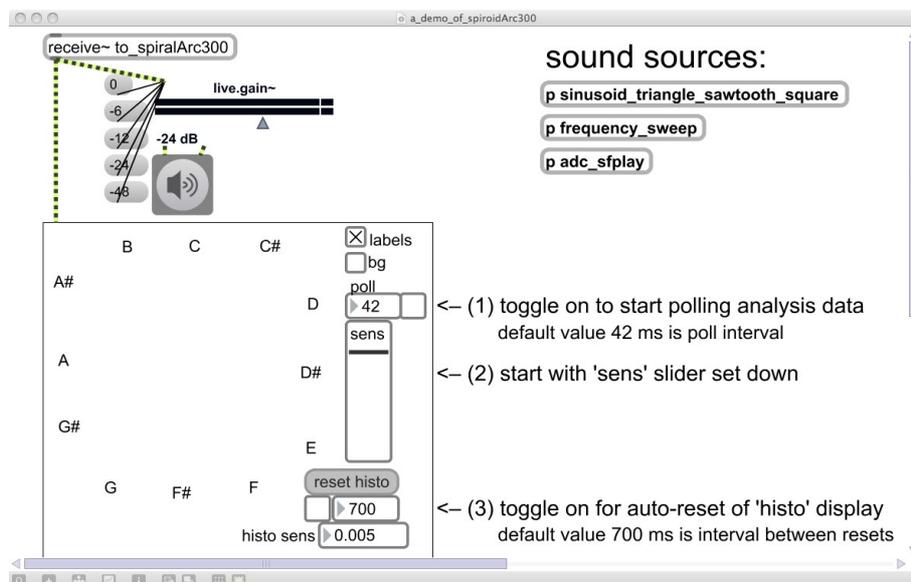


Figure 4.12: spiroidArc300demo1

The three sound source sub-patches are:

- `sinusoid_triangle_sawtooth_square`
  - the waveform oscillators, other than the sinusoidal, are antialiased; click on the graphic representation to switch between waveform shapes
  - frequency is set by combination of three parameters (octave-level, pitch-class, fine-tune)
  - panning is not in use: only right channel is output
- `frequency_sweep`
  - on/off toggle is master to toggles for both metro and gate~; in order to suspend the 'sweeping' of values, metro-toggle can be off while gate-toggle is on
  - **nslider** (CMN stave) on the right can be clicked on; the one on the left is display only
  - currently sounding pitch has its midi-name, midi-pitch number, and frequency Hz value displayed by objects in the patch
- `adc_sfplay`
  - using the input part of the standard Max IO utility patch
  - left and right channels are summed

### (4.3.2.b) Histogram of strongest pitch-classes

By implementing the spiroid curve to display the continuum from midi-pitch zero up there are several octaves of sub-sonic frequency values in the centre of the space. I decided to add a histogram of sorts that would draw radial lines, around the centre of the display, at the angles of the most present pitch-classes. A flow chart representation of the algorithm implemented for this is shown in Figure 4.13, and the histogram itself can be seen within Figure 4.14 (below) where the spectrum of a sawtooth waveform oscillator sounding at 65.406 Hz (midi-pitch 'C2') is displayed.

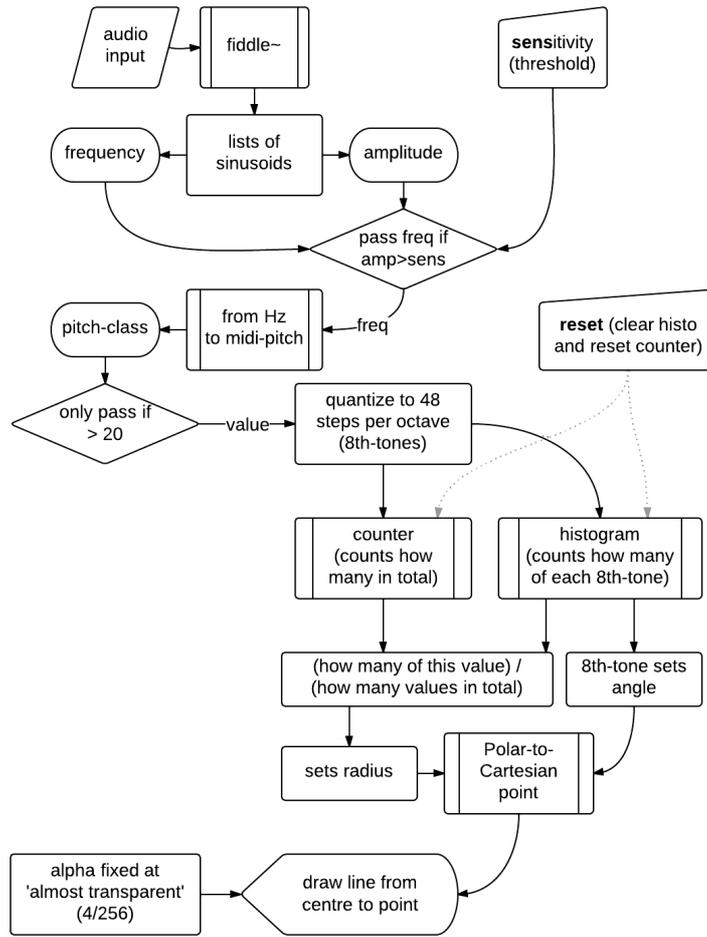


Figure 4.13: spiroidArc\_histoFlow

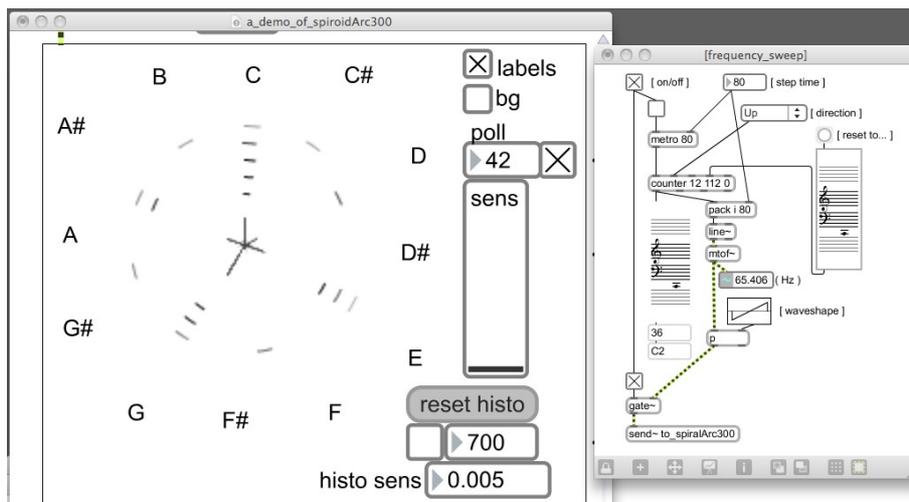


Figure 4.14: spiroidArc300demo2

### (4.3.2.c) Background option to show 12TET angles

In addition to the optional 'labels' – which show the 12TET pitch-class names, and are toggled on by default – there is also a 'bg' option that will show the angles of the twelve equal divisions in the circle. Rather than just drawing radial lines at the twelve angles, however, I decided to use divisions of the colour spectrum to fill each sector (see Figure 4.15); I now consider this as a poor design decision, and the reasons for that are discussed below.

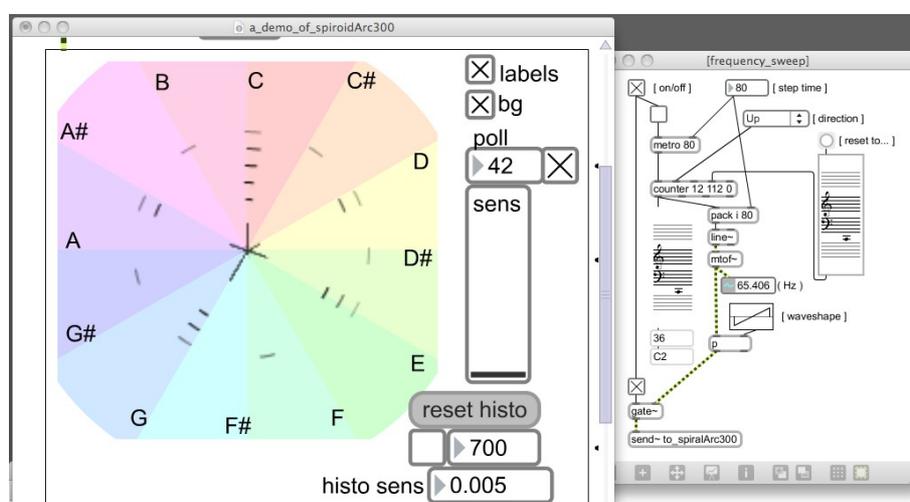


Figure 4.15: spiroidArc300demo3

## 4.3.3 Chroma paradox

### (4.3.3.a) (chroma ≠ chroma)

The analogy of pitch-class as colour is compelling but in many ways misleading. This is a complex subject to approach, and the discussion here is worthy of further work; in the process of contextualising my aesthetic engagement with these matters I have been humbled by the potential scope of this subject, and can only present here the stage of exploration that I have reached. It is exciting to but glimpse for now the potential possibilities of seeking to unravel the chroma paradox.

What I now call the chroma paradox (described in §2.2.2) is something that arises from the full octave pitch-class-circle being commonly known as the 'chroma circle', which thus associates

the angle attribute within a spatial representations of pitch- or frequency-space to colour; my suggestion is that this association is – if not paradoxical, then at least – contradictory to the consideration of morphophoric mediums and the respective (in)dispensability of colour and pitch, as outlined in §2.2.1. Clearly, this stance was arrived at only after the work of programming the spiroidArc display system described above.

Even within discussions of colour as colour, the word chroma can have different meanings depending on the context,<sup>92</sup> and to some extent the same seems to be true of the word hue. In this work hue is understood as a dimension of the HSL (hue, saturation, lightness) colour-space: creation of the 12TET angle background seen in spiroidArc300 (Figure 4.15) was achieved by stepping at twelfth divisions of the hue value in the 'hsl' message to the **swatch** object in order to obtain the RGB values to use for drawing on an **lcd**. That part of the patch is shown in Figure 4.16.

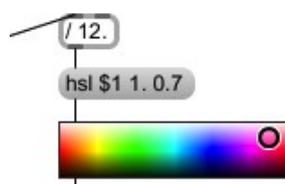


Figure 4.16: hsl\_swatch

One premise of the analogy between pitch-class and hue rests upon the measurement of wavelengths and inter-domain application of the octave concept from sound to light – from acoustics to photonics(?) – for example Daryn Bond, describing implementations of *the harmonic matrix*, writes (Bond, 2011):

Colorizing the matrix makes use of the fact that the human eye sees somewhat less than one octave of light – from 390 to 750 nm [1]<sup>93</sup> In figure 10, the visible spectrum is ‘artfully’ extended from 373 to 746 nm. Pitch class is associated with a color, octave with brightness

92 For example, results at <http://www.oxfordreference.com/search?&q=chroma> – even without getting through the paywall of the web site – show definitions of chroma both as 'One of the three variables of colour (hue and value being the other two)', and as 'The quality or hue of a colour irrespective of the achromatic ( 1 ) black/ grey/white...' (accessed 20130924)

93 [1] Cutnell, J. D.; Johnson, K.W. Physics. 7th ed. New York: Wiley, 2006.

The wavelengths of light beyond human perception are, in Bond's system, rendered as black; while this is a cunning compromise to one aspect of the chroma paradox, it does not help to remedy to ill-conceived situation of mapping pitch-class to hue in the spiroidArc. To get to the crux of this aesthetic crisis it is necessary to return to the subject of colour as an aspect of the medium being worked with.

The RGB colour-space has been discussed in chapter three (§3) because it is the method for defining colour values that is native to computer the screen medium. The RGB colour-space can be thought of in three-dimensional cartesian space as a cubic structure, as illustrated in Figure 4.17 (Otto, 2002); notice how the six colours used in the MoE *pixels* (§3.1) piece appear at the corners of this model.

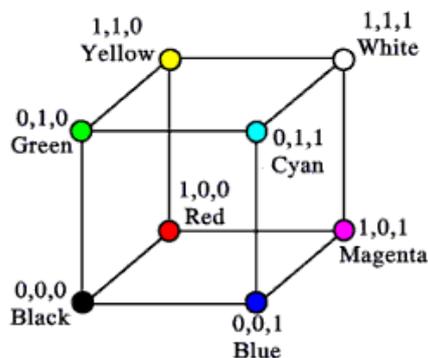
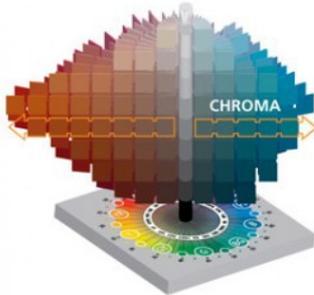


Figure 4.17: RGB\_cube

Many dictionary definitions for 'chroma' – particularly those that are, to me, the most convincing (cf. Colman, 2009) – make reference the 'Munsell colour system' which is named 'after Albert Henry Munsell (1858–1918)' (Colman, 2009), and 'is based on a three-dimensional model in which each color is comprised of three attributes of hue (color itself), value (lightness/darkness) and chroma (color saturation or brilliance)' (Munsell Color, n.d.). The Munsell colour system is thus closely related to the HSL colour-space, and when either of these is shown as manifest in three-

dimensional space it is with a polar coordinate basis (Munsell Color, n.d.(text), and (image), n.d.):

The neutral colors are placed along a vertical line, called the “neutral axis,” with black at the bottom, white at the top, and all grays in between. The different hues are displayed at various angles around the neutral axis. The chroma scale is perpendicular to the axis, increasing outward.



This amount of context is undoubtedly distracting from description of the spiroid, and yet – for the greater scheme – it is vital to bring mind to how the concept of colour is conceived of within the visual representation of aspects of sound.

#### (4.3.3.b) Double jeopardy

Not only is the chroma paradox casting heavy doubt on the use of colour described in §(4.3.2.c), but the assignment of the visual attribute of colour to aural attribute of pitch-class is also a superfluous doubling of visual attribute assignment: pitch-class equates to angle in the spiroid in a way that gains no benefit from a secondary identifier. The attribute of octave-level, however, would benefit from additional visual cues to help the viewer more easily perceive the relative radius values of the partials on display.

### (4.3.3.c) Suggested remedy

The better choice for use of the hue scale within a spiroid-frequency-space display would be to have its continuum mapped along the frequency-domain continuum curve. I sketched this solution in February 2011, as shown in Figure 4.18, using just the additive primaries red, green, and blue to imply the colour continuum. It is noted that the Photosounder *SpiralCM* uses colour in precisely this way.



Figure 4.18: 20110221\_0230

## 4.4 Spiroid as an interface for input

Discussion is now directed to use of the spiroid-frequency-space concept as the basis for specifying frequencies as input to software.

Three examples from my work are provided: first is another proof of concept type patch, *nRadii\_sawOsc* (§4.4.1); next (in §4.4.2) a spiroid based GUI widget is described; and the third example is an extension to the *spiroidArc* patch by which numbered frequency markers are placed upon the analysis display (§4.4.3).

The coiled presentation of a piano-keyboard type interface in the *Magic Piano* app by Smule Inc. (Hamilton et al., 2011, p. 61) is offered as an example a spiroid-like concept exhibited in software that has been released during the period of this project. Alignment of pitch-classes to angles in this input design, however, is only approximate, and the spiralling of the piano-keyboard is based on two octaves per turn ( $\approx 23$  semitones per 360 degrees are seen in Figure 4.19).



Figure 4.19: Smule Magic Piano

## 4.4.1 nRadii

### (4.4.1.a) nRadii\_sawOsc overview

The 'n' in title of this piece is used in the sense of 'any number', as in, any number of radius lines to be placed with equiangular spacing on a spiroid-frequency-space representation. The nRadii folder in the chapter4 directory of the portfolio contains nRadii\_sawOsc.maxpat (pictured in Figure 4.20) plus documentation of that patch.

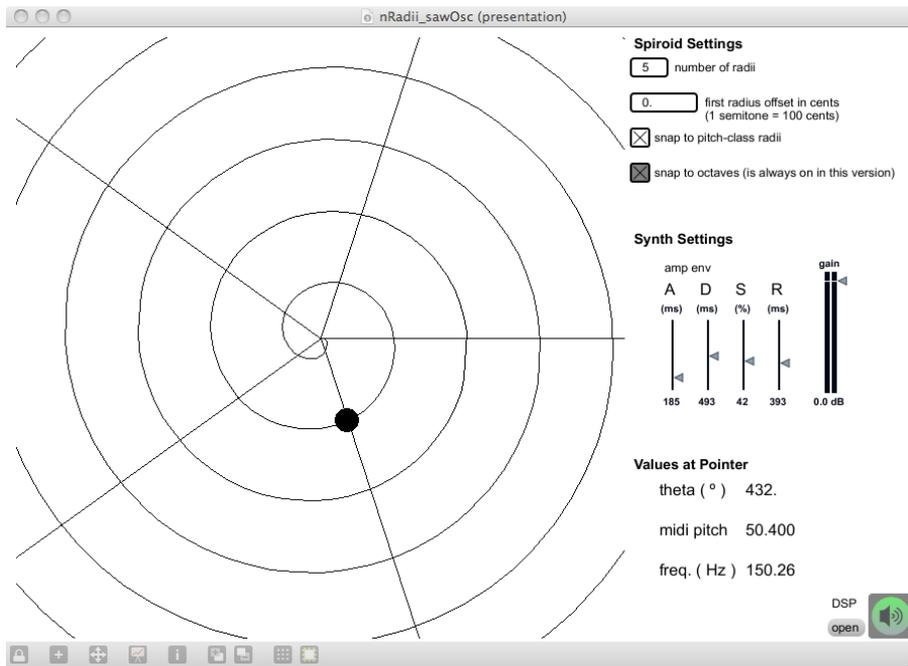


Figure 4.20: nRadii\_sawOsc.maxpat

The documentation in that folder is quite comprehensive and covers both the practical use of the patch, and precise details of its programming at a technical level; it was created, 2010, with FreeMind (Foltin et al., n.d.), and has been exported to this folder both as .png (see Figure 4.21) and .html. The following is a summary of that documentation with focus more on situating this work within the larger project.



Figure 4.21: nRadii\_sawOsc.mm

The *nRadii\_sawOsc* work is a proof of concept patch that has a sawtooth waveform oscillator (thus the '\_sawOsc') controlled my mouse interaction with a spiroid-frequency-space GUI. The patch will load in its presentation view showing the spiroid GUI to left of some settings above an information display reporting the value of the mouse pointer on the GUI in different units (*theta*, midi-pitch, and Hertz). At the top are spiroid settings where the number of radii to be placed upon the spiral is set; an offset value, in cents, can be used to adjust the angle of the first radius (see §(4.4.1.c), below, about angles in the version of spiroid). There is a **toggle** option for having the mouse controlled value selection on the GUI 'snap' to the currently displayed pitch-class radii. A second **toggle**, that is always on in this patch, brings attention to the fact that the input will always quantise to octave-level values that sit upon the curved continuum. The question of what might happen if we were to allow non-quantised octave-level values is significant, but discussion of that question would be in connection to synthesis experiments (involving granular approaches to octave shifting whilst maintaining pitch-class identity) that I chose to omit from this portfolio.

The test tone synthesiser has only the parameters of its fundamental frequency – set on the spiroid – and an (ADSR) amplitude envelope with attack time, decay time, sustain level, and release time being set by sliders. The mouse button acts as the attack and release trigger for the ADSR envelope.

#### (4.4.1.b) Construction of the nRadii GUI

As a developmental step beyond the spiroid-frequency-space work described in the previous sections of this chapter, there are a number of significant changes of method that can be seen to lead on directly to the works that follow. Examination of the spiroidArc patch described above will show that multiple lcd objects were used, with transparent backgrounds, to build up different layers of the display – the arcs are drawn to one, the histogram to another, and so on. In the nRadii work, jitter matrices are used for the spiral background, radii lines, and pointer spot layers that comprise

the GUI; a `jit.lcd` object is used to draw to those matrices that are then combined into a single matrix for on screen display with a `jit.pwindow` object which then provides mouse interaction data for the system.

The spiroid-frequency-space equations – for converting between *theta* and midi-pitch values, given at the end of §4.1.4 – are used in this piece of software; Figure 4.22 shows where this happens in the `mouse_interaction` sub-patch: the *theta* value having been derived from the mouse coordinates on the GUI, which may or may not have been quantised to a radius line (please refer to the above mentioned documentation for details of that process).

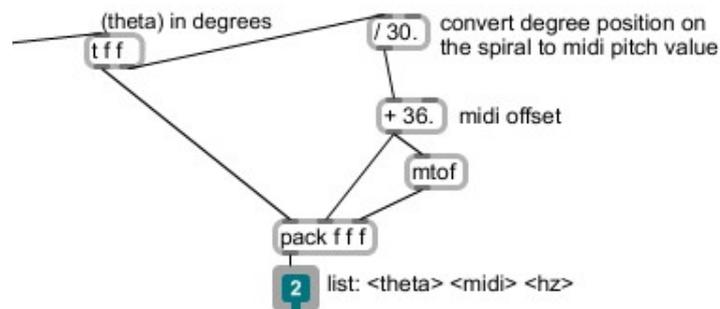


Figure 4.22: `theta_to_midi`

#### (4.4.1.c) Angular manifestation

The `cartopol` and `poltoacar` objects, that are used in this piece, natively place the value zero angle at the three o'clock position; when using a midi-pitch offset value that is a multiple of twelve, the pitch-class of 'C' is placed at that angle. Rather than adjust the algorithm to move 'C' to the twelve o'clock position, as was done for previous implementations of the spiroid, it was decided that the native manifestation was perfectly acceptable. Critical reflection of the situation thus created reveals aesthetic support for the placement of pitch-class 'A' which is now found at twelve o'clock conceptual reference point of a clock face, it the angle from which minutes are counted; one may

think also of how North is cardinal reference for magnetic compass points, and then of how 'A 440' is the standard reference point for mapping the midi-pitch scale to Hertz.

#### (4.4.1.d) A topic for NIME

Further work to nRadii system would appear to be relevant to the field of new interfaces for musical expression, typified by the annual conference of that name (NIME).<sup>94</sup> Within the archive of that conference similar works can be found; a pitch-class circle was, for example, used by Loïc Kessous (2002) as the basis for mapping the position of the stylus on 'the A6 Wacom graphire2 graphic tablet' to pitch-class, while octave-level was navigated either by button presses, or by crossing the 'start' of the circle to give the interface 'a boundless range'. An interesting feature of the pitch-class circle mapping in that work is the use of transfer functions across each 12TET semitone angle of the circle. Figures from that paper are included here as Figure 4.23, below; they show (Kessous, 2002, p. 114):<sup>95</sup>

**“Portions de Camembert” representation of the fundamental frequency angular control [and] Four examples for transfer function that can be used for fine-tuning (linear, sign (x)\*x2, x3, x5).**

[...] The transfer functions are designed to allow the user to have stable and efficient control of tonal pitch accuracy and force him or her to make a conceptual effort to have fine tuning variations.

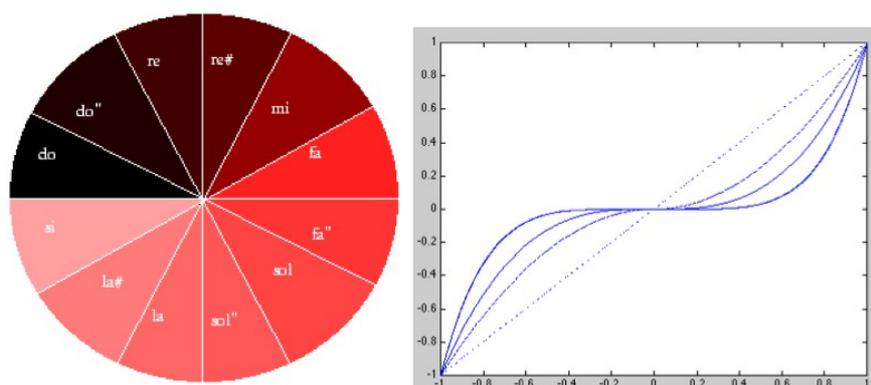


Figure 4.23: Kessous2002Mapping

94 'The International Conference on New Interfaces for Musical Expression [...] started out as a workshop at the Conference on Human Factors in Computing Systems (CHI) in 2001.' (<http://www.nime.org/>, accessed 20131217).

95 The pitch-class-circle in this work by Loïc Kessous appears to have the lightness attribute of the HSL colour-space mapped to 12TET divisions of the circle, while the hue is constant.

## 4.4.2 A spiroid widget design

### (4.4.2.a) A type of dial that is based on the spiroid

Both the spiroidArc display and the nRadii GUI are atemporal in their design: each is representing an instantaneous view of now in the frequency-domain (notwithstanding the paradoxical nature of that statement when the uncertainty principle is considered).

In §5.3, the working prototype of a time-space representation *CirSeq Zero* is described; sinusoidal oscillators are used within that work to make audible the divisions of time that are created by the visual representation. To control the frequencies and amplitudes of those sinusoids in time-space I created a spiroid-frequency-space widget, which is described here.

The spiroid widget works as a three-dimensional controller manifest on screen plane as similar to a dial type GUI element; whilst writing this document, the name 'spiroidial'<sup>96</sup> has presented itself to encapsulate the concept. The spiroidial is conceptually descended from the spiroidArc in many ways. Each spiroidial on screen represents three constituent dimensions of a partial: pitch-class as angle, octave-level as radius, and amplitude as opacity. In both the spiroidArc design and the spiroidial the spiral curve of the frequency-domain continuum is not itself shown; this is in contrast to the input method exhibited in the nRadii system, described above, where the spiroid-frequency-space is explicitly mapped on a visible curve on screen. Within the interface of *CirSeq Zero*, for which the spiroidial was invented, spatial location on screen is representative of temporal data, and there are many spiroidial controls distributed within the GUI. Each spiroidial, therefore, specifies pitch-class and octave-level according to its own polar origin at the place where it is located on screen.

### (4.4.2.b) Mouse interaction with a spiroidial

There are two conventional methods by which a dial type widget on screen may be affected by

---

<sup>96</sup> The word is a contraction of spiroid and dial; I thought to, perhaps, let the second 'i' be silent to put emphasis on the 'dial' at the end of the word, as in 'spiro-dial', but in practice I read the word as it appears.

mouse interaction: (1) after the mouse button is click on the dial – the mouse down (md) event – moving, or dragging, the mouse up or down will increase or decrease the value represented by the dial with the visual manifestation of angular change clockwise or anti-clockwise; it is common for the mouse pointer position on screen to be reset to where the md happened, after the mouse up (mu) event. (2) alternatively the mousing that happens between md and mu on the dial may be used to explicitly control the visual angle of the widget and thereby affect the value represented.

For the spiroidial widget, I chose to base the model of interaction upon the first (up/down) method, and extend this to the control of three discrete parameters with a spatio-visual metaphor. The metaphor – see Figure 4.24 below – evokes the operation of a manual transmission gearstick in a car: the md point on screen is comparable the neutral position of a gearstick, in that movement from there may optionally be left or right, before being forward or backward (up/down on screen) in order to affect the desired change in the system. The analogy is loose, however, because a manual transmission gearstick is used to access discrete values at each terminus. In contrast to that, the click-and-drag movement on a spiroidial widget uses the optional left/right stage of the action to select one of three parameter domains which will then be affect by relative change in proportion to the up/down movement.

To alter the amplitude value represented by a spiroidial, click and drag up/down; the opacity will change in response. To alter the pitch-class, click and then first drag to the left (to engage pitch-class control), and then drag up/down; the angle will be affected in the visual domain. To alter the octave-level, click and drag to the right and then up/down; the radius of the spiroidial will increase/decrease.

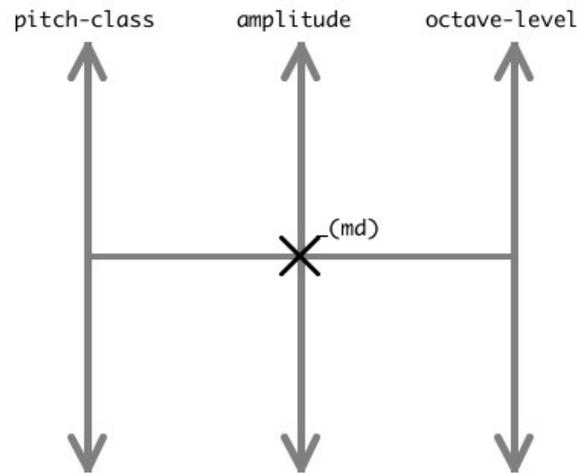


Figure 4.24: spiroidial\_mouse\_interaction

#### (4.4.2.c) Specific implementation of a concept archetype

The spiroidial concept has now been described as an archetype of a GUI element, but its only implementation in my work is as part of the *CirSeq Zero* system described below. The concept was developed in response to the challenge of wanting to specify frequencies and amplitudes at points in time-space. The implementation is thus deeply integrated to the *CirSeq Zero* system: there is no 'spiroidial abstraction' to be found, and that name was not used at the time of its programming.

There are no further examples of spiroidials being used in this project, but this is more the result of a change in approach imparted after the completion of the *CirSeq Zero* piece (see §5.3), as opposed to being an indication of a lack of potential. The concept may yet find use in future works, but if so then it would be likely that the radius to octave-level mapping would be reversed so that a larger radius would give a lower frequency in order to be more reflective of physical reality in which larger bodies tend to resonate at lower frequencies.

## 4.4.3 SpiroidArc plus sequencing

### (4.4.3.a) Frequency markers

The SpiroidArc patch described in §4.3 was modified to include an implementation of 'numbered frequency markers': a set of up to nine points on the spiroid-frequency-space that can be marked by pointing with mouse and pressing a number key on the qwerty keyboard. An automatic function has also been added which, at the click of a button, sets markers to the frequencies currently present in the partial analysis of the input audio; in this case the number for frequency markers placed will vary according to the audio input, up to a maximum of nine.

Within the programming, the marked frequency values are stored in a **coll** by which they are available to other patches. The format for each line of the **coll** is: <marker number>, <x-coordinate> <y-coordinate> <midi-pitch>; the coordinates are used for drawing the marker on an **lcd** overlay to the spiroidArc display, while it is the midi-pitch value that will be of most use to other patches. Read and write commands for the **coll** are available (see first annotation in §(4.4.3.c) below).

### (4.4.3.b) Sequencing with markers in a time-space

To make use of the spiroidArc frequency markers a simple step sequencer with a circular GUI has been implemented. This spatial mapping of a time loop on the plane is thought of a 'time-space' representation to be used alongside the spiroid-frequency-space representation. The output of the circular time-space discussed here is midi data: note-on and note-off messages.

For the purposes of this document, it is more convenient to consider the time-space of the current context independently from the CirSeq time-space that is presented in the next chapter (§5); they are, however, quite closely related. Cyclical traversal of time within both these time-space representations is conceived as happening in the same way that a standard clock hand is known function: as time passes the angle representing now is seen to move from one point on the

circumference toward the next, continuing around the circle and over again.

The time-space that I implemented for generating midi note messages is similar to *The Cyclotron* by Dan Trueman (2007). If this particular avenue of compositional exploration were to be taken further than it has been within this project, then the source code for *The Cyclotron*, that Trueman has made available, could be used as the basis for further work.

#### **(4.4.3.c) `_spiroidArc+timespace` annotation**

Demonstration of this work is found in the `spiroidArc+timespace` folder in the `chapter4` directory of the portfolio: Figure 4.25 shows `_spiroidArc+timespace.maxpat` in use with seven areas highlighted for annotation.

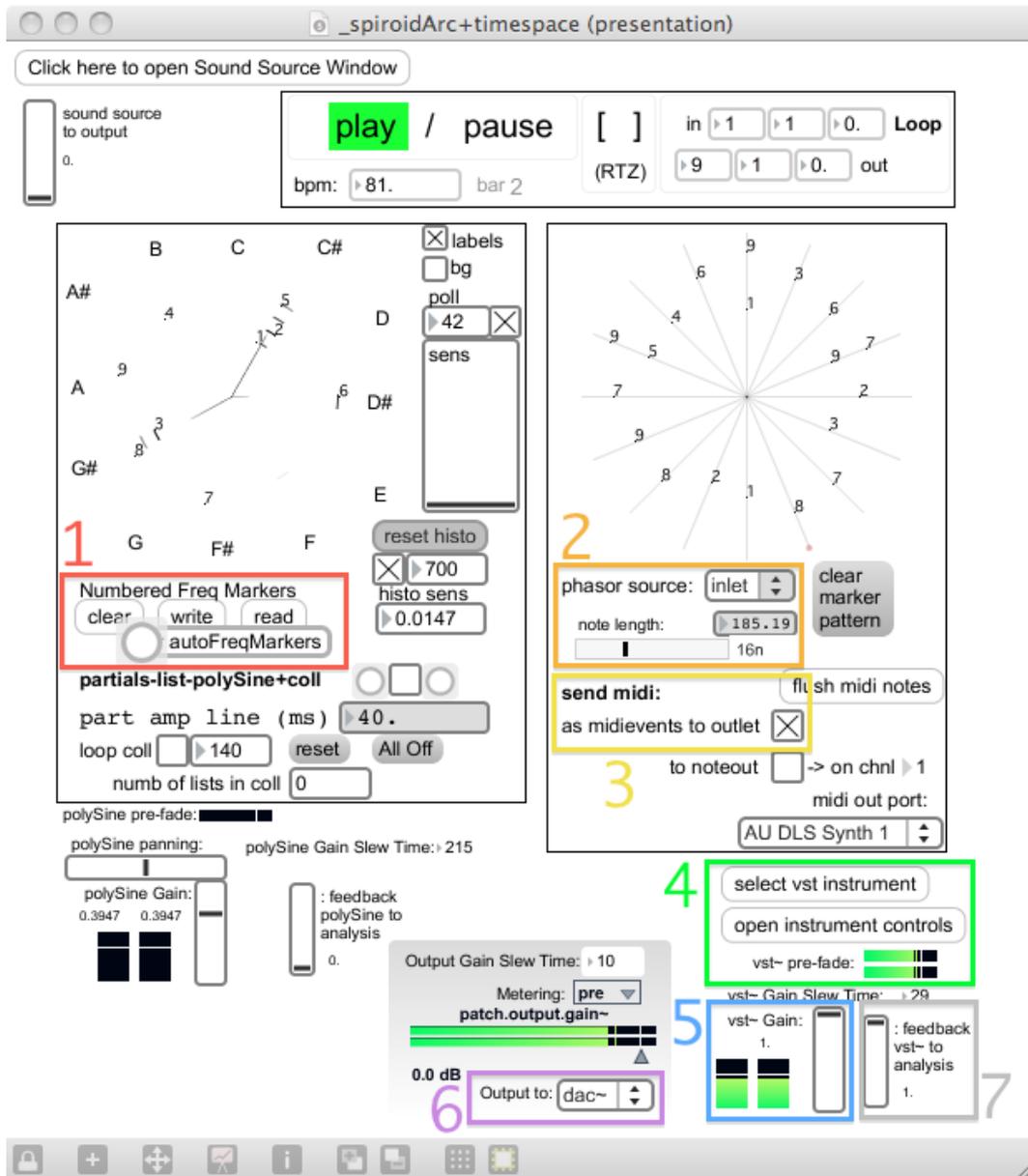


Figure 4.25: \_spiroidArc+timespace.maxpat (annotated)

To get started with using this patch, the seven areas indicated in Figure 4.25 can be navigated in the given order, but there are many other ways in which the system can be configured and used. For ease of reference, a copy of the .png file that forms Figure 4.25 is included in the folder of the portfolio, along with the maxpat. Annotation to correspond to the indicated areas:

1. The controls added to the interface of the spiroidArc patch that pertain to the numbered frequency markers; see §(4.4.3.a) for details. Note, also, that input to the spiroidArc module

patch can be provided via the 'Sound Source Window' which has the ADC and a sound file player as options; below the 'open' **textbutton** is a slide to set level for sending that source to the output (see annotation 6, below).

2. The 'phasor source' in the embedded time-space patch will default to being labeled as 'off', but it is actually set to receive **phasor~** input from its inlet when the patch is loaded; the input signal, ramping from zero to one drives, drives the motion of now around the time-space representation which is here quantised to 16 steps, represented as the spokes on the GUI. The transport interface module – linking play / pause to the qwerty space bar – above the time-space module, gives access to the 'GlobalTransport' timing system of MaxMSP; the object connected to the input of the time-space is declared as **phasor~ 1n @lock 1**.

Using the same 'point with mouse and press a number key' method described for placing frequency markers on the spiroid-frequency-space, numbers can be placed on the spokes of the sequencer. As the time-space is traversed, each spoke is evaluated; if there is a marker on the current spoke, and if that marker currently has a valid midi-pitch value stored in the **coll**, then a midi note message is generated for that pitch. The length (duration) of the generated midi notes is set by the horizontal slider that is shown within annotation area two of Figure 4.25: the convention for note duration value naming that is native to MaxMSP is used here, but the duration in milliseconds for that value at the current tempo (bpm) is also displayed.

3. For the setup being described here, the midi note data is set to be sent 'as midievents to outlet' which is the format used by the **vst~** object.
4. Select a suitable VST instrument to receive the midi note data; the pre-fade meter gives visual confirmation that audio is being generated.
5. Slider to set level of the audio signals coming from **vst~** towards the output (with 'post-fade'

metering).

6. The 'output to' option will default to 'off'; the other two options for this are 'dac~' and 'outlets' where the later is used in case of `_spiroidArc+timespace.maxpat` being embedded within another patch.
7. The final item of annotation here is the slider that sets level of gain for the audio output of `vst~` to go to input of the `spiroidArc` patch for analysis and display. This feedback option allows for frequency markers to placed and moved on the `spiroid-frequency-space` GUI in response to the audio that they themselves contribute to the control of.

#### (4.4.3.d) Evaluation

In retrospect it is now thought that there is great potential for this type of `spiroidArc` and time-space pairing; within the context of this project, however, its focus of the generation of midi messages (originally targeting a hardware synthesizer, but demonstrated with a VST instrument) is somewhat removed from the aesthetic motivation of working closely with the soundmaking processes at the data level of processing in software systems.

The continued prevalence of midi, three decades on from its introduction (MIDI Manufacturers Association, 2013), is perhaps in part due to its encapsulation of the CMN paradigm, that has several centuries of influence on Western culture, and in which music is predominantly conceived of as events called notes. My work is founded on a conception of music as being existent in the process of soundmaking, and my focus is on the visual representation of sound in software during the act of composition. That focus has led to a process of soundmaking that seeks to integrate multiple layers of abstraction within the creative work; the `spiroid-frequency-space` concept provides representation of the frequency-domain throughout my continuing practice.

## 4.5 Assessment of the spiroid

This chapter has introduced and contextualised the concept of a spiroid-frequency-space for the representation of frequency values on the plane. The use of a polar-coordinate-type mapping to align octave-equivalence creates an intuitively understandable representation of the frequency-domain.

### 4.5.1 Recapitulation

When the spiroid is employed as a display for analysis data – as in my `spiroidArc` implementations (§4.3 and §4.4.3), and in *SPIRAL* by Photosounder (§4.2.3) – the view is akin to that of an oscilloscope, only of the frequency- rather than time-domain. I am also inclined to suggest simile of this display to Cymatics, but whereas Cymatic figures give some indication of harmonic resonance within the physical system used, the abstracted mathematical system used by the spiroid gives an impression of the harmonic structure within the sound signal input.

Examples for spiroid-based interfaces that I have designed for human input to software have been described. Both the `nRadii` example (§4.4.1) and the example of frequency markers (§4.4.3) are based on the specification of a value as a point on the plane; the mapping of a cartesian point on the screen may, or may not, take into account quantisation to given pitch-class angles or octave-levels. This spiroid-on-the-plane method for inputting values has the advantage of being compatible with the visual output of frequency analysis within the same area of the computer screen. It is also advantageous that this method is able to simultaneously represent many values upon the same spiroid mapping. A spiroid-frequency-space mapping is one of several spatial mappings used for setting parameter-values in the main area of the `sdfsys` environment interface (§6).

For situations in which a large area of screen is not available for representation with a

spiroid mapping, the spirodial has been designed (§4.4.2). In that design, a 'local' spiroid mapping is defined at the centre of each dial-like widget; the angle of the 'needle' on, and the radius of that dial. While the spirodial has been introduced in this chapter, its use is described as part of *CirSeq Zero* in the following chapter.

## 4.5.2 Spiroid beyond this project

This chapter has shown that the general concept of what I call a spiroid-frequency-space is a long established form of visual representation; in addition to the Drobisch type of helix described and extended by Shepard (see §2.2.2), the pitch and frequency-domain aspects of sound are often represented two-dimensionally with a spiralling form.

There are many routes by which one may arrive at a concept the same as, or very similar to the spiroid. For me it was discovered after a compositional process involving hand drawn representations of pitch-classes, followed by scales of notes, and then continua, and since I began development of the spiroid (c. 2005), it has been a matter of some perplexity that this archetypical visual representation is not commonly found in computer music software. During the period of this project, however, there have been new software releases found to have spiroid-like interface elements, and I hope to see many more in the future.

The most recently found example of new software that would appear to be using a spiroid-based mapping was presented at the 'IRCAM Forum Workshop 2013' (November 22); it has not yet been released, but is described in abstract for that event (Picasso & Hélie, 2013):

### **Presentation of SNAIL iOS App**

The SnailAnalyser is a frequency domain sound analyser with an original chromatically aligned representation. Compared to standard analysers, this representation allows simple visualization of active zones in a sound (similar to a spectrum) and organises frequencies per notes on a tempered scale like a tuner.

At the time of writing, further details of this project were scarce; the authors, however, have

kindly directed me to one document that is public, and which gives 'the basic idea and a basic result with a picture' (Figure 4.26).<sup>97</sup> They state 'that the "true" application gives "better results"', but that they 'cannot give you any detailed information before (probably) June' (Hélie & Picasso, 2013).

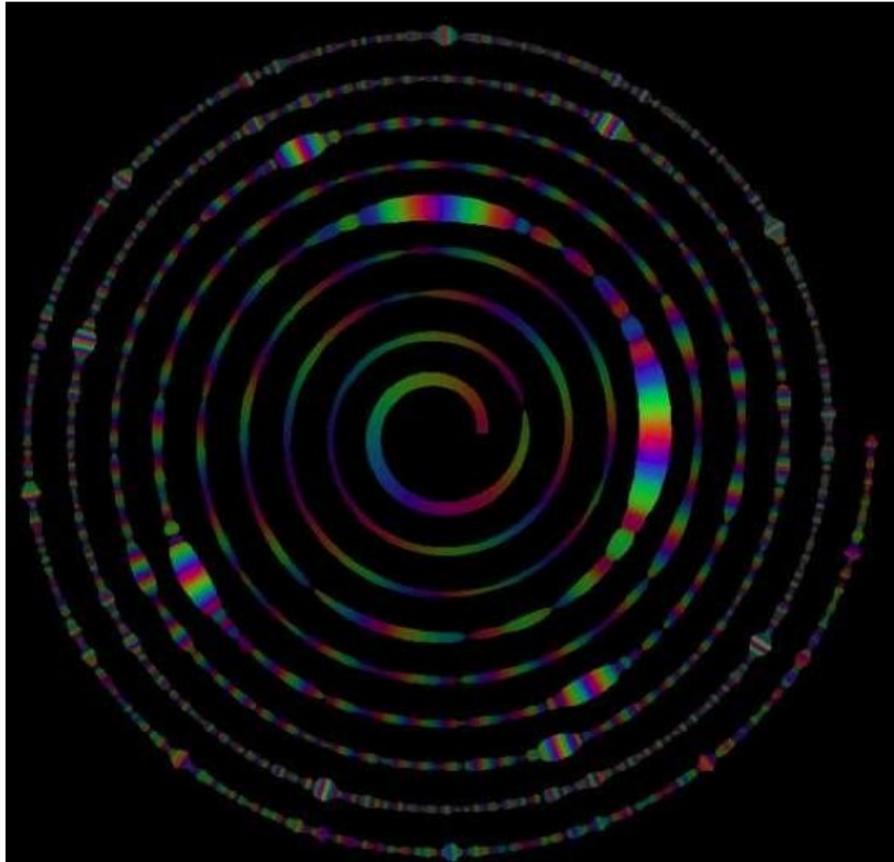


Figure 4.26: SnailAnalyzer

With software products such *SPIRAL* by Photosounder, and the *SnailAnalyser* becoming available, there is likely to be an increase of interest in the subject of spiroid-like representations. It remains to be seen whether or not the term spiroid will catch on in relation to the concept; I have registered the [spiroid.info](http://spiroid.info) web domain towards collating information on spiroid related things, both historical and contemporary.

### 4.5.3 Chapter conclusion

Spiroid-based systems could be utilised in pedagogy, entertainment, and scientific research

<sup>97</sup> Online at [http://recherche.ircam.fr/anasyn/helie/hdr/HdR\\_ThomasHelie\\_v1.5.pdf](http://recherche.ircam.fr/anasyn/helie/hdr/HdR_ThomasHelie_v1.5.pdf) (see page 90 → pdf page 102)

situations, as well as in the arts. Within this project the spiroid-frequency-space concept has techno-aesthetic significance. It is the basis of practical methods for interface design, being the default mapping assumed for the visual representation of frequency-values. It embodies key aspects of human perception, both in the auditory-domain (octave-equivalence of pitch-class), and in the visual-domain (evoking form constants originating in the visual cortex; see §3.4.2). It embraces non-rectilinearity with a Cymatic-like atemporality bringing focussed attention to the experience of now when working with sound in computer music software.

# 5: CirSeq

To complement the spiroid-frequency-space representation in new software for composition, circular representations of time have been explored. By separating the frequency- and time-domain conceptual aspects of sound, but then using visually similar representations of each, the idea was (in part) to be able to take data from one domain for use in the other, where the visual representations facilitate that transduction.

Near the end of the previous chapter (§4.4.3), a sixteen step circular sequencer was described for its use as part of a spiroidArc based system, and *The Cyclotron* (Trueman, 2007) given as a contextualising example for that model of cyclic time-domain representation. The investigation described in this chapter is closely related to that time-space system, but this exploration was derived from an approach of geometrical drawing in search of a logical stepwise genesis for a time-space model in a way that can be thought of as an attempt to replicate, for the time-domain, the process that begat the spiroid.

As with the introduction of the spiroid-frequency-space, at §4.1, CirSeq will be thus introduced through a sequence of geometrical drawings that demonstrate how it was that I approached the objective of developing a time-space representation as a counterpart to the spiroid (see §5.2). Additional contextualisation can then be given for the work, and indeed there are many precedents for the CirSeq concept; it will be shown, however, that my ground up approach to the circular representation of periodic time has founded an original contribution in the field: whereas visually and conceptually similar designs are plentiful, I have yet to find duplication of the logic and utility that CirSeq is based on and can offer.

Much of the thinking and practice going into the development of CirSeq is tied to the parallel thread that is called thisis: §5.1 describes the thisis concept, its connection to CirSeq, and

how these two elements of the project are intertwined, both prior to and during the implementation of *CirSeq Zero* (which is the software composition described in §5.3). Development did begin toward what would have been 'CirSeq Alpha', but the critical thinking coming out of the *CirSeq Zero* developments, which became foundational for the next stage of the project, did not include continuation of this investigative trajectory.

## 5.1 CirSeq and thisis

Development of CirSeq is closely related to my practice of geometrical drawing which has led to a software system that I call thisis. The shared origins and software overlap of CirSeq and thisis will be described here. These origins also form part of the roots to the model of composition that was exemplified in §3.7.3, particularly the idea of taking abstract thought and then specifying concepts as visual representations.

The idea of *technē* is again brought to bear because seeking knowledge and awareness of how things are made and done is central to the philosophy that underlies this work. Over periods of days, drawing for several hours each day, I would 'get to know' certain geometrical constructs in the same way that I had spent time 'getting to know' soundmaking systems or collections of recorded sound in the composition of music. To bring the growing sense of geometrical *technē* into play with computer music the objective became to make precise measurements of distances between points on the plane. If one begins with a particular length of line, then a host of different lengths of line can be created that are somehow related to the first, and at that time the idea was to use such sets of measurements, along with the positions of identified points on the plane, as parameter values in composition of music through software. To do this I needed a software representation of the drawing process.

Existing computer-aided design software, although powerful beyond my needs, did not seem

to offer the work flow that I had already internalised through paper based sketching, and to learn how to translate my processes into the processes that had been programmed into software – by someone else – would significantly alter the nature of the work that I had already started.

Introduction of new ideas about how to make and do things can be good, but it can also detract from the creative objectives at hand. Knowing exactly what data I was looking for, and also knowing what to do in order to find that data, the task was to program the computer to be able to receive and interpret the instructions that described that process.

The first version of the thisis system was therefore made to take textual input describing where to put points on a plane, either at absolute coordinates (using a cartesian system similar to that of OpenGL, with [0,0] in the centre and extending, square, to  $\pm 1$ ) or at positions relative to points that have already been put (such as at an angle on the circle defined by one point drawn around another). Commands for drawing lines from one point to another, and as one line around another were added along with 'where is?' and 'how far?' query commands that return precise numerical values. The resolution of the drawing did not need to be high because it was the numbers being stored by the system that were of main interest.

An **lcd** object – or in some versions, **jit lcd** – was used for the visual manifestation of the commands and the named point coordinates are stored in a **coll**. The first versions of the system employed a live coding paradigm with commands being typed in to the patch itself, but the use of external text editing software came to be preferred with a script being automatically parsed, by the system in MaxMSP, every time the file is saved in the text editor.

The figures presented in the following section were, as were many of the figures throughout this document, made with a version of the thisis drawing system; they are the png format exports from **lcd** renderings of text file scripts written in the thisis scripting language.

## 5.2 CirSeq inception

### 5.2.1 Basis for the search

After embracing the spiroid-frequency-space as a welcome alternative to rectilinear representations of the frequency-domain, the creation of a non-rectilinear representation of time was sought. More specifically, it was decided that a finite duration or period would be represented in a circular way; the data contents of that period would be cyclically looped, although that contents may itself change over time such that musical progression be possible through multiple cycles of the time-space. The data represented within this model would be for parameter control in a soundmaking system, rather than being audio data that could be directed to a DAC.

The spiroid-frequency-space is a representation that is very much based on polar-coordinates, and so – with the domain dichotomy described at §3.6.2 in mind – it seemed appropriate for a time-space representation to be created that, despite being of circular design, has its basis firmly upon cartesian-coordinates. Although this representation is being constructed with adherence to the cartesian grid, its non-rectilinearity is based on a clock-like angular motion for traversal of time, and so this conception of time-space has a polar aspect present at its core.

The CirSeq folder in the chapter5 directory of the portfolio contains [CirSeq Inception.pdf](#) (created October 2011) which details eleven illustrated steps of development of, and some notes pertaining to the background of, the CirSeq concept; below is a three step summary of how the concept is constructed as a geometrical progression.

### 5.2.2 A circle on a cartesian plane

With a cartesian grid of integer unit divisions as the basis of the time-space representation, a circle is drawn with a radius of one to represent a period of specific duration. This circle is, then, found to be touching integer coordinate points of the grid at four places.

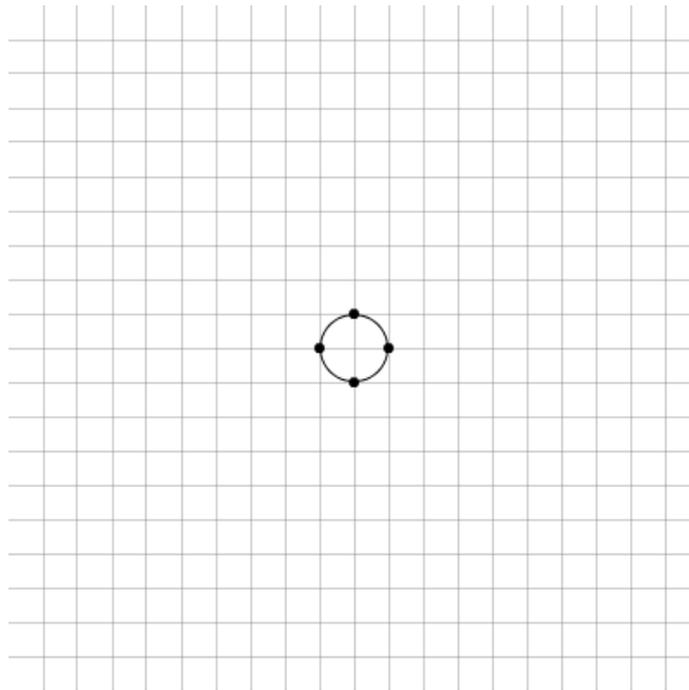


Figure 5.1: cirseq\_thisis\_01

Thus, the period represented by this circle on the time-space plane is immediately quantised to quarter divisions of its whole, and four equal durations are manifest between the points marked by the dots in Figure 5.1; if the direct path through these points is drawn, then a square (at 45 degrees of rotation relative to the orientation of the grid) is formed. The square formed by the quantisation of the first circle is shown in the centre of Figure 5.2 below.

### 5.2.3 Another circle on the plane

A second circle is then drawn on the grid, concentric to the first, and with a radius that is twice that of the first. It is important to note that each concentric circle is drawn on the time-space plane to represent the same period. As with the circle that has a radius of one cartesian-coordinate unit, this radius two circle is again found to only touch upon the grid at four integer coordinate points. Corresponding to the same four time-space angles that quarter the period represented by the radius one circle, these points are marked visually by a dot of the same type.

It is shown, in Figure 5.2, that on the straight lines that directly connect the four corner

points that create the 45 degree square within the radius two circle, there is found another set of points that intersect with the cartesian-coordinate grid; these occur at the time-space angle that is mid-way along each of the four line segments: each has been marked with an 'x-dot'.

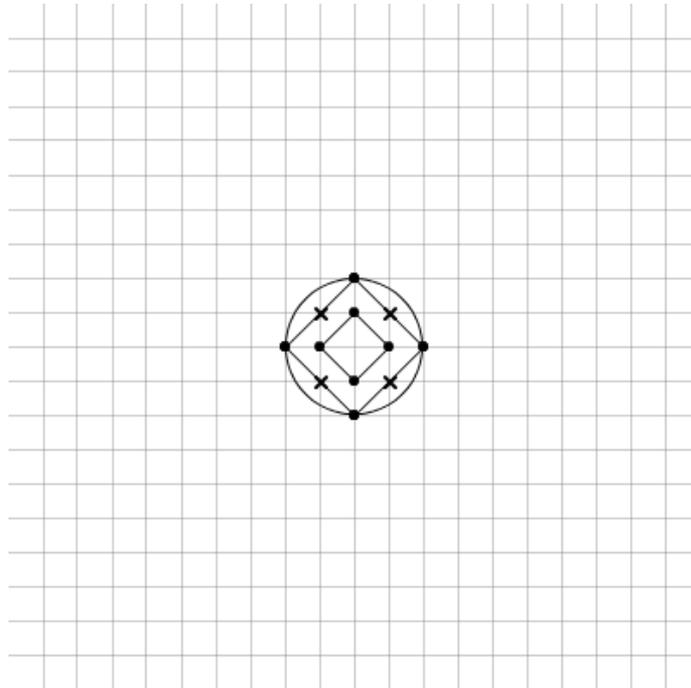


Figure 5.2: cirseq\_thisis\_02

## 5.2.4 Double and double again

After the radius two circle has been drawn, and those extra quantisation points found, the process of doubling the radius can be twice repeated: when the concentric circles of radius four and radius eight are drawn on the grid, the total number of grid-quantised points that are found to occur upon the 45 degree square line path inside each circle is also found to double for each doubling of the radius. Those two circles are shown in Figure 5.3 along with their 45 degree square line paths; 'open-dot' and 'square-dot' markings are used for the newly added time-space angle quantisation points that are introduced by the two new circles drawn.

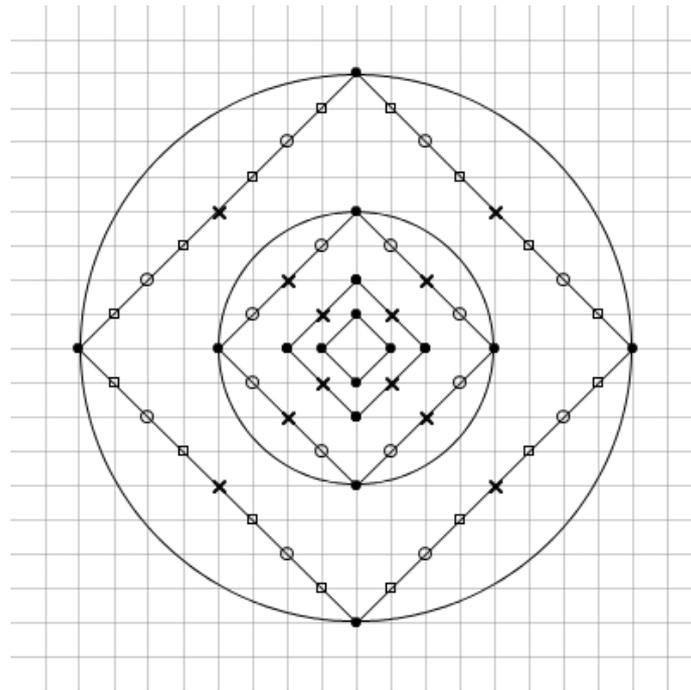


Figure 5.3: cirseq\_thisis\_03

Note that, along any square line path, while the linear distance between each adjacent quantisation point – or 'node' – is always the same (in the units of the grid, it is the square root of two), the duration of the period represented by that distance is dependant on the radius of the circle to which that path, and those nodes, belong. Whereas the notion of polar-coordinates on the spirioid-frequency-space equates angle to pitch-class and radius to octave-level, on the CirSeq time-space angle equates to temporal-phase and radius to temporal-resolution: a greater radius value creates more sub-divisions of time.

## 5.2.5 Common time

It will be apparent by now that my goal of designing a time-space representation that is based on the quantisation of a circle to the integer coordinates points of a cartesian grid has resulted in a literal reinventing of a very well known wheel: the scheme described above corresponds exactly to the common method rhythmic notation present in CMN; a sketch of that conformity is shown Figure 5.4.

If a radius three circle is drawn on the CirSeq time-space, then twelve equal divisions are found of the whole period (three nodes per quarter period), and with the parlance of CMN it can be said that these equate to triplet quavers; to then double from that to a circle of radius six is to find triplet semi-quaver divisions (six nodes per quarter period).

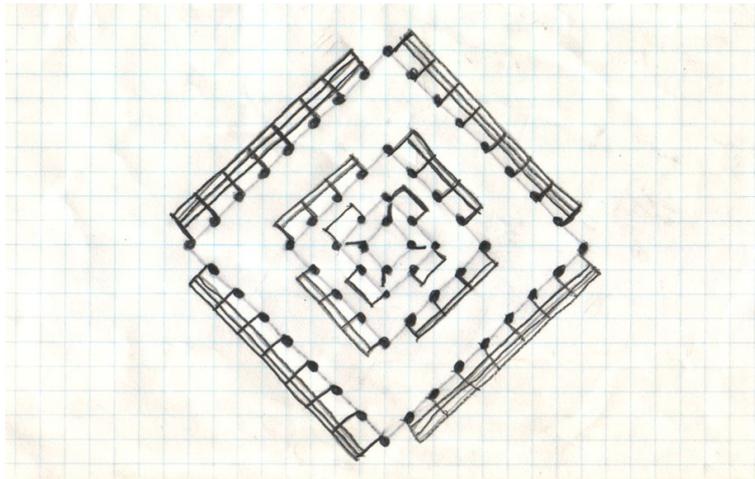


Figure 5.4: 4n8n16n32nNotes

Although it had not been my intention to arrive at such a conclusion, the occurrence of this formation now seems inevitable for a time-space that is devised by this method of geometrical interaction between circles on the squares of a grid; the square line paths created produce multiples of four nodes per 'system',<sup>98</sup> and it is used because the circle intersects at precisely four points.<sup>99</sup> The resultant structure that has been found through the geometry explored can thus be directly related to crochets, quavers, and so on, but the geometrical reasoning for the progression of sub-divisions, described in my work above, is not so common: for the notes of CMN, sub-divisions of duration value are more usually depicted by a hierarchical tree-type structure that begins with two minims below a semibreve, and above four crotchets. Some circular representations closely resembling

<sup>98</sup> 'System' is the term used to describe the square line paths of node that are created within the concentric circles; it is, I now think, an unfortunate choice of word, but it will, nonetheless be encountered in the description below because it is coded into the software.

<sup>99</sup> The circumference may, for larger radii, come close to (or maybe even touch) the cartesian-coordinate grid at other angles (especially if the radius was very large), but only those four quarter angles are present for every integer radius circle.

CirSeq have been identified: one in an educational worksheet by Susan Paradis (2008), see Figure 5.5 (this is, however, an abstract representation to aid cognition of theory, rather than a functional time-space traversed by angular motion); and another in two software applications for the Android platform (Google, n.d.) by One Man Band Studios S.A.S: both the *bComposer Metronome* and *bComposer Rhythm* apps (2013a, 2013b) use circular divisions in their GUIs that are akin to the CirSeq pattern, see Figure 5.6.

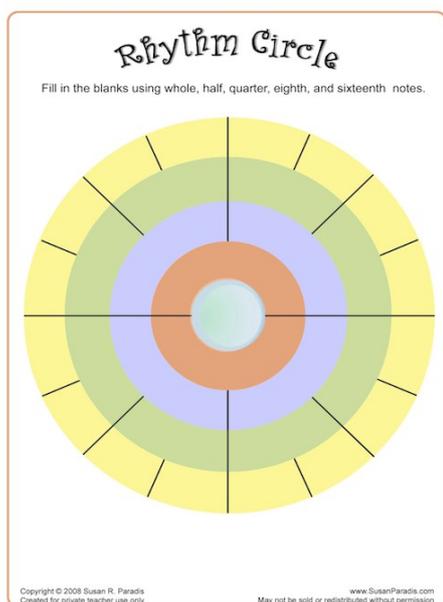


Figure 5.5: RhythmCircle

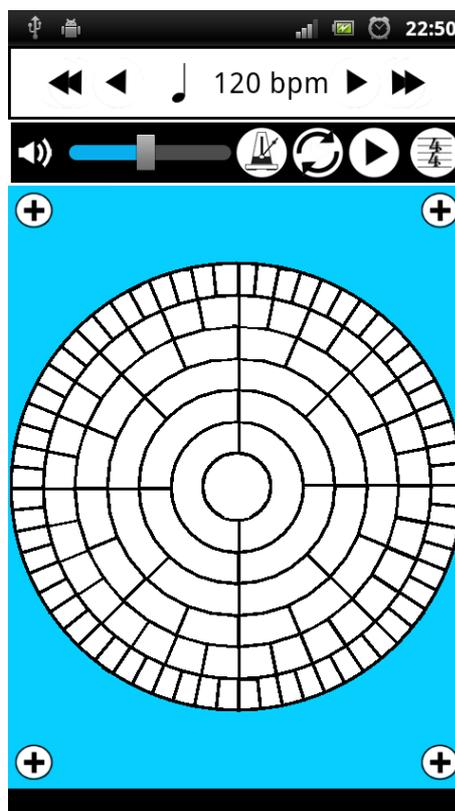


Figure 5.6: bComposerRhythm

In contrast to those similarly constructed examples, above, there are also many instances of circular time-space representations that do not include the radius-resolution aspect of the polar-type traversal of time on the plane. Of the step sequencer interfaces that arrange their discrete steps in a closed loop, those that have only one track/path/system of steps/nodes are of less interest here. It does, however – given the amount of historical context provided elsewhere – seem appropriate to

include some mention of *The Circle Machine* created by Raymond Scott in the 1950s: Bob Moog has written of Scott as an inspiration and friend, and it seems that Scott had been very secretive of his technological advancements in electronic music in those early years of its development as a field. *The Circle Machine* is now cited as one of (if not the) first electronic soundmaking sequencer systems (Winner, 2001, n.d.).<sup>100</sup>

One example of a circular interface that does include concentric systems of step sequencing tracks is the *Loopseque* software (Casual Underground, n.d.) which has four concentric systems, each having sixteen steps, or nodes, that can be toggled on or off (Figure 5.7). In *Loopseque*, the polar aspect of radius has been implemented to provide access to four different samples or pitches of the 'instrument' for which that wheel is sequencing. The visual consequence of maintaining the number of divisions at different radii is that the size of the nodes is much smaller for that inner system compared to the outer.



Figure 5.7: Loopseque\_Wheel

<sup>100</sup> Pre-electronic examples of soundmaking sequencer technologies include disc and cylinder music-box mechanisms, and the 'player piano' mechanisms that are related to the Jacquard loom and from which comes the 'piano roll' metaphor that is now a commonplace paradigm for the visual representation of midi note-on and note-off events.

The paradigm of mapping radius to pitch-class within a circular time-space representation is also seen in the GUI for the *AlphaLive* software that works with the *AlphaSphere* instrument (nudesine, n.d.); that 'circular sequencer interface' is illustrated in Figure 5.8 below. At first I thought the nodes of the systems were coloured light and dark to correspond to the keys of a piano for visual orientation within the 12TET pitch-space, but on closer inspection the twelve tracks seem more to just be alternating; where there is a keyboard layout within the GUI it has a set of twelve pitches highlighted blue, and this suggests that that is selecting the pitches that are mapped to the radial systems on the time-space.



Figure 5.8: AlphaLive

## 5.3 CirSeq Zero

The work presented here in §5.3, *CirSeq Zero*, has been described as both a working prototype, and as a software composition. It was created as an exploration, and demonstration, of an idea, but in its final state brought back the question of what may constitute a 'piece' of music. In this case it is the software configuration, embodied as a collection of maxpat (and supporting) files, that is taken to

be the piece named *CirSeq Zero*.

The piece comprises an implementation of the CirSeq circular sequencer concept to control a set of sinusoidal oscillators. There are at least four different ways to interact with the soundmaking aspect of the piece – or in other words to perform it – the most passive of which is to click the **toggle** labeled 'run random decide cascade', though in this case, still, the performer may wish to change the cycle duration, either before or during the performance. The performer must decide when and how to stop, and there are a great many other possibilities for variation of the piece within each performance. The software configuration may equally be thought of as something other than a composition per se; but surely the questionable-interpretation-of-what-the-thing-is is itself suggestive that it is a 'work of art' of some kind, else why question it?

This section is structured as follows: an overview of *CirSeq Zero* is provided in §5.3.1, and §5.3.2 gives an explanation of its technical workings, including details of how one may interact with the piece. The random decide cascade aspect is discussed in §5.3.3 under the heading: Making decisions to organise the soundmaking. Because that cascade is an example of automated decision making, there is discussion, also in §5.3.3, of further choices that are to be made in performance of the piece. To conclude this section of the document, §5.3.4 provides an example of how *CirSeq Zero* can be performed via meta-patch control; the exploration of *CirSeq Zero* through the meta-patch described, there, becomes the basis for discussion in §5.4.

## 5.3.1 Playing (with) the piece: an overview of CirSeq Zero

### (5.3.1.a) The CirSeq Zero patch

The reader is invited to open [CirSeq\\_Zero.maxpat](#) from the similarly named sub-folder within the CirSeq folder in the chapter5 directory of the portfolio; this maxpat is a slightly edited copy of [cirseq\\_021c.maxpat](#) – a patch that is also included, and was last modified September 2010 – just

four cosmetic changes were made to it for its presentation here.<sup>101</sup>

When `CirSeq_Zero.maxpat` is loaded it will look as shown in Figure 5.9 (minus the numbered annotation areas with which the image of the patch is here augmented). The annotation areas depicted shall be referred to as: (1) the output section; (2) master phase; (3) lcd-GUI; (4) cycle duration settings; (5) system selection section; (6) set all messages; (7) toggle to run random decide cascade.

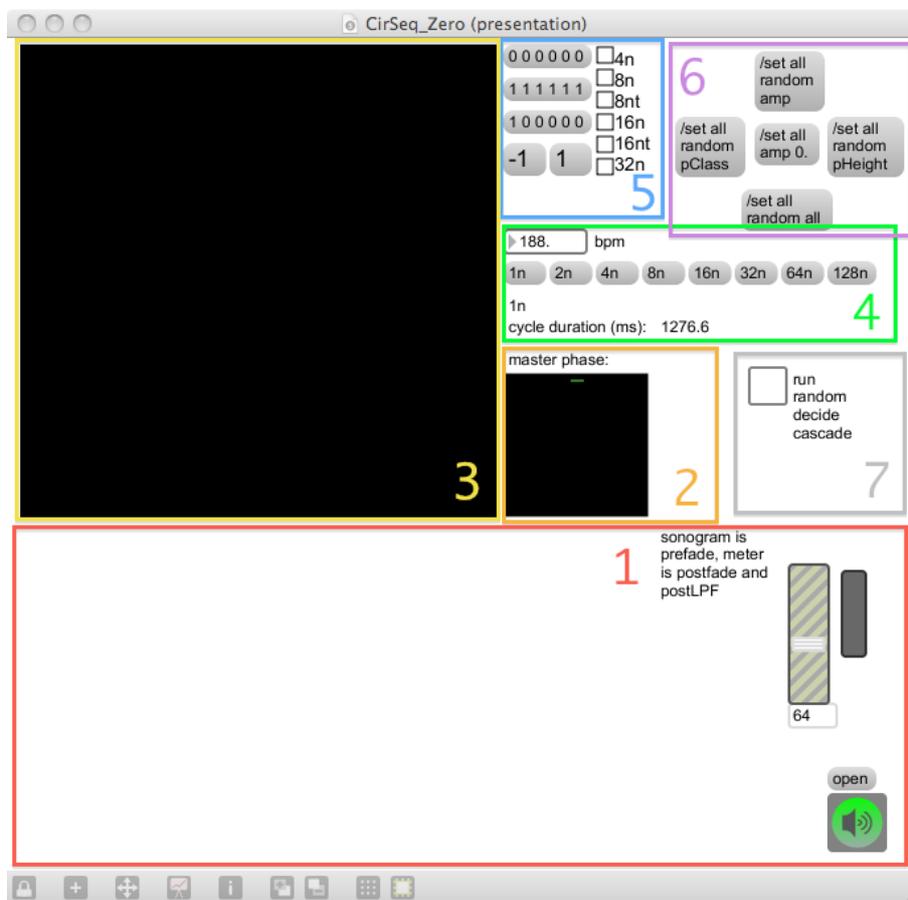


Figure 5.9: CirSeq\_Zero\_0a

### (5.3.1.b) CirSeq Zero patch annotation

1. The output section includes a scrolling sonogram display of the audio output, and a `gain~` object, as well as an `ezdac~` object. The master output level is initialised at a value of 64 on

<sup>101</sup> The only changes made to the patch were: (1) to hide the sub-patch windows; (2) to set presentation mode as the default view of the top-level patch; (3) to resize the window of the same; and (4) to make green the 'on colour' of the `ezdac~` object.

the **gain~** object; depending on the soundmaking settings a more nominal gain value of 110 is recommended.

2. The master phase display is a **scope~** object connected, as shown in Figure 5.10, to the **phasor~** object that drives the patch.<sup>102</sup> As time passes and the output of the **phasor~** loops zero to one over and again, the angle representing now in the time-space representation of the period is shown on the **scope~**; the **+~ 0.75** object offsets the phase value in order to let the period representation start/end at the twelve o'clock position.

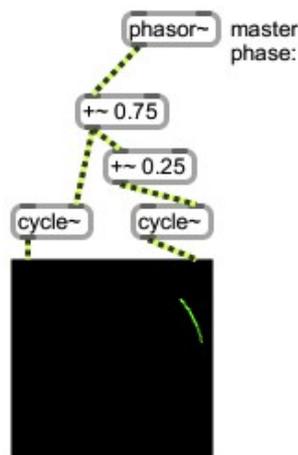


Figure 5.10: masterPhase

3. The lcd-GUI is the visual focus of the piece; it is where the interactive visual representation of the CirSeq concept is manifest. The coloured nodes of the all six quantisation systems are visible in Figure 5.11 below, with a pattern of non-zero amplitude spiroidial settings also shown. The visual manifestation of each spiroidial is allowed to overlap with those nearby because it is only the node at the centre that is clickable. The colouration of the nodes has been made to follow the same logic that was expressed as differently shaped dots during illustration of the CirSeq inception as given above (§5.2).

The spiroidial widgets (described in §4.4.2) can be moused at any time, though it is

<sup>102</sup> Note that the position of the **phasor~** object within the patch has been changed for the purposes of this illustration.

easier to locate them in the time-space when the nodes are visible (see annotation five, below). The currently selected node is framed by a green square, and its values are displayed in the top left corner of the lcd-GUI; click in empty space, within the lcd-GUI, to deselect. The parameter dimension of the spirodial that is active for adjustment during click-and-drag is indicated by the display of an ASCII character beside the parameter name on lcd-GUI; see table:

Indication char	Short name	Parameter name
	amp	Amplitude (linear)
<	pC	pitch-class
>	pH	pitch-height <sup>103</sup>

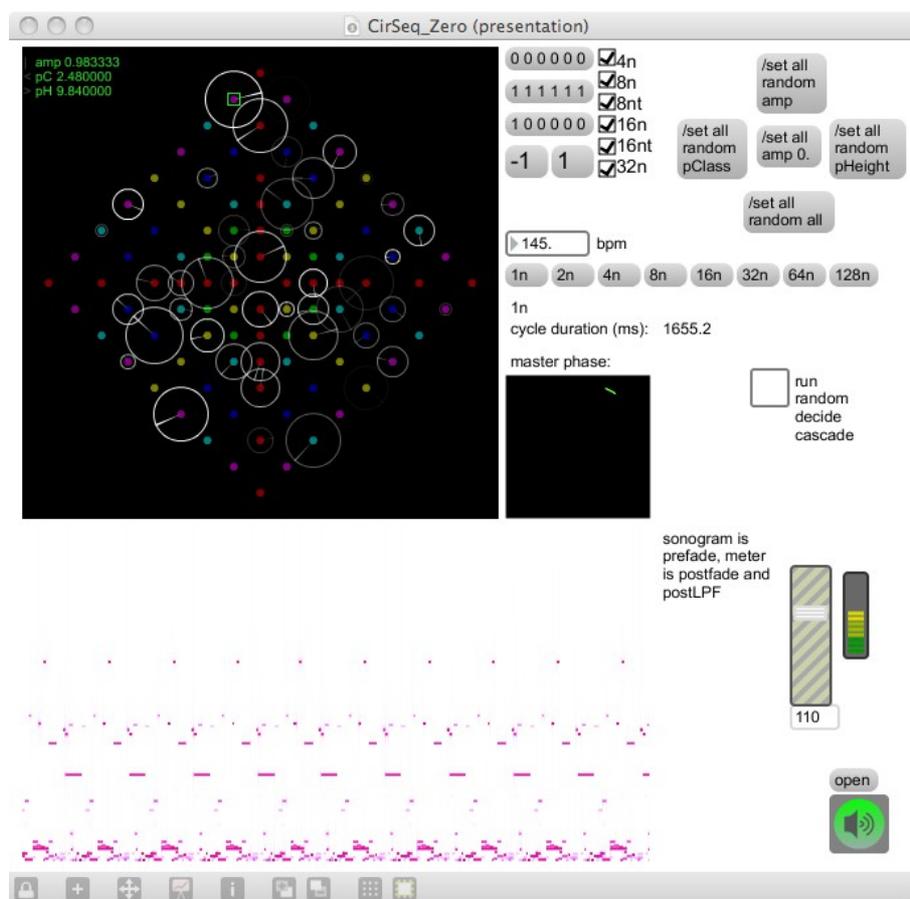


Figure 5.11: CirSeq\_Zero\_1

<sup>103</sup> Note the discrepancy of the term 'pitch-height' being used as opposed 'octave-level' that is more common in my work; and, also, that this value is not integer quantised.

4. The cycle duration settings make use of the 'Global Transport' aspect in the MaxMSP environment; the 'bpm' tempo value can thus be set from any patch active within the environment. Below the 'bpm' number box are messages for eight note-values given in the native format where **1n** is a semibreve, **2n** a minim, **4n** a crotchet, and so on; the currently selected duration value is displayed with the millisecond value of the period.
5. This section of the patch comprises a set of three plus two clickable messages and a group of check-boxes for selecting which of the six quantisation systems will be shown on the lcd-GUI. The word system is used often within this project as a whole, and its meaning is generally clear from the context of its use; its specific use as a term within this piece, however, adds an unfortunate complication to discussions – that which is a 'quantisation system' in *CirSeq Zero* has been described above as the set of quantisation points found along a 45 degree square line path in the CirSeq conception. Within this section the word system is used in this specific tense, to refer to those sets of nodes.

There are six quantisation systems, and these are the nodes created by the hypothetical circles of radii one, two, three, four, six, and eight; these are described, at the check-boxes, in terms of their CMN note duration values in the environment's native notation of 4n, 8n, 8nt, 16n, 16nt, and 32n. The omission of those systems that would be found at radius five and radius seven is a clear example of permitting influence of the environment over technē.

The three messages that are lists of six values can be used to set the check-boxes as all off, all on, or just one on with the others off. The messages that are the values **-1** and **1** can be used to perform a 'rotation' of the list representation of which systems are currently off or on. For a system to be on, in this context, means two things: (1) that it is visible on and lcd-GUI, and (2) that it is receptive to the '/set' messages that are described next.

6. There are five clickable messages in this section of the patch, each of which starts with the OpenSound Control (OSC)<sup>104</sup> namespace '/set' and is followed by the keyword 'all': these messages thus target all nodes that are currently visible on the lcd-GUI to set parameter values. The `/set all amp 0.` is the only message here to give a specific value; to the left, top, and right of that are messages to randomise the pitch-class, amplitude, and pitch-height parameters of the visible nodes; the bottom of the five messages is to randomise all three of those parameters.
7. Lastly is the **toggle** to run the 'random decide cascade' that is discussed further in §5.3.3 below. In brief, this feature of the piece is an automation of triggering the '/set' messages of annotation area six; it also automates the selection of which systems are currently receptive to those messages, such that data values for the parameters may be set at one time and remain unchanged during subsequently triggered messages that change the parameters values of other systems' nodes.

## 5.3.2 Technical structure

### (5.3.2.a) Original documentation

There is, within the CirSeq\_Zero folder in the chapter5 directory of the portfolio, a folder named cirSeq\_021\_doc which contains structured documentation of the work. That documentation was created in the FreeMind software environment. Exported versions of the structured information from the mind-map are also included. These files were created in 2010, and they provide details of what each patch, sub-patch, and abstraction is for, including notes about the arguments and input messages they take, and about the data that is shared between different patches. The png export of the mind-map is also included here, as Figure 5.12 below.<sup>105</sup>

---

<sup>104</sup> cf. (Freed & Schmeder, 2009)

<sup>105</sup> In print copies of this document this figure will only give an impression of the hierarchical manifestation, whereas in digital copies the text will be human readable after zooming.

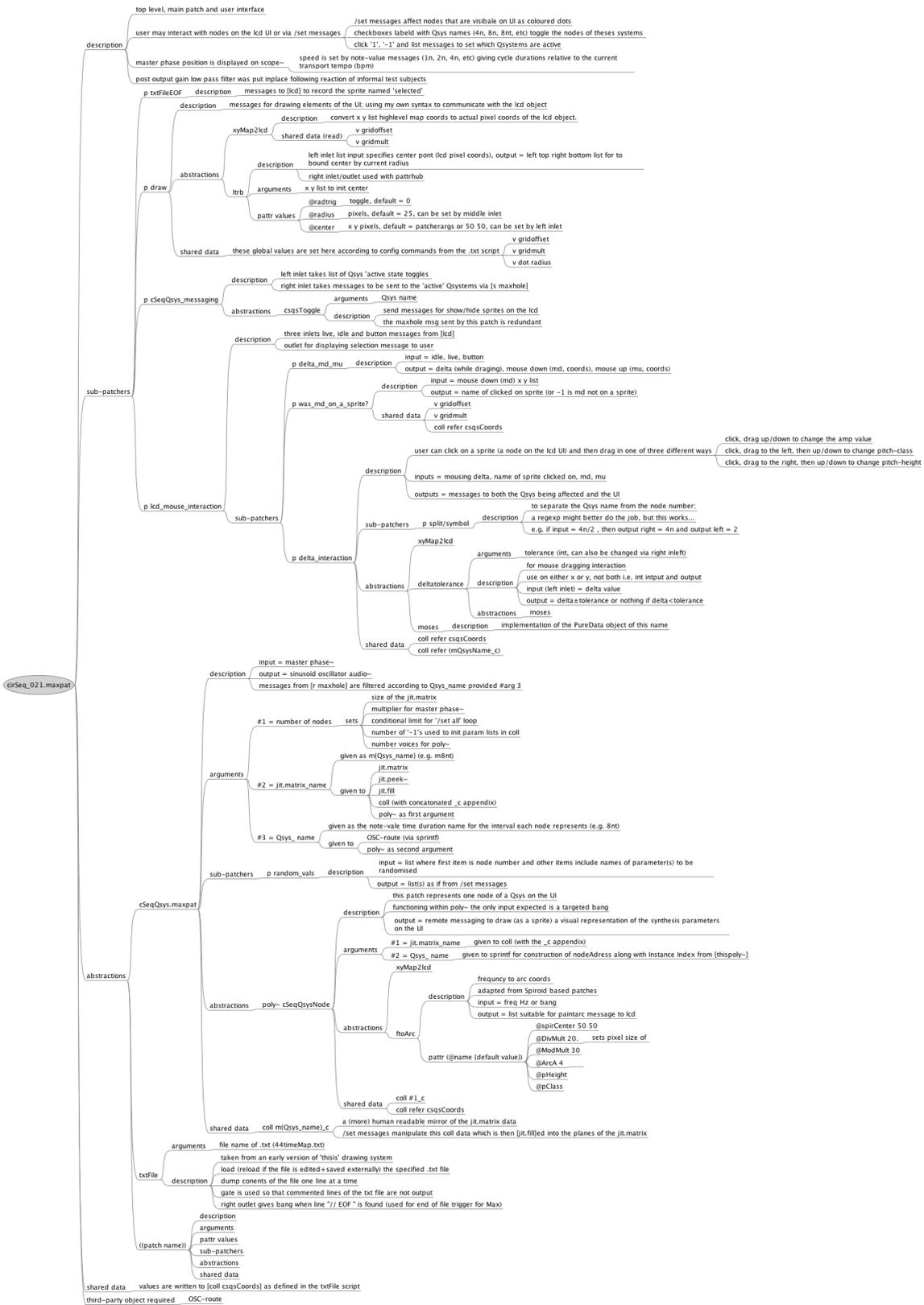
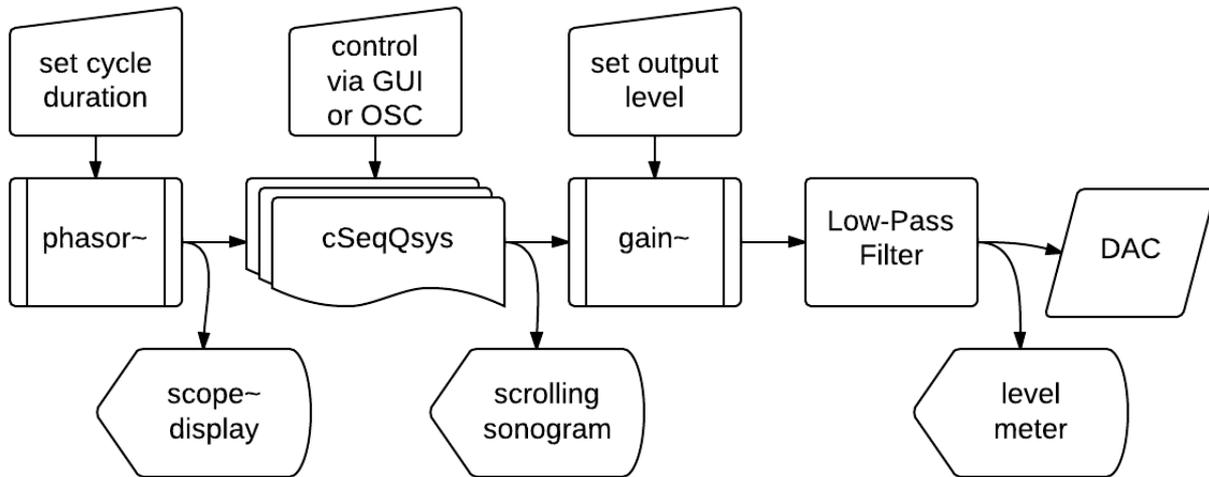


Figure 5.12: cirSeq\_021mm

### (5.3.2.b) CirSeq Zero structure

A flowchart representation for the *CirSeq Zero* patch is shown in Figure 5.13.



The CirSeq quantisation systems are implemented as an abstraction, arguments to which specify the number of nodes in the system, a name to be used for a **jit.matrix**, and a name used as an OSC namespace for the system. The abstraction is patch is called **cSeqQsys**, and a flowchart representation of its structure is shown in Figure 5.14, below.

The Jitter matrix in **cSeqQsys** has four planes of float32 data type, but only those planes indexed as 1, 2, and 3 are in use:<sup>106</sup> one plane each for data of the 'pClass', 'pHeight', and 'amp' parameters. The size of this one dimensional data matrix is determined by the specified number nodes in order to give, in each plane of the matrix, one cell per node. Messages to and from **cSeqQsys** are delivered via **send** objects, and the patch has one **inlet** (for signal from the master **phasor~**) and one **outlet** (for the audio signal of a sinusoidal oscillator).

<sup>106</sup> Perhaps there is reference within the programming here to the Jitter convention of ARGB matrices of which plane 0 (the alpha/opacity channel) is generally used in my work only as a secondary concern.

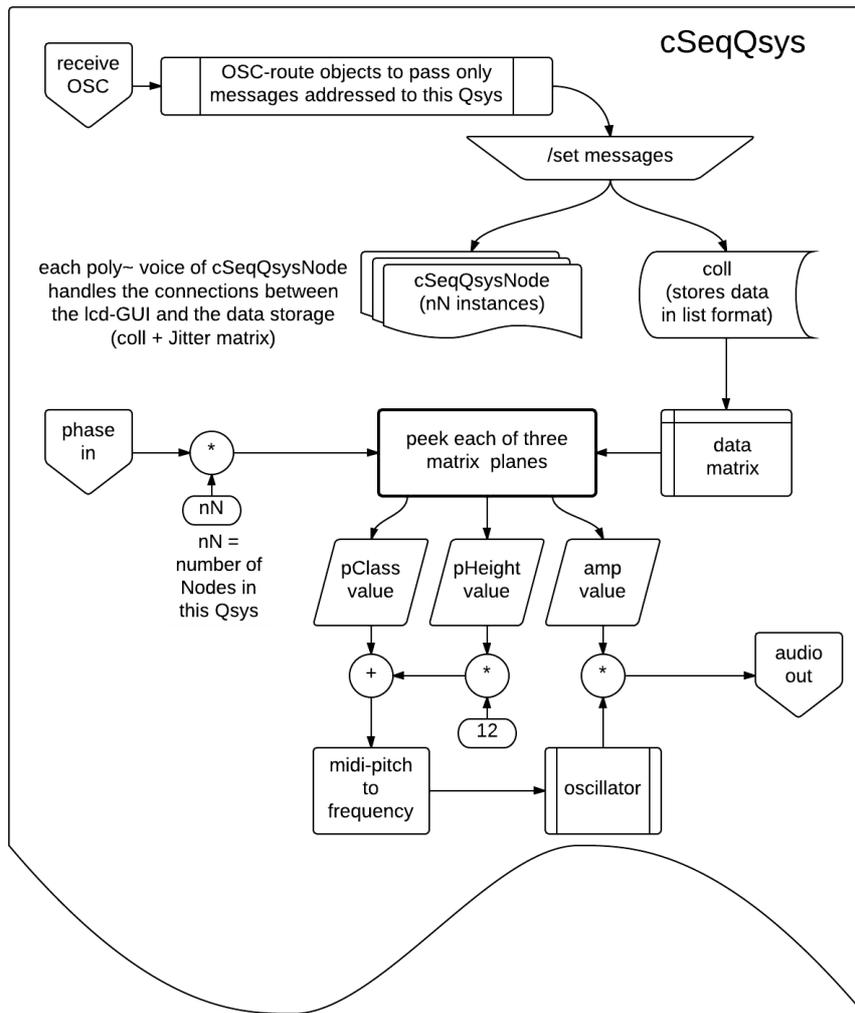


Figure 5.14: cSeqQsys\_flow

### (5.3.2.c) Methods for control

Spiroidials on the lcd-GUI provide one way to interact with the piece; another way is with the clickable messages in the patch; a third possibility is that of creating a meta-patch that would send suitably formatted OSC messages to *CirSeq Zero* via the **send** object namespace 'maxhole' in MaxMSP. Although *CirSeq Zero* is not network enabled, the idea behind using an **r maxhole** object in the cSeqQsys abstraction – through which all messages would pass, with only those that are locally relevant being processed – was to prepare the piece for being controlled by external software structures that may either be local or on remote computers. A meta-patch for *CirSeq Zero*

could, when running in the same MaxMSP environment as the piece, be as simple as two objects – **udpreceive** connected to **s maxhole** – which would facilitate reception of messages via network connections.

OSC messages that target an instance of cSeqQsys begin with /csqs, which is followed by the name of system (/4n, /8n, /8nt, /16n, /16nt, or /32n) and the final OSC namespace is /set.<sup>107</sup> The data part of the message is then a MaxMSP list: the flowchart in Figure 5.15 describes how valid messages can be constructed to either specify or randomise node parameter values. Mouse interaction with the lcd-GUI generates messages that are formatted – in the visual terms of the data section of this flowchart – by taking three 'either' routes (to give the nude number followed by the parameter name and a value). Four of the five clickable '/set' messages (see annotation six, §5.3.1) give access to all four possibilities that can be arrived at in the flowchart after two 'or' routes are taken.

---

<sup>107</sup> A /get feature was started, but then removed during development; only a suggestion of that remains in the maxpat.

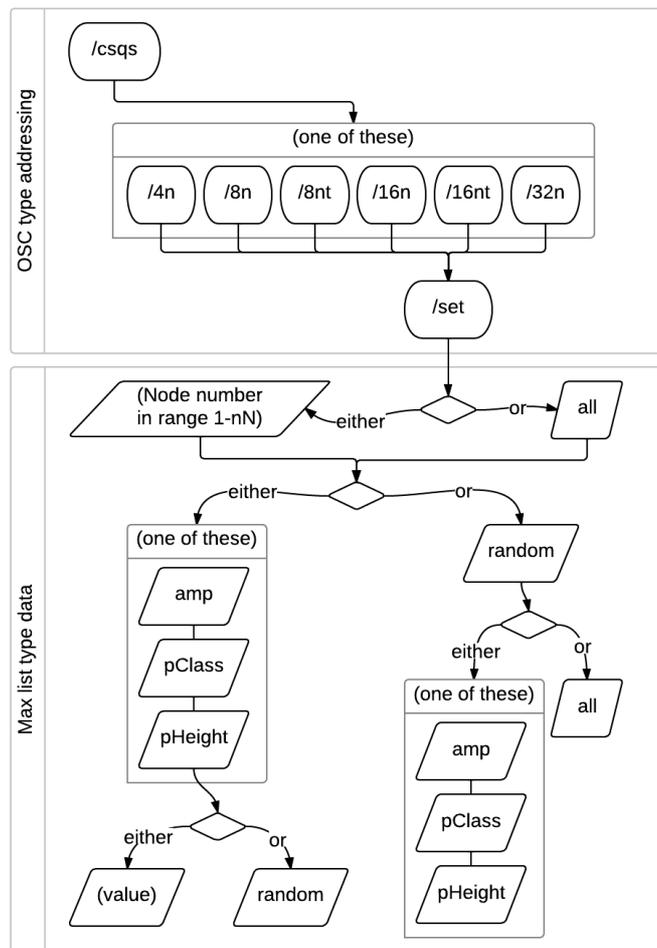


Figure 5.15: csqsMessages

### 5.3.3 Making decisions to organise the soundmaking

#### (5.3.3.a) Similar concepts at different layers of abstraction

The valid options that are implemented within the data part of the *CirSeq Zero* control message format are shown, in Figure 5.15, as a structure of either/or decisions. I have organised this visual representation of those code-level options so that, for each decision, the 'either' route is pointing to something-more-specific while the 'or' path leads to something-less-so. At a different layer of abstraction – at the level of decision making during real-time performance of the piece – there is a similar either/or structure that has been implemented to automate changes in the CirSeq pattern; the structure, within *CirSeq Zero*, that automates message triggering can be referred to, for short, as the

cascade.

The cascade of randomised decisions was added to the *CirSeq Zero* patch almost as an afterthought: the intention had originally been for messages to be received from beyond the patch in order to change the CirSeq pattern in real-time. When the patch was operational, however, and completed enough to be fulfil its remit as a working prototype, I chose to embed an example of the sort of external control I had had in mind.

### (5.3.3.b) An additional aspect to the piece

That the cascade aspect of this piece was a later addition to the work is reflected visually by its coding within the maxpat. For comparison, here, an overview of the layout of the *CirSeq Zero* patch in its non-presentation view (while the patch is locked) can be observed in Figure 5.16. When the patch is unlocked (as shown in Figure 5.17), it can be seen that a sprawl of objects have been placed into the available space that was remaining in the patch.

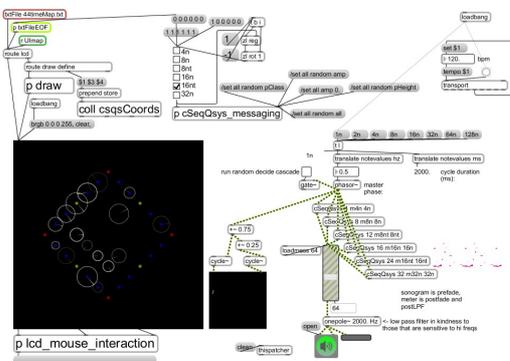


Figure 5.16: CirSeq\_Zero\_nonpres

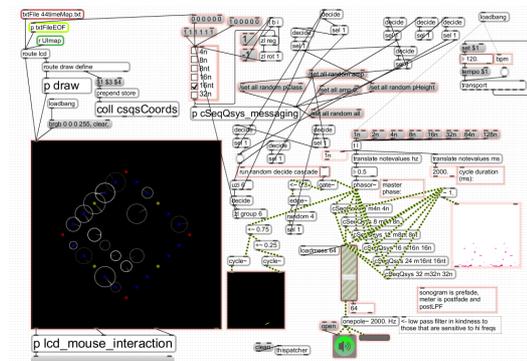


Figure 5.17:  
CirSeq\_Zero\_nonpres\_unlock

As a side note to the discussion, it is of interest that the process of analysing a work in this way brings to mind a concept of 'software archaeology'. The archaeological aspect could certainly be taken further by the inclusion of the many pages of paper based notes and sketches that were produced during the development of the piece. While in this case it is my own work that is being investigated – a narrative being constructed from a combination of memory and such archaeological

perspectives – such principles have wider reaching applications. To rein in the diversion, one may consider how some models of traditional musicological analysis take the CMN score as the basis of exploration, and that in software mediated music the code structures are artefacts through which the substance of the music can be explored (which is to again reinforce ideas already expressed). Ideas surrounding the role of the software itself as a musical artefact, and that the musical ideas being explored in composition are manifest in code before sound, have been reoccurring concerns encountered by this project.

### **(5.3.3.c) The cascade of randomised decisions**

The process that is active while the 'random decide cascade' is set to 'run' (see annotation seven, §5.3.1) begins with detection of when the cyclic traversal of the time-space crosses the phase position represented visually as the twelve o'clock angle. At that moment (at each of those moments) there is a one-in-four chance that the rest of the cascade will be triggered. In other words, every time the master phase ramp restarts – hypothetically when it has reached one and begins again from zero – there is a 25% chance of a change being made to the CirSeq control pattern that is represented on the lcd-GUI and heard manifest in the audio output.

The cascade itself could be compared to a Markov chain, but it was not particularly devised as such. In practice it is an intuited delegation, to the software, of what may otherwise be caused by human actions: the triggering of messages that introduce changes to the CirSeq pattern. The structure of the cascade is the manifestation of compositional choices that distribute changes in time with certain probabilities and interconnectedness. The sprawling appearance in the code should not be mistaken for haphazardness in the design of the algorithm: the connections were made with mindful awareness of their implications, and have undergone refinement while listening to the changes caused in real-time (a form of live coding during the composition of the software).

The cascade is based on a succession on binary choices: at each decision point in the chain,

the output of a **decide** object is used to trigger either an action, or another either/or-decision (there is, however, also a feedback path, and dual action decisions that make the cascade a little more complex). The actions are mostly those of triggering the same '/set all' messages that are clickable in the patch (top-right corner of *CirSeq Zero* maxpat in presentation view); remember that these messages will only be sent to the quantisation systems (cSeqQsys instances) that currently have their nodes visible in the lcd-GUI. For that reason there is also a chance, within the cascade, of enacting a change of which node systems are visible.

#### (5.3.3.d) To unravel the cascade

The object boxes and connections that comprise the cascade have been copy-pasted from [CirSeq\\_Zero.maxpat](#), rearranged in a new patch to make the structure they create more readable, and then saved as [random\\_decide\\_cascade.maxpat](#) (included in the portfolio folder). The additional annotation also put into that patch can be read in Figure 5.18: each decision point in the cascade has been highlighted.

Again there is a sense that some sort of musicological software archaeology is going on here, and the annotations in this study patch (Figure 5.18) are the product more of analysis than of recollection. My own interpretation of the structures exposed by these annotations include, again, observation of similarities across multiple layers of abstraction. For this aspect of the piece, however, I choose to present the work as is, without prescribed exegesis, so that the reader may draw their own conclusions; the technical details are described while the subjective matters of significance or aesthetic value in the configuration are left for individual interpretation.



### (5.3.3.e) Deciding the duration of the cycle period

An important decision to be made in the performance of *CirSeq Zero* is that of specifying the duration of the cycle period. By embracing the connection that the CirSeq concept has to 'common time' the piece, in its adoption of the note-value timing aspects available in the MaxMSP environment, has also inherited that de facto default tempo of 120 beats per minute (bpm). The GUI (message) controls provided in the presentation of the patch give access to a set of whole – as opposed to dotted or triplet – note-values (1n, 2n, 4n, ... 128n) for the duration to be represented by the CirSeq time-space. The duration defaults to the note-value of '1n', which at 120 bpm equates to a 2000 millisecond period. It is recommended that this be changed, either prior to or during performance. Other than selecting from those note-values for the duration, the only way to control the period represented on the time-space is to manipulate the tempo of the global transport in the MaxMSP environment, and the example meta-patch discussed in §5.3.4, below, is based on that manipulation.

To elucidate the next point it would help for the reader to follow the described actions in the *CirSeq Zero* software. Assuming that the patch has just been opened, the data matrices will not yet contain valid data values; to make the nodes of all six quantisation systems visible on lcd-GUI (either check each box, or just click the 1 1 1 1 1 1 message); then, to randomise the spirioial parameters, click the /set all random all message. Make sure DSP is on, and a repeating pattern will be heard; for nominal monitoring level, set the output gain to a value close to 100. Having started the piece we are now ready to begin a brief exploration of tempi.

Set the tempo to 93.75 bpm and notice that, with the duration '1n' selected, the period is of 2560 milliseconds; also observe the particular distribution of specific frequencies at different amplitudes that – having been defined as a CirSeq pattern, and displayed on the time-space lcd-GUI – are graphed over time on the sonogram.

Select '2n' and the duration is halved to 1280 milliseconds, and the pitch pattern is compressed in time. There is a doubling of speed/halving of duration for each consecutive subdivision at the set tempo: '4n', '8n', and so on to the right in the *CirSeq Zero* patch. At '64n' there is a 40 millisecond cycle duration, which is the period that equates to a frequency of 25 Hz, close to the threshold between rhythm and pitch in human perception of sound. At '128n' there is a 20 millisecond period and the temporal pattern of pitches that was (at the cycle duration of '1n') sounding as a rhythmic phrase, is now manifest as timbral texture. The distribution of energy in the frequency-domain that is prescribed by the CirSeq pattern remains while the temporal distribution is lost as an attribute of the sound.

A 20 millisecond cycle duration hypothetically gives a fundamental frequency of 50 Hz to periodic waveform being synthesised. In practice, however, the experience is not of a tone with that as its fundamental frequency; a personal observation that I ascribe to the fact that the spectrum of the sound produced is unlikely to contain the characteristics of a harmonic series based on that hypothetical fundamental frequency. Rather, it is that the specific frequency values defined by the spiroidials in the CirSeq pattern that are dominantly heard as the constituent partials of the output.

Figure 5.19 shows the sonogram display of a randomised CirSeq pattern that was played at 93.75 bpm while the duration values from '1n' to '128n' were selected in turn.

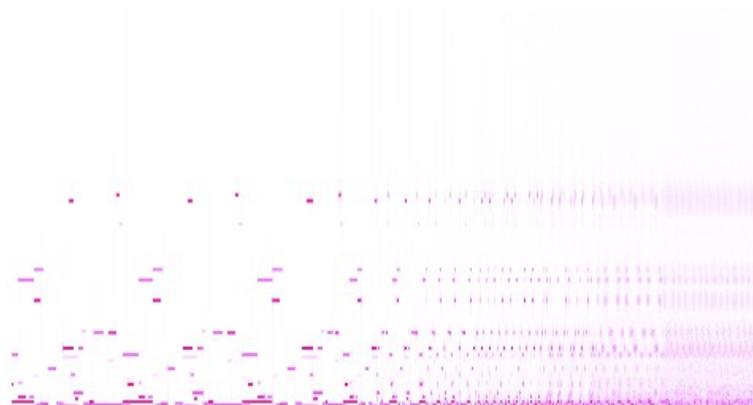


Figure 5.19: sono\_example

Another way to experiment within *CirSeq Zero* is alter to the bpm value while other parameter are kept the same. With '128n' selected, the tempo value can be manipulated to give a wide range of cycle durations: from 1874 milliseconds (at 1 bpm) to less than 1 millisecond when the tempo is set in excess of 1878 bpm. Strange and wonderful things can be heard when soundmaking at such extreme bpm. The screenshot in Figure 5.20 was taken with the tempo at 2000 bpm, and it shows the manifestation of a visual alias pattern in the master phase representation on the **scope~** object. The meta-patch described, below, in §5.3.4 explores this territory further.

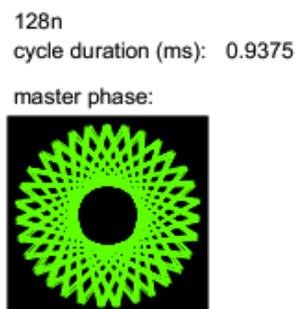


Figure 5.20: 2000bpm

### (5.3.3.f) Contextualising the blurring of lines on the time-domain continuum

CirSeq was devised as a concept to provide a visual representation of a time-domain period where the basis of that representation is comparable to the geometrical basis of the spiroid-frequency-space concept. There was a conscious decision to put out of mind that the concept of frequency is itself a property of the time-domain, and that any period duration can also be expressed as a frequency value. The continuity of those perceptual domains, however, could not be ignored.

A pattern of frequency and amplitude parameter values specified in the CirSeq time-space is experienced as a repetitive sound possessing attributes of rhythm and pitch when the period duration of the time-space is akin to a bar of CMN. Changes to the CirSeq pattern are heard as alterations of those attributes in the sound that is made. To create a working prototype of that modest specification was the primary objective for implementation of the CirSeq concept, but as

was the case with my model of a gramophone (§3.4), the software manifestation of the concept provides much more soundmaking potential than may have been suggested by the original remit. Changes that are made to the CirSeq pattern of frequency and amplitude parameter values – when the period duration specified for the time-space is a short enough – will be perceived as alterations to the frequency-domain aspect of the sound.

Such an exploration of the continuum between the rhythmic and pitched time-scales of music can be contextualised by the work of Karlheinz Stockhausen. Writing in a paper translated and published in *Perspectives of New Music* (Vol. 1, No. 1), Stockhausen (1962) describes the use, in the composition of *Kontakte*, of pulse generator, filter, and audio-tape manipulations to navigate that continuum. Stockhausen describes the 'concept of a *single, unified musical time*' (1962, p. 42, emphasis as in source):

the different perceptual categories, such as color, harmony and melody, meter and rhythm, dynamics, and “form,” must be regarded as corresponding to the different *components* of this unified time

With this concept of unification, Stockhausen sought compositional control of sound that did not treat such attributes as separate elements, but rather as arising from 'a single principle of ordering' (*ibid.*, p.41).

Writing for the same journal (35 volumes later) Michael Clarke (1998) reflects on how the unification described by Stockhausen was, however, mostly conceptual in the composition of *Kontakte*:

He does not here bring “all properties under a single control” (Stockhausen 1962, 40). Although pitch and rhythm are shown to be a single continuum, three different aspects of the sound are manipulated independently using different pieces of equipment (Clarke, 1998, p. 255)

The computer, by the end of the twentieth-century, was able to offer a greater sense of technological unity to the practice of electronic music making. Clarke employed the FOF method of granular synthesis 'to demonstrate that the techniques used by Stockhausen [were/are] still

available' through software, adding that 'the digital realization [of an extract from *Kontakte*] suggests new possibilities for the concept of unity' (Clarke, 1998, p. 230).

While this contextualisation is certainly applicable, it is not cited as a source of inspiration to the work. Just as it was a stepwise, ground up, approach to a quantized circular time-space specification that led, from the interaction of concentric circles with a cartesian-coordinate grid, to a structure that already had a well known existence (common-time), so too did my exploration of sound in *CirSeq Zero* bring to the fore that well trodden conceptual path of 'unity' within composition.

Such contextualisation could well be extended to inform many other aspects of this project, but in that way would direct attention away from the piece being described. Returning, therefore, to commentary which had previously made the transition from the rhythmic realm to that of pitch-like durations, the final illustration of performing *CirSeq Zero* (without extension) is of how the cascade is manifest at those high-speeds.

### **(5.3.3.g) Running the cascade with pitch-like durations of the cycle period**

When the cycle duration has a magnitude of several seconds, the automating cascade has a very gradual affect on the soundmaking of *CirSeq Zero*: there may be many iterations of the same CirSeq pattern before any change to that pattern is imparted. Letting the cycle duration encroach on the frequency-domain allows the changes made by the cascade to occur rapidly. The cascade triggered changes then become the aspect that is heard as rhythmic while the sinusoidal oscillators, that were intended to be the medium of rhythmic structure, become elements of timbre.

Running the random decide cascade at 256 bpm and '64n' (14.648 millisecond cycle period) results in a sound of almost vocal quality. The reader is invited to hear these settings manifest in sound through the software. For illustration, nevertheless, two screenshots of the sonogram in the

patch, taken while soundmaking with those given settings, are provided in Figure 5.21. Periods of silence can be seen in the upper of the two sonograms; these were introduced by the cascade, and such gaps in the sound punctuate the otherwise drone-like textures. The lower sonogram includes visual examples of where/when particular frequency bands have been sustained while the surrounding spectra has changed; this is caused by the cascade changing its selection of quantisation systems without zeroing the amplitude parameters of those nodes that are no longer targeted for change.

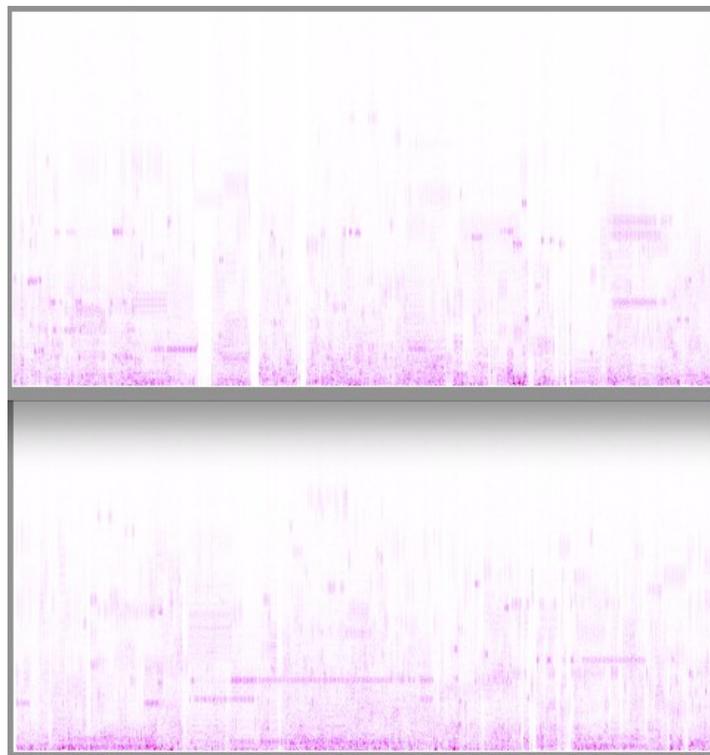


Figure 5.21: sono\_two\_up

### 5.3.4 Meta-patch: an example of external control

To demonstrate how a *CirSeq Zero* meta-patch may be designed the portfolio folder includes [CirSeqZero\\_example\\_of\\_control.maxpat](#); this patch focuses on changing the tempo to control the period of the time-space. It is designed to be used while *CirSeq Zero* is set to note-value '128n' in its cycle duration settings. This meta-patch requires human control, rather than being an automating

structure as was the cascade that has been discussed.

### (5.3.4.a) Annotating the meta-patch

Annotation areas have, once again, been added to an image of the patch, see Figure 5.22:

1. the `/set all amp 0.` message is meant as a human readable indication that the remote messages below, rendered in a tiny font, are formatted to send a `/set` message of that type to each of the six systems;
2. three messages to specify audible parameters for just one node in the time-space;
3. the dial can be used to set the pitch-class of that one node;
4. most of the clickable messages are connected to a `$1` message that then triggers input to the line object, and the output of that sets the tempo of the transport object;
5. non-negative integers set the ramp time in milliseconds for the line object;
6. as discussed below, it is important for this patch that MaxMSP is running DSP at a sampling rate of 44100.

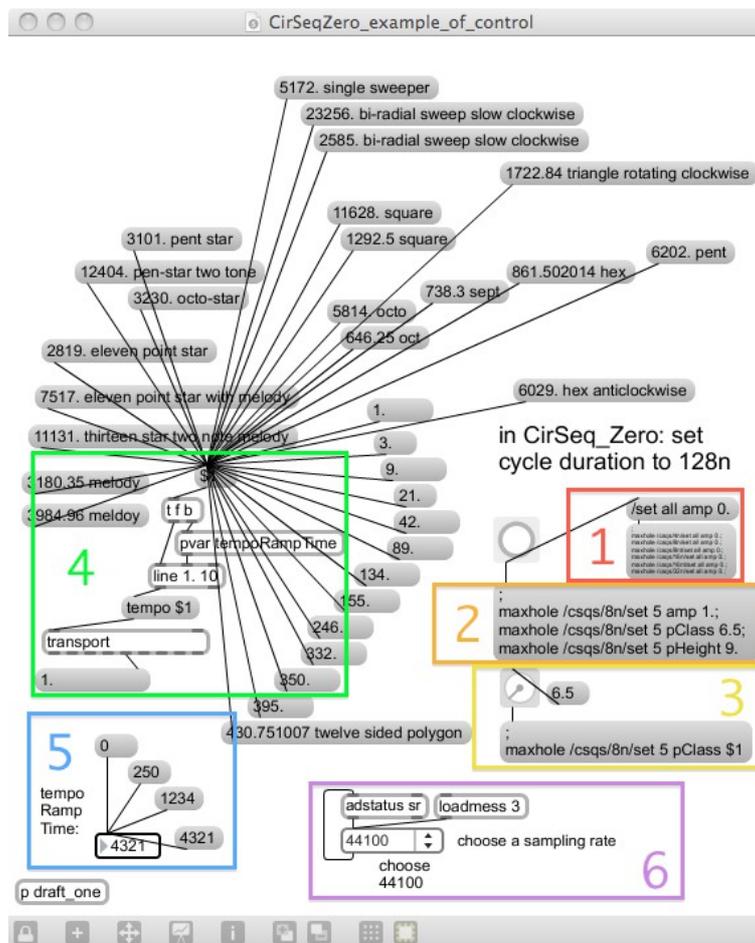


Figure 5.22: cs\_eoc

### (5.3.4.b) Exploring the effects of aliasing

It was noted in §(5.3.3.e) that, in addition to yielding strange and wonderful sounds from the CirSeq pattern, extreme tempo values would – in some cases – manifest on the master phase **scope~** display as aliased patterns; the example of 2000 bpm was illustrated in Figure 5.20. It is important to recognise that this visual aliasing is as much to do with the sampling rate of the system and the buffer settings of the **scope~** object as it is to do with the input parameter value. Whereas the **scope~** buffer settings are relatively inaccessible to the audience of the work, the sampling rate of the MaxMSP environment is more easily changed; indeed a popup menu for doing so has been provided in this meta-patch.

The tempo values that have been arranged in the clickable messages of this patch were

selected after playing with the tempo by mousing the number box in the *CirSeq Zero* patch: it was found that from a single soundmaking node in the time-space it was possible to attain a wide variety of spectra, but also it then became apparent that the **scope~** display would sometimes converge to geometrical figures. The meta-patch was developed by taking copies of the message box that is connected to the tempo outlet of **transport** object, and appending a text description (for human readers, not for use within the code of the software) to each bpm value. By repetition of that process, for successively identified alias patterns of interest, the collection grew to what is thought of as a control layer for the piece, with a particular focus, which could be the basis of further work beyond performance of *CirSeq Zero*. The focus of the exploration in this meta-patch, on selecting soundmaking parameter settings based on predisposition to favour simple geometrical forms, led to some interesting discoveries in the sound.

### (5.3.4.c) Explorations of soundmaking guided by shape

By starting with the one node beep at 1 bpm (remember this is a cycle duration at '128n'), and then slowly increasing the tempo, the specific frequency heard at discrete times becomes a continuous tone with an increasingly wide bandwidth in the frequency-domain. Figure 5.23 is a screenshot showing the *CirSeq Zero* presentation after performance of the described accelerando; to hear the effect, the reader may recreate the same thing by setting the ramp time of the meta-patch patch to 4321 milliseconds, and then clicking, quite quickly, the vertically arranged sequence of values from 1., to 3., 9., 21., and so on, to the message that reads 430.751007 twelve sided polygon. The dodecagon being described in that message can be seen in the master phase display in Figure 5.23; in real-time the polygon alias at this setting is found to be rotating, and 'fine-tuning' the tempo value can change the speed and direction of that rotation.

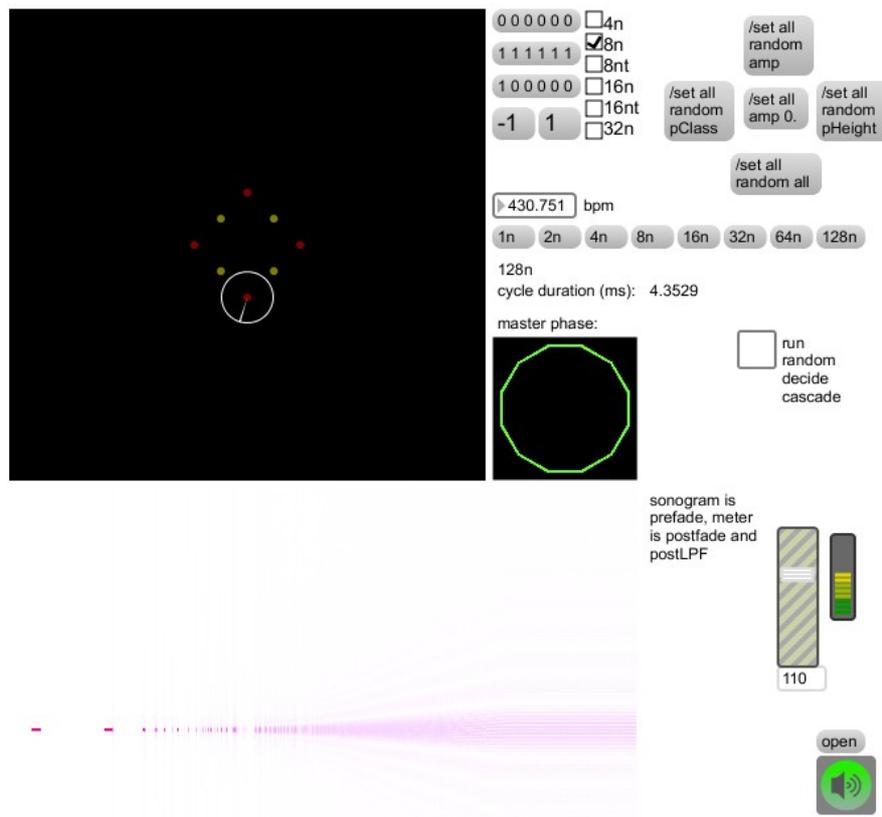


Figure 5.23: dodecagon

At 1722.84 bpm the form of an equilateral triangle was observed in the phase display, rotating clockwise. My understanding of this manifestation is that that triangle is made of the only three points at which the cycle phase is being sampled. I suspected that halving the tempo value would, by doubling the cycle period, double the number of points being sampled. A hexagon did, indeed, form by halving the tempo, and the dodecagon alias pattern that is illustrated above was found by halving that value again.

After more playing with tempo sweeps – enjoying the sonic aspects of aimless jumps in speed, quite expecting to break something, but not achieving any reproducible glitches (one probably could if one were to set that as an objective) – a two-ended alias formed on the phase display. From this, the rotating shapes of a square and an octagon were easily found.

A different bpm value was also found that formed a square alias pattern on phase display,

which led to a second octagon and another two-sided shape, again based on 'octaves' of the tempo value. As a general rule, for this CirSeq pattern with just one node having amplitude, it seems that doubling the number points in the master phase alias pattern also doubles the number of partials present in the sound.

Mathematical prediction of the aliasing in the audio and the phase display may well be possible, if one were so inclined to calculate such things. This project, however, is content to present this as an experiential phenomenon made possible by the *CirSeq Zero* composition, and the reader is encouraged to explore the possibilities offered by the work.

Screenshots of the phase display plus tempo value above it, and a portion of the sonogram below, were taken for most the tempi listed in the meta-patch; these screenshots were then compiled into a single image file, as shown in Figure 5.24. At either end of the very wide image is a lesser cropped screenshot of the patch showing time varying spectra obtained from that one sounding node in the CirSeq time-space.<sup>108</sup>

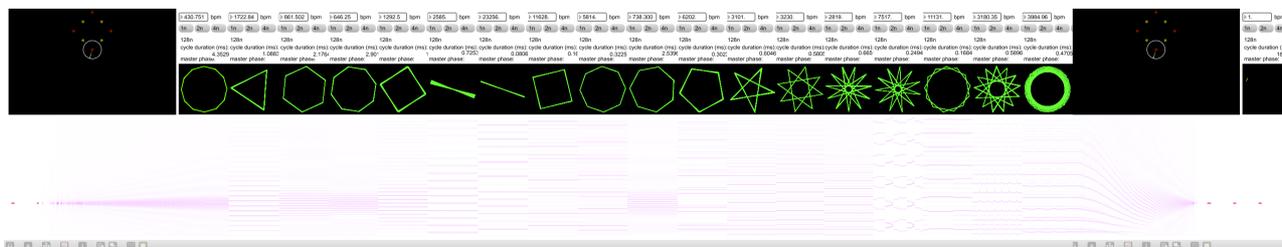


Figure 5.24: phase\_shapes

## 5.4 Aesthetic analogies in the exploration of CirSeq patterns

The idea of a domain dichotomy – contrasting time- and frequency-domains when we think about aspects of sound – has been discussed in connection to *saw~onpole~noise~click~*. That

<sup>108</sup> Again this is an image that, in digital copies of the document, can be zoomed to view more detail. It is also included, as [CirSeq\\_Zero\\_control\\_example\\_phase\\_shapes.png](#), in the CirSeq\_Zero portfolio folder.

discussion, at §3.6.1, illustrates how each of the concepts listed in the title of that composition can be thought of in terms of each domain. Exploration of *CirSeq Zero* has delivered clear demonstration that those two domains are but different parts of a single thing, which, in the historical contextualisation at §(5.3.3.f), is described as a concept of unity.

One way to comprehend this unity is to think of the property of duration as being common to both domain perspectives. The spiroid-frequency-space is an otherwise atemporal model for visual representation of duration; the CirSeq time-space is a visual representation of (harmonically quantised) sub-divisions of a given duration. Through the creation of, and extended experimentation with, *CirSeq Zero* I have gained experiential appreciation of one method for navigating the continuum between the two domains.

### 5.4.1 Exploration of the continuum – further detail

The example of a CirSeq pattern in which a single node is given amplitude has been described in §(5.3.4.c). In that pattern, a fixed-frequency sinusoid is heard for an eighth of the time-space duration; this is heard as the repetition of a simple 'beep'. As the momentum of angular traversal in the time-space is increased into the audible frequency range, the positional identity of the node as a sub-division of time becomes lost. The time-domain-like repetition of a beep soon becomes a continuous tone, and as rate of traversal in the CirSeq pattern increases further, the positional identity of the node's audible contents is gradually degraded in the frequency-domain as well. When extreme tempi are employed, the audio output is as much to do with DSP aliasing as it is with the CirSeq pattern being played, and the things that are found in that sound are far more rich than its humble specification may have suggested.

While experimenting with extreme tempi, to produce the meta-patch example described above, I chose to focus on finding geometrical shapes in the visual aliasing of the master phase

display. Because of this choice, a most intriguing discovery was made that again calls to mind the concept of layers of abstraction, as well as notions of circularity:

The four aliasing examples that are shown at the right of Figure 5.24 – starting from the second eleven-pointed star – are reproduced in Figure 5.25 to show better the sonogram details. Manifest in the audio output at these tempi are short looping melodic patterns, akin to those that CirSeq can produce at more regular tempi, but remember that these examples are all of the same one-eighth-node CirSeq pattern.

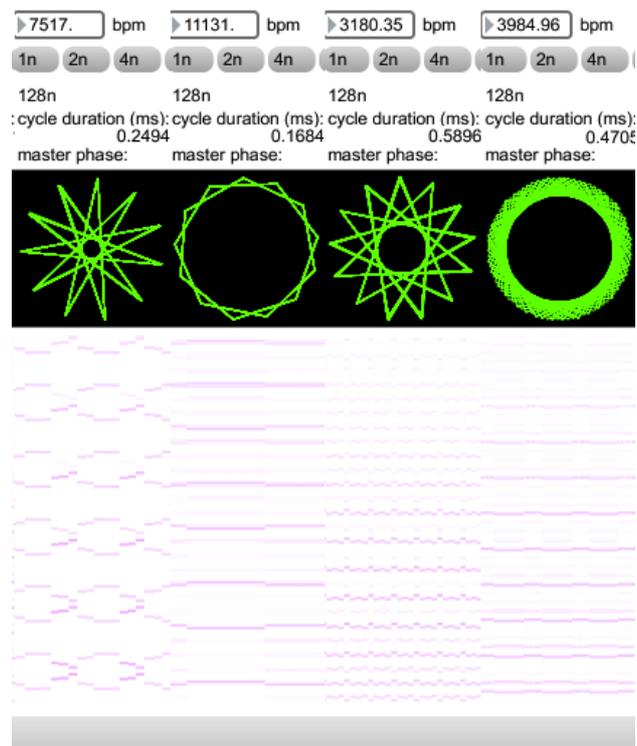


Figure 5.25: 4of\_the\_phase\_shapes

At 7517 bpm (as in the left-most example in Figure 5.25), the cycle duration is equivalent to slightly less than eleven samples, and the interaction of that rate and the specified (one node) CirSeq pattern produces a repetitive pattern of multiple pitches. It is almost as if the continuum, that exists between the pitch-rhythmic time-domain level of perception and the spectral-timbre frequency-domain level of perception, has been followed so far along that – having gone through

and beyond the frequency-domain range – its manifestation has reemerged as pitch-rhythmic.

At the risk of again invoking the chroma paradox – and to cite a contemporary cultural meme that I had not expected to feature in my research: the *Double Rainbow* viral video (Yosemitebear62, 2010; Anon, 2013) – the analogy of a double rainbow is suggested for this reemergence of properties thought to belong lower in the continuum, and with it the question: what does it mean?

Despite the irony of doing so, that analogy of refracted light will be extended just a little further in connection to the experimental exploration continua.

## 5.4.2 Observation of a specific configuration

For sunlight to refract through crystal glass in such a way as to project a rainbow on a given surface, there are many physical conditions (such as relating to various angles of orientation, and intensity of the sunlight) that must be met. Similarly, the specific examples of geometrical shapes, and the unexpected instances of time-varying spectra exposed in the given examples with *CirSeq Zero* and its meta-patch, are manifest only through particular conditions of the software configuration (DSP sampling rate, affecting both audio and visual outputs, and buffer setting in **scope~** being particular for the visualisations that were a focus of the exploration).<sup>109</sup>

If the tempo values exemplified in this meta-patch were to be used while the DSP sampling were set to a rate other than 44100, then interesting things would still be heard, but – because of the different aliasing – they would not be the same things for which the meta-patch was designed. One may expect there to be other tempo values that would be found to give rise to similar aliasing 'artefacts' at different sampling rates.

While keeping the recommended sampling rate for the tempi available through the

---

<sup>109</sup> In different experiments I have found that the DSP vector size settings affect the audio output, but this seem not to be the case for this example.

[CirSeqZero\\_example\\_of\\_control.maxpat](#) meta-patch, it is also stimulating to specify a CirSeq pattern that is rather more complex than the single node discussed so far. With arbitrary CirSeq patterns active in *CirSeq Zero*, there are numerous more time-varying patterns to be heard, as well as a multitude of richly complex spectra that could prove fertile ground for further research.

## 5.5 Critique and chapter conclusion

This chapter has introduced and contextualised the CirSeq concept as an archetype for the visual representation of a cyclical duration with harmonically quantised sub-divisions of the whole. It has also been shown that development of a software prototype to demonstrate the CirSeq concept in use as a step-sequencer, with elementary sinusoids as the sound source, led to a configuration (*CirSeq Zero*) that can itself be viewed as a software composition. Extended methods for performance of *CirSeq Zero* are suggested in the form of meta-patches. In the construction of, and experimentation with, one such meta-patch, a seemingly new domain of the unified-time continuum has been identified; a domain that is technically based on aliasing, and could be contextualised in relation to a glitch aesthetic.

To conclude this chapter the following discussion concerns where and why the CirSeq avenue of investigation fell short of my aesthetic demands, and how reaction to that helped to clarify specification for what came next in my research. That discussion begins with a description of known bugs in the *CirSeq Zero* software.

### 5.5.1 What you see is not always what you get

Spiroidials on the lcd-GUI can become 'out of sync' with the data values that they are a representation of. In particular, it may appear on the lcd-GUI that a node has an amplitude value greater than zero, when that value is actually at zero. This reproducible bug in the software can be

remedied within the existing configuration by randomising and the zero'ing the amplitude values of one quantisation system at a time. The cause of this visual asynchrony is generally attributed to the convoluted method of control that has been implemented to connect the lcd-GUI to the actual data that is used in the DSP of patch. While I consider the implementation to be of some technical merit – in most cases it does perform the desired task of allowing both OSC and mouse based interaction with the CirSeq pattern – the quality of the interaction can be undermined by the bug highlighted above. Indeed, the programming is rather 'Heath Robinson' in its ways; the way, for example, that the graphical front-end is linked to the oscillators that are being controlled. That front-end is drawn with an adapted version of the thisis drawing system, and is based on the **lcd** object; perhaps pushing the limit of what can reliably be achieved with that object as the basis for interaction. The oscillators, themselves, are intentionally simple in the implemented control; there is, for example, no enveloping or smoothing of control values, and this tends to result in audible discontinuities at the temporal boundaries of a node.

A second way in which what you see in the lcd-GUI of *CirSeq Zero* is not quite what it is that you get in the soundmaking that follows it is to do with the angle and radius mapping of the spiroidial widgets. The issue here is attributed to the origins of the work as a prototype, meant for further work after the proof of concept; it was at first imagined that the (intentionally basic) sinusoidal soundmaking part of *CirSeq Zero* would be a temporary 'place holder' for a more sophisticated synthesis subroutine. Indeed, there was a synthesis engine being developed in parallel to the CirSeq investigation, and a significant feature of that was to do with its interpretation of the pitch-class and octave-level (pitch-height) values: rather than quantising the octave-level to integer steps, the idea was to allow any value for the pitch-height and to synthesise sound in a way that would maintain a pitch-class identity while being able to move (on a radius in the spiroid-frequency-space) between octaves.<sup>110</sup>

<sup>110</sup> The synthesis work being referred to went under the name of 'harmonic grain tables' footnote continues on p.214

In the sinusoidal soundmaking that was implemented for *CirSeq Zero* – which was then embraced, rather than being replaced – does not quantise the pitch-height value when combining it with the pitch-class value for setting the frequencies of the oscillators (see Figure 5.26, below). The result is that the visual angle of the pitch-class indication in the spirooidal widget is not necessarily a true indication of the pitch-class that is heard for a given node.

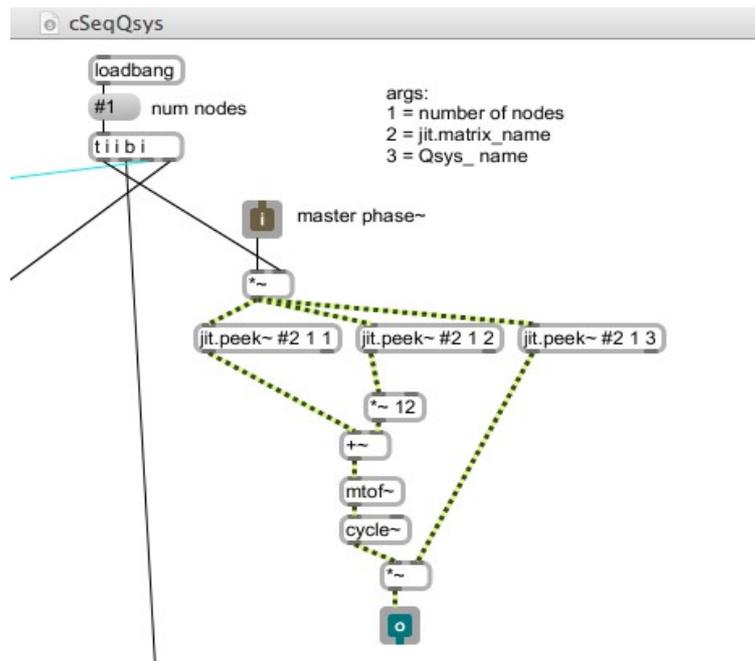


Figure 5.26: cSeqQsys\_synth

## 5.5.2 Remedy in moving on

Fixing these, or any other bugs, in the *CirSeq Zero* patch, or even reprogramming the entire thing in a more efficient or elegant way, would not alleviate the most significant issue which is that there are too many layers of abstraction at play within the design. The problem is that the layers of abstraction in the programming serve as obscurations for the actual processes that is being interacted with by the human player. The visual representations of the lcd-GUI are of a highly conceptualised nature that do not directly relate to the data that they control. Whereas the idea of

---

continued from p.213 and is one of many omissions from the portfolio; it is, however, discussed online at <http://sdfphd.net/a/doku.php?id=harmonicgraintables>

unity across layers has been discussed in different contexts within the work of this piece, there is little unity to be found between the on screen and the soundmaking aspects: the circularity shown on screen is only an illusion because the data that is being made into sound by the DSP algorithms is actually formatted in a set of one-dimensional matrices.

Bringing back to mind the conceptual model of a gramophone (§3.4), with its with two-dimensional matrices of data, I decided to move on from working with the CirSeq concept to find ways of working with data in a two-dimensions that could then be displayed – as directly as possible – on the computer screen.

Discussion of the gramophone model patches (at §3.4.5) has also highlighted the problem of interfaces that constantly split the attention of the human: often in computer music software, the eye is required to look away from one representation of an aspect of the sound being made in order to look at another representation of a conceptually separated aspect of the same sound.

Moving on from the work described in this chapter, it was established as an aesthetic priority that all interactive visual representations should be made to cohabit an 'integrated display'.

### 5.5.3 Fractal potential

Some final thoughts on the possibilities for further work, on the described concepts, that were not pursued within this project:

During the development of *CirSeq Zero*, one of the ways in which I imagined the work progressing was toward a fractal manifestation of the CirSeq concept. It has been shown that the CirSeq time-space representation is that of a given duration being sub-divided into sets ('systems') of smaller durations. Why not, then, allow a single node to 'contain' another CirSeq time-space to describe the happening within that period? The fractal potential of this idea is stimulating to the mind, and would certainly present a number of interesting challenges in programming, as well as

possibilities for composition.

## 5.5.4 Circle the square

Some preliminary steps were, however, made towards a successor to *CirSeq Zero*: it would have been called CirSeq Alpha or *cirSeq\_alpha*. The image in Figure 5.27 is called *cirSeq\_alpha\_preview* and it was made with a script written for and parsed by [thisis\\_112.maxpat](#), bringing the discussion of this chapter full circle;<sup>111</sup> it is also seen in this image that nodes have been placed on the circumference of concentric circles, rather than on the squares formed within those circles.

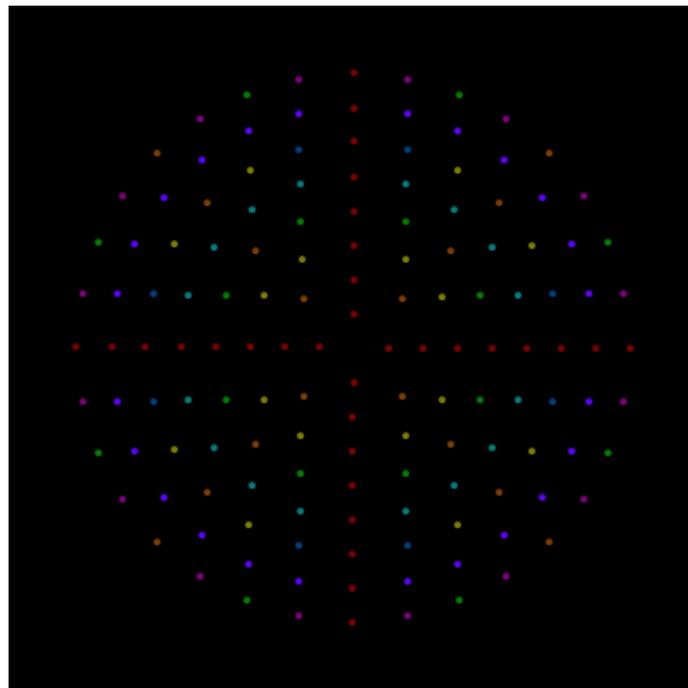


Figure 5.27: *cirSeq\_alpha\_preview*

The nodes of all the first eight 'grid-unit radius concentric circles' are shown in Figure 5.27; the number of nodes on each circle is equal to the radius value multiplied by four, so there are 144 nodes in total. A video<sup>112</sup> was made of the *cirSeq\_alpha\_preview* image being rendered by

---

<sup>111</sup> The origins of thisis were described in §5.1 and the portfolio includes example patches and related files.

<sup>112</sup> *cirSeq\_alpha\_preview.mov* (recorded 20110210) is included in the *\_alpha\_preview\_and\_thisis* folder of the *chapter5* portfolio directories, and is online at <https://vimeo.com/19794652>

modifying a thisis patch to introduce a 90 millisecond delay after the parsing of each line in the script; the video thus shows in 36 seconds what would otherwise be instantaneous.

Another short video (*spiral aliasing*)<sup>113</sup> shows a short thisis script that was written to draw an Archimedean spiral being edited, several times, to change the 'time' attribute of the implemented algorithm. In the video we see the spiral being drawn eight times – in millisecond durations of 15, 30, 60, 120, 240, 480, 960, and 19200 – without the lines being cleared each time the script is parsed. This particular sequence of duration values – octave interval steps, until the last step which is tenfold on the logical next – was chosen because of the particular aliasing patterns exhibited. It is up to the reader to decide, for themselves, whether or not this preplanned and rehearsed live coding is performance of a musical composition.

The return to the topic of thisis provides a perfect link from this work to that of the next chapter: sdfsys. Thisis is, indeed, a common element of most my musicking since 2008; having created a set of subroutines for drawing on screen via verbally styled syntax to describe the stepwise processes of putting points and drawing lines in absolute and relative terms, it (thisis) has become the software equivalent to the pens and paper that tend to travel in my pocket.

The title of this subsection conceals a pun, because to 'square the circle' is to 'do something that is considered to be impossible' (Stevenson & Lindberg, 2011b), and – though it may seem like I am trying to do so – it would be impossible to include every aspect of the practice-led research that culminates in sdfsys.

---

113 [thisis\\_a02\\_spiral\\_aliasing.mov](#) (recorded 20110220) is also included, and is online at <https://vimeo.com/20150095>

# 6: Sdfsys

Bringing together many aspects of the above described research and practice is a modular soundmaking system of which there have been two versions: the first was called *sdf.sys\_alpha* and is often referred to as 'the alpha build' of sdfsys; the second version is *sdfsys\_beta*, and it is to this version that the term sdfsys most generally applies.<sup>114</sup>

In the parlance of software development in general, a beta version is typically released for testing to generate feedback and bug reports that can be incorporated into the development process. That practice is common within computer-science based research projects, and although the idea of that model was entertained at the earliest stages of this project, the decision was made to release *sdfsys\_beta* with the publication of the thesis, and not before.

At some stage during the beta period of development, the dot was removed from the name of the work; a decision paralleled in the evolution of the previously named 'HELO.pg' ensemble (Hewitt, Tremblay, Freeman, and Booth, 2010) to its more well known name of HELOpg, a change on which we commented in our paper for the Symposium on Laptop Ensembles & Orchestras (Hewitt, Freeman, and Brooks, 2012).

## 6.1 Introducing sdfsys

Before describing, in turn, each of the two sdfsys builds, the following introduction provides orientation of the general principles that are common to both the *\_alpha* and *\_beta* versions.

### 6.1.1 Overview

In sdfsys, audio data is stored in two-dimensional data-matrices that can be visually rendered to

---

<sup>114</sup> Technically, the project was re-denominated *sdfsys\_beta1*, in February 2012, but the *\_beta* version is no longer identified as being separate to the *\_beta1*. The appended 'one' is, therefore, omitted from the name in all general discussion of the work; that extra digit may, however, be noticed within the programming at the code level.

screen in a very direct way: each data-matrix has one-plane of float32 data-type, and a number of view modes are implemented for the mapping of that data to the four-plane ARGB colour-space of the computer screen. In the main area of the jit.gl based user interface (gl-ui) the data-matrices are visually layered, each one atop of another. The modularity of sdfsys is based on a unit-generator paradigm in which data is written to, and read from, those matrices by specifically structured patches that are loaded into the sdfsys environment. The emphasis of the system is upon controlling those data-processes in terms of their visual manifestation within the display, and soundmaking occurs through the interplay of multiple processes acting upon different matrices.

Text is used both by the human (to instruct the system) and by the system (to inform the human), and a basic text editor for qwerty-based interaction has been implemented for sdfsys. This part of the system is also based on a data-matrix, but in this case it is the char data-type that is used. The ASCII-numbered glyphs represented by the cell values of the 'text-buffer' matrix are displayed as a layer of text over the audio-as-visual-data layers in the main area of gl-ui. The syntax implemented for text input to sdfsys includes several types of command: there are commands, inherited from thisis (introduced at §5.1), for specifying and drawing with points on a plane (drawing to a data-matrix); there are commands by which to load and control unit modules within the system; and there are commands by which to directly address aspects of the sdfsys environment and its host (MaxMSP).

Parameters of the loaded modules are represented visually as mouse-movable sprites. These sprites are layered, in the main area of the gl-ui, over the data-matrix and text-buffer displays. Various mappings of position-on-the-plane to parameter-value coexist on the same area of the screen. Each type of parameter sprite has a different mapping, such as Hz (spiroid), point (cartesian), radial (linear distance from a reference sprite), and others.

## 6.1.2 Construction aesthetics

Certain aspects of the system have been implemented with intentionally low resolution, as motivated by various aesthetic and pragmatic objectives. The data-matrix dimensions, for example, are set at 400 by 400 cells; the font used for the text editor is of a large size which not only allows the text to be read from some distance, but which also promotes focus on just a few command lines at a time. Awareness of, and concentration on now are encouraged when musicking with sdfsys.

The architecture of the system has evolved gradually over time. Although some factors were thoroughly planned before being programmed – and others (such as thisis) were imported, almost directly, from my previous works – some parts of the system emerged only through modifications being made, ad hoc, along the way.

The sdfsys environment emphasises certain aspects of the software medium in the creative process, and subdues (or even subverts) other aspects of contemporary praxis. Although the sdfsys environment has been implemented within the MaxMSP environment, when one is working in sdfsys one need not see any maxpat interface beyond the (small) top level patch, which includes a clickable shortcut to fullscreen viewing of the gl-ui.

## 6.1.3 Eight spaces

The modular aspect of sdfsys is implemented through conceptual 'spaces', of which there are eight, and into which can be loaded the available unit modules. A unit module is an appropriately structured abstraction maxpat. For each of the eight spaces there is a Jitter matrix that exists within the sdfsys environment independently of the structured modules that may act upon them. All module abstractions are expected to write data to the data-matrix belonging to the space they are loaded into, and modules may optionally read data from the matrix of another space. Interconnections between the eight spaces thus form 'the spaces network'.

## 6.1.4 Making sound

The primary soundmaking method of the sdfsys environment is based upon the control of shape and pattern in the visual domain while listening to the sonic results of the thus controlled processes.

Sound is made in sdfsys through exploration of the audiovisual synthesis methods made possible by the spaces network. The method is based mostly on the principles identified through the creation of my conceptual model of a gramophone (§3.4); the creative potential of this soundmaking technique is expanded by modularising the read- and write-head aspects of that model.

## 6.2 Alpha

The alpha build of sdfsys is described in this section, both to convey something of the techno-aesthetics of its design, and to provide a basis for understanding the more sophisticated beta version that follows (§6.3).

### 6.2.1 sdf.sys\_alpha

The top level patch of *sdf.sys\_alpha*, in the *sdf.sys\_alpha* sub-folder of the *\_alpha* portfolio directories, is *sdf.sys\_a51.maxpat*. The inner workings of the system are documented by the files located in the *sdf.sys\_alpha\_doc* sub-folder (cf. Figure 6.1).



Figure 6.1: sdf.sys\_alpha\_mm

## 6.2.2 Motivating factors for sdf.sys\_alpha

Of the aims motivating the alpha build, there are two objectives which stand out: (one) the primary goal of *sdf.sys\_alpha* was to establish the basis of an environment with which to then complete this project, and (two) after my proposal for a live coded video example of this system<sup>115</sup> was accepted for publication on a Computer Music Journal (CMJ) DVD, its construction was imperative. The curators of that CMJ DVD (Thor Magnusson, Alex McLean, and Nick Collins) allocated two minutes for my video contribution which, thus, was titled *sdfsys2min* (Freeman, 2011).

The timeframe imposed on the alpha build by the deadline for the CMJ DVD gave a particular incentive: the possible complexity of the alpha build was limited by the available time. Some of the technical requirements of the contribution for publication were also allowed to influence design of the system.

## 6.2.3 The gl-ui

To implement *sdf.sys\_alpha*, the first step was to establish the basis for its visual representation.

### (6.2.3.a) Decisions in the gl-ui design

It had become clear – not least after the experience of creating *CirSeq\_Zero*, and inspired by some of Oli Larkin's work c.2009–2010<sup>116</sup> – that the best option, for the construction of a custom GUI to a new system being programmed in MaxMSP was to use the *jit.gl* family of objects from Jitter.

My first experiments with the *jit.gl* objects had occurred at the time of first developing what became thisis; §5.1 described how existing software systems for programmatically drawing to the computer screen, although powerful beyond my requirements, did not seem to offer the compositional workflow that I did require. It was also described that a coordinate system like that of OpenGL (with a centrally located cartesian origin) was then taken within the specification of

---

115 Which was quite hypothetical at the time of writing the proposal.

116 <http://www.olilarkin.co.uk/>

thisis. It is now recognised that there multiple workflows within the process of composition. Roughly speaking these can be divided into the categories of preparation and performance, but by performance here I refer to performative aspects within the act of composition, rather than to the performance of an already composed work; of course those two types of performance may coexist, as in improvisation. See §6.5.1 for a more thorough discussion and contextualisation of the different cognitive modes of working with computer music software.

In the preparation of the sdfsys environment, toward performative composition of new computer music, processes that lay outside the realm of the ideal workflow of the performance domain are embraced. Programming in MaxMSP is now, for me, outside the realm of the ideal workflow for composing new computer music, but remains, nonetheless, the software environment within which I can create most proficiently. Because sdfsys is created in MaxMSP with the use of Jitter and JavaScript techniques, it is undoubtedly influenced by an associated technē. It is, however, the intention of this commentary to describe sdfsys in such a way as to make its important features cognitively unbound from the actual implementation, such that future versions of sdfsys (or sdfsys like systems) could be programmed in different environments.

Knowing that the interim goal of the system was to demonstrate how its key components interrelate via video recording, the front end of the software was conceived with the target medium of the CMJ DVD in mind. It had been requested that the video format of the submitted .mov file be NTSC, which is a 'US-originated transmission standard for analogue colour television' (Chandler & Munday, 2011), and has a resolution significantly smaller than contemporary computer screens. Many of my previous works were either fitted to the screen of my own laptop at the time of making,<sup>117</sup> or were made to adapt to the size of the hosting screen (as in *g005g*). For sdfsys it was decided that the native size would be accepted as small; by rendering the interface with the `jit.gl`

---

<sup>117</sup> Admittedly, this practice is rather egocentric and is future proof in of itself only if the trend of each new laptop having equal or greater screen size resolution is maintained.

objects, the destination display can easily be resized, making it almost a moot point. The point here, however, is that the aspect ratio of the interface should be preserved when resizing.<sup>118</sup>

At this point there is introduced a certain degree of irony caused by the differing technologies of the computer screen medium and analogue television broadcast standards for which DVD video is designed. When the resolution of the NTSC standard is described as being 720 by 480 – such as in the *Final Cut Pro 7 User Manual* (Apple, n.d.) – it is because the pixels of that system are not square: their height is greater than their width. The aspect ratio of NTSC is defined as 4:3, but the aspect ratio of 720 to 480 is 3:2.

Bringing back to mind those aspects of sound that were looked at in chapter 2, one may recall that harmonic ratios are cited as important to my musical language, and also that certain features of human perception of pitch have been given attention in this project. Ratios are used both to describe the two dimensions of a screen and the two frequencies of an interval. If one tilts one's head to the side then 4:3 and 3:2 are as 2:3 and 3:4, which in auditory terms are the intervals of a fifth and fourth (§2.1.3). If the choice is thus to choose between a fifth or a fourth to stand as an ever-present visual aspect of the compositional system that I am building, then it is the smaller-numbered ratio of the fifth that I choose. Shepard has already been cited (§2.2.2) as reporting ethnomusicological studies to 'have found that octaves and perfect fifths seem to be culturally universal'; the spiroid-frequency-space (one of the mappings that is used for parameter sprites in sdfsys) is based on octaves, and so it is fitting for the fifth to manifest in the framing of the space to be mapped. The native resolution of sdfsys is thus 720 by 480 measured in square pixels on screen, and a `jit.window` object, named 'sdf\_ui', is created with those dimensions.

### (6.2.3.b) Divisions within the gl-ui

The gl-ui for the alpha build of sdfsys is divided into two parts: on the left is the main area, and to

---

<sup>118</sup> It is not strictly necessary for that to be the case, but it is the way that I chose to think of the design.

the right of that is the sidebar. The main area occupies the largest possible square within the window, and is where the visual layering of data-matrices, text and parameter sprites takes place. The sidebar is sub-divided into eight framed squares. Each frame represents one of the eight spaces, and miniature views of the data-matrices belonging to the eight spaces are displayed within those frames.

Figure 6.2 shows the outline of the layout, but the lines shown here are not rendered on the gl-ui in *sdf.sys\_alpha*; instead, when the mouse point is within the side area, the frame of the space represented by the square under the pointer is highlighted. The following section (§6.2.4) describes the interaction with these frames and all other aspects of the gl-ui.

Numbering of the eight spaces starts at zero at the top-left of the two columns in the sidebar; space one is below space zero, space four is to the right of space zero, and space seven is represented by the framed square in the bottom-right corner of the gl-ui.

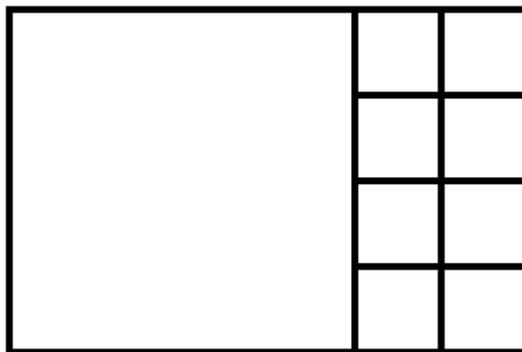


Figure 6.2: *sdf.sys\_alpha* ui layout

## 6.2.4 Interacting with sdf.sys\_alpha

### (6.2.4.a) Rendering the gl-ui and remapping the mouse position values

In *sdf.sys\_alpha*, the connection between the visual and the audible is emphasised by linking the rendering of the gl-ui to the DSP state: one cannot interact with the sdfsys environment unless the DSP in MaxMSP is switched on. When DSP is on, the gl-ui is rendered and interaction can be mediated by mouse and qwerty input.

When *sdf.sys\_alpha* is loaded, the gl-ui is seen at its native size in a window below the top level patch. The button in the top right corner of the patch can be clicked to enter the 'fullscreen' view of gl-ui, and then the [esc] key can then be used to exit. Fullscreen viewing of the gl-ui requires that the aspect ratio of the hosting screen be of a 'widescreen' format because, when the height of the gl-ui is maximised, on a 4:3 display, the left and right edges of the gl-ui would be rendered off the sides of the screen. Typically, fullscreen viewing of the gl-ui will include some null space to the left and right because 'widescreen' is rather wider than the 3:2 ratio used by sdfsys.

When the gl-ui is fullscreen, the mouse position is still reported by the `jit.window` object in terms of screen pixel coordinate values. I found that the 'screen-to-world' type translation built into the `jit.render` object did not give me the numbers that I would expect for mapping the actual screen pixel position to a point within the gl-ui.<sup>119</sup> To remedy this, I implemented an alternative solution for mapping the mouse position values to allow sdfsys to 'know' where the mouse is relative to its own gl-ui. A positive byproduct of this custom subroutine is that it generates a 'pixel scale' value that is then shared as a global variable via JavaScript for use in other parts of the system.

### (6.2.4.b) Frames in the sidebar

The 'active state' of all eight spaces is set by default to the value 0, which is represented visually by

---

<sup>119</sup> I have prior experience working with the `jsui` object and the 'screen-to-world' type method therein, which may have clouded by expectation of the `jit.gl.render` behaviour which was found not to be as expected. As is often my style, I simply chose to fix the problem, rather than worry about it.

the absence of the frame around that space in the sidebar. The active state of a space can be changed by clicking, or [shift]+clicking, within its frame in the sidebar grid. There are three possible active states; a normal mouse click will toggle between states 1 and 2, and [shift]+click will set the active state of the represented space to 0:

- active state 0 = 'Hidden'/'Off',<sup>120</sup>
  - frame in sidebar is hidden, and nothing of this space shown in main area;
- active state 1 = 'Visible',
  - frame in sidebar is yellow, and data-matrix of this space shown is in main area;
- active state 2 = 'Active',
  - frame in sidebar is red, data-matrix and parameter sprites of this space shown in main area, and mouse interaction messages will be received for use with the sprites.

### (6.2.4.c) sdf.sys.txt

Of the eight spaces in *sdf.sys\_alpha*, space zero is reserved for the text editor, and is thus unlike the other seven spaces in several ways. In contrast to the three active states described above, there are just two active states for text editor in space zero:

- 0 =
  - sdf.sys.txt not active, nothing shown in main area, and interaction inputs are ignored.
- 1 =
  - sdf.sys.txt is active, text buffer is shown in main area, qwerty input is received, and mouse click in the main area will set cursor location in the text buffer.

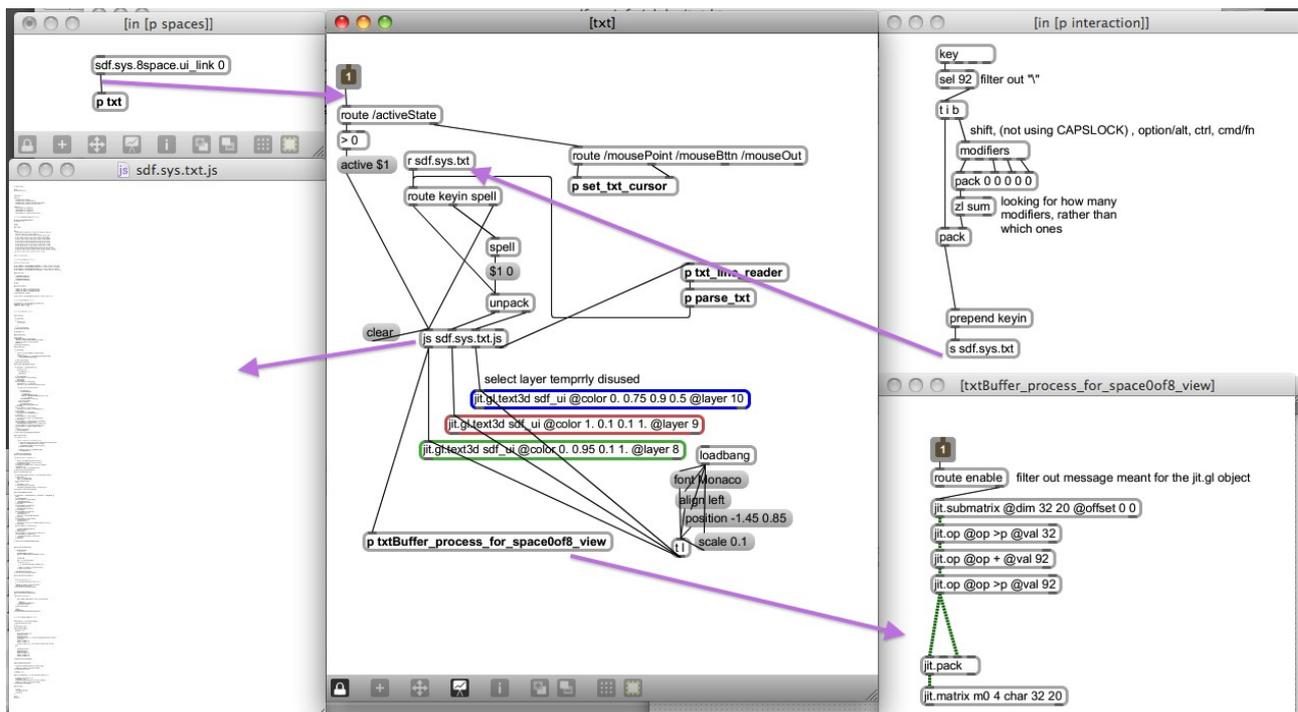
---

<sup>120</sup> The term 'Off' was originally used here, but may be misleading because the DSP processing of that space (if there is a module loaded) is always 'On' (as it were), and the data-matrix is always available to other spaces. Perhaps 'visible state' would be better than 'active state', but the latter is the term used in the code for both alpha and beta versions.

Another way in which the *sdf.sys\_alpha* space zero differs from the other seven spaces in the set of eight spaces is that, rather than having a float32 type data-matrix, multiple char type data-matrices are used by the 'sdf.sys.txt' sub-system.

The first matrix is the text-buffer and the char values stored in this are displayed as green-coloured text in the main area of the gl-ui. Second is a matrix that is used to display a single underscore glyph which, while blinking on and off, represents the sdf.sys.txt cursor location within the text-buffer; this matrix is rendered as red-coloured text in the main area. A third char type matrix, intended for use as indication of selected text, is not in use.

Figure 6.3 shows some of the many parts of the programming that contribute to the implementation of sdf.sys.txt in *sdf.sys\_alpha*.



The patch shown at the top-right of Figure 6.3 is an excerpt from the 'interaction' sub-patch, of *sdf.sys\_a51.maxpat*, showing the initial processing of qwerty-keyboard input to the system.

First, the “\” (backslash character) is filtered out because it is more trouble than that character is worth to process that as a string. Next, the modifiers object is triggered and the number of modifier

keys being held during the key press is determined. That number is then packed into a list with the ASCII number of the key-press, and a 'keyin' message is formed as input to `sdf.sys.txt`.

The concept of a 'double-modifier' is used in `sdfsys`; abbreviated to `[dblmod]`, the combination of `[shift]+[alt]` works on Mac OSX. Double-modifier key-presses are used as special inputs to `sdfsys`. For example, the key-combination `[dblmod]+[backspace]` is used to delete all characters in the text-buffer,<sup>121</sup> and return the text cursor to the start of the first line.

In description of how the thisis drawing system came to be (§5.1), it was told that an external (to MaxMSP) text editor is used to write thisis syntax scripts in plain text files, and that when the text file being used is saved the thisis maxpat would be triggered to read each line of that file in sequence and parse the commands contained. That concept is brought into the alpha build of `sdfsys` in two ways:

- The text-buffer of the internal (to `sdfsys`) editor is parsed when the key-combination `[dblmod]+[enter]` (or `[dblmod]+[return]`) is pressed; each line of the text-buffer is processed, one line at a time, and any valid `sdfsys` or `thisis` command is acted upon before the next line is read.
- As well as bringing into `sdf.sys_alpha` the subroutines that comprise thisis syntax parsing and drawing, the abstraction that was used to read the external text files (`txtFile.maxpat`) is also connected within `sdf.sys.txt`; the following walkthrough gives an example of it being used via input to the text-buffer.

#### **(6.2.4.d) A video walkthrough: commands and parameters in `sdf.sys_alpha`**

Adopting a similar structure to that of `sdfsys2min` (Freeman, 2011), the following text, that accompanies a video walkthrough of `sdf.sys_alpha`, aims to give concrete examples of concepts already introduced. The video file is [sdfsys9min2013alpha.mov](#) which will be found in the

---

<sup>121</sup> Actually sets all cell values to 32 which is the ASCII code representation for a space.

sdf.sys\_alpha\_video\_examples subfolder; the video was recorded with a live spoken commentary.

The first text command used in the video is to read from a pre-prepared text file that then sets the text-buffer with a number of other commands to initialise this within the environment:

```
| txtfile read 2013exmpl.txt
```

Those initialising commands from that text file are then parsed (Figure 6.4), and some basic this drawing commands are demonstrated (Figure 6.5) before the 'polarpeek' module unit patch is loaded with:

```
| setup space 2 load polarpeek
```

When that is parsed, and the module is loaded, the text-buffer is cleared and then set with a prompt:<sup>122</sup>

```
| /! set matrix to read from: !/  
| space 2 m_peekfrom m?
```

By changing the '?' for '!', and parsing the text-buffer again, the 'polarpeek' module – which functions like a read-head-stylus of the gramophone model – is set to read from the this drawing data surface.

The polarpeek module that is loaded actually comprises two parts: a maxpat, and a .js with corresponding filenames. When the setup command is given for the module to be loaded into a space, both files are loaded. First the maxpat is loaded into a **poly~** object; this patch contains a data-matrix-linked-DSP-algorithm with named parameters exposed via **pattr** objects and a **pattrhub**. The .js then acts as a bridge between those parameters and the gl-ui by creating a **jit.gl.gridshape** Jitter object for each: these are the sprites that are shown in the main area, and the .js that created them is also responsible for handling the mouse interaction with them.

---

<sup>122</sup> If a text-buffer line is empty or begins with // or /! then it is ignored; text parsing is based first on the first symbol in the line, but, for thisis commands in particular, the syntax is thereafter slightly more complicated than a simple left to right hierarchy.

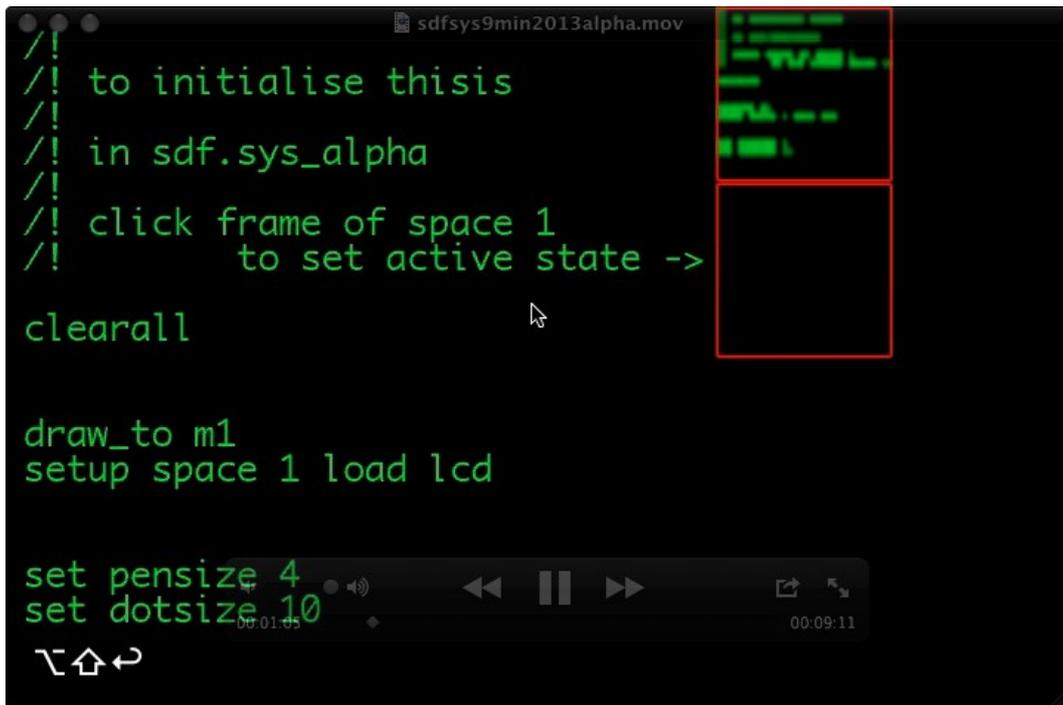


Figure 6.4: sdfsys9min2013a\_1m5s



Figure 6.5: sdfsys9min2013a\_2m12s

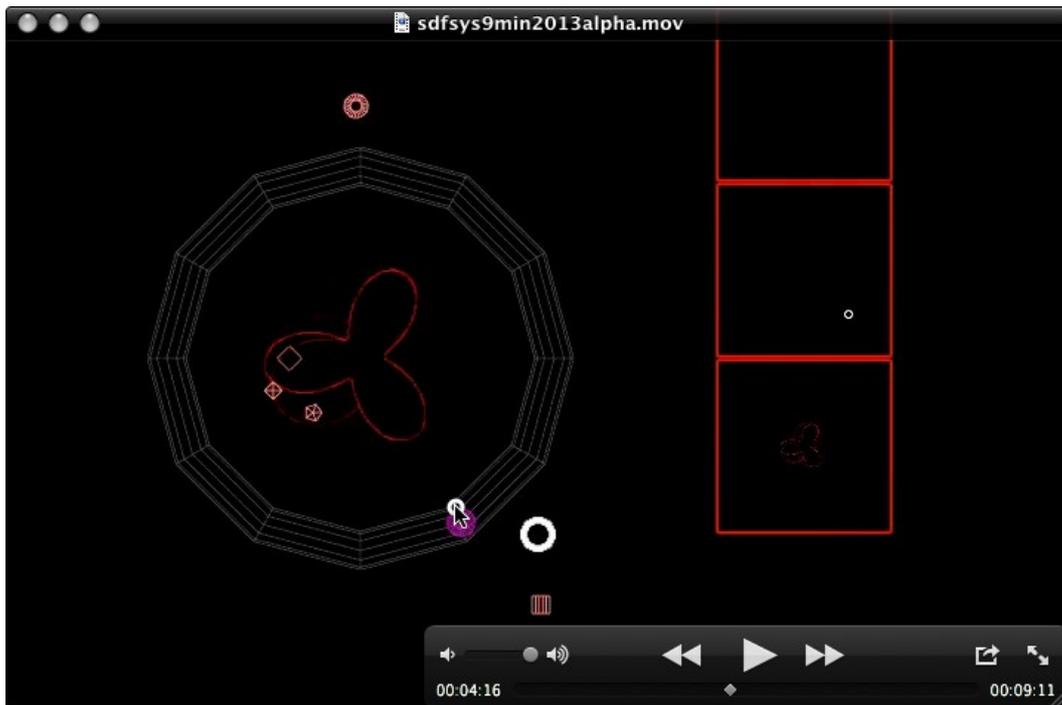


Figure 6.6: sdfsys9min2013a\_4m16s

At the moment of the video that is shown in Figure 6.6, the mouse is being used to drag one of the frequency type parameter sprites. This type of sprite is always circular (technically, it is using the 'torus' shape of the gl-gridshape as its glyph), and the inner radius size of the frequency type sprite glyph is used to help visually differentiate between the different frequency parameters belonging to the same space. Whilst a frequency type sprite is being moved in the main area, an overlay 'guide' is made visible. This guide is also a torus in the jit.gl world, but it is configured to appear as a dodecagonal shape; the twelve radial lines manifest in this guide give visual orientation for equiangular divisions of the spiroid-frequency-space mapping that is used on the plane for these sprites.<sup>123</sup>

<sup>123</sup> The spiroid-frequency-space guide in the *sdfsys\_beta* involves a thinner torus hoop and a spiral overlay, both still with dodecagonal corners though the spiroid mapping is not pitch-class quantised.

After a brief period of soundmaking with this configuration – polarpeek reading from the thisis drawn data – the module named tildegraph is loaded into space 3 (Figure 6.7), and then the command is entered to let polarpeek read from the data-matrix of that:

```
| space 2 m_peekfrom m3
```

The algorithm implemented in this tildegraph module was originally saved under the name 'cycle writer' because it uses three **cycle~** objects to write bipolar audio type values to the data-matrix cells. The output of two of the oscillators combine to control a path on the plane, and the third is used as the 'input' value for what is being written to the data-matrix.

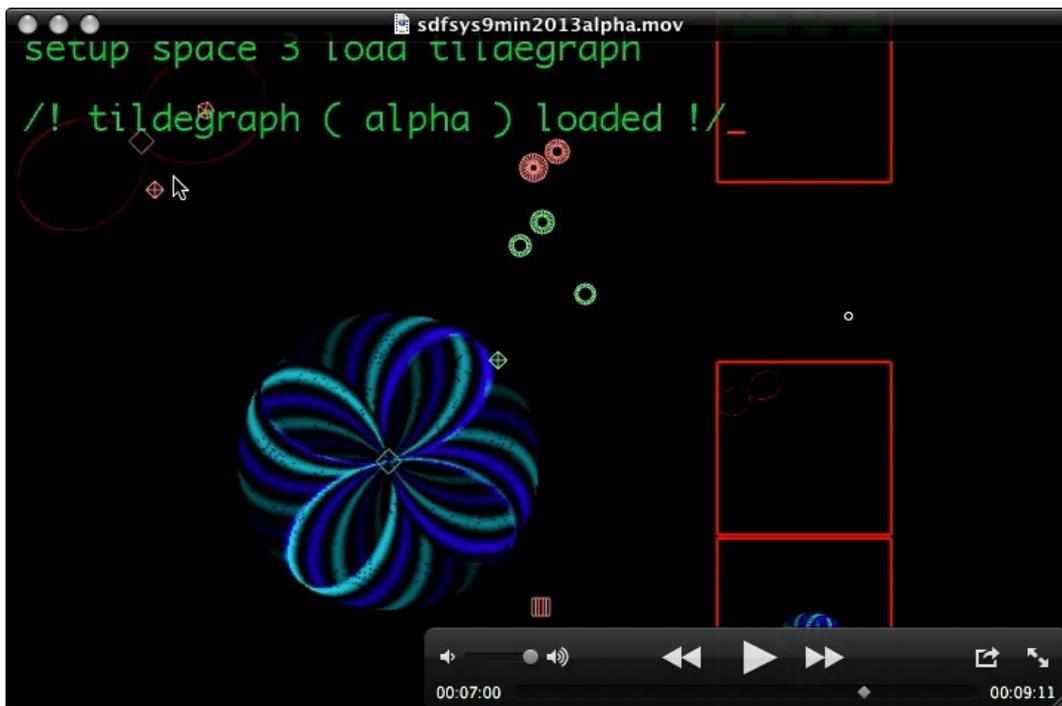


Figure 6.7: sdfsys9min2013a\_7m

## 6.2.5 Soundmaking in context

### (6.2.5.a) Colourisation: Looking at bipolar audio data-matrices on screen

With the tildegraph module writing audio data, and polarpeek reading that data in *sdf.sys\_alpha* – as demonstrated in the final few minutes of *sdfsys9min2013alpha.mov* – the soundmaking software configuration is very similar to that of my conceptual model of a gramophone. One may recall –

from §3.4.5 – that in the `gramophone_002_a` patch, only the positive values of the data surface were being visualised. That issue is resolved here by using blue and cyan for colourising bipolar data (see inset sub-patch in Figure 6.8).

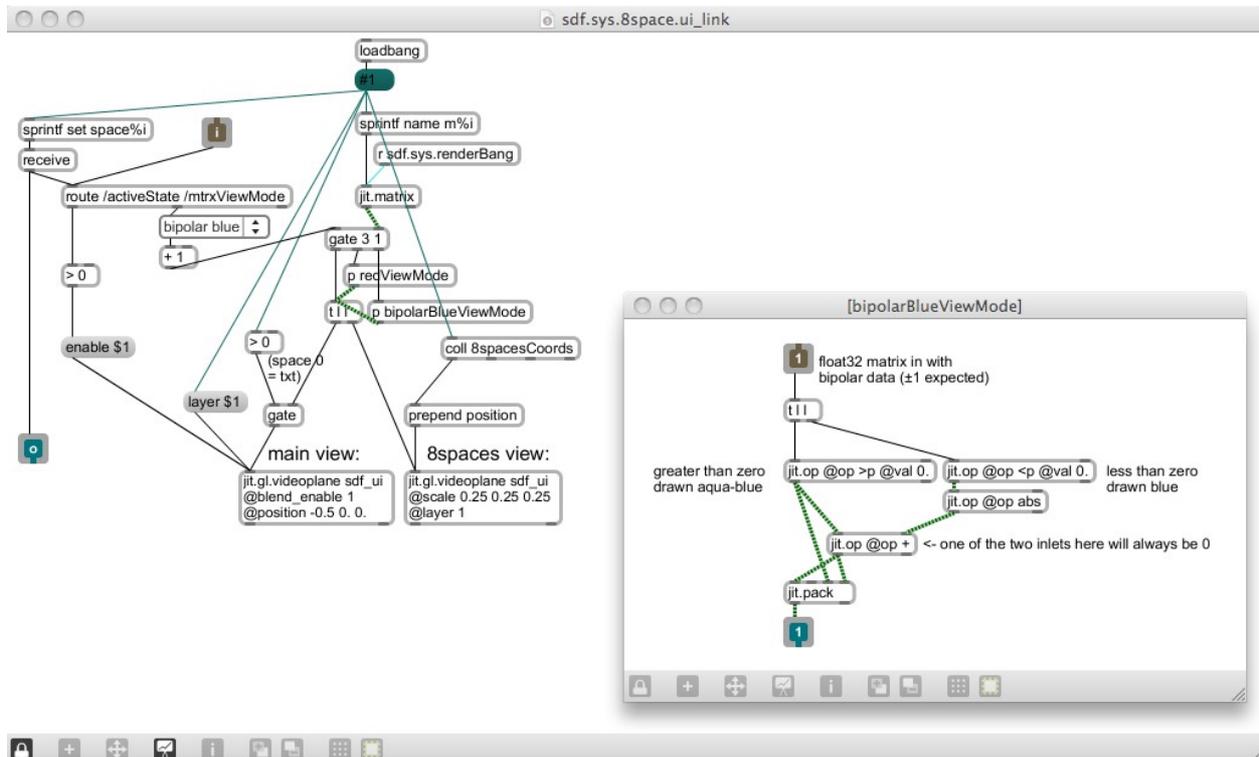


Figure 6.8: showing [p bipolarBlueViewMode] in `sdf.sys.8spaces.ui_link.maxpat`

### (6.2.5.b) Tildegraph: from a module to a concept

The range of possible sounds that can be made in `sdf.sys_alpha` with polarpeek reading tildegraph is very broad, especially when one considers how simple the underlying DSP actually is. One can go from harmonic organ-like tones to pitched and non-pitched noise by moving a parameter-sprite by just a few pixels on screen. I came to think of this soundmaking technique as 'tildegraphing', and so the word tildegraph was allowed to give name to a concept within the work, while the name of the audio data writing module in `sdfsys` was, for the beta version, reverted back towards its original name.

## 6.2.6 The tildegraph concept

### (6.2.6.a) Seeing sound

I presented an overview of the tildegraph concept, in the context of sdfsys and this project, at the 2013 Seeing Sound symposium;<sup>124</sup> a pdf version of my presentation is available online.<sup>125</sup> Whereas the primary focus of that symposium is to do with 'visual music', I explained that I do not think of my work as fitting that genre: I am not making visual music, I am making music visually. In visual music the screened aspect is treated as an independent element of the whole, often used as counterpoint to the sonic part of a work.<sup>126</sup> The direct relation of audio and visual data through the tildegraph concept in sdfsys seeks to establish an altogether different experience. The system has been conceived as a working environment for composing, and while audience observation is expected, that expectation arises from the context of live coding praxis. The abstract patterns that are typical of sdfsys tildegraphing, although independently stimulating in the visual domain, are controlled as an integral part of the soundmaking process, rather being a counterpoint or complement to it.

The animated visual works of John and James Whitney featured as a prominent theme in the 2013 Seeing Sound symposium programme, and there is strong visual link between much of that work and the visual character of my tildegraphing modules. Their 1944 *Film Exercises* used a 'subsonic sound instrument [that] consisted of a series of pendulums linked mechanically to an optical' system which was 'was synthetically exposed onto film' (Stamp, 2013).<sup>127</sup> This has obvious connection to the earlier described harmonograph. Later Whitney works were made with adapted analogue computer, and then early digital computer technologies; Zabet Patterson, for example, provide discussion of the Whitney brothers' use of these in an aesthetic context (Patterson, 2009).

---

<sup>124</sup> 23–24 November 2013, Bath Sap University; <http://www.seeingsound.co.uk/seeing-sound-2013/2013-papers/>

<sup>125</sup> [http://www.sdfsys.info/blog/wp-content/uploads/2013/11/seeingsound2013\\_tildegraph\\_in\\_sdfsys.pdf](http://www.sdfsys.info/blog/wp-content/uploads/2013/11/seeingsound2013_tildegraph_in_sdfsys.pdf)

<sup>126</sup> cf. Papers from the 2011 Seeing Sound symposium: <http://www.seeingsound.co.uk/programme-2/papers/>

<sup>127</sup> Richard Stamp (2013) cites: (John Whitney, 'Moving Pictures and Electronic Music', 1959)

### (6.2.6.b) The tildegraph concept in the context of wave terrain synthesis

To contextualise this soundmaking technique the wave terrain (WT) method of synthesis is cited (Roads, 1996, p. 163–164):

Several computer music researchers, including [Bischoff, Gold, and Horton (1978)], Mitsuhashi (1982c), and Borgonovo and Haus (1984, 1986), have explored the possibilities of techniques that scan a wave terrain using two indexes.<sup>128, 129, 130, 131</sup>

The traditional conceptualisation of a wave terrain is as a three-dimensional surface in which (ibid., p.164):

the *z*-point or height of the surface at each point represents a waveform value for a given pair (*x*,*y*). The waveform stored in such a table is a *function of two variables*, and the technique has also been called *two-variable function synthesis* (Borgonovo and Haus, 1986).

A scan over the terrain is called an *orbit*. Although the astronomical term “orbit” connotes an elliptical function, the orbit can consist of any sequence of points on the wave terrain.

The read-head-stylus type module of the tildegraph concept in sdfsys (such as the polarpeek module described) is analogous to the WT orbit, but there is a significant difference to be noted. The WT orbit is described as remaining always within the dimensional bounds of the WT: it is said that 'the orbit skips from one edge of the wave terrain to another edge' which is the same as 'the right-to-left wraparound that occurs in one-index wavetable scanning' (ibid., p.164–165). The **cycle~** object in MaxMSP implements a one-index wavetable that can be scanned by signal value input to its right inlet. An input value of 0 is (mentally) visualised as the left edge, and 1 as the right. If the input is greater than 1, then wraparound will occur such that, for example, an input of 1.2 gives the same output as does an input of 0.2. In the implementation of my conceptual model of a gramophone (§3.4), a Jitter matrix is used as a two-index wavetable with the **jitter.peek~** object facilitating access to that matrix data which is (actually) visualised with left, top, right and bottom

128 Bischoff, J., R. Gold, and J. Horton. 1978. “A microcomputer-based network for live performance.” *Computer Music Journal* 2(3): 24–29. Revised and updated version in C. Roads and J. Strawn, eds. 1985. *Foundations of Computer Music*. Cambridge, Massachusetts: The MIT Press.

129 Mitsuhashi, Y. 1982c. “Audio signal synthesis by functions of two variables.” *Journal of the Audio Engineering Society* 30(10): 701–706.

130 Borgonovo, A., and G. Haus. 1984. “Musical sound synthesis by means of two-variable functions: experimental criteria and results.” In D. Wessel, ed. *Proceedings of the 1984 International Computer Music Conference*. San Francisco: International Computer Music Association. pp. 35–42.

131 Borgonovo, A., and G. Haus. 1986. “Sound synthesis by means of two-variable functions: experimental criteria and results.” *Computer Music Journal* 10(4): 57–71.

edges on screen. In the soundmaking experiments that were based on that implementation I found creative use of the non-wraparound behaviour exhibited by the **jit.peek~** object: any input to the dimensional index inputs of **jit.peek~** that is beyond an edge of the wavetable-matrix gives an output value of zero. This behaviour is the basis for the controlled inclusion of silent periods within the audio output of the *g005g* composition (§3.5), and the tildegraph concept inherits directly from that work.

In description of WT synthesis, Roads continues to address the 'problem of generating predictable waveforms' with the technique (p.164):

systematic investigations of the technique have focused on wave terrains generated by relatively simple mathematical functions. [...] the advantage of using simple mathematical functions is that it is possible to predict exactly the output waveform and spectrum generated by a given wave terrain.

My work has been shown to favour simple mathematical functions, but exact prediction of the output is not my motivation for that. With the tildegraph concept I am interested in controlling sound – that is to say in generating an output waveform and spectrum – by manipulating the parameters of multiple interacting spatiotemporal processes. These processes are comprehensible both in terms of their visual manifestation<sup>132</sup> and in terms of how those processes, and the interaction between multiple processes, are manifest in sound. It just so happens that (aesthetically) pleasing patterns can be created, in both the visual and audio domains, from the (technical) implementation of simple mathematical functions. My investigation is focused these such factors.

The audio output possibilities of orbits on a two-variable function wave terrain were studied by Borgonovo and Haus as a specific case of a more general idea (1986, p. 57) :

The basic idea of the technique is that we can obtain an audio signal by the sampling of an  $n$ -variable function along an orbit determined by  $n$  expressions of time-dependent variables. Samples can be directly calculated or read by means of  $n$  dimensional table lookup.

[...] We have not considered  $n > 2$  cases because they need too much memory.

---

<sup>132</sup> One has control (either directly or indirectly) over visual attributes such as radius, rate of rotation, and shape.

The decision of these authors to study a two-dimensional structure is, thus, attributed to the technical factor of what their computer<sup>133</sup> was able to handle. On the 'continuous spectrum that connects aesthetics to technics' (Gilbert Simondon, 2012), it would appear to me that researchers such as Borgonovo and Haus are focused more toward the technics. My own work, however, is more focused on the aesthetic part of the techno-aesthetic.

My decision to work with a two-dimensional method that is similar to WT synthesis is derived from the fact that screens are flat (that is to say, two-dimensional), and that I am exploring visual representation of sound in software, and on screen. It is a technical observation that screens are flat; it is an aesthetic motivation that directs my practice to hold that as important.

The tildegraph concept is not restricted to the use of WT-like function-filled data-matrices. The cycle-writing module that first carried the tildegraph name can, indeed, be thought of as being based on a simple mathematical formula with time-dependent variables (and thus very WT-like). The first example of the orbit-like polarpeek module generating sound in sdfsyz, however, was with data being read from the matrix being drawn to via thisis commands. Thisis permits geometrical drawing, and geometrical thinking has been a part of human culture for much longer than has the algebraic conception of number (Barrow, 1993; Hart, 2012) through which data writing functions are expressed.

---

<sup>133</sup> Borgonovo and Haus (1986) 'implemented the technique on a digital signal processor (a Digital Music Systems DMX-1000) for real-time synthesis and on a Fairlight CMI Series IIX (controlled via a DEC VAX-11/750) for interactive experimentation and composition.'

### (6.2.6.c) Wave terrain and scanned synthesis

During the period of this research I have noticed a number of projects based on the 'scanned synthesis' method of soundmaking.<sup>134</sup> Of these, research by Robert Tubb has also made reference to WT synthesis; Tubb's 'Investigation into Scanned Synthesis and Multi-Touch' (2011) cites the WT technique as closely related that of scanned synthesis, pointing out (p.14) that Roads' discussion of WT synthesis 'pre-dates the first paper on scanned synthesis by a number of years when he says':

One can also imagine an extension where the orbit is fixed but the wave terrain is time-varying. In this case the wave-scanning process is equivalent to tracing the curves of an undulating surface, like wave motions on the surface of the sea. (Roads, 1996, p. 166)

Another contemporaneous work to employ scanned synthesis is the 'Tüb – Interactive Sonification of Water Waves' (Erlach et al., 2011). This is a system that uses a Cymatic-like configuration of a circular body of water over which specific lighting and a camera are positioned; human interaction with the water creates wave patterns, and DSP based on the camera images of these then generates sound.

---

<sup>134</sup> Described by Bill Verplank, Max Mathews and Rob Shaw (2001)

## 6.2.7 Concluding the alpha build

The flow-chart in Figure 6.9 illustrates, at a relatively high level of abstraction, how *sdf.sys\_alpha* is structured. Having arrived at this stage in the development of *sdfsys*, programming stopped, and attention was turned to questioning what was working well, and what could be made better. The alpha build, as its naming suggests, was always going to be a testbed for the ideas contained within it. Certain problems within *sdf.sys\_alpha* – such as those leading to all but three of the eight spaces module slots being removed – were addressed by the process of redesigning various aspects of the environment.

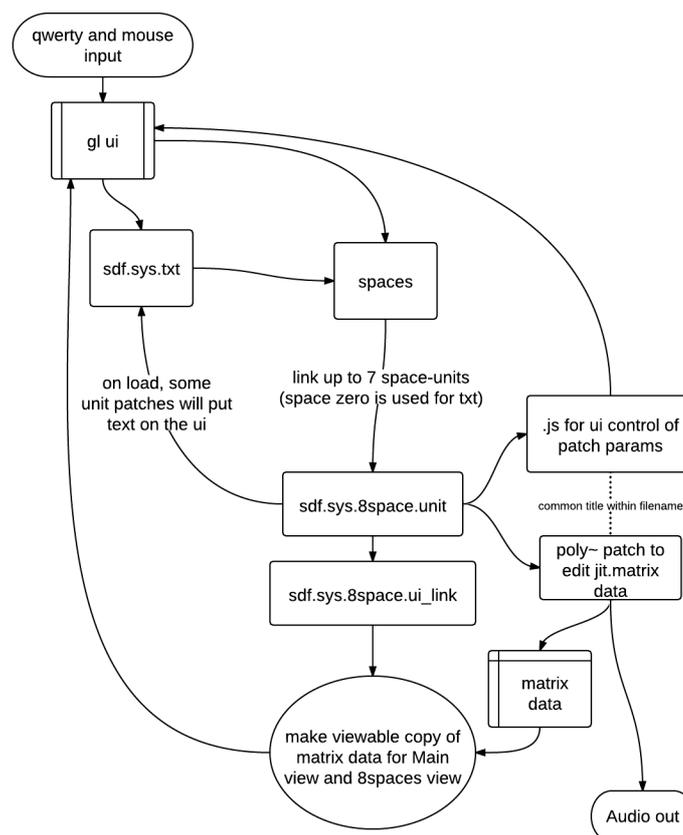


Figure 6.9: SdfSysAlphaStructure

## 6.3 Beta

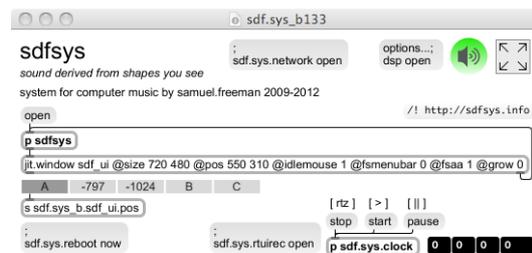
Building upon the foundations established in *sdf.sys\_alpha*, the beta version of sdfsys introduces frameworks for greater modularity (at various levels of abstraction), significant enhancements to the gl-ui, and many more module units for loading into the eight spaces of the environment. This section describes *sdfsys\_beta* in terms of the enhancements and additions made after the alpha build of the system.

### 6.3.1 sdfsys\_beta

The `_beta` folder, under `chapter6_sdfsys` of the portfolio directories, contains two sub-folders: the `sdfsys_b_doc` folder contains various documents that support description of *sdfsys\_beta*; and the `sdfsys_beta1` folder contains the files that comprise the work itself.

Most of the other works in the portfolio have had their directories 'tidied up' to some extent: for *sdf.sys\_alpha* – as a case in point – only the files needed to run the system, as demonstrated in the video walkthrough of §(6.2.4.d), are included. For *sdfsys\_beta*, however, I have decided to preserve the contents of the folder to the inclusion of everything 'as is'.

The top level patch, to be opened in order to run *sdfsys\_beta*, is [sdf.sys\\_b133.maxpat](#) (shown in Figure 6.10). To enter the fullscreen view of the gl-ui one can click on the icon in the top right corner of the patch; the key-combination `[dblmod]+[esc]` can also now be used to enter fullscreen, and – as before – the `[esc]` key is used to exit fullscreen of the gl-ui.



## 6.3.2 The beta design for the gl-ui

To accommodate new ways of working within the system, the layout of the gl-ui was redesigned.

The *sdfsys\_beta* design represented in Figure 6.11 was arrived at after process that took several perspectives of the work into consideration. As well as being directed by the technical needs of the new elements to be represented on the gl-ui, the aesthetic matter of aspect ratios was again allowed to influence the design. The five areas of the gl-ui, described below, are named: the main area; the spaces network area; the box area; the system toggle; and the text toggle.

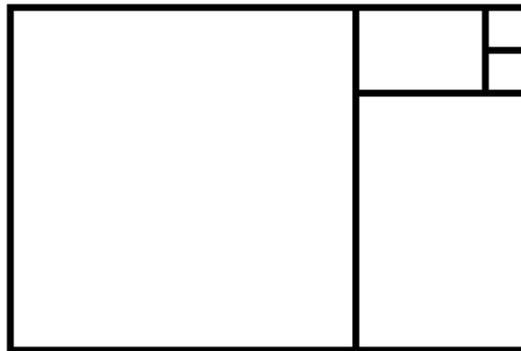


Figure 6.11: sdfsys\_beta ui layout

```

sdf_ui
-----
| sdfsys |
-----
(cc)by @samuelfreeman
-----
To run sdfsys,
  DSP must be on in Max
-----
Some qwerty key commands
  for the text editor:
-----
[dblmod]+[enter] to parse
[dblmod]+[backspace] to clear
try [shift]+[alt] as [dblmod]
url http://sdfsys.info

/! text from
info.txt
cpu:00 fss:23.21
vitein
spaces
matrix
network

```

The dividing lines that are drawn on the beta gl-ui have been chosen so that that the processes of drawing them, one at a time within the 3:2 window, will only create internal aspect ratios that equate to the intervals of unison, octave or perfect fifth (see Figure 6.13). The alternating horizontal and vertical orientation of the divides also allows for the simple coding of four if-else stages to be used for identifying which of the five gl-ui areas the mouse point is in.

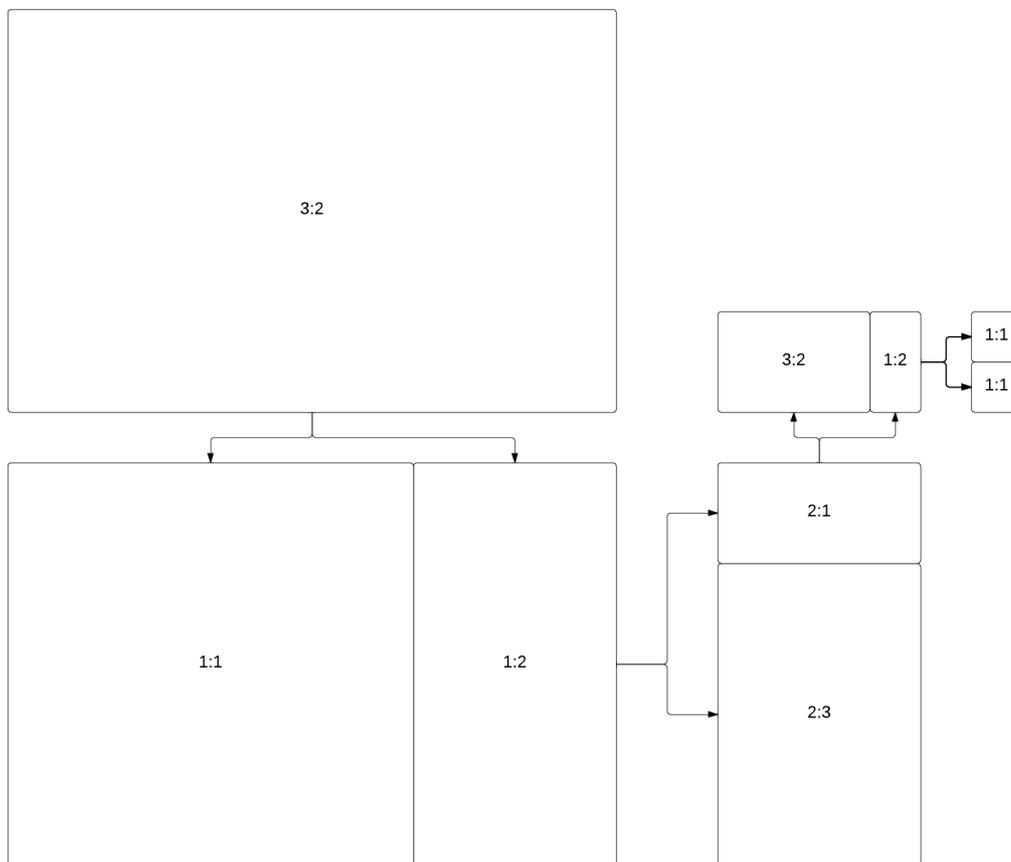


Figure 6.13: sdfsys\_b1\_ui\_ratios

Each of the five gl-ui areas has its own sub-patch within the coding of the system, and these will be described below in turn. With the exception of the clock, which is implemented in its own patcher at the top-level of the maxpat, all of the patching that comprises the system is found within the patcher named 'sdfsys' (Figure 6.14).

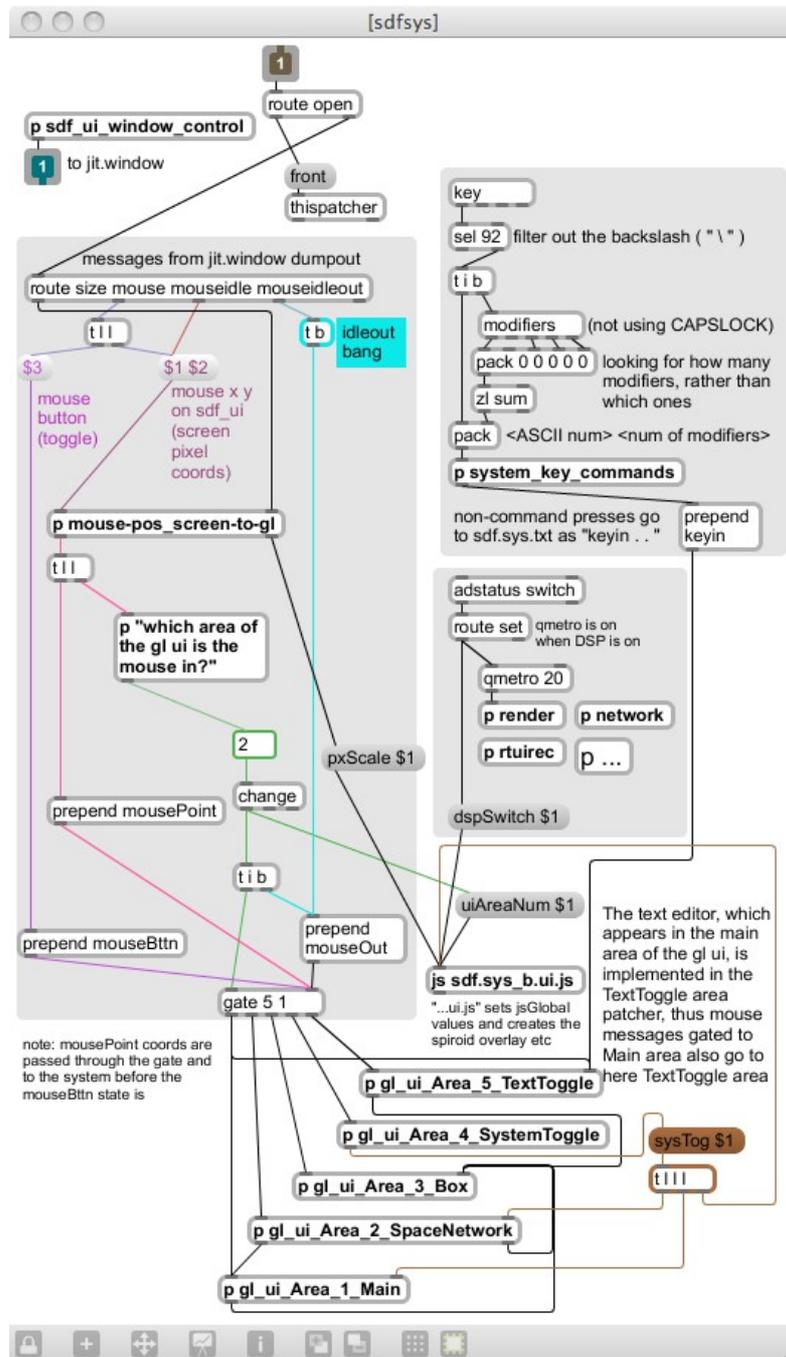


Figure 6.14: [p sdfsys]

### 6.3.3 Colour in sdfsys\_beta

Colour is used in three ways within the *sdfsys\_beta* environment:

- text: white for information output in the box and system toggle areas of the gl-ui, and green for the text-buffer display, with a red underscore to show the input cursor location;
- data-matrices: the 'matrix\_view\_type' sub-patch of the space structure has implementation for five types of matrix data colouration, see Figure 6.15;
- spaces: eight colours for visually identifying different aspects belonging to each of the eight spaces, see §(6.3.5.b).

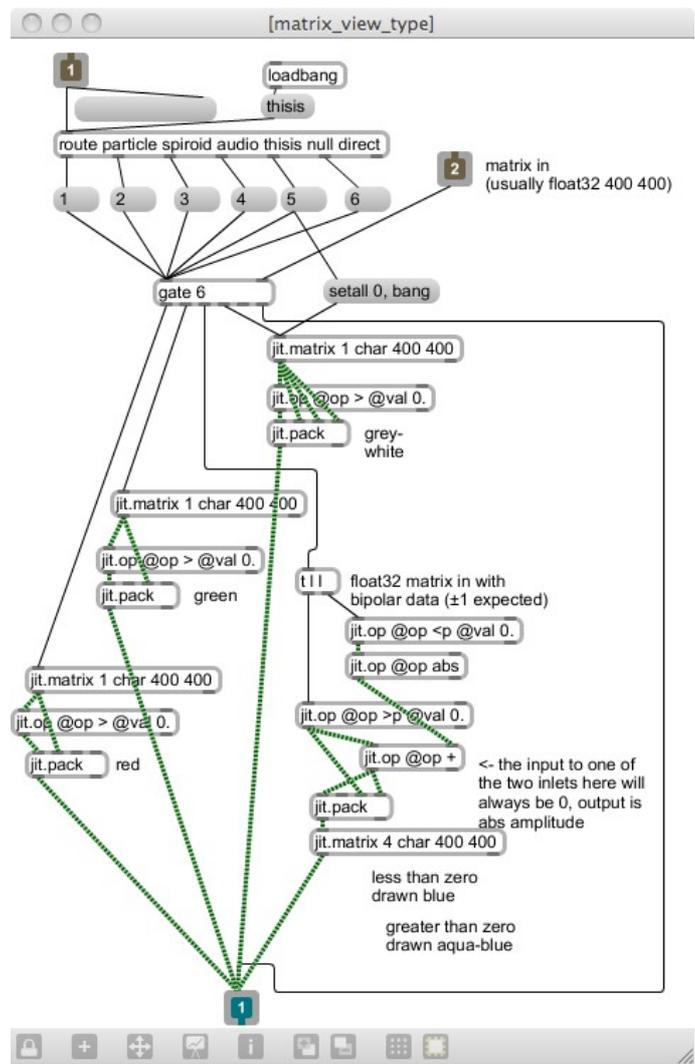
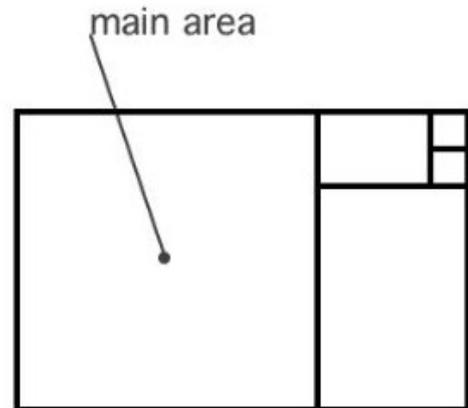


Figure 6.15: matrix\_view\_type

## 6.3.4 \_Area\_1\_Main

### (6.3.4.a) Mousing data

The first inlet of each `_Area` sub-patch is for mouse input data messages which will only arrive to one of these sub-patches at a time, according to where in the gl-ui the mouse is pointing.<sup>135</sup> In the main area patch, the mouse input is connected to a `poly~` patch that will forward the mouse data messages to any of the eight spaces that currently have their active state value set to 2.



### (6.3.4.b) Eight spaces

Before programming began for the beta version of `sdfsys`, a new 'space structure specification' was drafted on paper. This included a new way to work with the spaces network – involving the new spaces network area of the gl-ui; see §6.3.5 – and a better method for linking the parameter attributes of a module patch to their sprites on the gl-ui.

Eight instances of `sdf.sys_b.space.structure.maxpat` are created and connected in the `_Area_1_Main` patch. In `sdfsys_beta`, all eight spaces are the same because the text editor, rather than occupying one of the spaces, is conceptualised as a distinct part of the system (§6.3.8). Messages that target the spaces are received via `r sdf.sys.spaces` and routed according the integer (0–7) at the start of each message. The `send toAllSpaces` namespace is also being used. The 'space structure' patches have an outlet from which will come messages targeting the 'box area' of the gl-ui (see §6.3.6).

Whereas, in the alpha build, each module unit would comprise a maxpat and a `.js`, the beta structure for spaces is based on each instantiated space structure having its own instance of a

<sup>135</sup> Implementation of the text editor is found in the `_Area_5_TextToggle` patcher (§6.3.8), but because the editor appears in the main area of the gl-ui, mouse messages gated to `_Main` are also connected to go to the `_Text` patch.

`sdf.sys_b.space.structure.js` which handles the creation and control of gl-ui sprites. There is then a protocol by which a module unit maxpat – that will be loaded into a one-voice **poly~** object – can interact with its instance of the space structure JavaScript file. This protocol is embodied as an adaptive and re-usable block of patching: a configuration of objects that can be copy-pasted to any new module patch being made, and that requires a number of manual edits to be carefully applied.<sup>136</sup>

The flowchart in Figure 6.17 provides a simplified high-level overview of how the space structure specification fits into *sdfsys\_beta*.

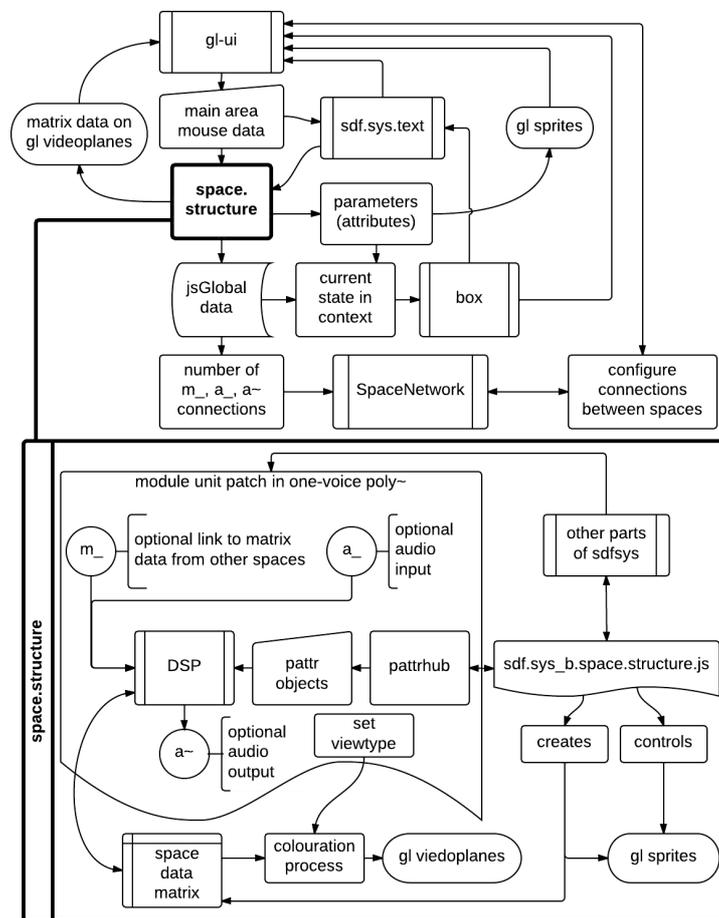


Figure 6.17: space\_structure\_flow

<sup>136</sup> There are notes within the existing module unit maxpat files to aid with creation of new modules; see `~g.null.maxpat` which is the default, parameterless, module that is loaded into each space structure when the sdfsys environment starts up.

**(6.3.4.c) Parameter-sprites**

The following table shows the six types of parameter-sprite that can be used by module unit patches for *sdfsys\_beta*. The parameter-sprites of a module unit that is loaded into a space will – when the active state is 2 – appear in the main area of the gl-ui with the colour of space into which that module is loaded.

Type	Glyph	Mapping from position on the plane to parameter-value			
Hz		Polar (spiroid)	$(r, \theta) \rightarrow$ (octave-level, pitch-class)	$\rightarrow$	(frequency value)
point		Cartesian	$(x, y) \rightarrow$ (left to right, top to bottom)	$\rightarrow$	(0–400, 0–400)
radial		Relative radius	(distance from a reference sprite)	$\rightarrow$	(0–400)
xdim		Horizontal slider	(left to right)	$\rightarrow$	(0–1)
ydim		Slider	(bottom to top)	$\rightarrow$	(0–1)
rotary		Dial (at a fixed location)	(down/left to decrease, up/right to increase)	$\rightarrow$	(0–1)

The rotary type of parameter-sprite is unique within this set of sprites because it comprises two parts: the first part is the needle of the dial is shown with the colour of the space to which the sprite belongs, and this is visible whenever that space is in active state 2; the second part is the circumference of a circle that is only shown (and always in grey) when the mouse clicks down on the area of the sprite.

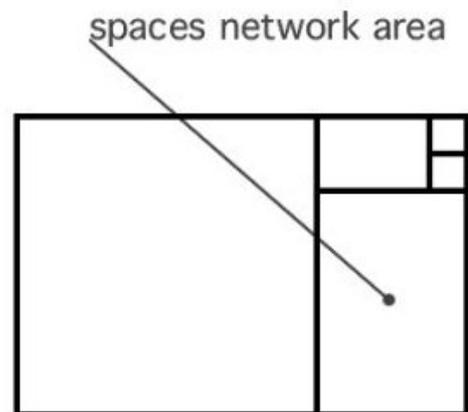
The `xdim` and `ydim` types can be moved anywhere within the main area of the `gl-ui` but only one of the cartesian dimensions is used to set the value of the parameter. The point and radial types use the 0–400 range in order to match the dimensions of the data-matrices, make simplifies certain matters for the visually-linked DSP algorithms that use them.

A radial sprite is created as 'referring to' another sprite. The parameter-value of a radial sprite is set by its distance away from its reference sprite; a radial sprite will thus be moved on screen when its reference sprite is moved (so to remain at the same distance). One radial sprite (let's call it `r2`) can refer to another radial sprite (`r1`), but only if the reference sprite of that radial (`r1`) is a different type of sprite (usually a point type). Better use of recursive programming techniques could probably allow for radials to refer to radials that refer to radials, and so on, but the system in place has been sufficient so far.

## 6.3.5 `_Area_2_SpacesNetwork`

### (6.3.5.a) Visual patching

A visual patching paradigm is added to the specification for the eight spaces aspect of `sdfsys_beta` for connecting matrix and audio data between the spaces. Rather than being part of the fixed the `gl-ui` layout, the eight frames – and the mini-views of matrix data within the frames – are now more like the sprites of the main area: the frames are smaller now than they were in the alpha build, and they can be moved via click-and-drag to be arranged within the spaces network area of the `gl-ui`.



In the `sdfsys` maxpat, mouse point and button messages coming into the spaces network part of `sdfsys` are processed to create inputs for `sdf.sys_b.space.network.js`; Figure 6.19 shows the message data flow in both the maxpat and `.js` elements of this sub-patch.

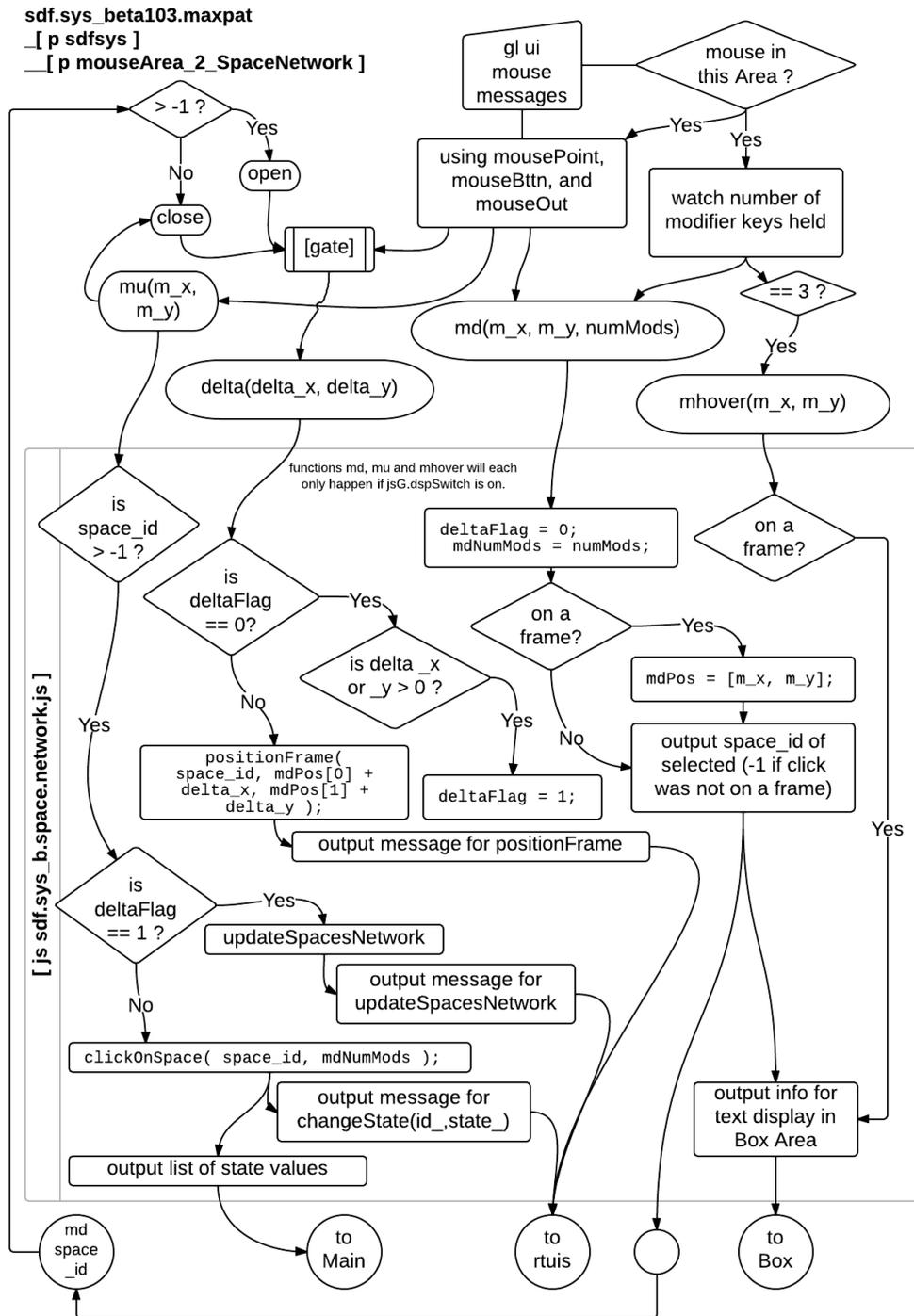


Figure 6.19: \_Area\_2\_SpacesNetwork\_flow

### (6.3.5.b) Eight colours for the eight spaces

The set of colours that are used to visually identify each of the eight spaces are defined in

[sdf.sys\\_b.space.network.js](#) as an array of RGBA values that is globally accessible to .js functions in different parts of the system:<sup>137</sup>

```
jsG.spacesRGBlist = new Array(9); // each [item] holds [r, g, b, a] value array
jsG.spacesRGBlist[0] = [1., 0., 0., 0.45]; // red
jsG.spacesRGBlist[1] = [1., 0.5, 0., 0.45]; // orange
jsG.spacesRGBlist[2] = [1., 1., 0., 0.45]; // yellow
jsG.spacesRGBlist[3] = [0., 1., 0., 0.45]; // green
jsG.spacesRGBlist[4] = [0., 1., 1., 0.45]; // aqua
jsG.spacesRGBlist[5] = [0., 0.5, 1., 0.45]; // blue
jsG.spacesRGBlist[6] = [0.375, 0., 1., 0.45]; // purple
jsG.spacesRGBlist[7] = [1., 0., 1., 0.45]; // magenta
jsG.spacesRGBlist[8] = [0.5, 0.5, 0.5, 0.45]; // grey
```

The frames in the spaces network area start with active state 0, which is represented visually by a thin grey line; states 1 and 2 are shown by thin and thick coloured lines using the colours of that .js global array.

### (6.3.5.c) Making connections in the spaces network

The spaces network area has two view modes:

- mode 0 is for matrix connections;
- mode 1 is for audio connections.

In each view mode, the frames can be arranged within the spaces network area, and clicking on the system toggle area (§6.3.7) will switch between the two arrangements of the frames.

Connections are made automatically when the frames are drag-dropped, and – referencing systems such as the *reacTable* (Jordà et al., 2007) and *Texture* (McLean, 2011, sect. 5.6) – the logic

<sup>137</sup> The colour comprising green and blue, indexed 4 in this array of arrays, is more commonly called cyan, but the slightly less precise name aqua became habit after the MoE *Pixels* project (§3.1) during which the latter was chosen for use because I thought it would be more accessible for the workshop participants.

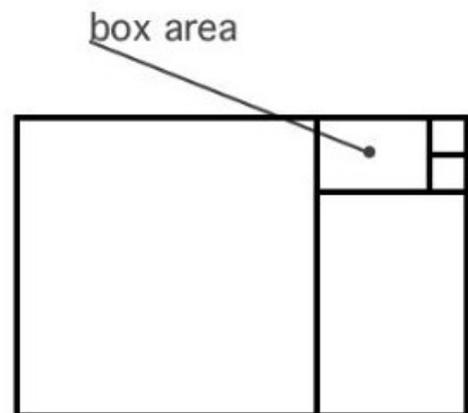
determining network connections is based on the proximity of outputs to inputs on the frames that represent the eight spaces. Outputs are represented by coloured dots on the bottom-right corners of the frames, and inputs by coloured dots on the top-left corners.

Each of the eight spaces has a data-matrix, and so all eight frames always have an output dot when the spaces network area is in view mode 0. If the module will read data from the matrix of another space then it requires a matrix input in the network representation. Matrix inputs, audio inputs, and audio outputs are all optional for the module units. When a module is loaded to sdfsys it will report which of the input and output options are to be used.

I decided to limit the space structure specification to only allow one optional matrix input, and module may have an audio output, or an audio input, or neither, or both.

### 6.3.6 `_Area_3_Box`

In the original prescription for the sdfsys environment, written before commencing to build *sdf.sys\_alpha*, it was stated that text would be used both by the human to instruct the system, and by the system to inform the human. In *sdf.sys\_alpha* this was achieved by letting modules alter the text-buffer, but I was not completely happy with that as a solution; a design in which the human input is being



overwritten by the system is at risk of undermining the interactivity. In *sdfsys\_beta*, the box area of the gl-ui is used for displaying textual information.

Information presented in the box area is most often related to the last thing that was clicked on, be that a frame in the spaces network area or a sprite in the main area, but the text display of the box area – comprising eight lines; 23 characters per line – is also targeted by the `sdf.sys_b.txt` sub-

system to report syntax errors and respond to query commands (see §6.3.8).

Whilst musicking with *sdfsys\_beta* there were a number of occasions upon which I wanted access to the box text, such as to be able to copy-paste parameter-values into a script. To facilitate this, I made it so that clicking three times on the box area<sup>138</sup> will bring up a standard text editor window with a copy of the box text. Although this external access to an element of the self-contained environment is not used often, it is useful to have it available. After the implementation of that feature, I had the idea for a context sensitive copy-paste inspired method – internal to *sdfsys* – for taking information out of the box area and into the main text-buffer. This an method is used much more often in practice than triple-clicking on the box area.

The key-command [dblmod]+[B] is used to access the text-buffer, and if the information currently displayed there is one of two recognised formats, then elements of it become available for pasting into the text-buffer with the key-command [dblmod]+[V]. If the box is displaying information about the space of the last frame to be clicked on, then the word 'space' and the appropriate number can be made available for pasting; this can be used as a shortcut to typing commands that target that space, without needing to think in terms of what number it is. If the information in the box area is about a parameter-sprite then the text that can be made available to paste provides the start of a command line for changing the value of that parameter; see §(6.3.8.c) for direction to syntax documentation.

A high-level representation of what is happening in the `_Area_3_Box` sub-patch of *sdfsys* is provided by the flowchart in Figure 6.21.

---

138 The count is reset if the mouse point leaves the box area.

sdf.sys\_beta133.maxpat  
\_[ p sdfsys ]  
\_[ p gl\_ui Area\_3\_Box]

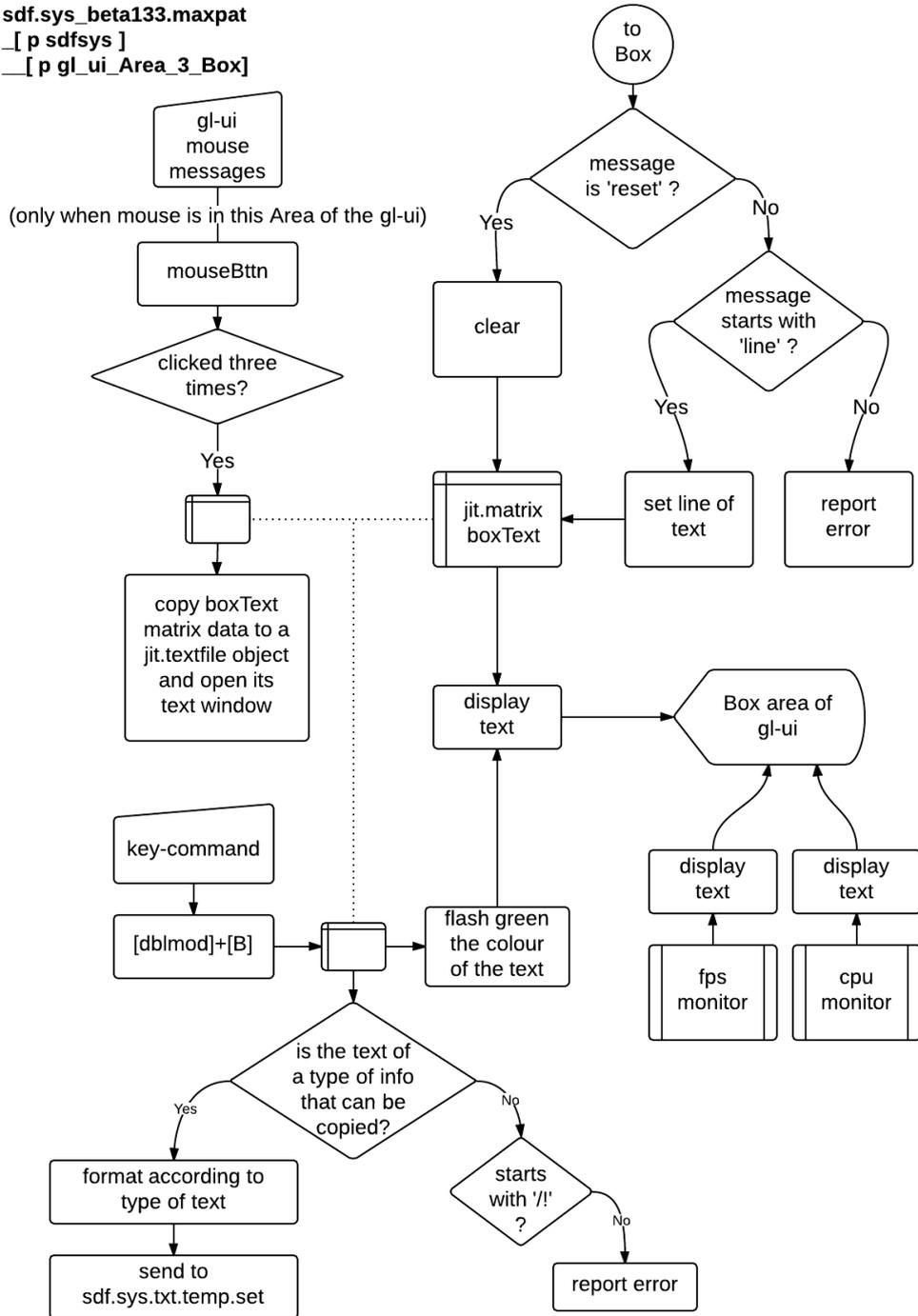
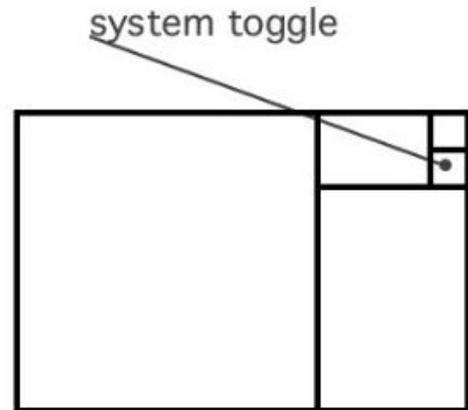


Figure 6.21: `_Area_3_Box_flow`

### 6.3.7 \_Area\_4\_SystemToggle

The system toggle area acts as a simple toggle to switch between the two types of view for the spaces network area (§6.3.5). When `sdfsys_beta` is loaded view mode 0 is selected, clicking on the system toggle changes it to mode 1. The most recent version uses the verbose text elements of 'matrix' and '\_audio' in the system toggle area of the `gl-ui`, but – as may be noticed in videos and photographs – previous beta versions only showed the mode number.

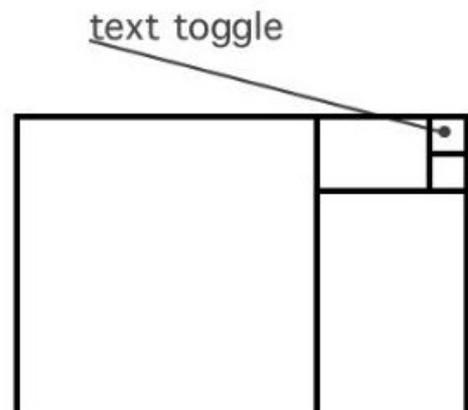


### 6.3.8 \_Area\_5\_TextToggle

#### (6.3.8.a) To toggle the text

The fifth area, in the top right corner of the `gl-ui`, is the text toggle. In terms of its own small square of screen space, it has three ways to respond to being clicked on, but because it represents the text editor of `sdfsys_beta` it has a lot of patching within its sub-patch.

Three types of click: (1) if the text toggle is clicked when DSP is off, then the DSP Status window will be opened; (2) in general operation, a click will toggle the active state of the text editor, the key-combination `[dblmod]+[T]` has also been added for the same purpose; (3) similar to triple-clicking on the box area, clicking four times on the text toggle area will open a windowed copy of the text-buffer so that the text can be taken out of `sdfsys`.<sup>139</sup>



<sup>139</sup> It is four clicks, rather than three, to have the active state of the text editor be back as it started; more than two needed to make sure it is intentional.

### (6.3.8.b) Editing the text-buffer

Advancing on the alpha build text system, the beta version adds more usability with additional features being implemented as a .js and maxpat hybrid, with the two parts interoperating to affect change on the text-buffer in response to qwerty-based input. For example, it was much quicker in a maxpat than it would have been in JavaScript, to add methods for inserting and deleting lines from the text-buffer; these are actioned via the key-combinations [shift]+[enter] and [shift]+[backspace].

### (6.3.8.c) Syntax in sdfsys\_beta

The number of commands available in the system has grown significantly. As well as some new system commands, the thisis drawing system has received many additions and improvements. A comprehensive overview of all available commands has been prepared in a .mm file, but it is perhaps best viewed in its .html export version: the sdfsys\_b\_doc folder has a sub-folder called sdfsys\_syntax within which is [sdfsys\\_parse\\_text3.html](#). The hierarchical structure of that syntax documentation is summarised here:

- Syntax of the text system in sdf.sys\_b130
  - About this document (sdfsys\_parse\_text3.html)
    - The lines of text from the text buffer are delivered as Max (list) messages from [p txt\_line\_reader], one at time, to [p parse\_text].
    - Interpretation of most commands will terminate with triggering the next line of the text buffer to be read.
    - This document shows how the lines of text are interpreted based on keyword routing: the list is routed first by the first symbol in the list.
    - Most keywords are followed by extra arguments, as represented by the hierarchical structuring below.
    - The order of presentation here is matched to that of the keyword routing in sdf.sys\_b130.maxpat
    - The keyword commands that are divided into conceptual groups: (commenting), (sdfsys), (thisis), and (other)
    - Examples of a command line are given [in bold]
      - **// like this, followed by a description of that command**
        - in this example the line is 'commented out'; note the necessary space after the '/'
  - [p parse\_text]
    - (commenting)
      - /\*

- \*/
- /..
- ../
- //
- /!
- (sdfsys)
  - **help**
  - **info**
  - url
  - pcontrol
  - txt
  - txtfile
  - setup
  - space
  - send
  - draw\_to
  - .txt
- (thisis)
  - delay
  - **clear**
  - lcd
  - **clearall**
  - thisis
  - unput
  - put
  - draw
  - get
  - set
  - macro
  - spawn
- (other)

## 6.3.9 Infrastructures and Meta-patches

### (6.3.9.a) sdf.sys.rtuis

A framework for what I described as real-time user interface/interaction sequencing (rtuis) has been put in place within *sdfsys\_beta*. It was partly an exercise in forward thinking towards the

possibilities of allowing multitouch control of sdfsys, and to make temporal recordings of interaction.<sup>140</sup> The sequence recording aspect is underdeveloped, but the rtuiss sends that go to the various parts of sdfsys have been utilised in other ways: they allow for scripting of elements that were previously only accessible by mouse, such as setting the active state of a space, and moving a frame in the spaces network area. Visible in the top-level patch of the *sdfsys\_beta*, there is a clickable message to open the 'sdf.sys.rtuirec' patch.

### (6.3.9.b) sdf.sys.network

Also at the top-level patch of *sdfsys\_beta* is a message that can be clicked to open the sdf.sys.network meta-patch that enables Open Sound Control (OSC)<sup>141</sup> based network communication for sdfsys. It was constructed to be part of the HELOpg SLIME system (Hewitt, Freeman, and Brooks, 2012). One should always launch sdf.sys.network via that message in sdfsys (rather than opening the maxpat in any other way) because there is a setup in sdfsys that uses the **onebang** object so that the first click on the 'sdf.sys.network open' message will load the meta-patch, whereas all subsequent clicks on the same message will merely bring that maxpat to the front. This functionality is important because the sdf.sys.network meta-patch has many sub-patches that will all overlap onscreen; having ease of access to the main patcher is crucial during its use in performance situations.

### (6.3.9.c) sdf.sys.reboot

The second most important piece of advice that I give when helping people to learn how to make things in MaxMSP – where the first is to do with regular saving of work – is to do with regular rebooting of the work. In theory – and unless programmed not to – a maxpat will load into the same state every time that it is started from scratch. Unexpected behaviour experienced in a patch can often be remedied by closing and reopening. Sometime restarting the MaxMSP environment

---

140 As commented on in a rare blog post: <http://www.sdfsys.info/blog/2012/03/02/rtuiss-and-the-spaces-network/>

141 cf. (Freed & Schmeder, 2009)

(or even the operating system of the computer) may be called for, but in general, a patch reboot is all that is needed to reliably reinitialise the work.

For sdfsys, a reboot is the quickest way to reset the spaces network, and when experimenting with different configurations, new modules, and especially when developing something that includes meta-patch scripting of sdfsys, rebooting is a frequent action in the workflow.

Clicking the `; sdf.sys.reboot now` message, at the top-level patch of *sdfsys\_beta*, will load a floating window meta-patch that performs three time-delayed actions: (1) after an initial delay period, the sdfsys patch is closed; (2) after another delay period, the sdfsys patch is opened again; (3) finally the reboot patch closes itself. Pressing [esc] will cancel these actions by sending a `stop` message to the **line** objects that drive a visual representation of a countdown in time towards each action.

If one is immersed in the sdfsys environment, then one may reboot without cognitively exiting that frame of mind by parsing the following command:

```
| send sdf.sys.reboot now
```

Note that the word 'now' at the end of that command could actually be anything because it is only used to trigger a bang to open the meta-patch.

#### (6.3.9.d) Loading other meta-patches

Any maxpat file that is in the MaxMSP search path can be loaded by name from the sdfsys command line with the 'pcontrol' keyword that passes the all of its arguments to MaxMSP object of the same name. The following example opens the meta-patch saved – in the *sdfsys\_beta1* folder – as [controlp.maxpat](#):

```
| pcontrol load controlp
```

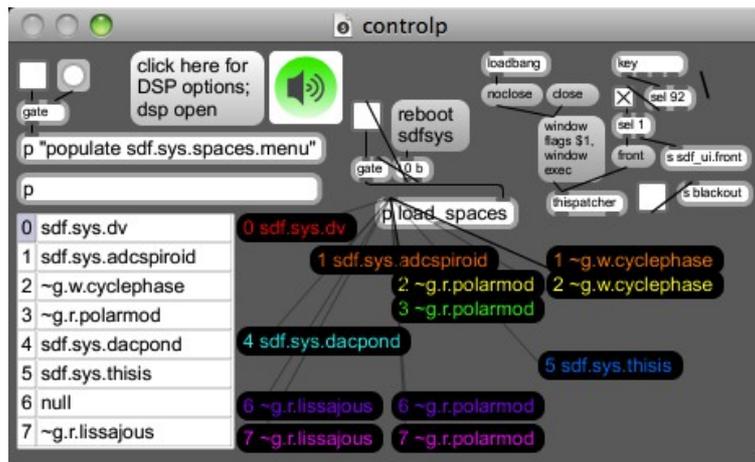


Figure 6.24: controlp after loading seven of the spaces

The controlp meta-patch (Figure 6.24) is based on the human\_seeking meta-patch that was created for as an aid to me playing in the sdfsys environment during a HELOpg performance at the Human Seeking event;<sup>142</sup> quotation after promotional image for that event (Angela Guyton, 2012):

Let's explore SONIC & VISUAL information through IMPROVISATION  
The role of TECHNOLOGY is considered, with each artist prioritizing it differently [re] their own practice  
& HUMAN CREATIVE POTENTIAL

MACHINE FUNCTIONING  
ANIMAL BEING  
**HUMAN**  
**SEEKING**  
SUBLIME TRANSMITTING

The sdf.sys.dv module was also created for that performance; it is a module unit that loads into a space but then also opens a meta-patch (the first module to do this). The meta-patch loaded by the sdf.sys.dv module has settings for using digital video (DV) input, and a single parameter-sprite in the sdfsys environment is used to switch between different mappings from the three-plane RGB char input to the one-plane bipolar audio-type float32 data-matrix. By using DV input to the data-matrix of a space in sdfsys, human movement and shape, beyond that afforded by qwerty and mouse, can be used in the soundmaking process.

<sup>142</sup> I wrote about the event shortly after; <http://www.sdfsys.info/blog/2012/06/06/human-seeking>: Other performances at the event included Richard Knight who had controlled feedback loops in and between two mixers without external input, and Takahashi's Shellfish Concern (Angela Guyton, and Rodrigo Constanzo) who combined live painting with both audio- and visual-domain DSP.

### (6.3.9.e) The sdfsys\_beta\_xnn\_video series

Five screencast recordings with live commentary form a short series exploring sdfsys\_beta by introducing the modules set out in the controlp meta-patch. The series begins with reference to walkthrough video of *sdf.sys\_alpha*; see §(6.2.4.d).

Located in the sdfsys\_beta\_xnn\_video sub-folder of the sdfsys\_b\_doc folder, the videos of this series (totalling approximately 42 minutes duration) are:

- [sdfsys\\_beta\\_x00\\_thisis.mov](#) (5 minutes 18 seconds);
- [sdfsys\\_beta\\_x01\\_tildegraphing.mov](#) (10 minute 54 seconds);
- [sdfsys\\_beta\\_x02\\_dacpond.mov](#) (8 minutes 50 seconds);
- [sdfsys\\_beta\\_x03\\_adcspiroid.mov](#) (10 minutes 55 seconds);
- [sdfsys\\_beta\\_x04\\_dv.mov](#) (6 minutes 23 seconds).

## 6.4 Composing with sdfsys

### 6.4.1 A composition with which to compose

#### (6.4.1.a) sdfsys as substance

In view of the perspective of software as substance (§3.7) the works described in §6.2 and §6.3 – *sdf.sys\_alpha* and *sdfsys\_beta* – are presented as compositional works in the portfolio of this project; thus the italics on their names, in contrast to the conceptual entity of sdfsys. The sdfsys environment exists without the software through which it is manifest, but it is only manifest through software. At the time of writing *sdfsys\_beta* (accessed, in the MaxMSP environment, via [sdf.sys\\_b133.maxpat](#)) is the most complete manifestation of sdfsys, and many other compositional works have been realised through it.

### (6.4.1.b) In the context of my compositional model

The compositional model that I drew in 2010, and that was discussed in §3.7.3, is revisited now.

The visual form of that model shown in §3.7.3 led to the observation that my perception of

composition as a process is very similar to the 'process of action and reaction in bricolage

programming' (McLean, 2011, p. 122). The first computer rendered version of the idea that was

originally scribbled on paper, however, had the conceptual stages arranged in a slightly different

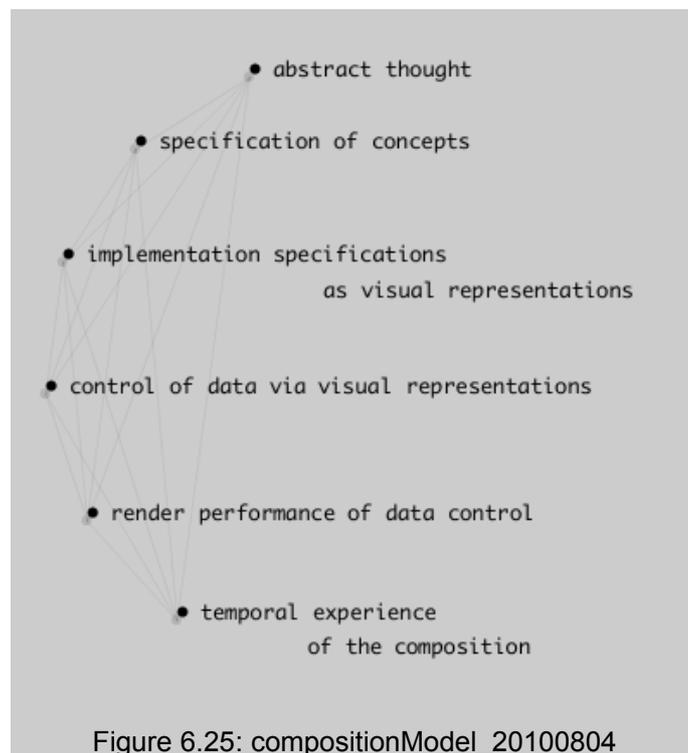
way. It is that version that is presented here, both in order that the model may be shown again

without repeating an image, and also to mention that it was rendered to .png from a thisis script;

Figure 6.25 is shown here on a grey background because some the thin lines that connect the

concepts in [compositionModel\\_20100804.png](#) , which has a transparent background, do not show

well upon white.



Abstract thought is the primary human factor at all stages in composition. Description of sdfsys in this chapter has focused mainly on the specification of concepts, and the implementation of those concepts as interactive visual representations. There have been some rendered

performances of control of data via those visual representations – §(6.2.4.d) A video walkthrough: commands and parameters in `sdf.sys_alpha`; §(6.3.9.e) The `sdfsys_beta_xnn_video` series – but these are more technologically than aesthetically motivated.

'Composing with sdfsys' implies completion of the conceptual stages shown in Figure 6.25 in order to express as music such things that cannot fully be expressed in any other way. Musicking with sdfsys is one part of it, but the sounds made as music through sdfsys can be experienced in much the same way as many other musics: either live, and I have performed sdfsys both in HELOpg and solo; or listening to recordings which may be acousmatic or with video, and this is as it was originally intended at the start of the project, though the question of whether or not to project the screen on to the acousmatic curtain continues to be explored.

Towards the objective of composing with sdfsys, there have been numerous approaches, and the pieces introduced below demonstrate the exploration of these.

#### **(6.4.1.c) On paper and on camera**

The visual appearance of the gl-ui can serve as a native notational form for sdfsys. Sketches made on paper can contain all necessary informational data to configure sdfsys either to realise something speculative, or to restore a specific configuration that was established while interacting with the system. The lines of text that form messages within the environment can be transmitted by any medium; this scripts are often written on paper or whiteboards that can then be photographed for portability. Sketches might include non-sdfsys-ean symbolic notation to indicate, for example, motions of parameter-sprites. Photographic or screencast images can be archived for the purposes of reproducing a specific configuration; examples are provided in the `sdf_ui_on_camera` sub-folder of `sdfsys_b_doc`.

#### **(6.4.1.d) Introducing the included compositions**

Documentation of the pieces described below can be found in the `_composing_with_sdfsys` folder

in chapter6\_sdfsys in the portfolio directories, but some elements of these works are, necessarily, located in the sdfsys\_beta1 folder, along with the workings of *sdfsys\_beta*. Alias links to such files in the sdfsys\_beta1 folder are provided for indicative purposes, but if opened are unlikely to find their target on the readers' copy of the portfolio directories; the filename of the alias will indicate where the target can be found.

The composition titles, given as sub-section headings below, match the folder names in the portfolio. The descriptions continue to be of a technical nature to allow the reader to arrive at their own aesthetic understanding of the works.

## 6.4.2 orbit8 reading cyclephase

A rather minimal approach to claiming composition with sdfsys is to state a relationship between two modules and let the performance of the piece be a free exploration of their parameters. That is the approach taken by *orbit8 reading cyclephase*, and two excerpts of it being played are provided in the portfolio; for each example there is a .mov screencast recording, a .wav of the audio extracted from that recording, and a .png showing the waveform view of that audio in Audacity which was used to extract it from the .mov (one can drag-and-drop a .mov on to an Audacity project window to do that).

### (6.4.2.a) 20120221\_0200\_cyclephase\_orbit8

(1 minute 25 seconds)

This recording was published online<sup>143</sup> with the following description:

```
sdf.sys_beta
space 2 load ~g.r.orbit8
space 3 load ~g.w.cyclephase
spaces network mode 0 (matrix connections): 3 => 2
```

143 At <https://vimeo.com/37150649>

~g.r.orbit8 has eight read-heads/peek-particles fixed to a circular path around a common center; each has radius size, frequency (cycles per second), and amplitude (multiplier applied to the peek'ed data) attributes/parameters. There are also master gain, slew time and center point parameters/attributes. None of these, however, are changed during this short video.

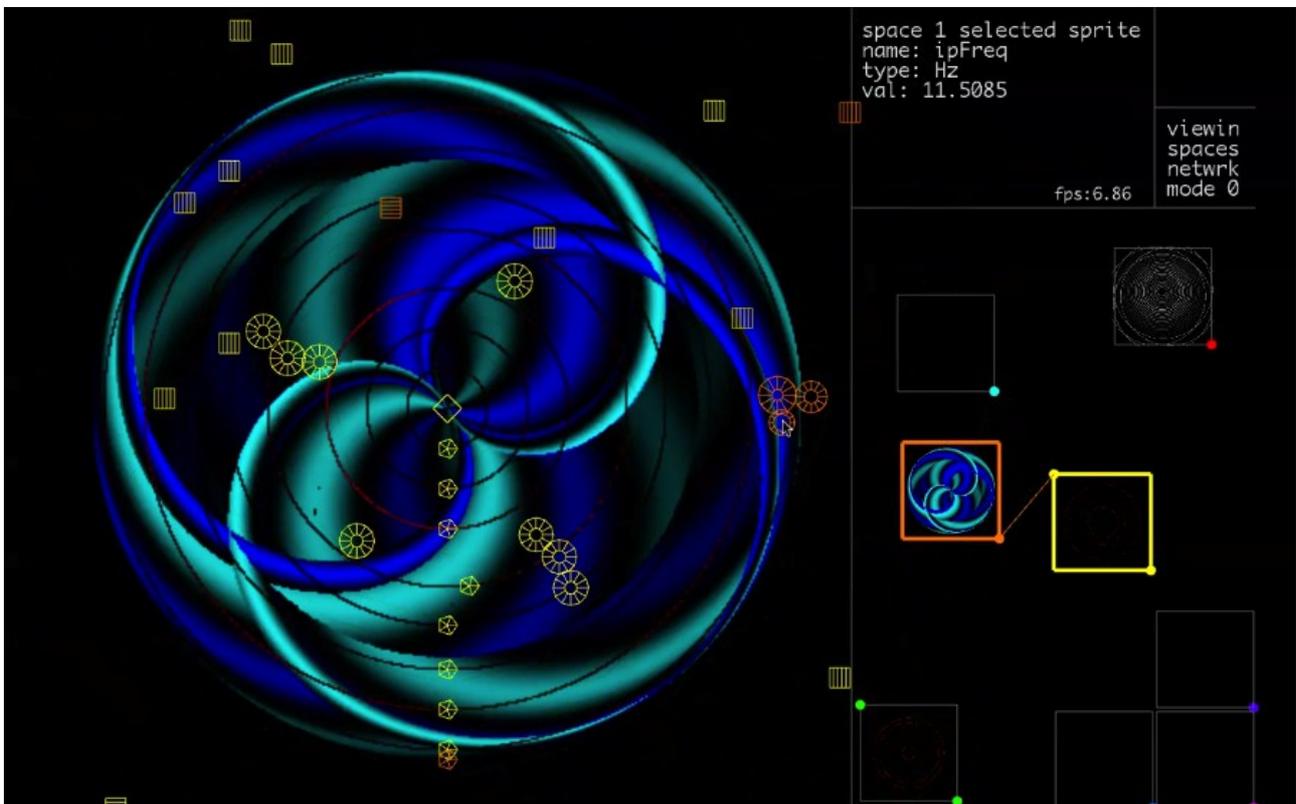
~g.w.cyclephase has one write-head/poke-particle which moves on the data matrix plane along a path determined by two frequency values: one to set the cycles per second of an circular orbit (this is the phase part of the name), while the second sets sinusoidal modulation of the particle path radius (the cycle part of the name). A third frequency type parameter is for another sinusoidal oscillator which provides the values to be written to the data matrix at the location of the particle. There are also slew time and trace parameters/attributes.

In sdf.sys\_alpha what is now ~g.w.cyclephase was called tildegraph but that name has since been reassigned to refer to the general concept at play here; that of graphical data read|write/peek|poke on a plane; the name inspired by the harmonograph of the nineteenth century plus the many meta- uses of the word, and symbol, tilde.

The frame rate in sdf.sys\_beta dropped by half when the QuickTime screen recording was started to record this video (too many other applications running is one reason for this). When the frame rate drops below about 10fps the interaction becomes less fun; thus the short duration of this video.

### (6.4.2.b) 20120222\_orbit8cyclephase

(2 minutes 22 seconds)



### (6.4.2.c) ~g.r.orbit8

There is also an example of *orbit8 reading cyclephase* within a more technical demonstration at the following file-path:

- .../chapter6\_sdfsys/\_beta/sdfsys\_b\_doc/sdf\_ui\_on\_camera/MVI\_0425\_trimmed\_h264.mov
  - (5 minutes 57 seconds)

The recording of that video identified a bug in the ~g.r.orbit8 patch: the xdim sprites that control the gain of each peek-operation-orbit allowed for negative values to be entered by dragging the sprite off the left edge; this has since been fixed so that the value is clipped to zero.

### 6.4.3 febandflow

(1 minute 39 seconds)

Another approach to composing with sdfsys is to allow sdfsys to function as an instrument, to record digital audio of it being played, and then to work with that digital audio as one may with recordings of any instrument.

In *febandflow*<sup>144</sup> a multi-track recording was made comprising four stereo tracks, and these are then combined to a stereo mix. My motivation for this recording of *febandflow* included a particular exploration of rectilinear representations of digital audio again. The four stereo-tracks that were recorded have been exported as an eight-channel .aif file which has been used in a prototype multi-track sonogram visualisation; this work is described in a video that includes presentation of the *febandflow* through my 'jit fft colour test' patch<sup>145</sup> (Figure 6.27):

- chapter6\_sdfsys/\_composing\_with\_sdfsys/febandflow/febandflow\_jitfftcolourtest06.mov
  - (5 minutes 53 seconds)

---

<sup>144</sup> The title *febandflow* was originally used to encompass a number of compositions that shared some common elements, but I have decided to apply the title to *febandflow* as described in this document, in spite of the potential confusion that this may cause if these pieces are revisited in future work.

<sup>145</sup> Also described online at [http://sdfphd.net/a/doku.php?id=jit\\_fft\\_colour\\_test](http://sdfphd.net/a/doku.php?id=jit_fft_colour_test)

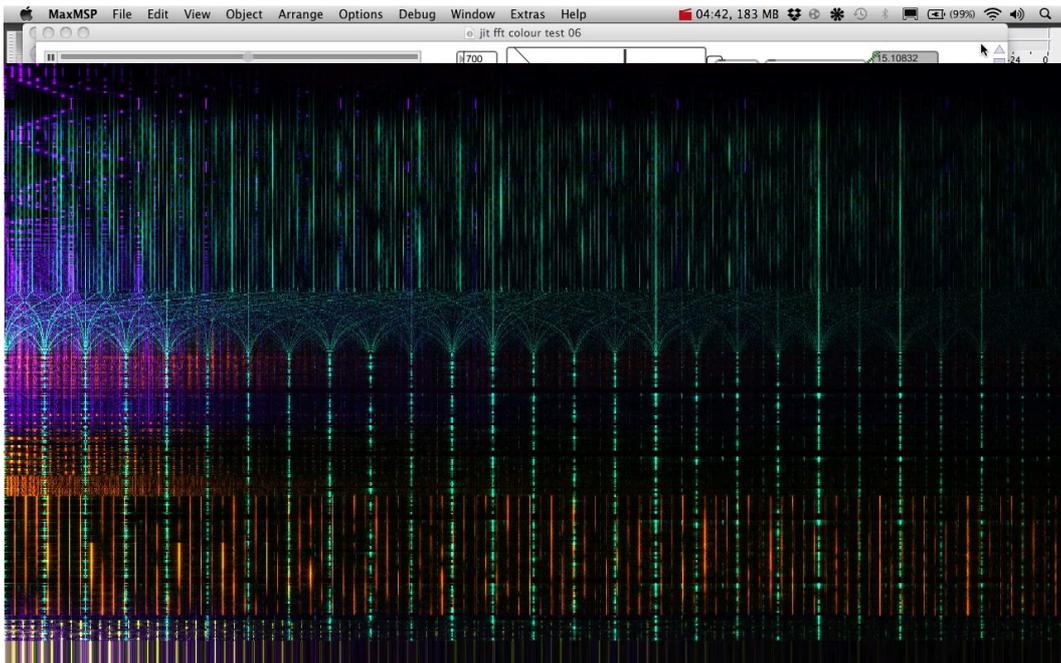


Figure 6.27: febandflow\_jitfftcolourtest06

## 6.4.4 five four three

### (6.4.4.a) Multiple layers of abstraction

The spatio-visual premise for *five four three* began as a sketched sequence of thisis commands; the sequence of sketches can be seen in the *fivefourthree\_whiteboard* sub-folder of this works folder in the portfolio directories.

To render that sequence of commands in time, rather than have the thisis drawing be rendered almost instantly in sdfsys, I implemented the typewriter model (see below).

As well as a scripted sequence for setting up sdfsys and drawing via thisis commands, this composition also includes meta-patch sub-processes that algorithmically control parameters of the loaded modules in sdfsys.

### (6.4.4.b) The typewriter model

Adding another layer of abstraction to working with sdfsys, I designed yet another text file based scripting system, and called it the typewriter model. The typewriter model is again based on the idea of a script being read one line at a time, but rather than parsing each line as fast as possible it will generate 'keyin' messages for the sdfsys text editor; the result is meant to give the impression text being typed, albeit at a mechanistic rate. The typewriter model also has syntax for sending messages directly to different parts of sdfsys and the meta-patch.

In the version of the typewriter model that is used for *five four three* the .txt file script is written with **coll** compatible formatting: that is that each line starts with an integer followed by a comma, and then ends with a semicolon. Writing, and editing, a script in this way reminded me of programming in BASIC,<sup>146</sup> and as such gives another opportunity to reflect on the history of computer music. Perhaps one day I will regress further to implement a hole-punch syntax, but in this project I did progress to improve the usability of the type writer syntax.

---

146 Although I did not go as far as to add a GOTO syntax method, there is syntax by which a line may cause the message of a different line to be parsed in its place.

(6.4.4.c) fivefourthree\_20120208.mov

(2 minutes 45 seconds)



Figure 6.28: fivefourthree\_20120208.mov

## 6.4.5 penhexsepoct

### (6.4.5.a) Scripted start

Adding a `text_to_coll` patcher to the typewriter model meant that scripts for that model could be written in `.txt` files without requiring line numbers to be manually maintained. The script saved as [penhexsepoct.txt](#) is written for the typewriter model and can be run by the implementation of that model in [penhexsepoct.maxpat](#); to run the `.txt` script, drag-and-drop the file from the operating system's file browser to the blue-bordered **dropfile** object at the top of [penhexsepoct.maxpat](#), reboot sdfsys, then click `go . . .` and watch the gl-ui.

The script is complete when the fourth group of points has been put (the gl-ui will look as in Figure 6.29). Performance of *penhexsepoct* then continues as a free exploration of the points that have been put.



Figure 6.29: penhexsepoct

**(6.4.5.b) penhexsepct\_20120427**

(2 minutes 45 seconds)

Audio recording from a performance of *penhexsepct* is provided – [20120427\\_0300.wav](#) – along with a .png (see Figure 6.28) to show the configuration of sdfsys, including the shape of the data being read as audio, when the recording was made. The rhythmic structure of the music in that recording could be recreated by programming sdfsys with the commands shown in that .png image.

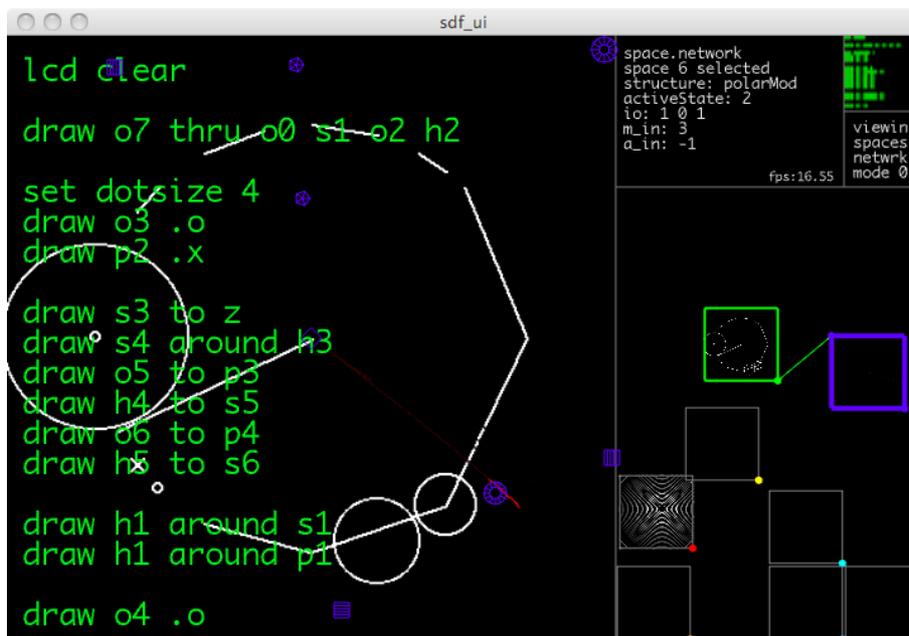


Figure 6.30: 20120427\_0311.png

## 6.4.6 more cogmos

### (6.4.6.a) Using rtuis

The typewriter model is based only on qwerty-like input to sdfsys, with some backdoor routes being used to do things like set active state values; it has no concept of the mouse, and scripts written with it, such as in *five four three*, would prompt the human to perform mouse actions for moving frames in the spaces network area to establish connections. It was after this that the rtuis aspect was added to sdfsys;<sup>147</sup> *more cogmos* began by exploring the use of the rtuis namespaces to automate the starting sections of a composition.

### (6.4.6.b) Recorded interaction

Using the rtuisrec sub-patch of *sdfsys\_beta*, the live coding of modules being loaded and configured in the spaces network area of the gl-ui was recorded. The recording can be seen in [rtuirec\\_cm00.txt](#) which can be read by the `seq~` object, and can be replayed – over approximately two minutes – to have sdfsys be configured the same again. That, however, was only the first step in the composition of *more cogmos*: the next step was to take only the necessary lines out of that .txt file and make a meta-patch based on them.

### (6.4.6.c) more\_cogmos.maxpat

Although it is clear from the appearance of this maxpat that it was a work in progress, it had fulfilled its first remit of automating the start-up sequence based on that rtuis recording. The composition, however, then took on a different trajectory.

---

<sup>147</sup> See §(6.3.9.a)

### (6.4.6.d) more\_cogmos.txt

A human targeted text score for *more cogmos* has been written and saved as [more\\_cogmos.txt](#), and a copy of it is presented here:

```
more cogmos
by samuel freeman, 2012--2013
using .maxpat files found in the sdfsys_beta1 folder
Mac OSX 10.6.8 and Max 5 recommended
begin
1. in the MaxMSP environment, switch on DSP
2. open sdf.sys_b133.maxpat
3. when ready, run more_cogmos.maxpat
4. wait whilst the spaces network is configured
5. when the text-buffer is cleared, the setup is complete
6. use thisis to draw a cog
7. use the shape of that cog as the basis for making sound
8. make a recording of the sound being made (either audio or audio-visual)
9. the recording can be split into sections, to be saved as separate files
10. if split into sections, then some sections can be deleted
11. if split into sections, then keep the original sequence of the sections
12. present the recording(s) with the date and time of when the recording was
made within the filename(s), using the format YYYYMMDD_HHMM
end
```

### (6.4.6.e) 20121120\_0040

Comprising three videos that are to be played in sequence:

- [20121120\\_0040\\_a.mov](#)
- [20121120\\_0040\\_b.mov](#)
- [20121120\\_0040\\_c.mov](#)

The videos are sections of a live screencast recording, recorded via QuickTime Player 10.0 (Apple); the originally recorded session was approximately 13 minutes duration, and the *sdfsys\_beta* performance had started some time before the recording began.

It is recommended that if one has time to experience this version of *more cogmos* more than once, then one should listen to the audio only version first. That way it is possible to hear the sound as sound in an acousmatic way. The audio-visual experience of this composition is quite different because the soundmaking process can be seen in full technical detail, and there is the human aspect of also being able to see the mouse pointer moving, which creates a different type of anticipation during the piece. The audio only version of those three parts in one file is included as:

- [20121120\\_0040\\_a\\_b\\_c.wav](#)
  - (9 minutes 34 seconds)

## 6.5 Evaluating sdfsys

Existing software did not provide the workflow that I required in order to continue making music with computers, so I made a software system that did and called it sdfsys. The preceding chapters of this thesis document the journey that led from questioning the basis for visual representation of sound in software to the inception of sdfsys, and this chapter has described its development, its present state, and how it has been used in composition.

### 6.5.1 Ready-at-hand

One of the reasons for creating sdfsys as a gl-ui fronted environment is that the visual appearance of a maxpat is always seen as an invitation to tweak, hack, and otherwise live code the software whilst musicking with it. Practice involving that type of interaction with software continues to be an important part of my art, and it is in that frame of mind that the module unit patches for sdfsys, the

meta-patches, and the workings of sdfsys itself are made. By establishing sdfsys as a non-Max-like environment, however, I have introduced a new and different way to interact with the processes that have been implemented as sdfsys module units. Context for this split perspective on creative praxis is found in a 2007 paper by ixi-software authors Thor Magnusson and Enrike Hurtado Mendieta. Martin Heidegger is cited<sup>148</sup> as a philosopher who 'talks about two cognitive or phenomenological modalities in the way we look at the world' (Magnusson & Mendieta, 2007, p. 98):

There is on the one hand the usage of tools, where the tools are ready-at-hand and are applied in a finely orchestrated way by the trained body, and, on the other hand, the moment when the tool breaks and it becomes present-at-hand, i.e. the user of the tool has to actively think what is wrong with the tool, and look at its mechanism from a new and critical stance. Heidegger takes the example of a carpenter who uses the hammer day in and day out without thinking about it. Only when the head falls off the hammer will the carpenter look at the hammer from a different perspective and see it in its true phenomenological light. As digital instruments/software are normally applied on the computer, we are more likely to have these phenomenological breaks. The tool becomes present-at-hand and we have to actively think about why something is not working or what would be the best way of applying a specific tool for a certain task. In fact, the way we use a computer and digital instruments is a constant oscillation between these two modes of being ready-at-hand and present-at-hand. We forget ourselves in working with the tool for a while, but suddenly we have to open or save a file, retrieve a stored setting, switch between plug-ins or instruments, zoom into some details, open a new window, shake the mouse to find the cursor, plug in the power cable when the battery is low [...]

That oscillation between ready-at-hand and present-at-hand cognitive modes has been in mind throughout much of this project, and sdfsys aims to provide a more consistently ready-at-hand experience than I perceive of MaxMSP in general.

In authoring this thesis, I also had that oscillation in mind when writing the term techno-aesthetic for the first time at §1.3; it (techno-aesthetics) has thereafter been found to be a subject of Gilbert Simondon's philosophical writing. To build on the start that has been made in this thesis, my intention is for future research-practice to explore these matters further in relation to computer music.

---

148 Heidegger, Martin. *Being and Time*. Oxford: Blackwell Publishers, 1995. Pp. 102-122.

## 6.5.2 What's next for sdfsys?

With the publication of this thesis, the software described will also be made public, and I will continue to develop sdfsys; perhaps as `_beta2` in the Max 6 environment at first, but the way that windows and fullscreen are managed in Max 6 is an obstacle to my workflow which could speed migration to a completely new implementation of sdfsys. For a long time I have wondered how different the sdfsys environment would become if it were implemented through something other than MaxMSP, and I have speculatively began exploring possible alternatives. I expect eventually to continue version naming in sequence through the Greek alphabet, meaning that the next significant redesign of the implementation would be *sdfsys\_gamma*.

# 7: Conclusion

## 7.1 Framing the project

My objective has been to document the creative process of developing an approach to computer musicking that is both broadly aware of its technological heritage, and specifically focused on the visual interface between human mind and soundmaking DSP algorithms.

This thesis has described and contextualised my compositional responses to questions about representing aspects of sound visually in interactive systems. The project began by asking the following primary research question which has been found to relate strongly to the concept of *technē*: How might the ways in which sound is visually represented influence the choices that are made while those representations are being manipulated and organised as music? Additional questions, indicated at the initial stages of the project (§1), were: Which aspects of sound are represented visually, and how are those aspects shown?

Taking those three questions as the starting point for investigation and compositional

programming towards new works of art, the project did not seek direct answers in a quantifiable sense, but instead pursued a techno-aesthetic dialogue which eventually took forms that could not have been predicted at the outset.

Already established at the start of this project were the bases of the spiroid-frequency-space concept, which had emerged through my practice of sketching on paper in the process of composition, and the scripted drawing system called thisis, which I had been developing to assist in that practice. Having grown weary of rectilinear representations and overt skeumorphism in computer music software I looked to the circular form of the spiroid, and the abstract geometry of thisis, as foundations to an alternative paradigm.

## 7.2 Contextual grounding

Recognising sound as a psychophysical phenomenon, the aim of §2 was to identify which fundamental aspects of sound were important to me in the context of music. Looking at how physical aspects, and abstracted representations psychological aspects, of sound can manifest visually, the following themes were highlighted: the sense of ever-present 'now' of Cymatics (in which one can see patterns being made whilst sound is happening); the concept of a recording sound as a temporal trace (in which one can see the patterns made after sound has happened); that the harmonograph is interesting (tracing harmonic interactions at a rate comprehensible to the human eye); and that pitch is a morphophoric medium, as is visual space (each being a medium in which forms retain identity through transformations), but colour is not a morphophoric medium (and herein lies the basis for what I call the chroma paradox).

Many of the works in the portfolio are conceptually grounded by aspects of live coding. As the project progressed this contemporary research-field was found to have wider reaching significance in understanding software as the medium of computer music than may have been

anticipated at the outset.

## 7.3 Techno-aesthetic development

The six study works that are described in §3 seek in different ways to understand better what visual representations in software are, and how they are being used. Starting from the recognition that the computer screen is but a matrix of pixels, works such as *visyn*, *sub synth amp map*, and the two gramophone model examples are all based on the manipulation of data-matrix cell values that are taken for display on screen. The connections in these works between aspects of sound and the data-matrices displayed on screen vary:

- for *visyn* the synthesised visual forms are abstract, while sound is made from the data of the visual display (limited success with this approach, but there is potential for further work if more computationally efficient methods were applied);
- in *sub synth amp map* the amplitudes of (any) two digital audio signals are mapped to spatial coordinates that create temporal traces on screen for direct visualisation of sound as motion (think Lissajous, lamp light, laser beams, and laptop screens);
- the gramophone model encompasses two types of data-matrix display:
  - the temporal trace type of display (as in *sub synth amp map*) is used again, in this case to show where a modelled stylus is now or has just been in the space; and
  - the other way that data-matrix cells are used is as memory for audio sample data-values in the modelling of the gramophone record surface (arranging a stream of sample values as a spiral path on the plane);

Of these three approaches to linking the screen and sound it was with aspects of the gramophone model that I found the most creative promise. Having developed the gramophone

model as a speculative curiosity, its key features would later become an integral part of sdfsys as the tildegraph concept.

## 7.4 Tildegraphing

Whereas it is sound input to *sub synth amp map* that makes patterns on screen,<sup>149</sup> the tildegraph concept, that eventually emerged during the project, allows parametric control of the visual patterns on screen which can then interact to make sound. This reversal of roles (sound before screen / screen before sound) was indicated as an objective, quite early in the project, in relation to Cymatics (see end of §2.1.1). In further work to extend that idea, one could implement a hybrid system to explore the interaction of those roles by using the sdf.sys.dv module with a digital video camera focussed on a Cymatic setup in order to take patterns generated by sound (in the physical world) into (the virtual world of) sdfsys; one could then use a ~g.r. module to make sound by reading data from the video matrix. Such a configuration would be similar the to Tüb installation (Erlach et al., 2011) described at §(6.2.6.c). In the hypothesised configuration a feedback loop between the physical world and sdfsys via audio and visual domains could be created by letting the sound input to the Cymatic system be taken from the output of the tildegraphing in sdfsys.

Exploration of tildegraph concept in sdfsys, as a synthesis technique founded on techno-aesthetic concerns, presents fertile ground for further research, both within sdfsys and with the potential for alternative implementations. The tildegraph concept thus stands as a significant contribution of this practice-based research.

## 7.5 Music via visual representations

In the act of controlling tildegraph modules in sdfsys there is a negotiation of cognitive balance

<sup>149</sup> A system that promotes close consideration of the primary research question because the visual manifestations in that piece certainly affect the choices that are made while controlling the input sound.

between targeting the visual and the audio consequences of parametric input; this leads to a particular sense of audio-visual synergy that is (seemingly) unique to the *sdfsys* environment.

Many of the parameters used in *tildegraph* modules for *sdfsys* are to set frequencies within their spatiotemporal algorithms, and a *spiroid*-frequency-space mapping is used to translate the location of a Hz type sprite in the main area of the *sdfsys* gl-ui to a frequency value.

The *spiroid* concept has its own chapter (§4) in which its genesis within my own practice is explained before the concept is contextualised as an archetypal form of visual representation. It is noted that although *spiroid*-like spirals are common in print, they are rare on screen; or at least they were: recent and forthcoming audio analysis softwares have been cited that use what I call the *spiroid*-frequency-space. In future work I hope to promote the use of this concept for input (of parameter values) as well as output (of spectral analyses on screen). My work on the *spiroid* concept within this project contributes both in the provision of a collated contextualisation, and also in the giving of that specific name which was formed after two observations: one, that a perfectly good word (*spiroid*) seemed not to be used much in contemporary language, and then also that there is a profoundly beautiful concept in the visual representation of sound that was both related to that word, and also not being used as much as perhaps it could be.<sup>150</sup>

Pursuing other circular representations to complement the *spiroid*-frequency-space in software for new musicking, and employing the *thisis* system for drawing, I developed the *CirSeq* time-space; this, and my explorations with it, are described in §5. Some elements of the *CirSeq* development that were not covered in this commentary include the work on *cirseq\_beta* (in the form of unit modules for *sdfsys\_beta*), and the exploration (on paper) of a triangular formation following the same harmonic series of systematic quantization maintaining angle-to-phase and radius-to-temporal-resolution correlations. Inquiries related to *CirSeq* could benefit further research

---

<sup>150</sup> Though that description puts the cart before the horse, somewhat, by putting *spiroid* before the frequency-space.

into new interfaces for musical expression.

The *saw~onepole~noise~click~* is a composition that connects to many of the themes that have emerged during this project whilst, of itself, taking a different approach to visual representation of sound. Bringing attention to the visual medium of source code in software, *saw~onepole~noise~click~* is composed in the form of a single maxpat such that nothing is hidden, as shown in Figure 3.37. The visual representation of soundmaking processes in the maxpat source code for this piece is both complete and human readable: the prescribed timing, timbre, and temporal processing of sonic events are all presented in full technical detail, including those aspects of the piece that are indeterminate. In this case the software is most definitely a score for the piece of music, but it is also the instrument used for, and the player of, that score.

As well as being derivative of a live coding exercise, *saw~onepole~noise~click~* is contextualised as an aesthetic exploration of the dichotomy of time-domain and frequency-domain thinking which is a common conceptual basis in computer music. Discussion both of *saw~onepole~noise~click~* (§3.6), and of *CirSeq Zero* (§5.3–§5.4) have shown that the distinction between those two domains is a matter of perspective and choice. The CirSeq (time-space) concept was made to be used in tandem with the spiroid (frequency-space) concept for controlling parameters of sound and making music, but I have subsequently found that the spiroid can be enough on its own to control both time- and frequency-domain aspects of soundmaking as durational elements of form.

## 7.6 The sdfsys environment

It is always 'now' when one is working in the sdfsys environment; there is no 'time-line' representation – unless one chooses to classify the spiroid under that term, but spiroid curve is a line that represents a continuum of durations, rather than representing the continuation of time within a

duration – there is also no undo feature in sdfsys, and very limited save and recall functionality. It can thus be described as a live coding environment that has been explored compositionally.

Different approaches to composing with sdfsys have been described (§6.4) in the presentation of *orbit8 reading cyclephase*, *febandflow*, *five four three*, *penhexsepoct*, and *more cogmos*, and many more approaches as possible.

With the tildegraph concept in sdfsys, one is able (for example) to use a combination of near-miss harmonic intervals when specifying frequencies to establish a slowly shifting timbral soundscape that may take minutes to evolve without requiring any further input control. The human is then free to enjoy the audiovisual experience with the option to affect further control over the soundmaking processes. There is also the dimension of geometrical drawing via thisis commands which may or may not also become part of the soundmaking, and promotes full-minded engagement with the present moment in a temporal process of creating form. To my knowledge, this workflow is a unique contribution to the art of computer music.

The sdfsys environment is an open ended techno-aesthetic entity that brings together a wide range of historically aware computer music concepts and techniques.

# Detailed Table of Contents

Abstract	3
Overview of the Portfolio Directories	5
1: Introduction	6
1.1 Motivation	6
1.2 Initial context	7
1.2.1 Representations of sound	7
1.2.2 Computer music software	10
1.3 Thesis structure	11
2: Looking at sound	13
2.1 Looking at physical aspects	13
2.1.1 Physical patterns	13
2.1.2 The temporal trace	17
2.1.3 Harmonic interactions	18
2.2 Looking at psychological aspects	22
2.2.1 Morphophoric mediums and (in)dispensability	22
2.2.2 Paradox and pitch perception	24
2.3 Looking at looking at sound	27
3: In software, on screen	28
3.1 Pixels	29
3.1.1 The sound of electricity	30
3.1.2 System architecture	30
3.1.3 Audience interaction	33
3.1.4 Inherent paradox	35
3.1.5 Towards programming with matrices	36
3.2 Visyn	38
3.2.1 Introducing visyn	38
3.2.2 Looking at visyn	39
3.2.3 Visual data to control sound parameters	40
3.2.4 Ever-changing and always the same	41
3.3 Sub synth amp map	42
3.3.1 Live Coding with MaxMSP	43
3.3.2 From passing thoughts to pivotal theories	45
3.3.3 Names and numbers	46
3.3.4 Looking at sub synth amp map	47
3.3.5 Three layers, three mappings	51
3.3.6 Input selections	54
3.3.7 Lamp-light, Laser beams, and Laptop screens	55
(3.3.7.a) Lissajous figures	55
(3.3.7.b) Lamp-light	57
(3.3.7.c) Laser beams	57
(3.3.7.d) Projected oscilloscopes	59
(3.3.7.e) Laptop screens	60
3.3.8 Connections and comparisons between study pieces	61
3.4 Conceptual model of a gramophone	63
3.4.1 Analysis of motion	64
3.4.2 A spiral on the surface of a plane	66

3.4.3 Stylus abstraction	68
3.4.4 Gramophone_002_a	69
3.4.5 Interacting with _002_a: limitations and inspirations	75
3.5 Gramophone_005g	76
3.5.1 Sequence generator	78
(3.5.1.a) List generation	79
(3.5.1.b) List value reordering	81
(3.5.1.c) Layers of abstraction	82
3.5.2 Manifestation of g005g	83
3.5.3 Audiovisual output: open outcome and recorded examples	88
3.5.4 Shapes in time: using rectilinear views of recorded audio output	91
(3.5.4.a) Difference (and the samenesses within)	91
(3.5.4.b) Sameness (and the differences within)	94
(3.5.4.c) Pre- and post-DSP	95
3.5.5 Contextualisation of the sound	97
3.6 saw~onpole~noise~click~	101
3.6.1 saw~onpole~noise~click~b2.maxpat	101
3.6.2 The domain dichotomy	104
3.6.3 Temporal structure	106
3.6.4 Sounding events	108
(3.6.4.a) Triggering sound	108
(3.6.4.b) Synthesis flow	108
3.6.5 Scored recordings	109
3.7 Software as substance	111
3.7.1 One-line programs	111
3.7.2 Expressivity and technē in programming and composition	114
3.7.3 Composing as an artist-programmer	116
(3.7.3.a) Bricolage programming	117
(3.7.3.b) Composition model	118
(3.7.3.c) Comparison of cyclical models	120
3.7.4 On the context of text in programming	121
(3.7.4.a) Programming with text	121
(3.7.4.b) Source code text as a layer of abstraction	122
3.7.5 The medium is the meta-medium is the tool is the message	124
3.7.6 In software, on screen: programming/composition	127
4: Spiroid	128
4.1 Spiroid	128
4.1.1 Circularity	129
4.1.2 Radially extending	133
4.1.3 On Spirals	135
4.1.4 Curved continuum	136
4.2 Some similar spirals	137
4.2.1 In the inner ear	137
4.2.2 In print	138
4.2.3 Photosounder Spiral	142
4.3 Spiroid display of partials as arcs on an lcd in MaxMSP	144
4.3.1 Analysis and display	145
4.3.2 SpiroidArc	146

(4.3.2.a) a_demo_of_SpiroidArc300	146
(4.3.2.b) Histogram of strongest pitch-classes	148
(4.3.2.c) Background option to show 12TET angles	150
4.3.3 Chroma paradox	150
(4.3.3.a) (chroma $\neq$ chroma)	150
(4.3.3.b) Double jeopardy	153
(4.3.3.c) Suggested remedy	154
4.4 Spiroid as an interface for input	154
4.4.1 nRadii	155
(4.4.1.a) nRadii_sawOsc overview	155
(4.4.1.b) Construction of the nRadii GUI	157
(4.4.1.c) Angular manifestation	158
(4.4.1.d) A topic for NIME	159
4.4.2 A spiroid widget design	160
(4.4.2.a) A type of dial that is based on the spiroid	160
(4.4.2.b) Mouse interaction with a spiroidial	160
(4.4.2.c) Specific implementation of a concept archetype	162
4.4.3 SpiroidArc plus sequencing	163
(4.4.3.a) Frequency markers	163
(4.4.3.b) Sequencing with markers in a time-space	163
(4.4.3.c) _spiroidArc+timespace annotation	164
(4.4.3.d) Evaluation	167
4.5 Assessment of the spiroid	168
4.5.1 Recapitulation	168
4.5.2 Spiroid beyond this project	169
4.5.3 Chapter conclusion	170
5: CirSeq	172
5.1 CirSeq and thisis	173
5.2 CirSeq inception	175
5.2.1 Basis for the search	175
5.2.2 A circle on a cartesian plane	175
5.2.3 Another circle on the plane	176
5.2.4 Double and double again	177
5.2.5 Common time	178
5.3 CirSeq Zero	182
5.3.1 Playing (with) the piece: an overview of CirSeq Zero	183
(5.3.1.a) The CirSeq Zero patch	183
(5.3.1.b) CirSeq Zero patch annotation	184
5.3.2 Technical structure	188
(5.3.2.a) Original documentation	188
(5.3.2.b) CirSeq Zero structure	190
(5.3.2.c) Methods for control	191
5.3.3 Making decisions to organise the soundmaking	193
(5.3.3.a) Similar concepts at different layers of abstraction	193
(5.3.3.b) An additional aspect to the piece	194
(5.3.3.c) The cascade of randomised decisions	195
(5.3.3.d) To unravel the cascade	196
(5.3.3.e) Deciding the duration of the cycle period	198

(5.3.3.f) Contextualising the blurring of lines on the time-domain continuum	200
(5.3.3.g) Running the cascade with pitch-like durations of the cycle period	202
5.3.4 Meta-patch: an example of external control	203
(5.3.4.a) Annotating the meta-patch	204
(5.3.4.b) Exploring the effects of aliasing	205
(5.3.4.c) Explorations of soundmaking guided by shape	206
5.4 Aesthetic analogies in the exploration of CirSeq patterns	208
5.4.1 Exploration of the continuum – further detail	209
5.4.2 Observation of a specific configuration	211
5.5 Critique and chapter conclusion	212
5.5.1 What you see is not always what you get	212
5.5.2 Remedy in moving on	214
5.5.3 Fractal potential	215
5.5.4 Circle the square	216
6: Sdfsys	218
6.1 Introducing sdfsys	218
6.1.1 Overview	218
6.1.2 Construction aesthetics	220
6.1.3 Eight spaces	220
6.1.4 Making sound	221
6.2 Alpha	221
6.2.1 sdf.sys_alpha	221
6.2.2 Motivating factors for sdf.sys_alpha	222
6.2.3 The gl-ui	222
(6.2.3.a) Decisions in the gl-ui design	222
(6.2.3.b) Divisions within the gl-ui	224
6.2.4 Interacting with sdf.sys_alpha	226
(6.2.4.a) Rendering the gl-ui and remapping the mouse position values	226
(6.2.4.b) Frames in the sidebar	226
(6.2.4.c) sdf.sys.txt	227
(6.2.4.d) A video walkthrough: commands and parameters in sdf.sys_alpha	229
6.2.5 Soundmaking in context	233
(6.2.5.a) Colourisation: Looking at bipolar audio data-matrices on screen	233
(6.2.5.b) Tildegraph: from a module to a concept	234
6.2.6 The tildegraph concept	235
(6.2.6.a) Seeing sound	235
(6.2.6.b) The tildegraph concept in the context of wave terrain synthesis	236
(6.2.6.c) Wave terrain and scanned synthesis	239
6.2.7 Concluding the alpha build	240
6.3 Beta	241
6.3.1 sdfsys_beta	241
6.3.2 The beta design for the gl-ui	242
6.3.3 Colour in sdfsys_beta	245
6.3.4 _Area_1_Main	246
(6.3.4.a) Mousing data	246
(6.3.4.b) Eight spaces	246
(6.3.4.c) Parameter-sprites	248
6.3.5 _Area_2_SpacesNetwork	249

(6.3.5.a) Visual patching	249
(6.3.5.b) Eight colours for the eight spaces	251
(6.3.5.c) Making connections in the spaces network	251
6.3.6 Area_3_Box	252
6.3.7 Area_4_SystemToggle	255
6.3.8 Area_5_TextToggle	255
(6.3.8.a) To toggle the text	255
(6.3.8.b) Editing the text-buffer	256
(6.3.8.c) Syntax in sdfsys_beta	256
6.3.9 Infrastructures and Meta-patches	257
(6.3.9.a) sdf.sys.rtuis	257
(6.3.9.b) sdf.sys.network	258
(6.3.9.c) sdf.sys.reboot	258
(6.3.9.d) Loading other meta-patches	259
(6.3.9.e) The sdfsys_beta_xnn_video series	261
6.4 Composing with sdfsys	261
6.4.1 A composition with which to compose	261
(6.4.1.a) sdfsys as substance	261
(6.4.1.b) In the context of my compositional model	262
(6.4.1.c) On paper and on camera	263
(6.4.1.d) Introducing the included compositions	263
6.4.2 orbit8 reading cyclephase	264
(6.4.2.a) 20120221_0200_cyclephase_orbit8	264
(6.4.2.b) 20120222_orbit8cyclephase	265
(6.4.2.c) ~g.r.orbit8	266
6.4.3 febandflow	266
6.4.4 five four three	267
(6.4.4.a) Multiple layers of abstraction	267
(6.4.4.b) The typewriter model	268
(6.4.4.c) fivefourthree_20120208.mov	269
6.4.5 penhexsepoct	270
(6.4.5.a) Scripted start	270
(6.4.5.b) penhexsepoct_20120427	271
6.4.6 more cogmos	272
(6.4.6.a) Using rtuis	272
(6.4.6.b) Recorded interaction	272
(6.4.6.c) more_cogmos.maxpat	272
(6.4.6.d) more_cogmos.txt	273
(6.4.6.e) 20121120_0040	273
6.5 Evaluating sdfsys	274
6.5.1 Ready-at-hand	274
6.5.2 What's next for sdfsys?	276
7: Conclusion	276
7.1 Framing the project	276
7.2 Contextual grounding	277
7.3 Techno-aesthetic development	278
7.4 Tildegraphing	279
7.5 Music via visual representations	279

7.6 The sdfsys environment	281
Detailed Table of Contents	283
Table of Figures	288
Bibliography	291
Total (inclusive) word count: 74810	

## Table of Figures

Figure 2.1: Chladni patterns	15
Figure 2.2: Fig. 1 and Plate V.(B) from 'Harmonic Vibrations and Vibration Figures' (Newton, 1909)	19
Figure 2.3: Benham's questionable table of harmonic intervals (Newton, 1909, p.36)	21
Figure 2.4: Penrose stairs optical illusion	24
Figure 3.1: MoE Pixels	32
Figure 3.2: MoE Pixels, image from Baldwin (2009)	34
Figure 3.3: visyn_10e	39
Figure 3.4: subsynthampmap_matrix_examples_4x2b.png	47
Figure 3.5: The 'sub synth' patch	48
Figure 3.6: The 'amp map' patch	49
Figure 3.7: The 'matrix clear metro' patch	50
Figure 3.8: 1.jpg	51
Figure 3.9: subsynthampmap_matrix_examples_2x4.png	53
Figure 3.10: 20100221_sfplay_random_flow	55
Figure 3.11: Four Lissajous	56
Figure 3.12: Lissajous mirrors	57
Figure 3.13: Visyn and sub synth amp map flow	62
Figure 3.14: Comparative poke flow	63
Figure 3.15: Vinyl Killer	65
Figure 3.16: Stylus abstraction flow	69
Figure 3.17: gramophone_002_a flow	70
Figure 3.18: g002a_x1.mov (at 39 sec)	72
Figure 3.19: g002a_x2.mov (at 3 min 16 sec)	73
Figure 3.20: g002a_x2.mov (at 4 min 58 sec)	74
Figure 3.21: set_g-mult_seq	79
Figure 3.22: g005g_reordering_paramVal	81
Figure 3.23: g005g_reordering_transDur	82
Figure 3.24: g005g_trigger flow	84
Figure 3.25: jitnoiseexpr	85
Figure 3.26: g005g_dsp	87
Figure 3.27: g005g_x1-x4(spectroLog)	91
Figure 3.28: g005g_x1-x4(spectroLog)mid7sec	93
Figure 3.29: g005g_x1-x4(wave)mid7sec	93
Figure 3.30: g005g_x5ab(spectroLog)	94
Figure 3.31: g005g_x6abcd(wave)4sec	95
Figure 3.32: g005g_x7(b+n)46-53(spect)	97
Figure 3.33: g005g_x7(b+n)46-53(wave)	97
Figure 3.34: phonicMX881_a	100
Figure 3.35: phonicMX881_f	100
Figure 3.36: phonicMX881_g	101

Figure 3.37: saw~onpole~noise~click~b2	103
Figure 3.38: saw~onpole~noise~click~(domainConcept)	105
Figure 3.39: saw~onpole~noise~click~(synthFlow)	109
Figure 3.40: saw~onpole~noise~click~b_midi_x	110
Figure 3.41: McLean_bricolageprogramming	117
Figure 3.42: CompositionModel	119
Figure 3.43: model of composition + model of bricolage programming	120
Figure 3.44: JacquardCards	122
Figure 3.45: layers of abstraction	123
Figure 4.1: Showing six pitch-classes	130
Figure 4.2: tone clock hours	131
Figure 4.3: BensonSymmetryEx3.maxpat	133
Figure 4.4: TwelveTones_ViHart	133
Figure 4.5: Three octaves of six pitch-classes (simple)	134
Figure 4.6: Sketch of the spiroid archetype	135
Figure 4.7: Storaasli Pat.5127056	139
Figure 4.8: chromatic pitch space	140
Figure 4.9: Showing the progression of different intervals	141
Figure 4.10: Photosounder Spiral CM	142
Figure 4.11: spiroid arc display flow	145
Figure 4.12: spiroidArc300demo1	147
Figure 4.13: spiroidArc_histoFlow	149
Figure 4.14: spiroidArc300demo2	149
Figure 4.15: spiroidArc300demo3	150
Figure 4.16: hsl_swatch	151
Figure 4.17: RGB_cube	152
Figure 4.18: 20110221_0230	154
Figure 4.19: Smule Magic Piano	155
Figure 4.20: nRadii_sawOsc.maxpat	156
Figure 4.21: nRadii_sawOsc.mm	156
Figure 4.22: theta_to_midi	158
Figure 4.23: Kessous2002Mapping	159
Figure 4.24: spiroidial_mouse_interaction	162
Figure 4.25: _spiroidArc+timespace.maxpat (annotated)	165
Figure 4.26: SnailAnalyzer	170
Figure 5.1: cirseq_thisis_01	176
Figure 5.2: cirseq_thisis_02	177
Figure 5.3: cirseq_thisis_03	178
Figure 5.4: 4n8n16n32nNotes	179
Figure 5.5: RhythmCircle	180
Figure 5.6: bComposerRhythm	180
Figure 5.7: Loopseque_Wheel	181
Figure 5.8: AlphaLive	182
Figure 5.9: CirSeq_Zero_0a	184
Figure 5.10: masterPhase	185
Figure 5.11: CirSeq_Zero_1	186
Figure 5.12: cirSeq_021mm	189
Figure 5.13: CirSeq_Zero_flow	190

Figure 5.14: cSeqQsys_flow	191
Figure 5.15: csqsMessages	193
Figure 5.16: CirSeq_Zero_nonpres	194
Figure 5.17: CirSeq_Zero_nonpres_unlock	194
Figure 5.18: random_decide_cascade	197
Figure 5.19: sono_example	199
Figure 5.20: 2000bpm	200
Figure 5.21: sono_two_up	203
Figure 5.22: cs_eoc	205
Figure 5.23: dodecagon	207
Figure 5.24: phase_shapes	208
Figure 5.25: 4of_the_phase_shapes	210
Figure 5.26: cSeqQsys_synth	214
Figure 5.27: cirSeq_alpha_preview	216
Figure 6.1: sdf.sys_alpha_mm	221
Figure 6.2: sdf.sys_alpha ui layout	225
Figure 6.3: showing_part_of_sdf.sys.txt_alpha	228
Figure 6.4: sdfsys9min2013a_1m5s	231
Figure 6.5: sdfsys9min2013a_2m12s	231
Figure 6.6: sdfsys9min2013a_4m16s	232
Figure 6.7: sdfsys9min2013a_7m	233
Figure 6.8: showing [p bipolarBlueViewMode] in sdf.sys.8spaces.ui_link.maxpat	234
Figure 6.9: SdfSysAlphaStructure	240
Figure 6.10: sdfsys_b133	241
Figure 6.11: sdfsys_beta ui layout	242
Figure 6.12: sdf_ui at load in sdfsys_b133	242
Figure 6.13: sdfsys_b1_ui_ratios	243
Figure 6.14: [p sdfsys]	244
Figure 6.15: matrix_view_type	245
Figure 6.16: main area	246
Figure 6.17: space_structure_flow	247
Figure 6.18: spaces network area	249
Figure 6.19: _Area_2_SpacesNetwork_flow	250
Figure 6.20: box area	252
Figure 6.21: _Area_3_Box_flow	254
Figure 6.22: system toggle	255
Figure 6.23: text toggle	255
Figure 6.24: controlp after loading seven of the spaces	260
Figure 6.25: compositionModel_20100804	262
Figure 6.26: 20120222_orbit8cyclephase.mov	265
Figure 6.27: febandflow_jitfftcolourtest06	267
Figure 6.28: fivefourthree_20120208.mov	269
Figure 6.29: penhexsepoct	270
Figure 6.30: 20120427_0311.png	271

Figure count: 145

## Bibliography

- Adobe (2009) ‘Adobe - Flash Player Statistics’, [online] Available from: [http://www.adobe.com/products/player\\_census/flashplayer](http://www.adobe.com/products/player_census/flashplayer).
- Agostini, A. and Ghisi, D. (2012) ‘Bach: An Environment for Computer-Aided Composition in Max’, In *Proceedings of the International Computer Music Conference 2012*, [online] Available from: <http://quod.lib.umich.edu/cgi/p/pod/dod-idx/bach-an-environment-for-computer-aided-composition-in-max.pdf?c=icmc;idno=bbp2372.2012.068> (Accessed 3 August 2013).
- Anon (1995) ‘Acoustical Society of America - Circularity in Pitch Judgement’, [online] Available from: <http://asa.aip.org/demo27.html> (Accessed 10 March 2012).
- Anon (2013) ‘Double Rainbow (viral video)’, *Wikipedia, the free encyclopedia*, [online] Available from: [http://en.wikipedia.org/w/index.php?title=Double\\_Rainbow\\_\(viral\\_video\)&oldid=579260099](http://en.wikipedia.org/w/index.php?title=Double_Rainbow_(viral_video)&oldid=579260099) (Accessed 30 October 2013).
- Apple (n.d.) ‘Final Cut Pro 7 User Manual: Frame Dimensions, Number of Lines, and Resolution’, [online] Available from: <http://documentation.apple.com/en/finalcutpro/usermanual/index.html#chapter=C%26section=6> (Accessed 9 December 2013a).
- Apple (2010) ‘iPad Available in US on April 3’, *Apple - Press Info - iPad Available in US on April 3*, Press Release, [online] Available from: <http://www.apple.com/pr/library/2010/03/05iPad-Available-in-US-on-April-3.html> (Accessed 3 August 2013).
- Apple (n.d.) *QuickTime*, [online] Available from: <http://www.apple.com/uk/quicktime/> (Accessed 5 August 2013b).
- Ashton, A. (2003) *Harmonograph: a visual guide to the mathematics of music*, Glastonbury, Wooden Books.
- Barrow, J. D. (1993) *Pi in the Sky: Counting, Thinking and Being*, London, Penguin Books.
- Bartetzki, A. (2011) ‘A Show Case for SC Tweets’, [online] Available from: <http://www.bartetzki.de/en/sctweets.html> (Accessed 4 July 2012).
- Benson, D. J. (2007) *Music: A Mathematical Offering*, Cambridge, Cambridge University Press.
- Bertin-Mahieux, T., Weiss, R. J. and Ellis, D. P. W. (2010) ‘Clustering beat-chroma patterns in a large music database’, In *Proceedings of the 11th International Society for Music Information Retrieval Conference, August 9-13, 2010, Utrecht, Netherlands*, [online] Available from: [http://academiccommons.columbia.edu/download/fedora\\_content/download/ac:148414/CO-NTENT/BertWE10-patterns.pdf](http://academiccommons.columbia.edu/download/fedora_content/download/ac:148414/CO-NTENT/BertWE10-patterns.pdf) (Accessed 24 March 2013).

- Blackburn, S. (2008) *The Oxford Dictionary of Philosophy*, Oxford University Press, [online] Available from: <http://www.oxfordreference.com/view/10.1093/acref/9780199541430.001.0001/acref-9780199541430> (Accessed 8 September 2013).
- Blackwell, T. and Young, M. (2006) 'Live Algorithms for Music Manifesto', [online] Available from: <http://igor.gold.ac.uk/~mas01tb/papers/AISB.pdf> (Accessed 13 December 2006).
- Bókkon, I. and Salari, V. (2012) 'Hypothesis about brilliant lights by bioluminescent photons in near death experiences', *Medical Hypotheses*, 79(1), pp. 47–49.
- Bokowiec, M. A. (2011) 'V'oct (ritual): An interactive vocal work for bodycoder system and 8 channel spatialization', In *Proceedings of the International Conference on New Interfaces for Musical Expression*, [online] Available from: [http://www.nime.org/proceedings/2011/nime2011\\_040.pdf](http://www.nime.org/proceedings/2011/nime2011_040.pdf) (Accessed 7 August 2013).
- Bond, D. (2011) 'The Harmonic Matrix: Exploring The Geometry Of Pitch', In *Proceedings of International Computer Music Conference*, pp. 288–291, [online] Available from: <http://quod.lib.umich.edu/cgi/p/pod/dod-idx/harmonic-matrix-exploring-the-geometry-of-pitch.pdf?c=icmc;idno=bbp2372.2011.057> (Accessed 25 September 2013).
- Borgonovo, A. and Haus, G. (1986) 'Sound Synthesis by Means of Two-Variable Functions: Experimental Criteria and Results', *Computer Music Journal*, 10(3), p. 57.
- Boulez, P. (1978) 'Technology and the Composer', *Leonardo*, 11(1), pp. 59–62.
- Bourke, P. (2008) 'Harmonograph', [online] Available from: <http://paulbourke.net/geometry/harmonograph/> (Accessed 6 March 2012).
- Bown, O., Eldridge, A. and McCormack, J. (2009) 'Understanding Interaction in Contemporary Digital Music: from instruments to behavioural objects', *Organised Sound*, 14(02), pp. 188–196.
- Bressloff, P. C., Cowan, J. D., Golubitsky, M., Thomas, P. J. and Wiener, M. C. (2002) 'What geometric visual hallucinations tell us about the visual cortex', *Neural Computation*, 14(3), pp. 473–491.
- Brümmer, L. (2008) 'Stockhausen on Electronics, 2004', *Computer Music Journal*, 32(4), pp. 10–16.
- Casual Underground (n.d.) 'Loopseque | See music, touch music, be music.', [online] Available from: <http://loopseque.com/> (Accessed 4 August 2010).
- Chandler, D. and Munday, R. (2011) 'NTSC', *A Dictionary of Media and Communication*, Oxford University Press, [online] Available from: <http://www.oxfordreference.com/view/10.1093/acref/9780199568758.001.0001/acref-9780199568758-e-1889?rskey=wZULSW&result=1> (Accessed 9 December 2013).
- Cheng, K., Cheng, V. and Zou, C.-H. (2008) 'A Logarithmic Spiral Function to Plot a Cochleaogram', *Trends in Medical Research*, 3(1), pp. 36–40.

- Chion, M. (1983) *GUIDE DES OBJETS SONORES Pierre Schaeffer et la recherche musicale*, English translation, 2009. Paris, Institut National & De L'Audiovisuel & Éditions Buchet/Chastel, [online] Available from: [http://www.ears.dmu.ac.uk/spip.php?page=articleEars&id\\_article=3597](http://www.ears.dmu.ac.uk/spip.php?page=articleEars&id_article=3597) (Accessed 3 March 2013).
- Chladni, E. F. F. (1787) *Entdeckungen über die Theorie des Klanges*, [online] Available from: <http://echo.mpiwg-berlin.mpg.de/MPIWG:EKGK1SP1> (Accessed 17 February 2013).
- Christensson, P. (n.d.) 'VGA (Video Graphics Array) Definition', *TechTerms.com*, [online] Available from: <http://www.techterms.com/definition/vga> (Accessed 5 May 2013).
- Clarke, M. (1998) 'Extending Contacts: The Concept of Unity in Computer Music', *Perspectives of New Music*, 36(1), pp. 221–246.
- Collins, N. (2012) 'Acoustics', *Leonardo Music Journal*, 22, pp. 1–2.
- Collins, N. (2006) *Handmade Electronic Music: The Art of Hardware Hacking*, Abingdon and New York, Routledge.
- Colman, A. M. (2009) *A Dictionary of Psychology*, Oxford University Press, [online] Available from: <http://www.oxfordreference.com/view/10.1093/acref/9780199534067.001.0001/acref-9780199534067> (Accessed 24 September 2013).
- Computer Music (2013) 'Our next CM Plugin, Spiral ...', *@computermusicuk*, microblog, [online] Available from: <https://twitter.com/computermusicuk/status/364672877632565248> (Accessed 6 August 2013).
- Cook, P. R. (ed.) (1999) *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics*, Cambridge, MA, The MIT press.
- Cross, L. (2005) 'Remembering David Tudor: A 75th Anniversary Memoir (1968-1969)', [online] Available from: <http://www.lowellcross.com/articles/tudor/1968-1969.html> (Accessed 12 May 2013).
- Cross, L. (1981) 'The Audio Control of Laser Displays', *db, The Sound Engineering Magazine*, 15(7), pp. 30, 34, 38–41.
- Cycling '74 (n.d.) *MaxMSP*, [online] Available from: <http://cycling74.com/products/max/> (Accessed 13 March 2013).
- Dust-to-Digital (2012) 'Pictures of Sound: One Thousand Years of Educated Audio: 980-1980', [online] Available from: <http://www.dust-digital.com/feaster/> (Accessed 10 January 2013).
- Erlach, B., Evans, M. and Wilson, M. J. (2011) 'Tüb-Interactive Sonification of Water Waves', In *Proceedings of the International Conference on New Interfaces for Musical Expression: 30 May – 1 June 2011*, Oslo, Norway, [online] Available from: <https://ccrma.stanford.edu/~mwilson/250a/tub.pdf> (Accessed 7 February 2014).

- Feaster, P. (2008) 'Édouard-Léon Scott de Martinville's "Principes de Phonautographie" (1857): A Critical Edition with English Translation and Facsimile', FirstSounds.ORG, [online] Available from: <http://www.firstsounds.org/public/First-Sounds-Working-Paper-01.pdf> (Accessed 5 October 2009).
- FirstSounds.ORG (n.d.) 'Édouard-Léon Scott in his own words', [online] Available from: <http://firstsounds.org/features/scott.php> (Accessed 5 October 2009).
- Foltin, C., Müller, J., Polansky, D., Novak, P. and Polivaev, D. (n.d.) *FreeMind*, [online] Available from: [http://freemind.sourceforge.net/wiki/index.php/Main\\_Page](http://freemind.sourceforge.net/wiki/index.php/Main_Page) (Accessed 25 September 2013).
- Frank, D. and D. (2010) 'Cymatics With Jodina Meehan', *The Frank Brothers*, [online] Available from: <http://cosmoctopus.com/2010/05/cymatics-with-jodina-meehan/> (Accessed 16 February 2013).
- Freed, A. and Schmeder, A. (2009) 'Features and Future of Open Sound Control version 1.1 for NIME', [online] Available from: <http://cnmat.berkeley.edu/node/7002>.
- Freeman, S. (2011) 'DVD Program Notes', Thor Magnusson, Alex McLean, and Nick Collins (eds.), *Computer Music Journal*, 35(4), p. 121.
- Freeman, S. (2007) 'S Freeman - MA ISD', [online] Available from: <http://collectivology.net/SamuelFreeman/academia/MAISD.htm> (Accessed 19 September 2013).
- Freeman, S. (2013) 'with Photosounder Spiral - YouTube Playlist', [online] Available from: <http://www.youtube.com/playlist?list=PLT6zTqvk6MqFbdZyfn0MJObXyi7a2R7FP> (Accessed 21 September 2013).
- Fuegi, J. and Francis, J. (2003) 'Lovelace Babbage and the creation of the 1843 "notes"', *IEEE Annals of the History of Computing*, 25(4), pp. 16–26.
- Garcia, J., Tsandilas, T., Agon, C. and Mackay, W. (2011) 'InkSplorer: Exploring Musical Ideas on Paper and Computer', In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 361–366, [online] Available from: <http://hal.inria.fr/inria-00600083> (Accessed 3 August 2013).
- Gilbert Simondon (2012) 'On Techno-Aesthetics (1982)', *PARRHESIA*, (14), pp. 1–8.
- Google (n.d.) 'Android - Discover Android', [online] Available from: <http://www.android.com/about/> (Accessed 6 October 2013).
- Grierson, M. S. (2008) 'Making music with images: interactive audiovisual performance systems for the deaf', In *Proc. 7th ICDVRAT with ArtAbilitation, Maia, Portugal, 2008*, Maia, Portugal, pp. 361–367.
- Guyton, A. (2012) 'Human Seeking', [online] Available from: <http://www.angelaguyton.com/2012/08/human-seeking/> (Accessed 21 December 2013).

- Hamilton, R., Smith, J. and Wang, G. (2011) ‘Social Composition: Musical Data Systems for Expressive Mobile Music’, *Leonardo Music Journal*, 21, pp. 57–64.
- Harman, N. (2008) ‘Chapter 3: Layers and Abstraction’, [online] Available from: <http://www.cs.swan.ac.uk/~csneal/ComputersComputing/Layers.html> (Accessed 17 April 2012).
- Hart, V. (2013a) *Twelve Tones*, [online] Available from: [http://www.youtube.com/watch?v=4niz8TfY794&feature=youtube\\_gdata\\_player](http://www.youtube.com/watch?v=4niz8TfY794&feature=youtube_gdata_player) (Accessed 10 July 2013).
- Hart, V. (2013b) ‘Twelve Tones Torrent’, *Vi Hart*, [online] Available from: <http://vihartvihart.tumblr.com/post/61642832809/twelve-tones-torrent> (Accessed 21 September 2013).
- Hart, V. (2012) *What was up with Pythagoras?*, [online] Available from: [http://www.youtube.com/watch?v=X1E7I7\\_r3Cw&feature=youtube\\_gdata\\_player](http://www.youtube.com/watch?v=X1E7I7_r3Cw&feature=youtube_gdata_player) (Accessed 13 December 2013).
- Head, T. (2008) ‘NOTES ON RECENT DIGITAL WORK’, [online] Available from: [http://www.ucl.ac.uk/slade/timhead/texts/th\\_notes.htm](http://www.ucl.ac.uk/slade/timhead/texts/th_notes.htm) (Accessed 11 October 2009).
- Healey, P. G. and Thiebaut, J.-B. (2007) ‘Sketching Musical Compositions’, In *Proc. CogSci*, pp. 1079–1084, [online] Available from: <http://csjarchive.cogsci.rpi.edu/proceedings/2007/docs/p1079.pdf> (Accessed 3 August 2013).
- Hélie, T. and Picasso, C. (2013) ‘Information on the SnailAnalyzer’, [online] Available from: [email to author] (Accessed 20 December 2013).
- Hennessey, M. and Giovannoni, D. (2008) ‘The World’s Oldest Sound Recordings Played For The First Time’, FirstSounds.ORG, [online] Available from: [www.firstsounds.org/press/032708/release\\_2008-0327.pdf](http://www.firstsounds.org/press/032708/release_2008-0327.pdf) (Accessed 10 January 2013).
- Herman, G. (1985) *Micro-music for the Commodore 64 and BBC computer*, London, Macmillan.
- Hewitt, S., Freeman, S. and Brooks, J. (2012) ‘HELOpg, lessons learned (so far)’, In *Proceedings of the 1st Symposium on Laptop Ensembles & Orchestras*, Baton Rouge, Louisiana, Louisiana State University, pp. 111–114, [online] Available from: [https://ccrma.stanford.edu/~ruviaro/texts/SLEO\\_2012\\_Proceedings.pdf](https://ccrma.stanford.edu/~ruviaro/texts/SLEO_2012_Proceedings.pdf) (Accessed 4 June 2014).
- Hewitt, S., Tremblay, P. A., Freeman, S. and Booth, G. (2010) ‘HELO: The Laptop Ensemble as an Incubator for Individual Laptop Performance Practices’, In *Proceedings of the International Computer Music Conference*, New York, ICMA, pp. 304–307, [online] Available from: <http://quod.lib.umich.edu/i/icmc/bbp2372.2010.060/--helo-the-laptop-ensemble-as-an-incubator-for-individual?view=image> (Accessed 13 December 2013).
- Iglesia, D. (2011) *24 Axes, for the Princeton Laptop Orchestra*, [online] Available from: <https://vimeo.com/27401835> (Accessed 19 June 2013).

- Iglesia, D. (2010) *Cardinality Aleph Two*, [online] Available from: <https://vimeo.com/11453161> (Accessed 19 June 2013).
- Iglesia, D. (2012) ‘Portfolio’, [online] Available from: <http://music.columbia.edu/~daniglesia/portfolio/> (Accessed 19 June 2013).
- Iglesia, D. (2008) ‘This is not a C#: What My Computer Means’, *Organised Sound*, 13(03), pp. 217–224.
- inclusive improv (2010) ‘ii @ Interlace · inclusive improv’, [online] Available from: <http://inclusiveimprov.co.uk/doku.php/events:2010-03-13-interlace> (Accessed 19 September 2013).
- Jansch, A. (2011) ‘Towards the open outcome record: a portfolio of works exploring strategies of freeing the record from fixity’, University of Huddersfield, [online] Available from: <http://eprints.hud.ac.uk/16730/1/ajanschfinalthesis.pdf> (Accessed 3 August 2013).
- Johnson, T. (1989) *The voice of new music: New York City, 1972-1982 : a collection of articles originally published in the Village Voice*, Digital edition (2002). Paris, Editions 75.
- Jordà, S., Geiger, G., Alonso, M. and Kaltenbrunner, M. (2007) ‘The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces’, In *Proceedings of the 1st international conference on Tangible and embedded interaction*, pp. 139–146, [online] Available from: <http://dl.acm.org/citation.cfm?id=1226998> (Accessed 17 February 2013).
- Karplus, K. and Strong, A. (1983) ‘Digital Synthesis of Plucked-String and Drum Timbres’, *Computer Music Journal*, 7(2), pp. 43–55.
- Kessous, L. (2002) ‘Bi-manual mapping experimentation, with angular fundamental frequency control and sound color navigation’, In *Proceedings of the 2002 conference on New interfaces for musical expression*, pp. 113–114, [online] Available from: <http://dl.acm.org/citation.cfm?id=1085214> (Accessed 26 September 2013).
- Kirn, P. (2010) ‘JazzMutant Lemur Controller is Dead; Long Live Multitouch - Create Digital Music’, [online] Available from: <http://createdigitalmusic.com/2010/11/jazzmutant-lemur-controller-is-dead-long-live-multitouch/> (Accessed 17 February 2013).
- Knight, C. (1858) *Knight's pictorial gallery of arts ...*, The London printing and publishing company, limited.
- Laurel, B. and Mountford, S. J. (eds.) (1990) *The Art of Human-Computer Interface Design*, Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc.
- Lexer, S. (n.d.) ‘I N T E R L A C E’, [online] Available from: <http://incalcando.com/interlace/> (Accessed 19 September 2013).
- Lieger, R. (2006) *Power64*, [online] Available from: <http://www.infinite-loop.at/Power64/index.html> (Accessed 25 July 2012).

- Loy, G. D. (2006) *Musimathics: A Guided Tour of the Mathematics of Music. (Vol. 1)*, Cambridge, MA, MIT Press.
- Lucier, A. (1990) *I am sitting in a room : for voice on tape*, New York, Lovely Music.
- Magnusson, T. (2011) ‘Confessions of a live coder’, In *Proceedings of International Computer Music Conference*, pp. 609 – 616.
- Magnusson, T. and Mendieta, E. H. (2007) ‘The acoustic, the digital and the body: A survey on musical instruments’, In *Proceedings of the 7th international conference on New interfaces for musical expression*, pp. 94–99, [online] Available from: <http://dl.acm.org/citation.cfm?id=1279757> (Accessed 9 December 2013).
- Manning, P. (2006) ‘The significance of techné in understanding the art and practice of electroacoustic composition’, *Organised Sound*, 11(01), p. 81.
- Manovich, L. (2012) ‘Software Studies: Continuum will publish my book Software Takes Command in July 2013’, [online] Available from: <http://lab.softwarestudies.com/2012/09/continuum-will-publish-my-book-software.html> (Accessed 7 April 2013).
- Manovich, L. (2008) ‘SOFTWARE TAKES COMMAND’, [online] Available from: <http://lab.softwarestudies.com/2008/11/softbook.html> (Accessed 19 February 2010).
- Mansoux, A. (2011) ‘bitop videos’, [online] Available from: <http://0xa.kuri.mu/2011/10/09/bitop-videos/> (Accessed 16 August 2013).
- Mauch, M., Noland, K. and Dixon, S. (2009) ‘Using Musical Structure to Enhance Automatic Chord Transcription.’, In *ISMIR*, pp. 231–236, [online] Available from: <https://www.eecs.qmul.ac.uk/~simond/pub/2009/ISMIR2009-Mauch-PS2-7.pdf> (Accessed 22 March 2013).
- McCartney, J. (1996) ‘SuperCollider: a new real time synthesis language’, In *International Computer Music Conference '96*, Hong Kong, [online] Available from: <http://www.audiosynth.com/icmc96paper.html> (Accessed 18 August 2013).
- McLean, C. A. (2011) ‘Artist-Programmers and Programming Languages for the Arts’, Doctoral Thesis, Goldsmiths, University of London, [online] Available from: <http://eprints.gold.ac.uk/6611/> (Accessed 7 August 2013).
- McLeod, J. (1994) ‘Tone Clock Theory Expanded: Chromatic Maps I & II’, [online] Available from: <http://sounz.org.nz/catalog/BO1095DS.pdf> (Accessed 10 September 2013).
- McLuhan, M. (2003) *Understanding media: the extensions of man*, Critical ed. Gordon, W. T. (ed.), Corte Madera, CA, Gingko Press.
- MIDI Manufacturers Association (2013) ‘30th Anniversary of MIDI’, [online] Available from: <http://www.midi.org/midi30/> (Accessed 28 June 2013).

- Moctezuma, A. (2011) 'TEAM 1 S11 TECHNICAL DETAILS', *TEAM 1 S11 TECHNICAL DETAILS - BIOE 414 Instrumentation Projects - University of Illinois - Engineering Wiki*, [online] Available from: <https://wiki.engr.illinois.edu/display/BIOE414/TEAM+1+S11+TECHNICAL+DETAILS> (Accessed 26 March 2013).
- Montfort, N. (2010) 'One-Line C64 BASIC Music', *Post Position*, [online] Available from: <http://nickm.com/post/2010/07/one-line-c64-basic-music/> (Accessed 29 July 2010).
- Munsell Color (n.d.) 'How Color Notation Works', [online] Available from: <http://munsell.com/about-munsell-color/how-color-notation-works/> (Accessed 24 September 2013a).
- Munsell Color (n.d.) 'Munsell Chroma', [online] Available from: <http://munsell.com/about-munsell-color/how-color-notation-works/munsell-chroma/> (Accessed 24 September 2013b).
- Munsell Color (n.d.) 'Munsell Color Space and Solid', [online] Available from: <http://munsell.com/about-munsell-color/how-color-notation-works/munsell-color-space-and-solid/> (Accessed 24 September 2013c).
- Newton, H. C. (ed.) (1909) *Harmonic vibrations and vibration figures*, London: Newton and Co., [online] Available from: <http://hdl.handle.net/2186/ksl:newhar00> (Accessed 19 February 2013).
- nu desine (n.d.) *AlphaLive*, [online] Available from: <http://www.alphasphere.com/alphalive/> (Accessed 6 October 2013).
- O'Reilly, M. (2010) *Chladni Singing*, [online] Available from: <https://vimeo.com/9986699> (Accessed 4 February 2014).
- One Man Band Studios S.A.S (2013a) *bComposer Metronome - Android Apps on Google Play*, [online] Available from: <https://play.google.com/store/apps/details?id=com.ombstudios.bcomposer.metronome> (Accessed 5 October 2013).
- One Man Band Studios S.A.S (2013b) *bComposer Rhythm - Android Apps on Google Play*, [online] Available from: <https://play.google.com/store/apps/details?id=com.ombstudios.bcomposer.rhythm> (Accessed 5 October 2013).
- Otto, G. (2002) 'Color in Computer Graphics', [online] Available from: [http://viz.aset.psu.edu/gho/sem\\_notes/color\\_2d/html/primary\\_systems.html](http://viz.aset.psu.edu/gho/sem_notes/color_2d/html/primary_systems.html) (Accessed 24 September 2013).
- Paradis, S. (2008) 'Rhythm Circle for 16th notes', *Music and Teaching Materials by Susan Paradis*, [online] Available from: <http://www.susanparadis.com/catalog.php?ID=SP750> (Accessed 5 October 2013).
- Patterson, Z. (2009) 'From the Gun Controller to the Mandala: The Cybernetic Cinema of John and James Whitney', *Grey Room*, (36), pp. 36–57.

- Photonicinduction (2012) *Extreme Speed Record Player*, [online] Available from: [http://www.youtube.com/watch?v=DNGxDaiTxaQ&feature=youtube\\_gdata\\_player](http://www.youtube.com/watch?v=DNGxDaiTxaQ&feature=youtube_gdata_player) (Accessed 25 July 2013).
- Picasso, C. and Hélie, T. (2013) 'Forum Workshop 2013 Program', [online] Available from: <http://forumnet.ircam.fr/ateliers2013/> (Accessed 7 December 2013).
- Razy Works (2008) 'VINYL KILLER.net', [online] Available from: <http://www.vinylkiller.net/> (Accessed 21 June 2013).
- Roads, C. (2004) *Microsound*, Cambridge, MA & London, The MIT Press.
- Roads, C. (1996) *The Computer Music Tutorial*, MIT Press.
- Roland Corporation U.S. and MIDI Manufacturers Association (2009) 'An Introduction to MIDI', [online] Available from: <http://www.midi.org/aboutmidi/intromidi.pdf>.
- Rouzie, M. (2013) 'Photosounder Spiral - Spectrum analyser', [online] Available from: <http://photosounder.com/spiral/> (Accessed 16 September 2013).
- Rushkoff, D. (2010) *Program Or Be Programmed: Ten Commands for a Digital Age*, OR Books.
- Saunders, J. (2003) 'Developing a modular approach to music', Doctoral Thesis, University of Huddersfield, [online] Available from: <http://eprints.hud.ac.uk/5942/> (Accessed 5 August 2013).
- Schat, P. (2002) 'Peter Schat - A Autobiography: opus website 2002', [online] Available from: <http://www.peterschat.nl/> (Accessed 10 September 2013).
- Schat, P. (1993) *The Tone Clock*, Contemporary Music Studies, Amsterdam, Harwood Academic Publishers.
- Shepard, R. (1999) 'Pitch perception and measurement', In Cook, P. R. (ed.), *Music, cognition, and computerized sound*, pp. 149–165.
- Stamp, R. (2013) 'Seeing Sound » 2013 Screenings', [online] Available from: <http://www.seeingsound.co.uk/seeing-sound-2013/2013-screenings/> (Accessed 5 May 2014).
- Stevenson, A. and Lindberg, C. A. (eds.) (2011a) 'Chladni figures', 3rd ed. *New Oxford American Dictionary*, Oxford University Press.
- Stevenson, A. and Lindberg, C. A. (eds.) (2011b) 'square', 3rd ed. *New Oxford American Dictionary*, Oxford University Press.
- Stockhausen, K. (1962) 'The Concept of Unity in Electronic Music', *Perspectives of New Music*, 1(1), pp. 39–48.
- Storaasli, A. G. (1992) 'Spiral audio spectrum display system', [online] Available from: <http://www.freepatentsonline.com/5127056.html> (Accessed 30 September 2009).

- Sullivan, J. (2005) *Vinyl Killer*, [online] Available from:  
<http://www.flickr.com/photos/jason0x21/3806307/> (Accessed 21 June 2013).
- Sumi, K., Arai, M., Fujishima, T. and Hashimoto, S. (2012) ‘A music retrieval system using chroma and pitch features based on conditional random fields’, In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 1997–2000.
- Sutcliffe, A. G. (1988) *Human Computer Interface Design*, Houndmills & London, Macmillan Education, Limited.
- Sykes, L. (2011) ‘The augmented tonoscope’, In *Proceedings of the 2011 international conference on Electronic Visualisation and the Arts*, pp. 281–286, [online] Available from:  
<http://phd.lewissykes.info/webdisk/UVM11-Lewis%20Sykes-TheAugmentedTonoscope.pdf>  
(Accessed 17 December 2013).
- Thiebaut, J.-B. (2010) ‘Sketching music: representation and composition’, Doctoral Thesis, Queen Mary, University of London, [online] Available from:  
<http://qmro.qmul.ac.uk/jspui/handle/123456789/406> (Accessed 4 August 2013).
- Treadaway, C. (2004) ‘Digital Imagination: the impact of digital imaging on printed textiles’, *Textile: The Journal of Cloth and Culture*, 2(3), pp. 256–273.
- Trueman, D. (2007) ‘The Cyclotron’, [online] Available from:  
<http://dtrueman.mycpanel.princeton.edu/Cyclotron/> (Accessed 8 October 2013).
- Tubb, R. H. (2011) *An Investigation into Scanned Synthesis and Multi-Touch*, MSc Project Report, Queen Mary University of London.
- Tufnell, B. (2003) ‘TIM HEAD’, [online] Available from:  
[http://www.ucl.ac.uk/slade/timhead/texts/bt\\_tim.htm](http://www.ucl.ac.uk/slade/timhead/texts/bt_tim.htm) (Accessed 11 October 2009).
- Verplank, B., Mathews, M. and Shaw, R. (2001) ‘Scanned synthesis’, *The Journal of the Acoustical Society of America*, 109(5), pp. 2400–2400.
- Wachowski, A. and Wachowski, L. (1999) *The Matrix*, Warner Bros.
- Wang, G. (2007) ‘A history of programming and music’, 4th printing 2010. In Collins, N. and d’Escrivan, J. (eds.), *The Cambridge Companion to Electronic Music*, Cambridge University Press, pp. 55–71.
- Wees, F. M. (1992) *Light Moving in Time: Studies in the Visual Aesthetics of Avant-Garde Film.*, Berkeley, University of California Press, [online] Available from:  
<http://ark.cdlib.org/ark:/13030/ft438nb2fr> (Accessed 11 December 2013).
- Weisstein, E. W. (2003) ‘Archimedes’ Spiral’, Text, [online] Available from:  
<http://mathworld.wolfram.com/ArchimedesSpiral.html> (Accessed 12 April 2010).
- Wells, D. G. (1991) *The Penguin dictionary of curious and interesting geometry*, London and New York, Penguin Books.

Winner, J. (2001) 'CIRCLE MACHINE | RaymondScott.com | Official Raymond Scott Archives Site', [online] Available from: <http://oldsite.raymondscott.com/circle.html> (Accessed 30 March 2012).

Winner, J. E. (n.d.) 'The RAYMOND SCOTT Archives', [online] Available from: [http://raymondscott.com/#293/custom\\_plain](http://raymondscott.com/#293/custom_plain) (Accessed 8 October 2013).

Yosemitebear62 (2010) *Yosemitebear Mountain Double Rainbow 1-8-10*, [online] Available from: [http://www.youtube.com/watch?v=OQSNhk5ICTI&feature=youtube\\_gdata\\_player](http://www.youtube.com/watch?v=OQSNhk5ICTI&feature=youtube_gdata_player) (Accessed 30 October 2013).