Aesthetically Driven Design of Network Based Multi-User Instruments

by

Curtis McKinney

A thesis submitted to the faculty of
Bournemouth University

in partial fulfilment of the requirements for the degree of
Doctor of Philosophy
in the School of Design, Engineering, and Computing

July 2014

Aesthetically Driven Design of Network Based Multi-User Instruments

Curtis McKinney

Abstract

Digital networking technologies open up a new world of possibilities for music making, allowing performers to collaborate in ways not possible before. *Network based Multi-User Instruments* (NMIs) are one novel method of musical collaboration that take advantage of networking technology. NMIs are digital musical instruments that exist as a single entity instantiated over several nodes in a network and are performed simultaneously by multiple musicians in real-time. This new avenue is exciting, but it begs the question of how does one design instruments for this new medium? This research explores the use of an aesthetically driven design process to guide the design, construction, rehearsal, and performance of a series of NMIs. This is an iterative process that makes use of a regularly rehearsing and performing ensemble which serves as a test-bed for new instruments, from conception, to design, to implementation, to performance.

This research includes details of several NMIs constructed in accordance with this design process. These NMIs have been quantitatively analysed and empirically tested for the presence of interconnectivity and group influence during performance as a method for measuring group collaboration. Furthermore qualitative analyses are applied which test for the perceived effectiveness of these instruments during real-world performances in front of live audiences. The results of these analyses show that an aesthetically driven method of designing NMIs produces instruments that are interactive and collaborative. Furthermore results show that audiences perceive a measurable impression of interconnectivity and liveness in the ensemble even though most of the performers in the ensemble are not physically present.

Contents

List of Figures

List of Tables

Accompanying Materials

A DVD should be found accompanying this document which contains a portfolio with audio and video recordings of performances of the pieces *Leech*, *Yig*, *Flow*, *Simulacra*, *Renditions*, *NeuroMedusae I*, and *NeuroMedusae II*. The DVD also contains the complete code repositories for the projects *Leech*, *Yig*, *Simulacra*, *Mutagen*, *OSCthulhu*, and *Azathoth*. An online back-up of the portfolio material may be found at `http://curtismckinney.com/portfolio.zip`.

Acknowledgements

I am extremely grateful for Dr. Alain Renaud's unwavering support during my journey from initial ideas to a fully completed study. His knowledge and wisdom have been a guiding light. I would like to thank Dr. David John for his input and oft required knowledge which has helped paved the way for me to finish. I cannot say enough to thank Chris Brown and John Bischoff who, over the years, have been a constant source of awe and inspiration.

Of course, none of this would have been possible without my fellow bandmates and co-conspirators Cole Ingraham, Chad McKinney, and Ben O'Brien, whose musical bravery has constantly pushed me to strive to unlock new ideas. This goes doubly so for Chad who has also served the role of eternally supportive brother (though I think we can all agree that I come up with the better song titles). Many thanks to my mother who has not once doubted what I could do. Lastly I would like thank Stacy for encouraging me to simply finish what I started.

Author's Declaration

The written work presented here has been carried out solely by the author. Several passages in this work have been previously published, in whole or in part. These portions include:

- Chapter 3 contains passages which were part of a paper published in the proceedings for the Symposium for Laptop Ensembles and Orchestras, 2012, as the paper *Glitch Lich: Evolution of an Intercontinental Network Band*.
- Chapter 4, section 3, is largely from a paper published as part of the International Computer Music Conference, 2012, titled *OSCthulhu: Applying Video Game State-Based Synchronization to Network Computer Music*.
- Details concerning NMI visualisation that are discussed during chapter 4 were published as the paper *Visualization of Network Based Multi-User Instruments* for the Live Interfaces, Performance, Art, Music Symposium, 2013.
- Most of chapter 4, section 7 is derived from the paper *Leech: BitTorrent and Music Piracy Sonification*, published in the proceedings of the Sound And Music Computing Conference, 2011. This paper was nominated for best paper of the conference.
- Chapter 2, section 5, subsection 7 contains details from works which were created during the course of the author's thesis presented for completion of a Master of Fine Arts degree at Mills College (completed in 2009), and is published in the school's library.

Furthermore all of the musical works created by the author for this research were created in close collaboration with Cole Ingraham, Chad McKinney, and Ben O'Brien as part of the band *Glitch Lich*.

Definitions

Aleatory - Random chance or luck.

Digital Audio Workstation - or DAW, is a piece of software created for the specific purpose of creating and manipulating audio files. The design of this software is usually greatly influenced by the manner in which audio engineers have worked with magnetic audio tape for musical creation in the past (Nahmani, 2009; Avid Audio Inc., 2011).

BitTorrent - A peer to peer networking technology often used to share copyrighted material illegally (Cohen, 2011).

Connectome - A relatively new term in neuroscience which refers to an accurate map of neural interconnections which reside in the brain of a specific individual. This term is applied metaphorically in this dissertation to the network connections that occur in a NMI (Hagmann, 2005)..

Dimension Space - In the research conducted here, a dimension space is a kind of chart used for analysis. This chart depicts several characteristics of an instrument plotted along different axes. This chart naturally presents instruments as distinct shapes that may easily be compared and contrasted to other instruments (Hattwick and Wanderley, 2012).

Fast-Fourier Transform - or FFT, is an algorithm for the efficient transformation of a signal from the time/space domain to the frequency domain. It may be used during sound synthesis to apply various effects to an audio signal which affects that audio signal's spectral content directly (Brigham, 1973).

Viscerality - Having the quality of strongly affecting a person, as if affecting the viscera (internal organs) of that person,

Musical Instrument Digital Interface- is a communications protocol that was designed to allow digital musical instruments to communicate with each other over specialised hardware. As it was created before the proliferation of The Internet It is designed in such a way as to minimise bandwidth requirements. Packet sizes range from one to three bytes in length (MIDI Manufacturers Association Inc., 1995).

Network based Multi-User Instrument - or NMI, is a digital musical instru-

ment which exists across multiple nodes on a network, wherein multiple performers share musical control over a single music producing entity. Furthermore an NMI should treat this networking infrastructure as a fundamental aspect of making music with the instrument. Often NMIs encourage collaboration amongst a musical ensemble as part of their design by creating novel interactions between the musical instrument's various input mechanisms.

Network Music - is a kind of electronic music which has as its focal point the novel usage of communications networks during the production of said music. For the purposes of this research this is meant to cast a wider net than *Network based Multi-User Instruments*, and encompasses other musical usages of networks, such as classical musical performance that make use of telepresence.

Open Sound Control - is a network communications protocol built on top of User Datagram Protocol, and which has been specifically crafted for usage between software systems that are related to the production of sound and/or music (Wright, 2002).

Remotely Rendered Synthesis - is a technique for software synthesis in a geographically displaced network ensemble. This prescribes that no actual audio is sent over network connections, and instead low-bandwidth/low-latency control information is sent instead. The entire sound of the ensemble, modulated by the control information that is being shared, is synthesised on each end-point of the network ensemble.

Similarity Matrix - A similarity matrix is a two-dimensional chart which depicts the similarity of two signals, plotted over time. In some cases one signal is compared to itself over time; this would be called a "Self-Similarity Matrix" (Collins, 2011).

Sonification - A technique akin to visualisation that attempts to transform some kind of information or signal form one domain to the sonic/musical domain.

Stochastic - A stochastic system is one whose state is non-deterministic such that any following state is derived in a probabilistic manner (Xenakis, 2001).

Synthdef - A construct used in the SuperCollider programming language which is used to store and recall specific sound synthesis engines (Wilson et al., 2011).

Transmission Control Protocol - or TCP, is a protocol for network communi-

cations that has built into it several mechanism that attempt to guarantee delivery of a packet of information. The emphasis is on reliability over low-latency and low-overhead (Stevens, 1994).

User Datagram Protocol - or UDP, is a protocol for network communications which prescribes for a connectionless communication system that has lower overhead and lower latency than Transmission Control Protocol, but which is less reliable during transmission (Ford et al., 2005).

1 Introduction

The advent of the Internet and the influx of modern digital technologies has had an immeasurable impact on the composition, performance, consumption, and conception of music. While there have been efforts to utilise older methodologies for performing and composing music in this new framework, others have striven to develop new approaches for musical expressivity that utilise the unique capabilities that these technologies have to offer. The ability to instantaneously share musical information with others in real-time allows for compositional and performative techniques not possible before.

One novel method for harnessing the power of digital networks is through the usage of Network based Multi-User Instruments (or NMI). Specifically these are digital instruments who have no central corporeal point of existence and instead exist along a network of musicians engaged in a collaborative real-time musical performance. These musicians manipulate NMIs through a software interface that has at its core a network layer that keeps the instrument synchronised on each performer's computer, sharing the performative gesture of each musician with the entire group. NMIs have several peculiar and interesting properties. The first and most obvious is that since the performers operate the instrument via network messages the players engaged in the performance need not be in the same physical space. Another interesting aspect of NMIs is their incredible ability for collaboration. Since the instrument is operated by multiple individuals simultaneously, the performers are almost forced to navigate a musical space wherein they are constantly reacting to the performance gestures of the other musicians in the ensemble, as the instrument they themselves are manipulating is being modulated right underneath them by their fellow ensemble members.

Furthermore, the physical nature of digital networks means that latency will always be a part of any constructed NMIs. In a sense, NMIs are in a constant state of schizophrenia, with multiple slightly different states instantiated on each member's computer. Latencies as low as 50 milliseconds have been demonstrated to have debilitating effects on a musician's ability to perform coherently with another musician along a network (Chew et al., 2005) . A designer of NMIs may choose to either fight this through sophisticated synchronisation technologies, or embrace it as a new performance medium, akin to a twenty-first century concert hall. NMIs also serve as a kind of social contract over shared, limited resources. This brings ensemble politics into the forefront of the very design of the NMI itself.

The League of Automatic Music Composers (followed by The Hub) is generally considered to be the first ensemble to explicitly exploit digital networks as a musical-resource for real-time musical performance, operating from the late

1970's to present day (Brown and Bishcoff, 2002). The ensemble interconnected their individual computers, initially via direct serial port connection, later via Musical Instrument Digital Interface (MIDI), and then Open Sound Control (OSC) networking protocols (MIDI Manufacturers Association Inc., 1995; Wright, 2002). Through these interconnections they were able to construct a *meta-musical-instrument*, whose behaviour was more than the sum of its parts, controlled equally by the members of the ensemble and by the neuronal-like interactions between their computers.

Following The Hub, there have been many efforts to harness the capabilities of NMIs, including the establishment of networked ensembles such as the Princeton Laptop Orchestra (PLOrk), and network-specific installations, such as Atau Tanaka's *Global String* (Trueman et al., 2006; Tanaka and Bongers, 2001). There have also been many efforts entirely outside of the arena of networked electronics that have also explored the possibilities of multi-user instruments, including traditional acoustic instruments such as the organistrum (Brauchli, 2005). A detailed survey of these instruments, may be found in Chapter 2.

There are several open issues that are paramount when designing a new NMI, given current techniques and technologies. One of the interesting capabilities of networked music is the ability for the performers to be geographically displaced during performance. However this capability is a double-edged sword, as dislocative network systems must solve serious technical issues to guarantee a smooth musical performance. The technical complexities involved with network music often require large bandwidth and a high quality of service to meet the rigorous demands of the music (Renaud et al., 2007). However, the presence of networks of this quality are not always possible when performing in non-academic settings. Furthermore, consumer grade networking further degrades the possibility of networked ensembles through overly-resistant network firewalls and routers which block incoming traffic (Ford et al., 2005). There have been efforts to address this issue. Bencina created the software OSCGroups to help networked ensembles establish musical performances behind consumer grade firewalls and routers (Bencina, 2013). However this technology has its limits, only solving the issues of routers (and not solving consistent networking, due to its reliance on UDP). As well, it only solves the router issue roughly 85% of the time. Is there another networking system or infrastructure that might make performing with NMIs on consumer grade networks a possibility?

Ostertag (2002) highlights another open issue with electronic music in general, stating that the absence of the human body in electronic music has negative effects on the visceral appreciation of the art by performers and audience members. Performances by NMIs with members that are geographically dispersed exacerbate this issue. In these performance scenarios, the instrument and even

the performers themselves are physically missing from the performance space. Are there means to combat this lack of embodiment and visceral appreciation for NMIs? Much research has been conducted to investigate methods for creating embodiment in electronic music, such as through the usage of gestural controllers (Wanderley and Battier, 2000). While gestural control is certainly a fertile and interesting area of research, this does not have immediate application in distributed network performance, unless real-time video streams of each performer would be used, otherwise only a the physically present performer's gestures could be seen by the audience. However, this would severely compound the previous issue of reliable networking during performance. Is there another way to combat this issue?

The complexities and technical rigours of building NMIs requires a large amount of technical knowledge by anyone who would attempt to make one, requiring knowledge of music theory, digital audio, digital sound processing, software programming, networking, instrument design, music composition, and musical performance. Furthermore, there is a serious time-requirement for designing and implementing NMIs, with a large amount of repetitive "boiler-plate" code (generic and repetitive code that must be programmed to begin construction of a new project) required to start the implementation processes. Finally, as NMIs are relatively young concept in music performance, one might ask, how is a NMI, or multi-user instruments in general, defined at all, and how might one set out to design one in the first place?

1.1 Research Questions

These issues may be summed up in the following research questions that the research conducted during this study has investigated:

- What is a multi-user instrument, and how is it defined?
- Are there distinct morphologies of multi-user instruments, and may a taxonomy be created to organise them?
- Is there a manner in which to examine the collaborative capabilities of a multi-user instrument?
- Is a methodology for designing NMIs based on the aesthetics of a specific ensemble able to generate NMIs which satisfy the design requirements generated by that ensemble?
- May the issues of liveness and disembodiment inherent to NMIs be addressed in some manner?
- Are there better techniques for overcoming the technical difficulties involved with networking geographically displaced ensembles?

- Is there a way to streamline the development processes of creating NMIs?

1.2 Aims and Objectives

In an attempt to answer these research questions the research conducted in this study aimed to:

Aim 1 - Establish clear definitions and methods of analysis for a class of digital musical instrument termed a 'Network based Multi-User Instrument' as well as 'Multi-User Instruments' in a broader sense. *Related objectives are to:*

1. Conduct a survey of Multi-User Instruments.
2. Create a taxonomy for organising Multi-User Instruments.
3. Create modes of analysis for Multi-User Instruments.
4. Employ these new modes of analysis and taxonomies on the instruments found in the survey of Multi-User Instruments.

Aim 2 - Create new, and refine old, tools and techniques for composing, performing, and designing such instruments. *Related objectives are to:*

5. Establish a methodological framework for designing NMIs.
6. Using this methodology, initialise a design space for creating new NMIs.
7. Use this design space to establish technical requirements for designing new NMIs.
8. Identify short-comings in previous technologies for accomplishing the technical requirements of the initialised design space.
9. Create new tools, NMIs, compositions, and performances with the established methodology and initialised design space, taking into account the shortcoming of established technologies, and overcoming them by creating new technologies where necessary.

Aim 3 - Determine the effectiveness of these newly created, or refined, tools and techniques. *Related objectives are to:*

10. Quantitatively study the effectiveness of new tools created to overcome shortcomings of previous technologies for usage by NMIs.
11. Analyse a live performance of several NMIs, examining the quantitative and qualitative effectiveness of the techniques established in the research.
12. Use the taxonomy and analysis tools deployed in the survey to dissect the new NMIs designed in this research.

1.3 Contribution to Knowledge

To extend the previous research that has been conducted in this area a number of possible answers are proposed. A definition of multi-user instruments, as well as a taxonomy to structure and organise them has been created. The application of this taxonomy to the survey of multi-user instruments previously discussed may be found in Chapter 2. To give some structure to the design process of NMIs, a methodology for constructing new NMIs is proposed, based on establishing design goals which are initialised by the aesthetics of a musical ensemble or band. This is discussed in Chapter 3.

Several new techniques and software systems have been created to resolve the technical networking issues of NMIs, as well as to combat the issues of performer and instrument disembodiment. To allow for the consistent networking of NMIs in arduous and real-world contexts, a new networking system, entitled OSCthulhu, has been created that takes inspiration from video game networking technology (Sweeney, 1999). As well, to combat the issues of disembodiment, a series of techniques for the visualisation and projection of virtualised instruments and performers has been created. To reduce the amount of technical complexity and boiler-plate code required to construct a new NMI a "network music engine" is proposed, as an analogue to the concept of a video game engine. Finally a series of new NMIs have been created following the methodology established, using the techniques and tools created. The design and implementation of these systems, techniques, and NMIs are discussed in Chapter 4. A quantitative analysis of the technical rigours of the network techniques established is shown, as well as quantitative and qualitative analysis of the software tools and NMIs involved in a musical performance are found in Chapter 5. Table 1 contains a description and time line of development for each software system that was developed during the course of this research. All of this software is open-source and is freely available from the author's source code repository, currently found at `https://github.com/CurtisMcKinney`.

Project	*Timeline*	*Description*
OSCthulhu	2010-2014	UDP-based networking system. All of the network based projects depend on this.
Medusa	2010-2011	Networked GUI front-end for musical interaction
NeuroMedusae I	2010-2011	Network piece based on single-sample feedback, using the Medusa System.
NeuroMedusae II	2010-2011	Network piece using convolution based feedback, using the Medusa System.
Renditions	2010	Visualised network music score for acoustic performers.
Flow	2010	Sound art installation using a stream as a musical sequencer.
Yig	2011-2012	Descendant of Medusa, an improved Networked GUI front-end for music.
Curse Of Yig	2011-2012	Musical piece developed using Yig, with visuals, based on feedback synthesis.
Leech	2011-2012	Bit-torrent visualisation and sonification.
Mutagen	2012-2013	Networked DAW/sequencer.
Simulacra	2012-2013	Network music piece based on forced performer interactions. Using Mutagen and Azathoth.
Azathoth	2012-2013	Network music engine using OSCthulhu.
Necronomicon	2014-2015	Network music engine using Haskell. A more developed descendant of OSCthulhu and Azathoth.

Table 1: Software projects developed during the course of this research.

2 Multi-User Musical Instruments

This chapter details research pertinent to aim #1 as defined by the Introduction chapter. To reiterate, aim #1 is to establish clear definitions and methods of analysis for Network based Multi-User Instruments (NMIs). To achieve this aim several objectives (numbered one through four) were established and have been completed. These objectives are as follows:

1. Conduct a survey of Multi-User Instruments.
2. Create a taxonomy for organising Multi-User Instruments.
3. Create modes of analysis for Multi-User Instruments.
4. Employ these new modes of analysis and taxonomies on the instruments found in the survey of Multi-User Instruments.

What follows in this chapter is a detailing of the completed objectives and how they relate to the established aim.

2.1 Definitions

To aid in this it is important to determine exactly how one defines a multi-user musical instrument. Traditional musical instruments are easily identified. They are self-contained physical bodies that have various methods for exciting resonant spaces, and which are manipulated by persons for auditory enjoyment. However, digital technologies can blur and confuse this definition, as a software based musical instrument has no physical body, may be comprised of several different articulated systems, and the methods for performing them are as varied and flexible as the human mind can conjure. However despite these complications, the central premise is of a singular entity that a person manipulates in a real-time performative fashion (Gurevich and Fyans, 2011).

For this study, a definition for multiple-user instruments has been created, which is as follows: A multi-user instrument is a musical instrument, piece, or ensemble, wherein multiple individuals have shared performative control over a single sound-producing source or engine, or where the connections in a network of discrete sound producing sources or engines controlled by separate individuals achieve a sufficient level of interconnectivity that it is difficult to differentiate between those discrete sources and a group whole. That is to say that there must be a sense of fusion between the performers of the instrument. This concept is more abstract in scenarios where there are multiple performers with multiple different sound producing sources. What makes one group of laptop performers an ensemble and another a multi-user instrument? It is this sense of fusion. If the multi-performers have shared control and have a high degree of interconnectivity

and interactions, to such a point that there could be said to be a single instrument, perhaps not a physical instrument but an instrument in the abstract, then that ensemble is said to have crossed the threshold and is deemed a multi-user instrument.

An instrument, piece, or ensemble has been included in the survey that has been conducted if it is deemed to meet this definition. In some of these instruments there are various levels of temporal asynchronicity that performers experience while playing the instrument. This could be due to structural characteristics, as with network latency in network based instruments, or could be intentional features of the instrument, for example through the use of time delays. For some instruments these temporal asynchronicities are considered a musical resource. This temporal divergence is exploited as a means to alter the relationship between the performer and the instrument, enforcing inherent rhythmic signatures onto the performance, or to sonify a physical characteristic of the instrument's medium (Chafe and Leistkow, 2008) (Renaud and Câeceres, 2001). However, systems that allow multiple users to interact with each other, but whose interactions are separated by large spans of time (measured in hours or days, as compared to network lag times measured in milliseconds), are considered to be a different concept and are thus not included here, as their interactions are not readily apparent in a performance setting, unlike the rest of the instruments covered in this survey.

Furthermore, for the purposes of this research, an Network based Multi-User Instrument (or NMI) is considered to be a multi-user instrument, as defined by the definition given in the previous paragraph, who has as one of it's central structural or aesthetic focuses the usage of digital networks, often, but not strictly, related to The Internet.

2.2 Structural Properties of Multi-User Instruments

Jordà (2005) provides a taxonomy for characteristics of multi-user instruments, which defines three major properties.

- *User-number and user-number flexibility*: The number of performers for the given instrument. This may be variable. Theoretically, the more performers on the instrument, the more simultaneous musical information may be manipulated, increasing "musical bandwidth".
- *User-roles and role flexibility*: Many multi-user instruments feature different roles for each performer. For example, one performer may determine the pitch of the instrument, while another determines the amplitude. Some multi-user instruments also allow for the performers to dynamically change

what their role is during performance. Different roles in an instrument allow for multiple musically intense or demanding tasks to be executed simultaneously and with full attention.

- *Interdependencies and hierarchies*: The degree to which performers interact with and affect each other. Also, the manner in which influence is shared and exerted in the instrument(i.e. democratically, anarchically, dictatorially). The more inequality in the capabilities that each performer has, the more pronounced the hierarchy that emerges from the system.

The first two of these deal mainly with structural concerns of a multi-user instrument. The third pertains to collaborative and political nature of multi-user instruments. The third of these will be expanded upon later in section 2.4, however the first two of these properties are of initial concern. To help categorise these instruments the author has extended upon these properties by adding four more structural properties to describe multi-user instruments.

- *Geographic group distribution*: Some multi-user instruments have the unique capability to allow performers to be distributed across different geographic locations.
- *Incidental versus coordinated group formation*: Some instruments are constructed so that they may be performed at any point in time by users that incidentally arrive at the instrument within the same time span. These users may have never been associated with each other before the performance. In some instances, these performers may not even be aware that they actively performing *at all*. Other instruments are comprised of groups of performers that coordinate when they perform on the instrument, such as a regularly practising ensemble.
- *Number of sound sources*: Some multi-user instruments are voiced through multiple articulated sound sources, such as the different laptops in a laptop ensemble. The key difference between an ensemble of laptops simply playing with each other, and a multi-user instrument comprised of a group of laptops is the issue of interconnectivity.
- *Medium*: Some instruments are inherently tied to a certain medium, such as acoustic, electro-acoustic, and digital instruments.

2.3 Models for Multi-User Instruments

A series of models for multi-user instruments is presented based on the structural properties laid out above. These models are meant to encapsulate methodologies and design that instruments makers have followed constructing multi-user instruments in time.

- *Utilitarian Multi-User Instruments*: These are acoustics instruments that employ multiple performers due to some logistical reason, such as the unwieldy size of the instrument. The user number and user roles in these instruments are fixed, and the performers are located in a single space. Traditionally these instrument relied upon coordinated group formations. These instruments only have one sound source, and are acoustic in nature.
- *Extended Traditional Instruments*: Instruments that extend an acoustic instrument, often times through the use of electronic sound processing. The user numbers and roles tend to be more fixed, with performers assuming specific duties, such as exciting the acoustic instrument versus processing the instruments output. The performers of these instruments are locally located, and have coordinated group formations. There may be multiple sound sources, and are electro-acoustic in nature.
- *Surface Instruments*: Instruments that employ the use of a surface as communal medium for multiple performers to perform on, as well as to provide visual feedback. The user number and user roles on these instruments tend to be rather flexible and egalitarian. The groups that play on these instruments tend to be local, though incidental group formations are often possible as these instruments may be situated as an installation for public interaction. There is usually a single sound source, and they often use a digital medium.
- *Interconnected Laptop Ensembles*: Ensembles that are enmeshed to such a high degree, often through the use of digital networking technologies, that they could be identified as a sort of meta-multi-user instrument. User-number and user-roles are very flexible, the groups that play these instruments may either be located in a single space, or geographically displaced. They are structured as an ensemble, and thus use coordinated group formations. There may be multiple sound source, and they use the digital medium.
- *Cloud Instruments*: These instruments live in "The Cloud", as it were, usually on the Internet and accessible by any individual with a connection to that network. User numbers and roles are very flexible for these instruments as they are intended for use by the general public, thus they have highly incidental group formations. They may have many different sound sources and are of either an analogue or digital medium.
- *Kinetic Group Instruments*: These instruments map the movements and gestures of a group of individuals to control a singular instrument. User numbers and roles in these instruments are often well defined. These groups are

formed locally and are usually coordinated. There may be multiple sound sources, and are digital in nature.

- *Game Instruments*: These instruments map the actions of multiple individuals engaged in some kind of game to performative controls over a musical instrument. The user number and roles in these are fixed, and the groups are locally formed in a coordinated fashion. There may be multiple sound sources, and may be either analogue or digital.

2.4 A Dimension Space for Collaboration

While it is possible to group these instrument based upon their structural characteristics and physical make up, the way in which the various instruments allow their performers to interact with each other may differ vastly from one instrument to the next, or even from one piece on an instrument to another piece on the same instrument. Presented here is a dimension space used to analyse the collaborative, social, and political aspects of multi-user instruments. This dimension space is an extension of the model proposed by Hattwick and Wanderley (Hattwick and Wanderley, 2012). There are seven axes in the dimension space which describe the various characteristics of an instrument's collaborative capabilities. To discern an instrument's location on these axes a series of questions may be asked:

- *Texture* - Homogeneous to Heterogeneous: Are individual parts uniquely discernible, or do they blend together and/or sound similar?
- *Equality* - Unequal to Equal: Do performer have equal capabilities, or are there multiple roles with differing functions? Is there a hierarchy or uneven distribution of power over the instrument?
- *Centralisation* - Centralised to Decentralised: Is there a single server or other source the player must use? Is there a conductor? Do performers have access to the same information/data or is it fire-walled between performers.
- *Physicality* - Fixed to Free: Is there a physical manifestation to the instrument, or is it virtualised? How important is physical gesture and communication?
- *Synchronicity* - Synchronous to Sequential: Do performers play simultaneously to each other, or is there a substantial lag-time to collaboration? Do performers take turns?
- *Dependency* - Interdependent to Independent: Do performers interact with and depend on each other, or are they independent of outside influence? Do performers rely on each other to produce sound at all?

- *Cognisability* - Obscure to Cognisable: Are the interactions that occur on the instrument easily cognisable to an audience member, or is the behaviour of the instrument obscured?

The dimension space analysis for all of the instruments in each of the models discussed in this survey may be found at the end of each model's section. The values of the various axes in the dimension space are assigned to each instrument based on the author's judgements and are determined by subjective analysis of the instruments. They are also relative to the set of instruments that have been collected in this survey. Therefore they should not be viewed as quantitative data but instead a qualitative comparison of the qualities of the instruments. Arguments for why the values for the various axes were chosen may be found in the sections where they are presented.

2.5 A Survey of Multi-User Instruments

A survey of multi-user instruments has been conducted, using the structural properties and models as defined in the previous section. Furthermore, each instrument in this survey has been analysed using the dimension space that has been presented here. The instruments in this survey are grouped into subsections according to the model they are associated with.

2.5.1 Utilitarian Multi-User Instruments

There have been several traditional examples of multi-user instruments in the acoustic realm. The oldest of these instruments tend to employ multiple performers for more utilitarian reasons, such as the unwieldy size of the instrument. The user numbers, and user role are fixed, with clear hierarchies of power and differences in abilities emerging. The performers are locally distributed, with coordinated group formation, a single sound source, and operating in the acoustic medium.

An early example of a multi-user instrument is the medieval organistrum. A predecessor to the hurdy-gurdy, this stringed instrument was so large that it required two performers. One performer cranked a wheel to excite strings that vibrated over a resonant chamber, while another performer depressed keys across the neck which shortened the length of the vibrating strings, changing the pitches being played. For full functionality the number of users had to be exactly two. Though theoretically possible for the two users to switch roles, in traditional practice this never occurred. The organistrum demonstrates a clear example of interdependency, as both performers must rely on each other for the instrument to operate (Brauchli, 2005).

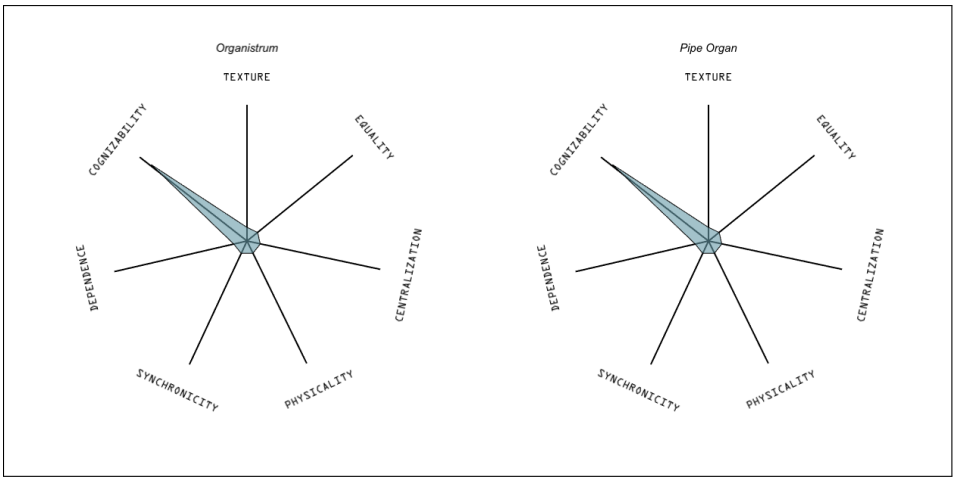

Figure 1: Collaborative dimension spaces for Utilitarian Multi-User Instruments.

Another example of a utilitarian multi-user instruments are traditional pipe organs. Pipe organs require a constant stream of airflow through their pipes to produce sound. Before the advent of electricity or steam engines, the pipe organ required a secondary user called a *calcant*, derived from the Latin word *calcare* which means "to tread". The calcant operated a set of bellows which pumped air through the organ, while the keyboardist operated the manifold to produce pitches. In older organs the calcant was also required to alter the configuration of the *stops*, so called because they literally stopped air from flowing into certain pipes on the organ. By changing the configuration of the stops on the organ the calcant altered the timbre that the organ produced. The players are dependent on each other, though one of the performers is clearly hierarchically more important, to the point that one participant is not even considered to be a performer proper. The instrument demonstrates low agility, as often times the calcant was not in close proximity to the keyboardist, and the only means of communication was a bell to signal the start and stop of the instrument. Eventually the calcant was entirely replaced by steam and electric engines (Bush and Kassel, 2006).

The interactions that take place between the two performers of either an organistrum or pipe organ look very similar when viewed through the lens of the dimension spaces found in Figure 1. These instruments, with their fixed acoustic construction and unequal performer roles both naturally enforce a similar dimension space with localised centralisation, tight synchronicity, and almost complete interdependency. Furthermore, their acoustic sound production and clear performer roles makes for high Cognisability.

2.5.2 Extended Traditional Instruments

Extended Traditional Instruments either take a traditional instrument and extend its capabilities, or take the inherent design of a single-user instrument and design

a modernised multi-performer version of it. These instruments share many of the same properties as their traditional counterparts. They often times have fixed user numbers and user roles, with pronounced hierarchies emerging out of the interactions from the defined roles. The groups are usually locally distributed, with coordinated group formations, a single sound source, and operating in the acoustic, analogue, or digital mediums.

A more recent example is the contemporary classical usage of multiple performers on one piano. Though two-hands piano is a traditional mode of performance, and could be considered a form of multi-user instrument, albeit one in which the musical interactions that occur during performance are very similar to a standard inter-ensemble performer relationship. However, starting in the 20th century composers such as Ben Johnston, Harry Partch, and Henry Cowell employed multiple performers simultaneously using various extended techniques on the piano. Multiple performers utilised scratching, striking, muting, plucking the strings, depressing the pedals, and knocking on the wood of the instruments, simultaneously in various combinations, with various levels of interactivity between the performers (Ishii, 2005).

Unlike the previous acoustic multi-instrument examples, the presence of extra performers is not required. However, the nature of the extra performers' presence is also more creative, as they are not relegated to slavishly pumping or turning a mechanism with little possible musical expressivity. The organistrum, and the pipe organ all require another performer for mechanical reasons, whereas extra performers are employed on piano for more creative purposes. Altogether this multi-user piano performance demonstrates a large amount of flexibility.The number of users may be as low as one, and can be up to as many as can fit around a piano. There are generally two performance roles, being either situated at the keyboard, or inside the instrument. These roles are flexible and may be switched dynamically. The level of interaction is very high. A performer playing a note on the keyboard will alter the sound being produced by a second performer manipulating the same string with his hands or various implements. Also, the sound produced by the first performer will be altered by the actions of the second. Thereby their actions and sounds produced by each are inherently linked. The various performers may be organised in many different structures and hierarchies. However, the actions of a performer manipulating the strings of the instrument will generally affect the sound of someone playing a note on the keyboard more so than vice versa.

The composer Karlheinz Stockhausen created a multi-user electronic instrument system centred on electronic manipulation of a Tam-tam for his piece *Mikrophonie I*. There are six performers in the piece, two performers acoustically activate the Tam-tam in various manners, two performers shift the placement of a

microphone around the tam-tam, and two performers operate analogue band-pass filters that affect the previous performer's microphone, with the resultant signal amplified quadraphonically (Manning, 2004). This structure constructs two sub-groups, each comprised of an excitation point, a microphone point, and a filter manipulation point, with the two groups situated on either side of the Tam-tam. This leads to a high degree of interaction within each sub-group, as each sound produced is the result of manipulation from individuals at all three points. In fact, it is actually impossible for a sound to be produced without it being affected by all 3 performers in the sub-group.

The Tooka is a novel two-person digital wind-instrument created by Sidney Fels and Florian Vogt, consisting of a pair of tubes that each performer blows into, and a set of buttons. The Tooka does not produce a specific sound but is instead a controller that produces MIDI information that can then be sent to various arbitrary sound producing devices. Air pressure produced by each performer is measured, and the two measurements are summed to produce a single control value. This control value may be used to control a number of musical parameters, though according to Fels and Vogt (2002) amplitude is the most obvious choice . Each performer has three buttons used to control frequency data. The first button is controlled by the index finger which is used to determine octave, while the middle and ring buttons are used to control intervals within the octave. However, once again, it is the summation of the two performers' control data that is used to produce a single frequency.

The dimension spaces for the collaborative interactions of these instruments may be found in Figure 2. In comparison with the traditionally utilitarian instruments the extended traditional instruments show more variability in collaborative interaction. These instruments while not entirely physically fixed still exhibit a more physically fixed than free idiom. Their localised physical construction likewise enforces a more centralised approach to interaction. However their level of equality, synchronicity, dependency, and Cognisability differ.

2.5.3 Surface Instruments

The concept of a communal surface that brings together several performers on one instrument has been investigated by some instrument builders. These instruments use visual feedback and direct tactile manipulation as a means for a group of people to cooperatively sculpt a musical performance. The benefit of this approach is its direct appeal to human senses and comprehensibility in how the group is interacting. The performers are usually locally located, manipulate a single sound source, allow for incidental group formation, and operate in the digital medium. The user number and role are usually very flexible, utilising communal resources

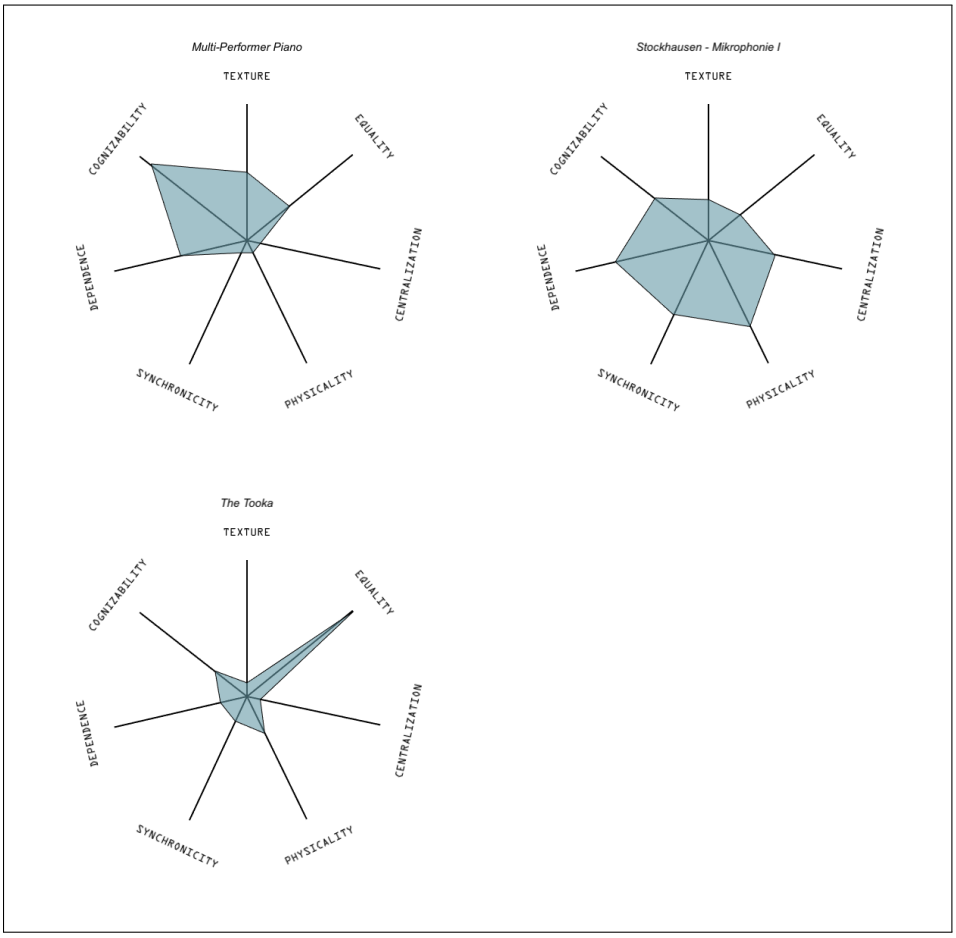

Figure 2: Collaborative dimension spaces for Extended Traditional Instruments.

that can be shared among the performers.

A well known example of a surface based multi-user instrument is *The Reactable*, a luminous table top instrument developed by Marcos Alonso, Gunter Geiger, Martin Kaltenbrunner and Sergei Jordà (Jordà, 2009). The instrument employs infrared visual analysis to track the position, orientation, and velocity of individuals pieces that are placed about the surface of the instrument. Multiple performers may stand around the table moving the piece around, taking pieces off, and adding new ones to change the character of the music. These pieces are affixed with fiducial markers that inform the system of what each piece on the table is, and how it affects the music. Oscillators, filters, mixers, decimators, and ring-modulators and other digital synthesis components are controlled by individual pieces. The proximity of these pieces from each other also plays a large role. When two pieces are placed next to each other a connection is made between the components that they represent. If a band-pass filter is placed next to a square-wave oscillator, then the oscillator is routed through the filter. The connections and audio data are projected onto the table to give real-time visual feedback of the digital synthesis that is being produced (Jordà, 2009).

Nan-Wei Gong, Mat Laibowitz, and Joseph A. Paradiso designed a multi-user instrument called the *MusicGrip*. This instrument, created in 2007 uses a specially constructed pen as a controller device, using stroke pressure and direction to modulate a digital instrument. The MusicGrip is able to played in solo but may also be calibrated to be used in group settings, with up to four performers at a time. In group settings the different pens control different parameters of a single synthesised sound, such as pitch, rhythm, amplitude, phase, and filter envelope (Gong et al., 2009).

Interval Research Corporation in collaboration with Tina Blaine and Tim Perkis designed a surface based drum-like instrument dubbed the *Jam-O-Drum*. Developed in 1999, The concept of the Jam-O-Drum is to extend the concept of a drum circle into the realm of electronics, and to encourage passers-by to engage in musical collaboration with each other. Furthermore, a game like graphics system is projected onto the Jam-O-Drum to serve as an interface for participants to interact with each other. Through the use of bouncing balls, turn indicators, and virtual "drawing", the spontaneous collaborators share influence over the sounds that the Jam-O-Drum produces (Blaine and Perkis, 2000).

JamSpace, created by Michael Gurevich in 2006, is a similar concept to the Jam-O-Drum that allows for multiple individuals to "jam" with each other by using a drum-pad based interface for electronics. However JamSpace is designed for participation by individuals over a distributed network, thus local-area connections or interactions are not required. JamSpace's hardware interface consists of a drum-pad like surface with twelve raised button that may be pressed or hit to

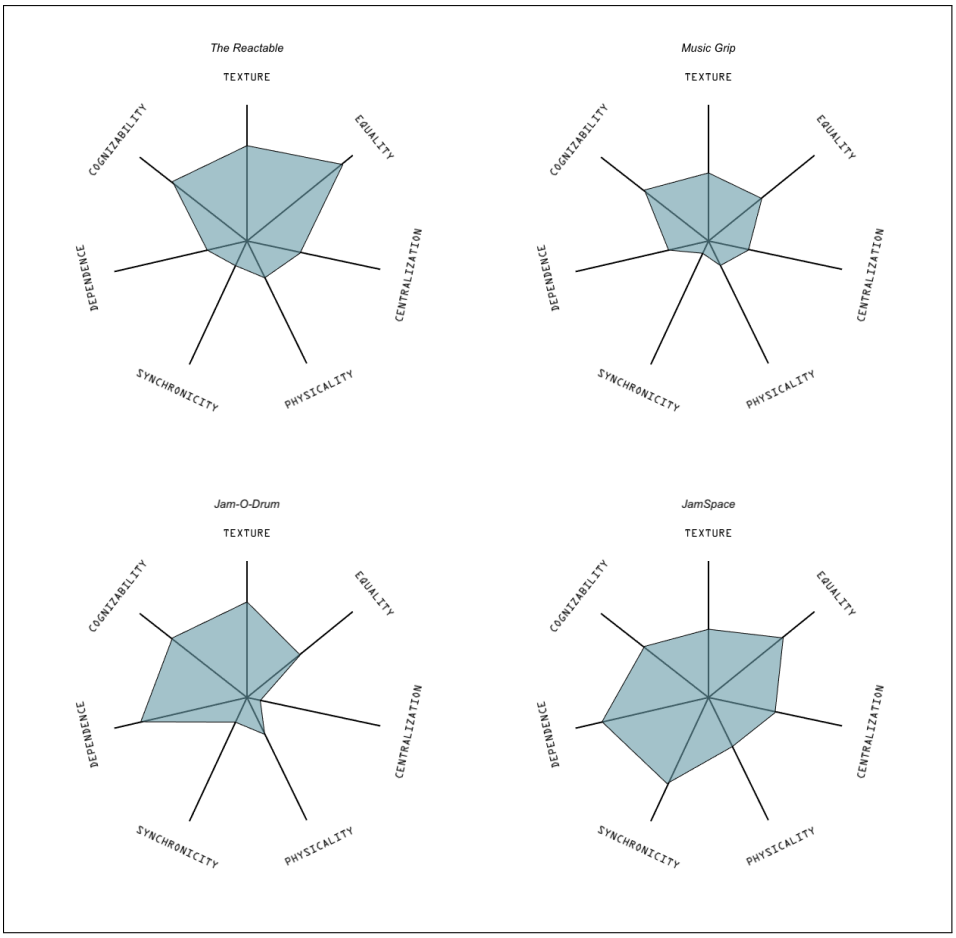

Figure 3: Collaborative dimension spaces for Surface Instruments.

activate different sound or pitches. The software component of JamSpace consists of a chat-room like interface that allows Individuals to create and share loops, as well influence and mix other participant's contributions (Gurevich, 2006).

The collaborative dimension spaces for each surface instrument may be found in Figure 3. These spaces show that there are some similarities in the collaborative interactions that occur when using a surface type instrument. The emphasis on physically activated interactions with others makes for high physical fixedness, a large amount of performer equality, heightened Cognisability, tight synchronicity, and more heterogeneous soundscapes. However there seems to be a degree of variance in the amount of interdependency in these systems. The *Reactable* and the *MusicGrip* both utilise performance systems where performers do not even necessarily "own" a sound of their and instead collaboratively alter a soundscape. This stands in comparison to the *Jam-O-Drum* and *Jam Space*, where performers may effect each other's sounds, but still have major control over their personal sound production.

2.5.4 Interconnected Laptop Ensembles

Interconnected Ensembles are placed in-between being an ensemble of discrete musicians and a completely self-encapsulated instrument. These instruments are comprised of multiple individuals, each with their own sound producing devices, who share information with each other over a digital network in such a way as to provide a strong interconnectivity in the overall system. Given enough interactivity and interconnectivity one could describe these ensembles as a sort of meta-multi-user instrument. Describing the properties of these "instruments" can be difficult as they usually reconfigure the system drastically from piece to piece. Often times, but not always, these are regularly rehearsing ensembles or bands, with strict user numbers. User roles can either be flexible or strict depending on the group, and there are wildly divergent systems for the distribution of power. The groups are often locally distributed, though the technology allows for distributed performance groups. Given the ease of use of this instrument groups may be incidentally formed by individuals newly introduced to the instrument. There are multiple articulated sound producing sources in the instrument, and the instruments utilise the digital medium over some kind of network.

The network computer music band The Hub, established in 1986, is an early (and possibly earliest) example of this model. Consisting of members John Bischoff, Tim Perkis, Chris Brown, Mark Trayle, Phil Stone, and Scott Gresham-Lancaster, The Hub grew out of the first network computer music band, The League of Automatic Music Composers. Utilising many different setups over their 25 year span, The Hub's main focus is interconnecting their individual computers to each other to produce spontaneous music that is not simply a summation of their individual inputs, but is instead created through the constantly shifting influence of the group as a whole. The Hub's philosophy equally emphasised individuality and interconnectivity. When creating a piece the group draws up a "spec" which defines the manner in which the network is to operate. Then each individual member is left to implement this spec in his own personal way, coding it in the programming language they chose, using their choice of peripherals(such as MIDI controllers, microphones, or instruments), and constructing their own synthesised sounds (Brown and Bishcoff, 2002). The aesthetic and music of The Hub will be covered in more detail in Section 3 as part of a larger discussion of the author's personal aesthetics and influences in creating music with network based multi-user instruments.

The Princeton Laptop Orchestra (PLOrk) is a collection of 15 networked laptops that function as a sort of interconnected orchestra. Formed in 2005 at Princeton University, PLOrk took much of its inspiration from earlier network music ensembles such as The League of Automatic Music Composers and The Hub. A

meta-instrument in PLOrk consists of a laptop running software synthesis programs Max/MSP, SuperCollider, and ChucK a rack of audio equipment, and a six -channel sound speaker array. These 15 meta-instruments are networked utilising the Open Sound Control communications protocol, enabling the individual stations to communicate with each other (Trueman et al., 2006; Manzo, 2011; Wilson et al., 2011; Wang, 2002). This same concept was later used to establish a similar ensemble at Stanford University's Centre for Computer Research in Music and Acoustics (CCRMA) called Stanford Laptop Orchestra (SLOrk).

In addition to the efforts by Ge Wang and Perry Cook with PLOrk and ChucK, there has been work to create a distributed interface for audiovisual collaboration entitled Co-Audicle. Based upon the work done with the audicle, the audio interface used by PLOrk to interface with ChucK, the Co-Audicle seeks to extend this to multimedia control, as well as to open it up to the possibility of collaborative control over a distributed network. Co-Audicle allows for several configurations including Server/Client and peer to peer connections, all the while maintaining synchronisation, security, and data consistency. Furthermore, Co-Audicle comes with a GUI engine for user-interaction called CHUI (Wang et al., 2005).

After the rise of PLOrk and SLOrk there have been a slew of academic ensembles that have arisen with a wide array of acronymed names including: Boulder Laptop Orchestra (BLOrk), Huddersfield Experimental Laptop Orchestra (HELO), Laptop Orchestra of Arizona State (LOrkAS), Moscow Laptop Cyber Orchestra (CybOrk), Laptop Orchestra of Louisiana (LOL), the Linux Laptop Orchestra (L2Ork), The Seattle Laptop Orchestra, The Tokyo Laptop Orchestra, The Berlin Laptop Orchestra, and the Birmingham Ensemble for Electroacoustic Research (represented by the humorous acronym BEER) (Boulder Laptop Orchestra, 2012; The Huddersfield Experimental Laptop Orchestra, 2012; Siwiak, 2012; Moscow Laptop Cyber Orchestra, 2012; Laptop Orchestra of Louisiana, 2012; Linux Laptop Orchestra, 2012; Birmingham Ensemble for Electroacoustic Research , 2012).

In 2005 Alain Renaud constructed a system for shared control over a laptop ensemble's overall output called *Frequencyliator*. Frequencyliator uses a shared time that is contained on a central server that modulates the properties of the group's overall sound. This time line may either be predetermined and created manually, or it may be algorithmically generated in real-time. This time line affects the allocation of frequency bandwidth in the ensemble, with each of the members only having an allocated chunk of the overall 22khz to create sound in. This produces roles that are akin to the different members of a traditional ensemble with treble and bass instruments, only these instrument morph and change over time. Furthermore, several systems are in place for group synchronisation and cuing, including a countdown notification system for section changes, and a *Sync Event* system that coordinates synchronised moments in the music (Rebelo

and Renaud, 2006).

Renaud's network ensemble The JacksOn4, consisting of Renaud, Tom Davis, Jason Geistweidt, and Jason Dixon, created network based multi-user instrument in a piece entitled *The Loop*. This instrument consists of four acoustic nodes that are geographically distributed across the globe. These acoustic nodes are tied together through a networked audio feedback loop. At each node this audio feedback loop is amplified through sheet of metal using a transducer, which is captured using a contact microphone attached to the same sheet of metal. This contact microphone feeds the resultant audio back into the loop. The sounds that emerge from this range from quiet rumblings to intense howls of feedback. The performers have a physical interaction with this sound and with each other through touching, bowing, striking, and scratching the metal sheets. This soup of sonic feedback provides a single strand that no singe performer has complete control of, and in some ways emerges as its own entity (Davis, 2012).

In 2004 George Hajdu created a dynamic score creation environment called QuintetNet that allowed for the possibility of up to five different musicians to connect to each other over the Internet and interact with each other. Audio is produced either through a sampler or MIDI, as well as through the usage of granular synthesis and VST plug-ins for sound processing. Each member of the ensemble has control over their individual score creation engine, and over the sound producing systems. These capabilities alone would not suffice to call this a multi-user instrument, but the added role of a "Conductor" in the group changes this. The conductor is a member of the group who has the extra ability to manipulate other user's instruments, including their timbre, sound processing, and tunings. The conductor may also create trigger sequences on the streaming time-line to change other users instruments over time (Hajdu, 2004).

The collaborative dimension spaces for several pieces from these interconnected ensembles may be found in Figure 4. Comparing these dimension space several themes for collaborative interactions in an interconnected ensemble become clear. Equality seems paramount in ensembles such as these, in an almost utopian fashion, with all of these pieces exhibiting greater equality than inequality. Even in groups such as PLOrK which utilise hierarchical organisation, there is still an emphasis on creating scenarios in which the members are sharing some kind of capability and influencing each other's systems in a democratic fashion. This emphasis on equality also spawns soundscapes that are more homogeneous than heterogeneous, with each of the members somewhat blending into each other, though not always to the point of indistinguishability. Likewise, given that these are performing ensembles that are locally located, they exhibit more centralisation than decentralisation.

A ramification of this emphasis on equality and homogeneity is a decreased

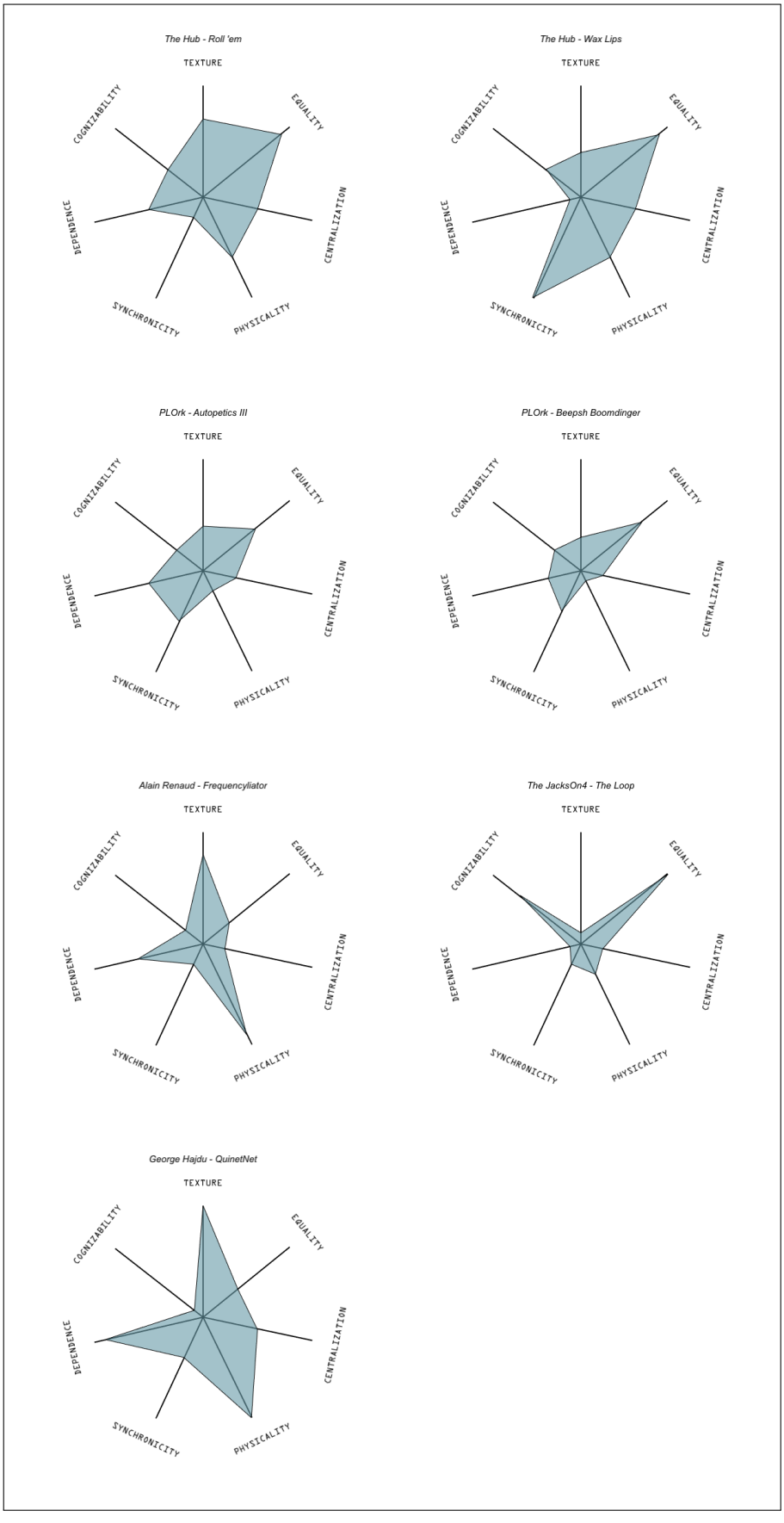

Figure 4: Collaborative dimension spaces for Interconnected Laptop Ensembles.

ability to cognisably distinguish the output of these systems. The homogeneously interdependent soundscape are made more amorphous by these ensembles general lack of physical fixedness (with a clear exceptions made by *The Loop*, where members physically handle sounds passed around the network. These ensembles also place an emphasis on synchronous relationships, where performers are free to interact simultaneously. However, *Wax Lips* provides a clear counterpoint, where the network is made purposely as sequential as possible, even to the point of network failure.

2.5.5 Cloud Instruments

Cloud Instruments are ephemeral entities that exist entirely on some kind of network, allowing complete strangers to dynamically connect to each other. These incidental group formations are core to the ideals of this approach, allowing anyone(with a network connection), anywhere, to collaboratively perform on a multi-user instrument of high interactivity. Often times there are multiple sound sources, one at each network node, each rendering a mirror of the current virtual instrument's properties on the local computer. The user number and user roles for these instruments are dynamic, with a wide range of hierarchies that emerge, from groups of completely equal peers to dictatorial formations of power.

An early example of this manner of multi-user instrument is Max Neuhaus's piece *Public Supply*, created in 1966 and its larger scale follow up *Radio Net*, created in 1974. In *Public Supply*, Neuhaus collaborated with the radio station WBAI in New York to allow him to have an hour of radio time where callers on up to ten lines could phone in to the station and make any sounds they desired. Neuhaus also told participants, without initially realising the full depth of the consequences, that they could have the radio on in the background as they performed, to have some of the sound of the station feed back in. The result was more than Neuhaus had hoped for, and an otherworldly sound emerged that was a conglomerate of up to ten different sites across New York feeding back into each other, excited by the sounds that participants fed into the system.

Neuhaus developed this concept further, eventually resulting in the piece *RadioNet*. Similar in structure to *Public Supply*, however instead of there being only one radio station that routes audio Neuhaus interconnected five National Public Radio (NPR) stations across the United States of America, each receiving hundreds of call. Furthermore, Neuhaus processed these sounds at each site, using frequency shifting and audio mixing to affect the total sound. Neuhaus also persuaded participants to whistle into their phones, which when fed into the system created clouds of moving pitches that ebbed and flowed over the network (Neuhaus, 2010).

A 21st century take on the concepts explored in *RadioNet* can be found in Pedro Rebelo's piece *NetRooms: The Long Feedback*. In much the same way as *RadioNet*, *NetRooms* creates a feedback loop between multiple sites, only instead of a radio network, audio is shared over the internet. Central to the concept of NetRooms is a shared environment. A feedback loop is created through the use of an open air microphone capturing the sounds of the participant as well as the sounds of the loop being produced by a loud speaker. This instrument may contain a flexible number of sites and is open to any individuals who wish to participate, the requirements being a laptop with a microphone and speaker, Pure Data with a custom patch, and a broadband Internet connection. Participants are encouraged to improvise with each other using any acoustic sounds or instruments they like. The sounds range from the sounds of free improvisation displaced and delayed over time, to swells of tones and pitches that emerge when resonances cause the feedback loop to oscillate (Rebelo and King, 2010).

WebDrum is a web based multi-user instrument developed by Phil Burk that allows multiple users to simultaneously create drum loops. The system employs the TransJam Server and JSyn systems developed by Burk, which allows separate clients to log into a central server which controls remotely rendered synthesis on each individual client computer. The usage of synthesis only severely cuts down on bandwidth requirements, as no actual audio is shared between the server and the clients. Furthermore the system is designed to be cyclically perceived and performed, based on the paradigm of a drum loop, so that issues of client synchronicity are alleviated Now updated to WebDrum II, the system allows for up to six users at once (Burk, 2000). Each user takes the role of editing one instrument at a time, with the system being comprised of eight instruments, six drum based instruments, and two melodic instruments.

Max Neuhaus himself also has created a more modern take on this concept with his piece *Auracle*. Conceived by Max Neuhaus and realised by C. Ramakrishnan, Jason Freeman, and Kristjan Varnik, *Auracle* is an online multi-user instrument that uses the human voice as a means of control over the system. Utilising the java based JSyn synthesis engine and TransJam server systems developed by Phil Burk, the system allows performers to control a synthesiser using their voice. High level gestural analysis of streams of vocal data is collected from multiple performers with the aggregate data shared across the network and used to drive synthesis that is remotely rendered on each performer's computer.

This system does not require large amounts of bandwidth due to it only requiring analysis data to be sent across the network. Also, issues of synchronicity are sidestepped due to the instrument being controlled by a higher-level gesture analysis, as opposed to real-time analysis. The system can support up to 100 users, with there being one role that each user takes on. Each user has essentially

equal say in the system's output, and due to it's web presence the system allows for spontaneous groups of strangers to form and make music together (Ramakrishnan et al., 2004).

Peer Synth is an online multi-user instrument developed by Jörg Stelkens. Like WebDrum, PeerSynth employs remotely rendered synthesis, however it does so through peer to peer connections instead of relying on a centralised server. This allows PeerSynth to be more decentralised and less reliant on a single node for it to function. With regards to latency, instead of trying to eliminate or hide it's effects, PeerSynth attempts to aestheticise it. PeerSynth constantly keeps track of the latency between each peer of the peers and uses this as a musical parameter to modulate characteristics of the sounds being generated (Stelkens, 2003). Each performer controls a single synthesiser, which can use either an oscillator or a sample as a base sound. This sound can then be routed through effects, and modulated using physically modelled control schemes. Each instances of this user's synthesiser is replicated on each other user's computer.

An instrument that is similar in use and execution to PeerSynth is Juan Pedro Bolivar Puente's software version of the Reactable called *Psychosynth*. Psychosynth, just as the actual Reactable, contains various synthesis modules that interact on a virtual surface to create sounds and music that changes based on user manipulation. Unlike the Reactable, Psychosynth allows for any user to download the software and connect to other users on the Internet to construct and control a single instrument together spontaneously. This capability is notable as the official *Reactable Mobile* software does not support this functionality (Puente, 2011).

Another instrument in the same vein as PeerSynth and Psychosynth is *MOLS*, or Multiple Performer Online Synthesizer, is a web-browser based online performance instrument created by Jorge Herrera. With MOLS, multiple user may simultaneously manipulate a synthesis engine that is capable of FM Synthesis, Granular Synthesis, and various filters, all based on the Synthesis Tool Kit. For networking MOLS utilises Adobe's *Stratus* and *Real Time Media Flow Protocol*. The GUI in MOLS is modelled after Pure Data and Max/MSP, being essentially boxes that represent UGens connection to other boxes through lines. The extra feature here of course being that multiple users may edit the same patch simultaneously (Herrera, 2009).

The collaborative dimension spaces for each cloud instrument may be found in Figure 5. When viewed together these instruments clearly place emphasis on decentralisation (with a clear exception made by the server based *Auracle*. This type of interaction seems natural, as these instruments seem to emphasise an ideal of performance by anyone, anywhere, at any time. Following from this is an ideal of equality amongst the members which free them to pursue synchronous

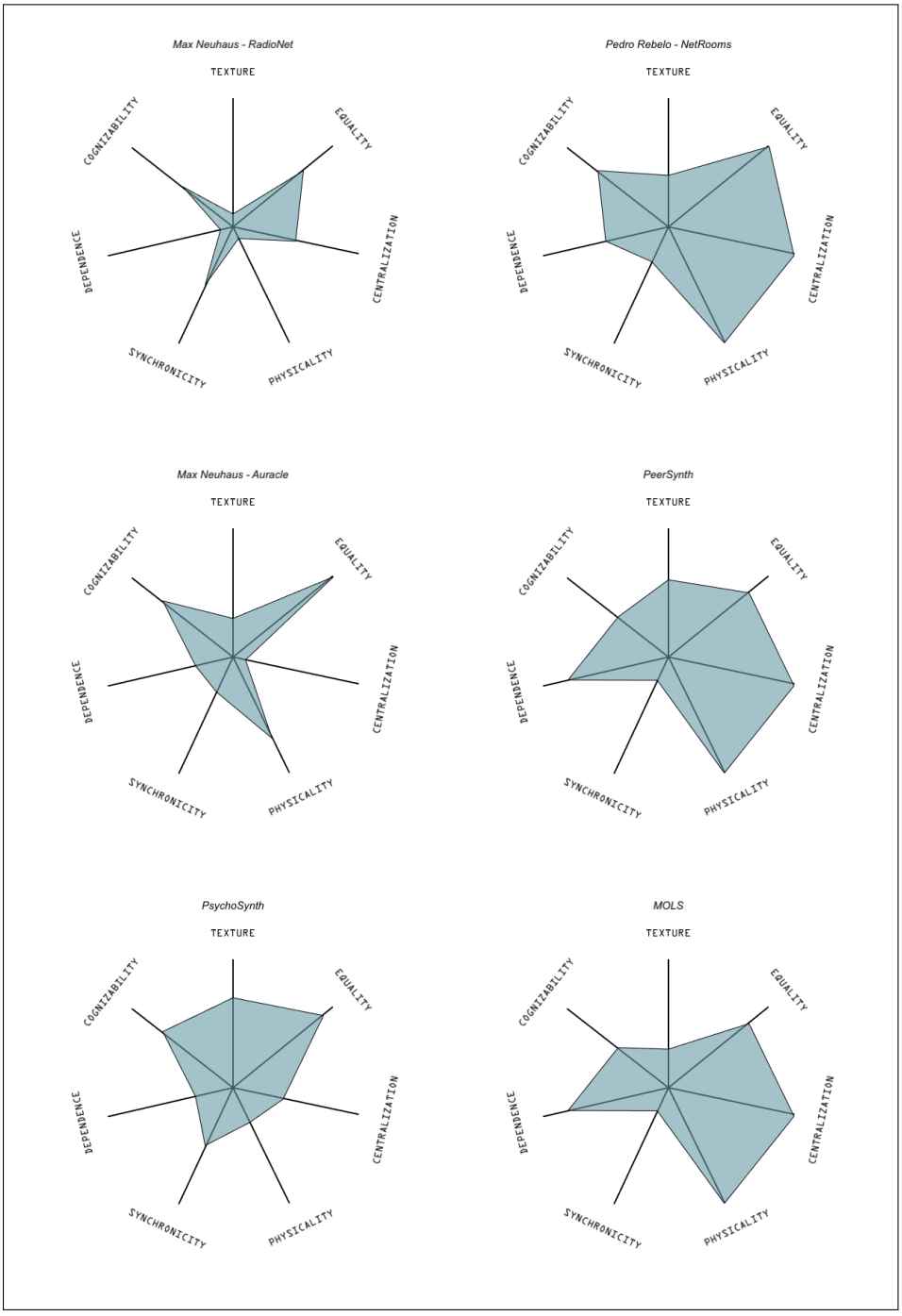

Figure 5: Collaborative dimension spaces for Cloud Instruments.

interactions. These instruments usage of GUI interfaces tends to increase their cognisability. However, there seems to be a large amount of variance in the degree of interdependency that these instruments choose to employ.

2.5.6 Kinetic Group Instruments

Some instrument makers have investigated instruments that allow multiple people to exert physical control over them. These instruments analyse the individual and group movements of the performers and use this data to modulate aspects of the music being produced. A shared sense of physicality and tactile connectedness are central themes for many of these types of instruments.

Sensorband, an experimental electronic music band focused on gesture controller based instruments comprised of members Atau Tanaka, Zbigniew Karkowski, and Edwin van der Heide, created an instrument called *SoundNet* in 1996 that focused on this sense of group physicality. In *SoundNet* the three members of Sensorband crawl, jump, and climb along a large stainless steel web that is affixed with eleven tension measurement sensors that measure that load on each of the steel wires. These sensors feed the tension data into a central computer that synthesises sound based on the incoming data. Each of the three members have the ability to exert some individual control over the instrument, however the performers are fundamentally linked, as all the wires are interconnected, and thus each of their individual movements and weights are distributed, summed, and differenced as a whole across the web. Performances of SoundNet are extremely physical and somewhat chaotic, as the musical interconnections the performers experience are literally played out as physical oscillations on a web (Bongers and Sensorband, 1998).

Tanaka, along with collaborator Kasper Toeplitz, later created *Global String*, a work that embodies some of the principals of *SoundNet*, but takes them to a wholly different place. *Global String* is a network based multi-user instrument that creates a "String" that virtually spans a large geographic distance and interconnects remote sites onto a singular sonic strand. A steel cable is installed at each site that is connected on one end to the floor, and the other to a point high on a far wall. Vibration sensors are installed on each site, and the physical vibrations from one site's string are streamed to and transduced to the other site's string, creating a virtual tactile bond between the two strings. Video of the string and the participants engaged with it are shared between sites. Furthermore, physical modelling is used to synthesise the sound of a string whose length is the distance between the two geographic sites. This synthesised string sound is activated and modulated by the participants interaction with the string at each site via the vibration sensors. Despite the distance separating the individuals at each site during the

piece, *Global String* serves as a means of human connection via a single shared instrument (Tanaka and Bongers, 2001).

The *Sensemble* is a system created by Ryan Aylward and Joseph A. Paradiso to be used by a dance ensemble. The Sensemble is a series of wireless inertial sensors that are used to allow the gestures of the dancers to influence music that is being produced as they dance. What makes the Sensemble interesting is that unlike other gestural controllers that provide direct one-to-one mapping of some axis of control to a musical parameter, the system instead creates a cross-covariant average of all of the sensors worn by all of the dancers simultaneously. This novel approach seems rather logical given the context of the instrument: a choreographed group of dancers attempting to perform in synchronicity. Of course it is impossible for a group of human dancers to keep their motions perfectly synchronised, so the system accounts for this, and the average of the overall group movement is used to influence music (Aylward and Paradiso, 2006).

Another example of theatrical kinetic group instrument is MIT's efforts to create a juggling controlled multi-user instrument. In 2001 the MIT Media Laboratory developed a musical sound stage that reacted to the incidental movements of the juggling performers The Flying Karamazov Brothers. The instrument consists of a complex amalgamation of sonar emitting headgear, accelerometers attached to wrists, wearable computers, and a master computer with special tracking algorithm software. The summation of all of this data is then mapped to MIDI to trigger musical notes, as well as to cue visuals projected on to screen, and to dynamically change the colour of the clubs the Karamazov Brothers were juggling (Reynolds et al., 2001).

The four person Australian ensemble *Metaform* explores virtual immersion and interaction on a virtual terrain in their work *Ecstasis*. In this work each of the four members wear head-mounted displays equipped with motion-tracking sensors. There are four screens that display the views of each member into a virtual terrain. The spatial relationship of the four avatars on this terrain modulate various aspects of the virtual world including its colour, shape, transparency, and affecting the sound of the world by mixing 16-channels of audio, as well as applying various processing to that audio(Kim-Boyle, 2009).

The rising popularity and usage of mobile phones in networked instruments could be seen as a sub-genre, if you will, to kinetic group instruments. In 2007 Ge Wang and others at CCRMA established the mobile equivalent to the Laptop Orchestra: the Mobile Phone Orchestra (MoPho). MoPho is similar to SLOrk and PLOrk, but instead of laptops with six-channel speaker arrays MoPho wields accelerometer-sensored iPhones and specially created "Glove-speakers." These mobile devices allow members of MoPho to traverse across the stage and even through the audience to provide dynamic acoustic specialisation (Wang, 2010).

Nathan Bowen created a four person mobile-phone based musical instrument catered to non-musicians entitled 4Quarters. 4Quarters makes use of technologies common to touch-based controllers including iPhones, OSC/touchOSC, and Max/MSP for a sound synthesis engine. Multiple participants may manipulate a single layer or multiple layers of sound by assigning modulation of various parameters of the layer, such as sound selection, pitch, panning, and equalisation, to the motion sensors of their phones. Furthermore an overview of the state of all the layers and the current role of each member is projected on a central screen for all the users to see (Bowen, 2012).

The concept of a shared physicality and social musical interaction is something that composer and luthier Gil Weinberg has explored with several multi-user instruments. The Beatbug was developed by Gil Weinberg, Roberto Aimi, and Kevin Jennings in 2000. The design of the instrument emulates the anatomy of a bug, with a speaker for a mouth, two bend sensors for antennae, and a drum trigger on the back of the bug body. Each member in the group has their own Beatbug, with each Beatbug wired into a centralised equipment rack and computer system. The computer system interprets the actions of the different performers and generates music through a series of sharing and manipulation algorithms. The music system for the Beatbug emphasises rhythmic motif sharing and development. Players create their own rhythmic motif by playing rhythms on the body and by altering timbre by manipulating the bend-sensors/antennae. These motifs are then automatically sent to other members in the group who develop the rhythm further (Weinberg et al., 2002).

Gil Weinberg and Seum-Lim Gan developed a multi-user instrument called The Squeezables. Physically the instrument consists of 6 squeezable gel balls attached to a table. Inside each of these ball are five pressure sensors, providing multiple axes of control in each ball. The pressure sensors in each of the balls control different parameters of the same synthesiser, a Nord Lead2, including pitch, arpeggiation, amplitude modulation, and different timbral controls (Weinberg and Gan, 2001). Each ball has one role, such as melody, rhythm, or timbral controls. However, individuals may trade balls with each other, which allows for shifting roles and leading to hierarchical scenarios where negotiation is required between members to produce different musical effects. Another unique characteristic of this instrument is its ability to give the performer a tactile relationship with the sound being produced, often times missing in other digital instruments.

Weinberg and Gan state that one design goal was to create an instrument that inhabited a region in between direct control of the sound by each individual, and interdependency in the group. For example, squeezing the melody ball alters the frequency of the melodic voice on the synthesiser, but squeezing it will also change the tonality of the accompanying arpeggio voice being controlled by

a different ball. Unlike the interconnections in an instrument such as those in Mikrophonie I, where the instrument is a complete melding of individual members, these types of interconnections are meant to allow each performer have a feeling of ownership of a specific voice in the music, while also providing a sense of communal interaction with those around them.

The collaborative dimension spaces for these kinetic group instrument may be found in Figure 6. These ensembles seem to naturally enforce a certain kind of collaborative interaction. Looking at these dimension spaces it becomes readily apparent that equality, homogeneity, and interdependency. These instruments involve groups of individuals in one physical location physical interacting with each other, and thus there is also a large degree of centralisation, physical fixedness, synchronicity, and cognisable are all greatly emphasised. In this way they are actually rather similar to the utilitarian traditional instruments, however replacing the completely hierarchical relationship of the pipe organ and organistrum with much more equal performer relationships.

2.5.7 Multiplayer Game Instruments

Multiplayer Game Instruments use game structures as a central mode of musical performance. The distributions of power in these systems are interesting in comparison to other instruments, as the relationships between the multiple players can range anywhere from collaborative to combative (Shim et al., 2011).

On March 5th, 1968 John Cage, David Tudor, Gordon Mumma, David Behrman, Lowell Cross, Marcel Duchamp, and Teeny Duchamp engaged in a performance entitled *Reunion*. At this performance a special chessboard was retrofitted by Cross with photo-resistors and audio mixing electronics to act as a multi-channel matrix mixer for incidental electronic music being generated by Tudor, Mumma, Behrman, and Cross. Two games of chess were played on the board by Cage and the Duchamps (first against Marcel, then Teeny). As the pieces moved around the board the 16-channels of audio provided by the four sound generating members were gated, ungated, and spatialised around the 8-channel sound system placed around the audience (Cross, 1999).

The author's ensemble, *Glitch Lich*, had a previous incarnation whose name, *LAG*, was an acronym for The League of Art-Game composers. Starting off as a tongue-in-cheek joke based on their professor's band, The League of Automatic Music Composers, the group did eventually create several multiplayer game pieces. The first of these pieces was *Samurai Showdown*, composed by Chad McKinney and Curtis McKinney. *Samurai Showdown* sought to explore the compositional possibilities of combative performer relationships. In the piece two performers play a classic two-player NeoGeo arcade fighting game called

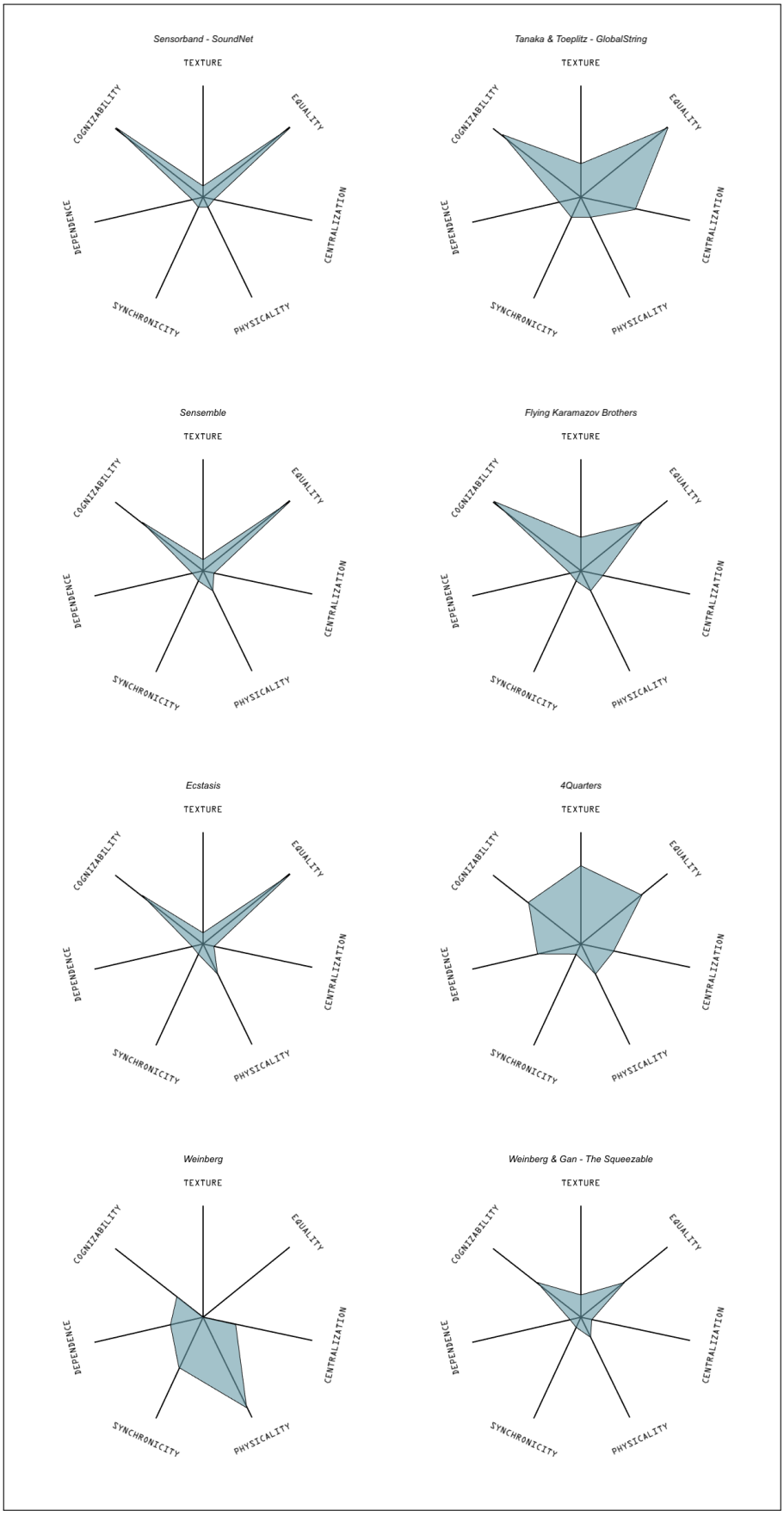

Figure 6: Collaborative dimension spaces for kinetic group instruments.

Samurai Showdown (SNK Playmore Corporation, 1993). This game is emulated on a computer using Mame, and the commands for the player controls are input via two USB game pads (Salmoria, 1997). These control inputs are also simultaneously routed to Max/MSP, which converts the player movements into synthesised sounds. These synthesised sounds are split into two groups. In the first group, there are sounds that are keyed off the players input action directly, and map one to one, one button push to one sound activation. On top of this runs a longer more atmospheric sound that was based upon the progress of the game and the life totals of the virtual fighters in the game. This "game progression" metric and the life totals of the fighters were measured by a Jitter application. This Jitter application analysed the screen output by Mame and could measure the life totals of each fighter in the game at any point in time. Each fighter's life total had their own background associated with them, and as that fighter's life total diminished the output of the background sound became less ambient and more erratic. The game progression metric was tied to the number of rounds the two fighters played, which was usually decided as best two out of three. At the very end when a player decided the final striking blow a massive noise volley signalled the winner.

The second of these pieces was LAG's version of *Pong* composed by Cole Ingraham. *Pong* followed in the vein of *Samurai Showdown* in that it pitted two performers against each other in a meeting in games and music. However, while *Samurai Showdown* had two performers playing a game and then using those incidental actions to control music, Ingraham approached the problem from the flip side of the concept, by having two acoustic instrumentalists play music, analyse the acoustic properties of those instruments, and use that data to control a virtual game of Pong, the two person paddle game originally created by Atari. To control the paddles the two performers play a stream of notes up and down different registers of their instruments. The the pitch a performer was playing would dictate the position of the virtual paddle. Thus high notes resulted in the paddle being on the screen and lower pitches resulted in the paddle being lower. Furthermore there is a synthesised sound that tracks the position of the ball as it travels across the playing field.

Red-king Snoring vs The Octopus Knight, composed by Chad McKinney and Curtis McKinney was the ensemble's take on a chess game, influenced by Cage's *Reunion*. In this piece, a magnetic sensored chessboard tracks the position of each individual piece on the board. This locational data is sent to SuperCollider which analyses it and outputs sounds based on that analysis. Each piece on the board has a single sound associated with it that stays dormant until the piece is initially moved. The location of a piece on the board modulates two parameters for that piece's sound, dictated by the X and Y coordinates of the piece. When a piece is captured there is a sonic flourish and then the piece's sound disappears from

the sonic texture. Whenever a player's King is put into check by the opposing player, a foreboding clang is produced, increasing in shrillness as the number of available exits are reduced, until finally Checkmate occurs, at which point a final noise flourish is summoned. *Red King Snoring vs The Octopus Knight* produces a visceral sonic representation of a Chess game, the density of movements that have been made, and the pieces in motion, as well as physically manifesting some of the strategic aspects of the game.

Finally, the group created a piece based on the Japanese card game *Koi-Koi*, a traditional betting game based on matching cards and card types that has similarities to poker and matching games such as *Memory* (Japan Publications, 1979). In a similar vein to *Samurai Showdown* and *Red King Snoring vs The Octopus Knight* this piece sonifies the incidental actions of two performers playing a game. To analyse these actions in *Koi-Koi* authors Chad McKinney and Curtis McKinney constructed a home-brewed version of the *Reactable* system, using the open-source *Reactivision* software used by the *Reactable* to track positions of their specialised bar codes called *fiducials*. The cards used in the game were affixed with individualised fiducials and as the game is played the cards are placed onto a glass table where their position and orientation are tracked via an infrared camera placed underneath the glass. Similar to the *Reactable*, each card in the deck has an associated sound producing or processing system. As the game progress, a signal chain is dynamically created out of the sound systems associated with the individual cards. Thus the unique progression and combinations of cards produced a new and often surprising signal chain every match (McKinney, 2009).

Sound Pong, created by Jon Bellona and Jeremy Schropp, is a Multiplayer-game instrument and Kinetic-group instrument combined. Through the use of *Wii-motes* and projected visuals a group of four individuals play a game of pong. During the course of their game their incidental game movements are mapped to parameters of a sound producing engine. The various sounds controlled include four banks of four samples, modulated individually by each player, incidental sounds such as opening/closing theme music and crowd noise, and a sound whose properties are mapped to the x/y coordinates of virtual ball being bounced by and forth by the actions of the player (Bellona and Schropp, 2012).

The collaborative dimension spaces for these multiplayer game instruments may be found in Figure 7. Multiplayer games instruments seem to exhibit many of the same collaborative group interaction characteristics as kinetic group instruments. Like the kinetic group instruments these game instruments employ interactive relationships that have large amounts of equality, due to the zero sum nature of the game play, centralisation, as the groups play these games locally, more fixed physicality, with specific game actions triggering specific sounds, and

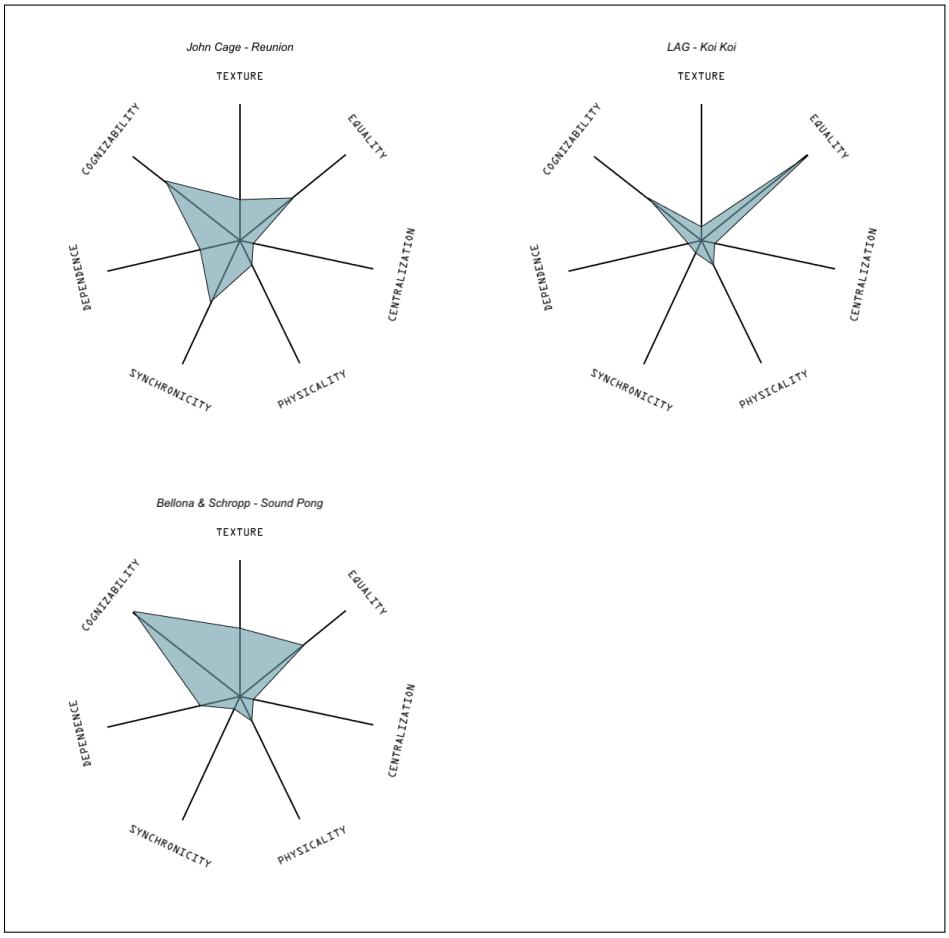

Figure 7: Collaborative dimension spaces for multiplayer game instruments.

interdependency, with the sum total of the game actions producing the soundscape instead of individuals playing individual instruments. This seems logically as both categories of instruments involve a group of people physically interacting with each other to create locally networked music. The cognisability of these instruments is emphasised to a large degree by the inherit relationships established between the game being played out visually and the sounds that these games produce in reaction to the game actions. There does seem to be somewhat more variability in the textures and synchroncity of these instruments in comparison to kinetic group instruments, though this is a small amount of variance and they still tend towards homogeneity and synchronous interactions.

2.6 A Final Word on Dimension Spaces

Interestingly the dimension space values mostly do not seem to have a clear connection with the categories that have been created for this survey. However there are two categories that seem to show some connection between the dimension space values of the instruments in those categories. First are the utilitarian multi-user instruments. The physicality of these instruments dictates that they have similar qualities as they largely avoid the pitfalls of fully electronic instruments. It is understandable then that they should share characteristics such as high centralisation (the members must be physically present), high dependence (the instrument won't make sound without both performers), high physicality (they are acoustic instruments after all), high synchronicity, high dependence, and high cognisability. Another group that displays a pattern in their dimension spaces are kinetic group instruments. Once again it seems to be the physicality of the instruments that produces a kind of natural form to their collaborative dimension spaces. The physical displays inherent to the instruments make for high cognisability (as there are one to one interactions between physical actions and sound), high centralisation (they are physically present), high physicality, tight synchronisation, and high synchronicity (as all performers perform simultaneously). It seems that physicality in instrument design seems to create a kind of collaborative template for multi-user instruments.

2.7 Conclusions

In this chapter research related to aim # 1 was discussed. A clear definition of multi-user instruments in general, and NMIs in particular, was established. To repeat, the definition of a multi-user instrument is: a musical instrument, piece, or ensemble, wherein multiple individuals have shared performative control over a single sound-producing source or engine, or where the connections in a network

of discrete sound producing sources or engines controlled by separate individuals achieve a sufficient level of interconnectivity that it is difficult to differentiate between those discrete sources and a group whole. The definition of an NMI in particular is: a multi-user instrument who has as one of its central structural or aesthetic focuses the usage of digital networks, often, but not strictly, related to The Internet.

To achieve this aim several objectives were completed. These objectives (numbered one through four in the Introduction chapter), and the work conducted to achieve them, are as follows:

1. *Conduct a survey of Multi-User Instruments* - A survey of instruments ranging from traditional instruments to cutting edge technology was conducted.
2. *Create a taxonomy for organising Multi-User Instruments* - A taxonomy was created, yielding an organisation of multi-user instruments into seven discrete types: Utilitarian Multi-User Instruments, Extended Traditional Instruments, Surface Instruments, Interconnected Laptop Ensembles, Cloud Instruments, Kinetic Group Instruments, and Multiplayer Game Instruments.
3. *Create modes of analysis for Multi-User Instruments* - A dimension space has been created that may be used to analyse the properties of multi-user instruments. This dimension space includes seven axes of analytical concern: Texture, Equality, Centralisation, Physicality, Synchronicity, Dependency, and Cognisability.
4. *Employ these new modes of analysis and taxonomies on the instruments found in the survey of Multi-User Instruments* - The instruments covered during the survey were presented in sections according to the groups associated with the taxonomy created during this research. As well, many of these instruments had dimension spaces generated for them. Furthermore, these dimension spaces were compared and contrasted over the course of the survey.

3 Aesthetically Driven Iterative Design Methodology

The second aim of this research is to ”Create new, and refine old, tools and techniques for composing, performing, and designing such instruments”. This chapter details the establishment of a methodological framework for designing NMIs, in accordance with objective number four (in accordance with the objectives presented during the Introduction). Using this methodological framework a design space and several digital software NMIs were created. That work is covered in Chapter 4.

The software created during this research has followed an iterative methodology similar to that found in the *Agile* software development framework as laid out by the Beck et al. (2001) in their Manifesto on agile software development. In a traditional design approach, often called the *Waterfall Model*, a requirements specification is predetermined, followed by a design plan to satisfy these requirements. These designs are then implemented in software, followed by verification and maintenance (McConnell, 2006). Each of these phases follow sequentially from one to another until the software is considered to be finished. Alternatively one may employ an iterative approach. According to Cockburn (2008) an iterative design process is one in which throughout the development cycle time is periodically set aside to review and improve parts of a system in a cyclical fashion until a final product is deemed worthy. The development cycle pursed during the course of this research is executed in such a fashion that it is iterative in both a self-contained method inside of a single project, as well as being iterative from one project to the next, with each subsequent project gaining insight from the previously constructed projects.

Furthermore, the design requirements of the software developed have been directed by the specifics of the aesthetics and performance practices of the band Glitch Lich. This ensemble serves as the vehicle for development and test bed for the software produced in this research, including several NMIs as well and the software tools that support them. This is a similar mode of working as taken on by John Bowers in his Master's thesis *Improvising Machines: Ethnographically Informed Design For Improvised Electro-Acoustic Music*. Bowers's methodology involves first the creation of a musical aesthetic specific to his idiomatic manner of musical creation. According to Bowers, this musical aesthetic *configures the design space and enables certain preferences to be articulated where, without, one might face a crisis of motivation and indecision in the face of equally appealing yet faulty alternatives* (Bowers, 2008).

Glitch Lich is a networked ensemble of individuals and thus it is naturally

collaborative in nature. Each member has their own somewhat different aesthetics and approaches, and the pieces that each individual composes for the band attacks the creative problems of NMIs differently. This chapter will cover some of the aesthetic and technical approaches made by Glitch Lich. This aesthetic has been cultivated by the ensemble over the course of five years of existence, and is partially documented in the paper "Glitch Lich: Evolution of an Intercontinental Network Band" in the proceedings of the first Symposium for Laptop Orchestra and Ensembles (SLEO), which includes insight into the ensemble's approach to music making from the point of view of each individual member. (Ingraham et al., 2012). Glitch Lich's approach to creating NMIs, and composing music for them, explores four conceptual areas:

- *Linearity, Improvisation, and Aleatory*: The opposing yet synergistic compositional tools of linearity, improvisation, and aleatory are three fundamental forces whose effects create ripples in both the immediate visceral intake of the music Glitch Lich is attempting to create, as well as the deeper conceptual ideas that drive the pieces as a whole.
- *Beauty, Play, and Viscerality*: Fundamentally, these are the ideas or perhaps energies that the ensemble is attempting to project with its music. These are in essence the things that the band has to say and are essential to why the ensemble makes music.
- *Collaboration and Camaraderie*: While this might seem mechanistic, the collaborative method of constructing these works themselves become an integral factor of the performance of much of Glitch Lich's music, and thus become part of the aesthetic itself.
- *Performance and Liveness*: The problems of live performance in electronic music go well beyond the scope of this dissertation as it touches any composer who attempts to "perform" a piece of electronic music. The band has cultivated its own methods for dealing with these issues in the especially troubling scenario of intercontinental network music. However, many of these "solutions" are in fact aesthetic approaches, which are rooted in how the ensemble would like its music to be perceived.

3.1 Linearity, Improvisation, and Aleatory

The three forces of linearity, improvisation, and aleatory are some of the most fundamental forces that drive Glitch Lich's compositions. Each of these forces can be seen as fundamentally beautiful, but also wrought with peril. It is for these reasons that Glitch Lich attempts to bag them up, force them to make nice, and attempt to coalesce a composition who's character is derived at through equal

measures of each. It's an attempt to harvest the merits of each, and utilises these merits to patch over what could be seen as problems that these forces exhibit in isolation. In essence, it's compositional alloy.

According to Kramer (1988), linear music is music in which one event implies another, providing a succession of gestures that flow from one point to the next. A simple example of this is the leading tone of a scale naturally implying its resolution to the tonic. This is essentially what is taught in classical music composition courses, being the distillation of intuitive ideas and methodical concepts into form which is sculpted endlessly, picked over meticulously, crafted, bent, chopped, and screwed, until the parts align like the stars to produce a pristine specimen replete with intention. At every step of the way the composer is in control, a god in his own microcosm. In classical music this concept runs even deeper as the end product is not even the music itself. Instead the end product is the directions for achieving the most ideal form of the music, which may never be truly achieved.

It is this struggle for perfection, that most summarises the ensembles attitude towards linearity. All of the members of the ensemble have studied Western classical music performance and composition. To a large extent it is something that has been ingrained into each of us. At its best it provides beautiful form and energetic direction to a composition, soaring it to great heights. This form of composition however begets a certain form of thinking of the world. Linearity is inherently teleological, and an overly composed work could be construed as implying that there is some form of plan underlying the machinations of the things around us. However, the world is in reality rarely this composed, and is instead filled with imperfection, inconsistencies, inefficiencies, ambiguities, and humanity itself. At its worst, when compared to fundamental truths, the lofty heights that linearity attempts to push us can seem flimsy, kitsch, overwrought, and disingenuous. It's draconian struggle for perfection can also seem overly restrictive for performers in an age where everything is interactive, responsive, and changing. However, if used in the right measure, there is nothing that can quite replace its powers to move individuals.

In stark contrast to linearity is improvisation. In Glitch Lich's approach, improvisation is the efforts to empower the performers to creatively respond in the moment itself, adapting to changes in the music, the audience, the surrounding, and changes in the performers themselves. It is the equivalent of musical tactics versus the musical strategy of linearity. This sense of immediate action and unpredictability also engenders a sense of danger in improvisation, as you remove the safety net of predictability. Borgo (2007) states that when improvising there is a "leap into the unknown or the uncharted, the adrenaline rush that can accompany the excitement and danger of an uncertain future, the mandate to make

something happen - to swim - or else..."

Improvisation allows for unmatched mobility in music, giving the performer a sense of agency that without which could otherwise make performing a composition seem robotic. Ancient Grecian philosopher Heraclitus states that "No man ever steps in the same river twice, for it's not the same river and he's not the same man (Kahn, 1980) ". Improvisation gives the performer the ability articulate that sense of constant change. However, at times improvisation can become aimless, with no clear direction driving it, no destination in mind, meandering and amorphous. At its very worse it can seem pointless (most likely more so for the audience than the performer). And it is for these reasons that Glitch Lich uses linearity and improvisation as natural partners, despite their differences. Combining these elements can produce musical pieces in which there is a clear sense of direction driving the music, while some of the details may be massaged or wholly changed on the spot, to give the audience and the performers the sense that this is "actually" happening, that the music is alive, that it is breathing.

Aleatory is by far the newest technique of the three here, mainly developed and used since the 20th century by such composers as John Cage, Christian Wolff and Morton Feldman (Pritchett, 1996). In Glitch Lich's performance and compositional practice, the ensemble defines aleatory as the techniques utilised which produce non-deterministic results, and which are outside of direct control (but not necessarily influence) of the composer or performer. Some of the effects may be essentially stochastic in nature, being that there is a deterministic framework, a bounded scope of possibility, but the actual details are not set in stone, and therefore each performance exhibits its own characteristics. Some algorithmic techniques may offer wholly unpredictable and emergent behaviour. Techniques that produce effects of this nature can be found by harnessing the extremely powerful forces of recursion and feedback; the idea that something bears itself, folds in on itself, produces itself. This is the concept of Ouroboros (the snake eating its own tail) and the eternal return (Neumann, 1954). For Glitch Lich these forces are more than simple techniques that produce novel results, they are primal forces that may be central to life , and to the fundamental make up of humanity itself.

Douglas Hofstadter in his book *Godel, Escher, Bach: An Eternal Golden Braid*, makes the case that certain forms of recursion (which he calls *Strange Loops*) are the fundamental basis for cognition itself, and thus the seed for humanity and culture (Hoftstadter, 1979). He posits that self-awareness and cognitive emergence is a natural outcome that arrives from the presence of extremely complex and paradoxical recursive systems. It is this concept of emergence that the ensemble finds most exciting; the idea that one can fashion a system which may produce results that you do not expect and whose presence can emerge as almost another performer in the ensemble. However, interesting algorithms on

their own a performance does not make. Aleatory could be conceived as a form of machine improvisation (though, improvisation with preconceived intention), and thus suffers and enjoys some of the same effects of improvisation. In the music of Glitch Lich, the ensemble finds that aleatory is most useful when embedded within the confines of a linear structure. As well, the band finds that stochastic systems are more effective when a performer is constantly and performatively manipulating them, sculpting the output in an improvisational manner to gain the most interesting results out of the algorithm and to match its character to what is predetermined by the agreed upon linear structure of a piece.

The fusion of compositional direction, spontaneous energy, and emergent structures serve as the core basis for Glitch Lich's aesthetic approach to composing and performing music. When setting out to make a new instrument, the ensemble is looking to create instruments that allow a composer to create a linear structure, that have tools for the performers to execute that structure, but which also have embedded inside of them the capabilities for allowing performers the freedom to break off from the pack and to make spontaneous changes to their performance as they see fit. Stochastic and emergent structures also play an important rule, in multiple ways. Firstly, these are usually ingrained in a fundamental manner into the system, through devices such as interconnected audio feedback chains. Secondly, the manner in which the performers themselves interact, by sharing musical information along a network and by incidentally influencing each other's musical output via the intrinsically linked structure of the instrument, create emergent properties. These properties are outside of the direct control of the individual performers, but at the same time are formed as a direct effect of their sum total. It's an attempt to marry the pulsating chaos of David Tudor's feedback experiments with the emergent properties of the networked behaviour that occur during The Hub performances.

3.2 Beauty, Play, and Viscerality

When the members of Glitch Lich set out to compose music for the ensemble, they do not set out to memorialise a person, proselytise a political viewpoint, or articulate a literary narrative. Instead, it is preferred that the members compose music that evokes more basic responses, with the sounds existing of their own accord, without the need of higher level concepts for them to be enjoyed. The ensemble has an affinity for distilled sounds in music, unmuddied by oppressive ancillary ideas. This line of thinking is articulated beautifully by John Cage:

> "They say,'you mean it's just sounds?' thinking that for something to just be a sound is to be useless, whereas I love sounds just as they are, and I have no need for them to be anything more than what they are. I don't want them to be psychological. I don't want a

> sound to pretend that it's a bucket or that it's president or that it's in love with another sound. I just want it to be a sound. And I'm not so stupid either. There was a German philosopher who is very well known, his name was Immanuel Kant, and he said that there are two things that don't have to mean anything, one is music and the other is laughter. Don't have to mean anything that is, in order to give us deep pleasure (Sebestik, 1992)."

However, while members of the band does not set out to make pieces of music "about" things, it is still a significant goal to create coherent and elegant works. It is important to hone in on the central concept of a piece and pare away all of the unessential portions. To bring out the most crystalline and pure foundations and to produce that as minimally and powerfully as possible. So how does one go about making these decisions, what is it that is being expressed with a composition? Fundamentally, Glitch Lich does not look to invoke particular reactions in audiences. However, amongst whatever the audience may feel in reaction to the music, it is hoped that somewhere in that reaction is thoughtfulness and deep pleasure.

With all of this said, there have been themes that the band gravitates towards. These themes could best be described as beauty, play, and viscerality. Beauty in that the members of the ensemble care deeply about sounds (just as Cage exhibits in his quote), and hold a special reverence for them. This reverence for sound may leak out into the band's aesthetics, as the ensemble has gravitated more and more towards sounds that evoke serene, slowly changing, and dark atmospheres. It isn't any one particular scale, or synthesis technique, instead it's a summation of a compositional practice.

Even though there is a reverence for purity of sound in the ensemble, the band attempts to never take itself or the music it produces too seriously. Glitch Lich is not setting out to make the next Fifth Symphony, and no one lives or dies based upon the clang of tones in the band's music. Everyone is here to enjoy themselves, and the band intends to do so thoroughly. There are streaks of humour and play in the band's music that manifest themselves in many ways, ranging from ridiculous names of compositions, to evocations of video game music, arbitrary abandonments of long developing sections of music for something wholly incongruous, to the actual making of jokes during live performance.

Violence seems to be another common theme that emerges time again in the group's music. It's also the one that's taken the most time to understand, and perhaps even come to grips with. It is not in fact violence, as the members of the ensemble have no personal inclination to want to do damage to anyone. Instead it is viscerality in the purest sense. Viscerality, in the way that one might experience an uncontrollably powerful force, such as large earthquake, or a powerful tornado (both of which members of the ensemble have lived through on multiple occasions, being from Oklahoma, and California), and the flood of instinctual

gut reactions that are immediately experienced. The human body responds to incredibly destructive events such as earthquakes and tornadoes by producing various hormones, such as epinephrine, norepinephrine, and cortisol, in a response dubbed "Fight-or-Flight" that amongst other physical reactions gives a person a sense of heightened excitement and awareness (Harari and Legge, 2001). It's this same reaction that the ensemble attempts to pull out of music. This heightens the mundane to the surreal, catapults an individual evening to an epic proportion, and in the process may take the venue as a whole away to some other realm.

The concepts of other realms and primal forces are the reason that early 20th century horror author H.P. Lovecraft often serves as inspiration for the band's music. An astute reader may have taken note of the fact that many of the software projects listed in Section 1.3 are either directly from or are largely inspired by his works. Lovecraft's stories include an in-depth pantheon and mythology surrounding ancient beings from far-flung quarters of the universe. The mere presence of these ancient beings can be enough to drive a person insane (Lovecraft, 2014). This concept of the unimaginable and the unknowable is a great inspiration for noisy electronic music. Instead of hiding the inherently alien act of making music without physical instruments these inspirations provide a springboard for exploring this alien nature. The band is charting unknown primal places, using sound as a medium (as primal a force as any), and asks the audience to join them to make an attempt at the unimaginable nature of the universe.

3.3 Collaboration and Camaraderie

Glitch Lich is a unique ensemble in that two of the four members (Curtis McKinney and Chad McKinney) are identical twins. One consequence of this is that collaboration, camaraderie, and even friendly competitiveness comes very naturally to the ensemble. This preference for collaborative work has resulted in particular aesthetic ramifications for band's work with network based multi-user instruments. There is a deep sense of camaraderie that is felt when the ensemble attempts to wrangle complex networked compositional systems into a coherent performance, navigating the hectic morass through the use of tight communication and empathetic musicality. The band has more and more attempted to showcase this camaraderie as part of the design of its network instruments, and to showcase it as part of the performance itself. For Glitch Lich, it is much more exhilarating to see a drama unfold of four individuals working together to produce a coordinated chaos, than if the same music were produced with nothing else to show but four glowing apples affixed to the back of laptops, and an atmosphere devoid of urgency or communication. The band has taken this further in recent compositions and attempted to bring the audience members themselves into the performance (which will be covered further in Chapter 4).

It should be made clear that the work that the produces does not fit into the

standard orchestral model in either musical sensibility or in performance practice. There is a clear effort made by the band to steer clear of what many other network ensembles have sought out to do, which is to replicate the hierarchy and organisation of a traditional orchestra. These ensembles, usually incorporating "Ork" somewhere into their names, often (but not always) are academically supported, have specifically delineated roles such as composer, conductor, and performers, and operate with a clear hierarchy of power. Glitch Lich instead seeks to emulate the dynamics as espoused by The Hub to create a group dynamic comprised of individuals collaborating on equal ground. Furthermore the ensemble is a long-running band that has maintained the same membership over five years, as opposed to academic ensembles that rotate members as students graduate or progress. This same attitude also informs the performance arena that the band prefers to play in. Preferably Glitch performs in informal settings, if possible in a relaxed atmosphere, such as at music venues, bars, warehouses, or outside settings.

A unique attribute of Glitch Lich is that its four members are each geographically located in a different part of the world. Cole Ingraham lives in Boulder, Colorado, Chad McKinney lives in Brighton, England, Curtis McKinney lives in London England, and Ben O'Brien lives in Gainsville, Florida. However, the ensemble is actually not very interested in exploring the nature of telepresence and divergent performance spaces in these pieces. The ensemble tends to agree with their mutual former teacher Chris Brown rather wholeheartedly when he stated, while discussing The Hub's approach to network music, that: "We were interested in the sound of idiosyncratic, personal computer music instruments that could influence, and be influenced by each other. The Hub became a way to extend compositional ideas from the solo electronic performer to an ensemble, creating a new form of chamber music. (The fact that the chamber could be expanded in distance was not entirely irrelevant, but never really the point)" (Brown and Bishcoff, 2002). Distance has never been something that the ensemble has emphasised in its instruments or in its music. However, it is of course logistical constraint and concern that must be overcome technically. However, in a perfect world all the performers would always perform in the same room (and in fact, The Hub will only perform in the same room). The main reason that Glitch Lich does not always perform in the same room is simply logistics. Being that the ensemble is so geographically dispersed networking is required for the ensemble to continue to exist. In this sense, the band simply could not exist without the existence of current networking technologies.

For Glitch Lich, it is imperative that a NMI allow for meaningful and expressive relationships between performers. The promise of NMIs, is the ability to create ensembles that have even greater abilities to share and express musical ideas amongst a group of performers, as well as to explore a whole new realm

of making music, where the output is not the output of any one individual, but is instead a sum total of all the constituent parts and their interactions. It's for this reason that feedback loops drawn out across modules that are manipulated by multiple individuals has been a common approach for the band. In these scenarios, the sounds take on lives of their own, creating a microcosm of cascading synthesis. In this kind of music, while each member has an input into the direction those sounds go, it's not under the bidding of anyone performer, or even all the performers combined, as the sounds take on lives of their own.

3.4 Performance and Liveness

The final component to the aesthetics of Glitch Lich is an emphasis on performance and liveness as part of the music created. As stated previously, the members of the band came into music through playing traditional instruments and composing music for traditional ensembles (both in popular and classical idioms). Performing has always been something that's part of the life of each of the band's members. The band has never quite understood those composers who are happy to hand a piece off to an ensemble and let them have all of the fun. Instead, the group prefers to have a hands on experience in realising music. When the ensemble began composing electronic works, initially it started like many others by using Digital Audio Workstations (DAWs) to create fixed "tape" pieces. While these tools afforded the capability to shape interesting soundscapes to my heart's content, they also stripped away any possibility for appealing live performance. Now performances of music consisted of nothing more than putting a compact disc into a CD player and pressing play, something the ensemble found to be incredibly unfulfilling. For reasons that are not too strange to understand many audience members that the members of the band spoke with relayed how awkward, uncomfortable and unexciting it was to sit in a dark room and listen to a fixed media piece laboriously make its way through a fixed journey. The band wanted a way to regain the excitement of performing with traditional instruments, but still have the interesting sonic capabilities of electronics. Fixed media electronic music techniques offer ways of moulding and sculpting timbre and rhythms, through sound synthesis and sampling techniques, that simply are not possible acoustically. However, the fixed media itself transformed performance into presentation, performers became CD players. This was not desirable.

However, this all changed once the usage of SuperCollider became the standard mode of composition in Glitch Lich. By using SuperCollider the band was now able to create dynamic and responsive software instruments that could actually be performed in a live setting. This brought back all of the energy and anxiety of live performance; something that was missed dearly. This kind of performance must be said to be different in some ways with traditional methods of performance. In general the members of the ensemble do not play each individual

note that is created during the performance, in contrast with a piano performance where the pianist must depress a key for every note that is produced. Instead the music could be said to be *parametric* in nature. In other words performances consist of the members of the ensemble performatively altering various characteristics, or parameters, of the music, as it is being played. And in fact it is generally these parametric changes that are actually propagated over the network. However, there were still issues with live performance. As the band set about making NMIs and performing them for audiences, it has often found that people truthfully want to understand and engage with the performance, but there is a literal wall (in the form of the back of a laptop screen) to an understanding of the underlying concepts of the piece. For Glitch Lich, making electronic music, and network music in particular, as approachable and cognisable as possible can actually accentuate appreciation for that music. Simply put, electronic music in general, and network music in particular, does not have the same history with individuals that more traditional instruments do.

Most people are very familiar with how a guitars sounds, what its musical roles are, and what it looks when being played. Seeing a live electronic music performance throws all of that out of the window. These issues were explored specifically in a study that questioned several network musicians on their thoughts about liveness in network music by McKinney and Collins (2012). In this study respondents shared the sentiment that many network music performances are lacking in a sense of liveness. As one respondent stated "It's a bit ironic; the performance practice we have embraced in order to make electronic music that is very, very live, can look very, very dead from the audience's perspective." According to Sanden (2013), the concept of liveness is derived from the concept of music as a performed medium. Sanden states that it therefore follows the perception of liveness in a musical experience is the perceived level of performance involved, which may differ from the actual level of performance involved (as is demonstrated by the perceived "deadness" of network music to the previous respondent).

Glitch Lich attempts to build in a method for understanding a network music instrument into the very design of that instrument. This built in method of instrumental understanding may then be utilised to engender a sense that the performers are indeed performing this instrument for the audience. It is a goal to build in as many mechanisms for increasing the sense of liveness, presence, and explanatory measures as possible for any new NMIs that are built for the band. A common way of approaching this is through the usage of graphics to visualise some sort of activity in the network and the music, so that the audience may have a better understanding of what is happening, and why a piece is unfolding in the manner it is. Furthermore, by building into these systems some form of performer embodiment (such as through projected ensemble chat or avatars,) that the audience is more ready to buy into the "reality" that the band is fully present and performing

together, even though members may be physically absent. Without these mechanisms, the audience may instead believe that the one person on stage is simply sitting and playing an iTunes play list, or checking e-mail.

3.5 Limitations of Approach

The approach of composition and performance as laid out in this chapter is geared towards a specific performance practice and as such comes with some limitations. Since performing in non-academic environments is paramount to the group this restricts the kinds of pieces that may be created. Performances that would require large-bandwidth requirements, such as real-time audio and video streaming are generally not possible due to the networking realities of live performances in unknown spaces. This is one of the reasons the group has adopted the parametrised music approach. This approach is both low-latency and low-bandwidth. Furthermore should a network failure occur, it is less obvious; There isn't an actual picture of a person at the other end of a video stream that suddenly cuts out. Instead perhaps their visualised avatar becomes unresponsive. This also means that it has better recoverability, i.e. their avatar starts responding again and they regain control of whatever musical subsystem they are in charge of.

The parametrised musical approach also calls for a manner of performance where the performer is one stepped removed from performing the music. It is a kind of meta-performance. The performer does not simply play every note as it occurs in the music. Instead the composer designs semi-autonomous musical subsystems that may be controlled parametrically. For instance a musical melody may be activated and deactivated by the performer. Furthermore the performer may be able to sculpt the melody by changing its register and timbre, or even rearrange it rhythmically during performance. This kind of performance can seem distant when compared to the immediate control afforded by a traditional musical instrument.

A serious limitation of this approach is simply the amount of development time required to develop the underlying technology. It has been common for the ensemble to put six months to a year just into the technical systems required to perform a new piece. Only after the technology has been fully developed may a new piece be created using that technology. Furthermore this approach, due to its complexity, is certainly error prone. In response to this there have been efforts towards creating more long-term dependable technologies. This includes OSCthulhu, a general network system geared towards musical performance, Azathoth, and later Necronomicon, both of which are network music engines which are akin to a video game engine, providing boilerplate code and functionality required by most network music works.

3.6 Conclusion

This chapter has covered objective four of this research: the creation of a methodological framework for the creation of NMIs. This objective was established to achieve aim #2 of this research, which is the creation of new, and the refinement of old, tools and techniques for composing performing and designing NMIs.

4 Design, Development, and Composition

In this chapter, aim #2 of this research, ”the creation of new, and the refinement of old, tools and techniques for composing performing and designing NMIs”, is addressed in detail. In particular, the following objectives are covered:

6. Using the methodology established in Chapter 3, initialise a design space for creating new NMIs.
7. Use this design space to establish technical requirements for designing new NMIs.
8. Identify short-comings in previous technologies for accomplishing the technical requirements of the initialised design space.
9. Create new tools, NMIs, compositions, and performances with the established methodology and initialised design space, taking into account the shortcoming of established technologies, and overcoming them by creating new technologies where necessary.

Several new pieces of software tools and software based multi-user instruments have been created throughout the execution of this research. The design of these pieces of software are directly informed by the aesthetic methodology established in the previous chapter. The design space has been initialised by these aesthetic choices and create certain requirements for the software to fulfil.

4.1 Initialising the Design Space

The software should allow a group of users to tightly and intricately create musical performances despite the fact that the members of the ensemble may be at vastly dislocated geographic locations. Normally the band operates with members in Colorado, Florida, and England, however members of the band have performed as far apart as from Hawaii to London. Therefore any software created for the ensemble must be able to effectively handle lag times and packet loss in a sensible manner. Furthermore, there has to be a means of meaningful musical interactions for the users, despite these distances. If possible, the system might also structure itself in such a way that lag times may be obfuscated or their impact minimised. The emphasis on informal performance environments exacerbates this issue, as the systems produced have to be able to create these networked connections in informal environments with often times sub-optimal network conditions.

Due to the long standing membership in the band, a certain amount of virtuosity is attainable with any software produced, thus ease-of-use, while important, does not need to restrict the users to simplistic interactions, and instead focus can be placed on attempting to produce the most meaningful and elegant of musical interactions. The instruments produced should have capabilities that allow for musical compositions to be composed for them, such that these pieces may

Aesthetics Demands	*Design Requirements*
Dislocative collaborative interactions	Networking infrastructure
	Minimised and obfuscated lag time
Shared musical resources	Network Synchronisation
Informal performance settings	Low-bandwidth networking
	Disconnection recoverability
Composability and performability	Intricate and commandable controls
Virtuosic performability	Complex interface interaction
Long-standing ensemble	Less need for low learning curve
Improvisational capability	Variety and variability of controls
	Interperformer interaction
Algorithmic musical material	Support for algorithms
Sense of liveness in performance	Interperformer interaction
	Emergent sonic behaviours
	Visual projection
Egalitarian/Socialised distribution of power	Homogeneous/shared user capabilities

Table 2: NMI Design Requirements as dictated by the aesthetics of Glitch Lich.

be performed in a repeatable manner, while allowing for enough flexibility that improvisation within the ensemble is possible.

As well, the instruments require a means for algorithmically generating musical material, and allowing for the musicians to interact with these algorithms, and for the algorithms to respond to the musicians. In this way algorithmic systems can emerge as proxy members of the band and deepen the sense of connection between machine and musician, and musician to musician. Finally, live performance is paramount, and all software should seek to give a strong impression of liveness to the audience, the sense that things are "happening" and someone is not simply pressing play on an audio file. Special consideration should be given on how best to evoke the dislocated members of the ensemble in the minds of the audience during dislocated performances.

4.2 Strategies

Given these requirements several strategies have been formed to address these issues and have been used throughout the creation of the software tools and multi-user instruments created during the course of this research. The first and most obvious is the usage and transfer of control information in lieu of networking audio signals. This is modelled after the set-up of The Hub, and puts an emphasis on interaction between musicians in the band and in-between the musicians and their

machines. Emphasis is placed on interactions not possible in traditional ensembles, such directly as by directly affecting each other's audio output. This makes the networking involved more akin to a Real-Time Strategy video game, such as WarCraft or StarCraft, than to a conferencing or video chat program (McShaffry, 2012).

Audio is not networked amongst the ensemble (all the networking relies upon sending parametric data), however there have been discussions about the possible usage of shared video streams between network nodes. While there are certainly benefits to sharing video, such as creating a sense of physics connection between the remote nodes, there are also some issues. Logistically sharing video is bandwidth intensive and depends upon a strongly consistent connection. Glitch Lich prefers to play outside the confines of academic halls in places such as venues, bars, warehouses, and often the facilities for this networking that intensive is not present. This may change in the future, but historically for the ensemble it has not been feasible. Furthermore, the ensemble has chosen to focus the aesthetics of the band's networking on the interactions between the performers in a digital environment. Thus in lieu of sharing video across nodes and emphasis is placed on visualising the members digitally, as well as visualising the musical elements in the music, and the members interactions. These networked visuals give the audience a much clearer picture of what is happening, and gives a sense of presence to the dislocated members that would otherwise not be possible. As well, this method places emphasis on the digital quality of the performance and allows each piece to take on its own visual character, something that is aesthetically desirable for the band. For the first few instruments created, simply showing the GUI of the NMI was used as a means of establishing the presence and musical interactions of the performers in the music. Later instruments sought to create rich custom made visuals using OpenGL that aesthetically presented these concepts.

The NMIs created do not rely upon one to one interactions with performers, so that in general a single action by the performer does not equal a single sound made by the instrument. Instead, focus is placed on creating semi-self-sufficient systems that may be modified by performer interaction. As well, often times these systems have interlocking elements that allow each performer's individual sounds to interlock and influence the other sounds the performers are modulating. Likewise, when it is desired to have synchronous temporal and rhythmic elements in the music, these are also not established through one-to-one performer controls. Instead repeating or semi-repeating systems are put into motion that are modulated by the musicians. A major benefit of this manner of composition is that it drastically minimises the effects of network latency and general network issues on the performances, as any small delays in control times (which general range in milliseconds, to micro seconds) are not so noticeable when it is used to transmit control information of semi-autonomous instruments. Comparatively these delay

times may have drastic effects on the ability of musicians to synchronously play together using networked audio (Kleimola, 2006).

An often used method of creating these semi-autonomous systems is through the usage of audio and control data feedback, which plays strongly into the aesthetics espoused by the band. Feedback loops philosophically fit into the groups themes, but also sonically tends to produce sounds that fit into the sonic character the band is attempting to cultivate, ranging from screeching howls, to pounding impulses, to serene atmospheres. Feedback based systems also give the instruments themselves an organic feel, and certainly enhance the feeling of liveness and that "anything can happen" during performances. Even after performing some of these pieces for multiple years on numerous occasions, they still manage to present surprising results due to these networked feedback systems.

4.3 OSCthulhu

Work on a new networking platform for collaborative electronic music began in June 2010. This platform uses Open Sound Control (OSC) messages, a protocol for sound specific networking based upon User Datagram Packets (UDP), as the basis for networking. The project, named OSCthulhu, was inspired by the program OSCgroups created by Ross Bencina, which enables users to share OSC messages with each other over a network (Bencina, 2013). OSCgroups accomplishes this by creating a central rendezvous server that uses Network Address Translation (NAT) hole-punching techniques to enable individual users to bypass firewall and router restrictions normally placed on peer-to-peer communications.

4.3.1 NAT Hole-Punching and UDP multicasting

Normally, a router will block any message that is received unless a previous message has been sent out by the user to that specific IP address and port. This is done to prevent nefarious traffic from reaching the user's private network and computer. Furthermore, the IP address and port of an application behind a user's router is obfuscated by NAT, a system utilised by routers to preserve IP address real-estate on the open Internet (largely addressed by the upgrade from IPv4 to IPv6) (Hagen, 2006). The server works around this by noting the private and public IP Address and port pairs, known as an endpoint, of each user that logs into a group. The server distributes this information to everyone within the group, at which point all of the users then asynchronously send messages to all of the public and private endpoints that it has received from the server. The first messages received at either end will be discarded by their respective routers as they have not been met with a matching outgoing message. However, now the user has punched a hole in their firewall by sending an outgoing message to each of the recorded endpoints. Now when the user receives messages at their endpoint, be it from any

of the public or private IP pairs that they have received, it will successfully be accepted as valid traffic by the router, allowing for full bi-directional peer-to-peer communication between the users (Ford et al., 2005). A graph exhibiting this process may be seen in Figure 8.

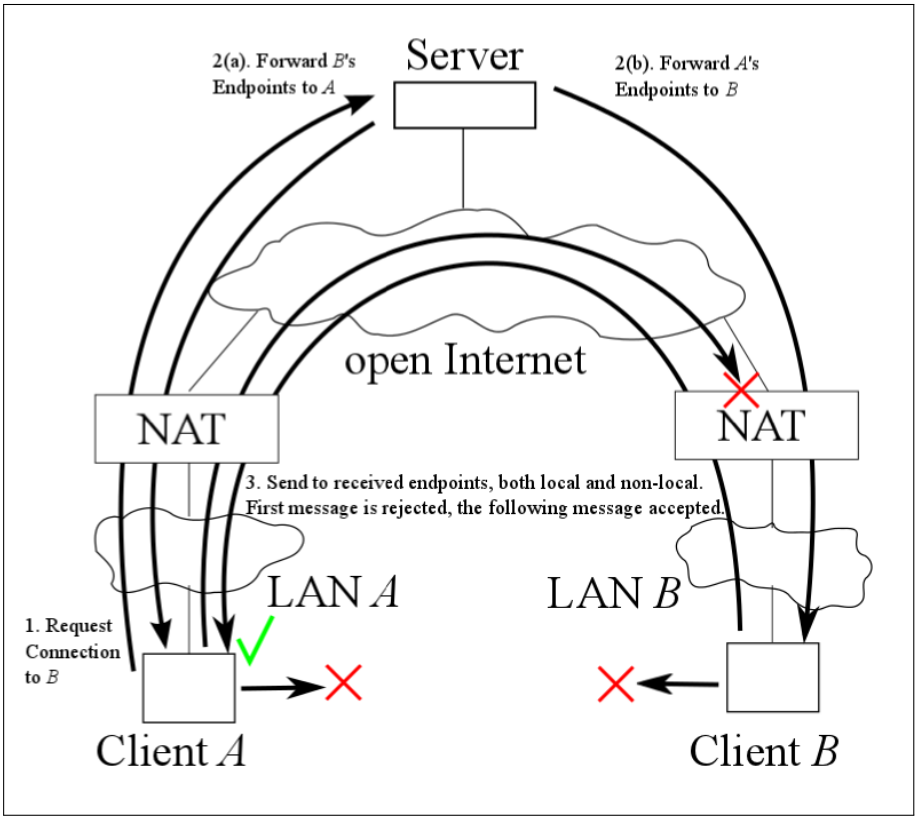

Figure 8: NAT hole punching process (Ford et al., 2005).

Once external communication has been established the client application on each user's computer opens up an internal UDP port that parses any incoming OSC messages it receives and multicasts that message to everyone in that user's group. Due to the flexible nature of OSC the origin of these messages could be from any application that has OSC capabilities, including programs such as Max/MSP, SuperCollider, or Reaktor (Manzo, 2011; Wilson et al., 2011; Sasso, 2002).

The benefits of this approach are that users can easily and dynamically form groups to share messages between while being in completely different places in the world, without having to note the individual public and private IP endpoints of each person within the group. The strain on the server is also minimal, as it only serves as a rendezvous point for users, and none of the actual OSC messages are passed to the server. The system's multicasting architecture is quite appropriate for network music systems that require musically significant gestures to be shared over a network with the utmost speed and low overhead granted by UDP messaging systems, but do not require the reliability of a slower Transmission Control Protocol (TCP) based system (Stevens et al., 2004).

4.3.2 Reality of the Internet: Packet Loss

After extensive usage of this system within the context of the author's network based computer music band Glitch Lich, several shortfalls became apparent. While OSCgroups is straightforward and robust, the usage of UDP meant that systems that relied upon extensive synchronisation between peers would often suffer from potentially fatal errors during performance due to packet loss. Packet loss, when a packet is sent at one endpoint but not received at the other, is an unfortunate reality of networking that every networked program must address in some way.

Many programs overcome this by utilising TCP for reliability, which has built-in systems to handle packet loss, re-transmitting lost packets after a certain time-out period (Stevens, 1994). However, for systems that require the utmost speed, such as gaming and musical applications, TCP is deemed inappropriate due to its sluggishness. Furthermore, the nature of TCP's re-transmission mechanism means that critical real-time gestures in a game or piece of music may be transmitted out of order, negatively impacting the quality of play. Thus, TCP is too slow and unwieldy, and UDP is too unreliable for pieces which rely upon stringent accuracy.

The similarities between multiplayer gaming and network based music are rather striking. Both require raw speed to ensure that multiple peers can react to each other's actions as realistically as possible. In both raw speed is considered more important than absolutely receiving every packet sent. In both out of order information and information based upon an old state of the system should be avoided if possible. However, both can be fatally affected by the loss of certain important packets of information. It seems only natural then for the network musician to look to video game networking techniques to learn how they deal with such an important issue.

Tim Sweeny, a game programmer and creator of the Unreal Engine, wrote an in depth analysis of the history, difficulties, and techniques involved in programming multiplayer games titled "Unreal Networking Architecture". Written in 1999 at the veritable dawn of modern multiplayer First-Person Shooter(FPS) games the document is rather striking in its presentation of a problem that sounds remarkably similar to the plight of the network musician. Sweeny eloquently states that "Multiplayer gaming is about shared reality: that all of the players feel they are in the same world, seeing from differing viewpoints the same events transpiring within that world (Sweeney, 1999)."One may easily replace multiplayer gaming with network music in that sentence to describe the promise of network based collaborative electronic music.

Figure 9 shows an example of this: a player looks out at the world in a multiplayer sessions of (Sweeney, 1999).

Sweeny describes a system that overcomes the short falls of packet loss in UDP by utilising what he describes as a "Generalised Client-Server Model "(GCSM).

Figure 9: Screen capture of the online multiplayer game *Unreal*.

In the GCSM whenever a client makes an action the client simultaneously updates its own internal game state(the exact state in which all objects in the world are in at any given time), as well as sends its action as a message to the server. The server then updates its own internal game state to reflect these actions. After a period of time of receiving action messages from clients, referred to as the Delta Time the server will update its game state, using predictive analysis to correctly account for lag time in message transmission. The server then issues an update to all of the clients, called a Tick. This game state may differ from that of any of the clients' due to several issues, including packet loss, and the inherent asynchronicity of client actions due to lag. However, as Sweeny put it "The Server is *The Man*"and the server's game state takes precedence over that of the client. Thus, when a client receives an update after Delta Time, its internal game state is replaced with that of the servers. This solves both of the previous problems stated in the previous section: The clients actions are perceived to be immediate(as it updates its own internal game state immediately), it receives peers actions in a swift manner due to the usage of UDP, but if a packet is lost in transmission a system is in place to handle it in a sensible and reliable way.

To the user the only perceived anomalies are when there is a discrepancy in the server update, usually due to packet loss or lag: He may have perceived himself as shooting another player in the head, but the server states that the player is still alive. While this can be vexing, it is preferable to the alternative: the client kills the other player on their computer, but in the other player's game-state they are still alive, effectively creating simultaneous alternate realities and immediately

ruining any notion of a shared experience. On the other hand, there is always the lovely scenario where the client believes they missed, when in fact, the server states they hit their target.

OSCthulhu was created as a musical analogue to the GCSM approach. After testing several implementations of the GCSM as described by Sweeny, some tweaking was required to produce a model that was appropriate for usage in the context of a network music environment. The core of OSCthulhu is the way it represents data, which is very similar in approach as the GCSM. Data is represented in the system as a series of networked entities called *SyncObjects*. These SyncObjects contain an arbitrary amount of modifiable values, called *SyncArguments*. SyncArguments may be Strings, Integers, Floats, or Doubles. While in the original GCSM SyncArguments were accompanied by a fixed name, in OSCthulhu they are referred to by index. This change was made to preserve bandwidth.

Another change that was made was the behaviour of client actions and ticks. In OSCthulhu when a client action is received it is immediately multicast to all of the clients instead of the server waiting for Delta Time and issuing a Tick to update the clients. This was done to make the system as fast as possible, though at the expense of more bandwidth. This is considered acceptable for musical purposes, as the average network music server will deal with significantly less traffic than a gaming server, and thus can afford to be faster at the expense of being less efficient. This also means that there are two ways that a client may be updated in OSCthulhu, either by a *setSyncArg* message, which updates a single SyncArgument, or by a *serverSync* message which wholly replaces the clients state with the servers. A graph depicting the organisation of an OSCthulhu network is show in Figure 10.

OSCthulhu 1.0 was constructed as a Java based library, usually accompanied by the Processing programming library used for visuals (Harold, 2004; Reas et al., 2007). This would eventually change in OSCthulhu 2.0, which will be covered in section 3.

One key point to keep in mind when using OSCthulhu is that it is fundamentally a different way of organising the manner in which a networked composition or software system is constructed. Often times we as composers think of networking as a series of commands: change sections, get louder, stop playing, switch timbres. To network with OSCthulhu, a composer must think of his or her composition instead in terms of a series of objects. These objects may be manipulated in similar fashion to the objects of an object oriented programming language. Objects may be created, destroyed, or have their values altered. So instead of our previous example where we gave commands to modify music, instead a composer would have an object that represents a synthesis unit generator, including variables that represent that unit generator's amplitude, and timbre. To

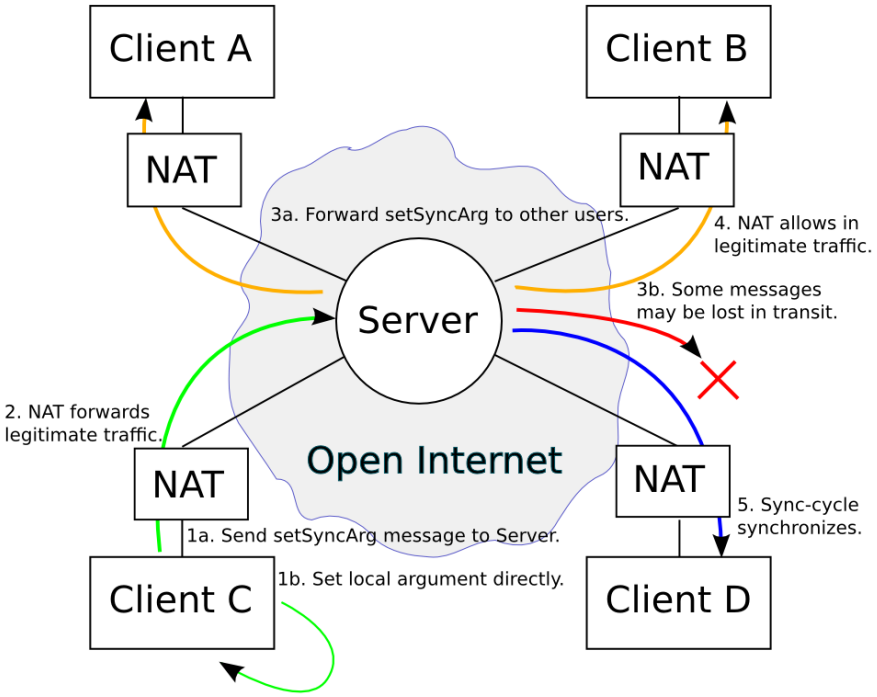

Figure 10: Depiction of sending a *setSyncArg* message using OSCthulhu.

create another instance of that unit generator, perhaps with a different set of arguments, one would simply add another instance of that object to OSCthulhu. This approach may require a bit more forethought, but the structure lends itself well to networking musical contexts, especially in remotely rendered synthesis configurations.

4.3.3 OSCthulhu 2.0

Initial research and testing of OSCthulhu 1.0 showed that the basic structure of a Generalised Client-Server Model with synchronisable Sound States was promising. However, there were several ways in which it was believed that the system could be improved. While the java based library made implementing OSCthulhu 1.0 into Java based projects straightforward and simple, it made it nearly impossible to use with projects that did not include some kind of Java based component. Also, OSCthulhu's automatic synchronisation API, originally intended to streamline implementation, actually made the process more restrictive and counter-intuitive by forcing the programmer to adopt a specific programming style. Furthermore, the manner in which the centralised server was implemented required that a user remotely start the server for the group to use. This proved to be disastrous on one occasion in performance when the terminal session for the server timed out, killing the server process and effectively destroying performance by preventing the group from being able to communicate with each other.

To improve upon these issues, and to add additional features, it was decided that a second version of OSCthulhu should be created with the lessons learned

from working with version 1.0. Development of OSCthulhu 2.0 began in spring of 2011. Instead of creating a software library to be implemented into projects at the code level, OSCthulhu was developed to be more like OscGroups and to operate as a separate application that is communicated with via an OSC API. C++ was chosen as the programming language for version 2.0, due to its efficiency and deployability. As well, the Qt framework was utilised for its GUI functionality, and slots/signals system (Summerfield, 2010).

Many improvements and additional features were added into OSCthulhu 2.0. Operating as a separate application means that it is much easier to implement OSCthulhu into projects that utilise different combinations of programming languages and software packages. Thus far OSCthulhu 2.0 has been used to develop projects that use SuperCollider, Java, Processing, C++, and Quartz Composer. This set-up even allows for asymmetrical scenarios where different members of the group use different programming languages or software to create a piece of music. Also, the server architecture has been implemented so that it may be run as a self-sustaining *Daemon* process(an application that runs completely without the aid of human intervention, handling situations where the application stops due to error, or when the machine the process is running on shuts down or restarts) on Unix machines. This greatly improved reliability of the system. This also greatly increased usability, as to use the system a user no longer needed to log in to the physical machine via Secure Shell(SSH, used to remotely operate a computer) and manually start or stop the server. Instead the server is always running on the internet.

The API has been greatly streamlined and made more flexible. The goal of OSCthulhu 2.0 was to simplify usage and to reduce the number of commands required to operate the system as much as possible. There are three standard and six auxiliary OSC commands that are used to control OSCthulhu 2.0. The */addSyncObject*, */removeSyncObject*, and */setSyncArg* commands create, destroy, and alter the parameters of SyncObjects on the server. These SyncObjects are completely symbolic, and the manner in which they are used and what they represent is completely up to the programmer, providing a much more flexible and simple system. OSCthulhu dynamically configures each SyncObject as it is added to the server, assigning the type, sub type, and argument types based upon the default values provided by the user.

In addition to the previous three standard command addresses, there are six commands that may be implemented by the user for more functionality, but are not required. The address spaces */addPeer* and */removePeer* are for the user to implement a logging scheme for their project. These are useful if ownership is an important part of the system, such as in scenarios that include shared GUI elements. The */chat* command is used to send text based chat to the whole group.

Many network based compositions start with building some kind of chat el-

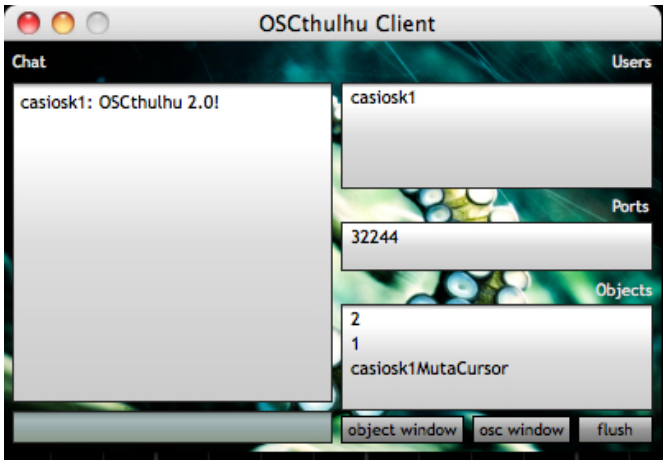

Figure 11: Screen capture of the SyncObject and Chat window in *OSCthulhu 2.0*

ement, so adding an automatic text propagation system has shown to speed up initial development of new projects. This chat is also echoed in the GUI for OSCthulhu 2.0, so users are really only required to create a visual representation of chat for pieces that require full screen real-estate. Figure 11 shows a screen capture of the chat window in OSCthulhu.

The */ports* command is used to change the ports that OSCthulhu echoes incoming data to. Version 2.0 is an improvement over both OSCthulhu 1.0 and OscGroups in this regard, as it allows a user to remotely change the ports that are being communicated on, as well as to allow the system to echo data streams to multiple outputs. This feature facilitates pieces that require multiple different software systems to receive the same data on the same computer, as is common in projects that have both a visual and sonic element. The */login* command is for situations where the users wants to easily have the system send all of the previous data that has been added to the server. This facilitates situations where multiple individuals are asynchronously performing, or in situations where a crash necessitates a restart of the client software or hardware. The final command */flush* removes all the current SyncObjects from the server and clients. This is useful for cleaning up after a piece has concluded, or for removing errant SyncObjects, that for one reason or another cannot be removed from the performance GUI. The final command,

A newly implemented feature in version 2.0 that has proven to be even more useful than first thought is the establishment of two added descriptors to SyncObjects, referred to as an object's Type and Subtype. These feature allows a user to much more easily create systems in which groups of objects react in particular ways to incoming messages.

4.3.4 OSCthulhu and OSCgroups: A comparison

OSCthulhu has not been designed to supersede OSCgroups. Instead, it is meant to be an alternative approach, useful for a set of situations that OSCgroups may be deemed to be less suitable for. The advantages of OSCgroups over OSCthulhu are a simpler interface with less overhead that doesn't restrict the structure in which it is used. Also, due to OSCthulhu using a GCSM a server is required, with all pertinent traffic being directed through that server. Although a central server is required for OSCgroups as well, this server can handle multiple groups simultaneously, and none of the multicasted traffic is forwarded through the server.

However, OSCthulhu is appropriate for projects that require both a close degree of synchronisation as well as the speed and simplicity of UDP multicasting. This allows for the construction of new kinds of network based electronic music systems or pieces. These systems can rely upon shared resources to coordinate a network music performance, with the knowledge that this information will be transmitted in the most musically sensitive way, while always being safely accounted for. This capability makes it much more technically reliable and easier for network instruments and ensembles to reliably interact across consumer grade network connections in remote parts of the world, making a touring network ensemble a much more feasible possibility. One benefit of this is the capability to cast a much wider net for performance opportunities. The author's ensemble Glitch Lich has managed to create truly world tours in which consecutive dates of completely disparate locations was possible. OSCthulhu has been used as the networking infrastructure for most of the NMIs created in this research.

4.4 Medusa

A set of new pieces were created in the Summer of 2010 to test the newly developed OSCthulhu system. These pieces were created with a custom designed GUI system entitled *Medusa*, which uses OSCthulhu for networking and the Processing graphics library to create the musical interface. The design of the software system is influenced by the design of the Reactable multi-user instrument created by Sergei Jordà, however instead of a physical interface it is purely software based, and supports performance with others over a network (Jordà, 2009). Medusa was designed to specifically take advantage of OSCthulhu's Sound State synchronisation capability and relies upon the accurate transfer of persistent network objects.

The design process for Medusa was driven, initially, by the aesthetics of the piece Neuromedusae I and then was later reused for a follow up piece entitled Neuromedusae II. The aesthetics of this piece were largely inspired the "no-input" feedback circuits designed by David Tudor. To create a similar system for usage along a network required the exact kind of musical sensibility and reliability that

OSCthulhu required. Furthermore, the system mandated that there be a system for creating and removing feedback generators, each with their own set of individual controls that may be configured and altered by any member of the group spontaneously.

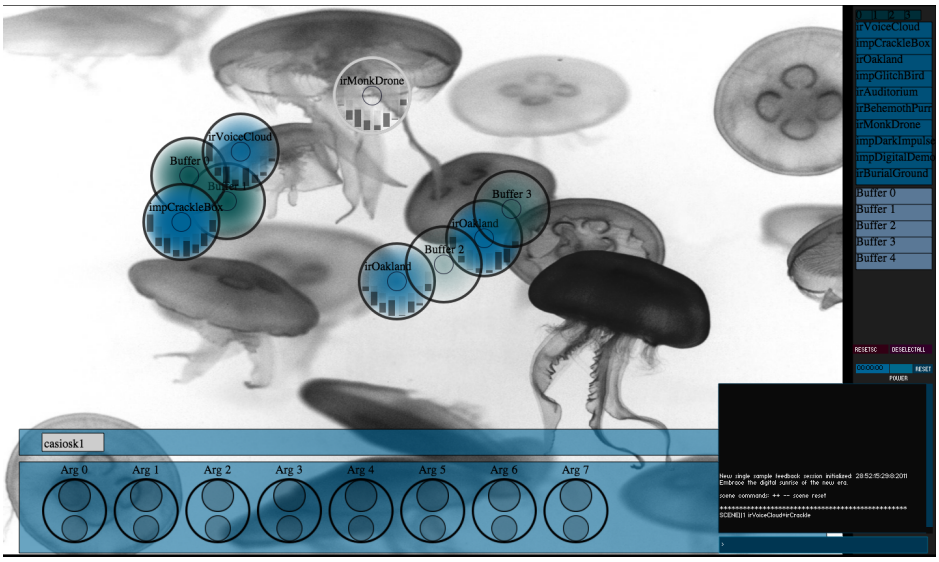

Figure 12: Screen capture of a performance using the Medusa System

The Medusa System organises these feedback generators into individual modulatable SyncObjects that exist along the network and on the OSCthulhu server. Medusa contains two different types of SyncObjects, both represented as ellipses on a two-dimensional plane, as can be seen in Figure 12. The first type is a Sound Ellipse, which symbolises a specific sound emitting synthesiser in SuperCollider. All of these Sound Ellipses are designed to interlink in some way with each other, usually to form a kind of audio processing feedback loop. To accomplish this interlocking, Buffer Ellipses are used. Buffer Ellipses symbolise specific audio buses which the Sound Ellipses output into.

Placing a Sound Ellipse next to a Buffer ellipse causes the Synthesiser in SuperCollider to output and input sound from that audio bus. By placing multiple Sound Ellipses next to the same Buffer Ellipse, it is possible to have the individual Sound Ellipses feedback into each other, creating a sound processing chain. At the bottom of the GUI is a series of controls that manipulate the synthesis parameters of the currently selected Sound Ellipses. The specific parameter that is manipulated changes depending on the design of the Synthesiser represented by the Sound Ellipse. Any time a member of the Network Ensemble creates, moves or destroys a SyncObject, or manipulates their parameters, this change is synchronised and shared with the whole group via OSCthulhu. Only OSC is passed along the network, and due to the use of Remotely Rendered Synthesis, no audio transmissions are required between the network nodes.

At this point, Glitch Lich began the performance practice of displaying the GUIs of the musical systems being employed during concerts. This was an early

attempt to address some of the issues found with network performance, mainly that of disembodiment and confusion. While not particularly aesthetically pleasing, the presence of any visual cue to inform audiences of the inner workings of the music and the band was deemed a positive development.

4.4.1 Neuromedusae I

Neuromedusae I was the first piece created using OSCthulhu and the Medusa System. The piece focuses on the synthesis technique known as *Single-Sample Feedback*. Single-Sample Feedback is a special form of digital feedback that attempts to overcome one of the shortcomings of creating feedback loops in a digital environment. In a normal digital feedback loop, there is a delay that occurs in the audio processing chain at the output and subsequent input points. This delay is equal to the block size set on the audio hardware being used at the time, often 1024 samples. In an analogue feedback loop no appreciable delay exists at this juncture. This delay impacts the nature of the feedback, and prevents certain feedback effects that would require less delay time.

One way to overcome this delay is to replace the manner in which audio feedback loops transmit audio over the feedback network. Instead of using audio buses, one can read and write to an audio buffer of a single sample in size at audio rate. This reduces the delay time from block size to a single sample. In a typical hardware set-up, this may be a reduction of as much as 1000%. In a Network Ensemble, to achieve a similar feedback effect, often times audio transmissions are used, which increases the delay time even further. In tests it was shown that the average round trip time for an audio signal from London to San Francisco was approximately 300 ms. At a standard Sample Rate of 44.1Khz, this delay time would be approximately 13,000 times slower than using Single-Sample Feedback.

To perform Neuromedusae I members of the Network Ensemble create combinations of these Single-Sample Feedback synthesisers that interlock and influence each other. These chains may be controlled by multiple members at once, creating dynamically formed network based multi-user instruments. The advantage of this system, is that members may have closely interacting sounds, that influence each other much faster than sending audio over a network would allow due to network and hardware latency. The sounds produced by this system tend to be incredibly turbulent and tumultuous, often times reacting in incredibly unexpected ways. Figure 13 shows an example of this Single-Sample Feedback in the Synthdef code used in Neuromedusae I.

In this chunk of synthesis code two buffers are used to write to and read from for feedback, called buffer1 and buffer2. These buffers are read from and written to by this specific synth, but may also be written to by other synths simultaneously, creating feedback chains. These buffers are only a single sample long, and

```
buffer = bufNum;
buffer2 = bufNum2;
rd = Dbufrd( buffer, 0 );
rd2 = Dbufrd( buffer2, 0 );

dRead = Duty.ar( 1 / SampleRate.ir, 0, rd );
dRead2 = Duty.ar( 1 / SampleRate.ir, 0, rd2 );
dRead = Slew.ar( dRead, 1, 1 );
dRead2 = Slew.ar( dRead2, 1, 1 );

lf  =  LFNoise0.ar(
        dRead.abs.linlin( 0, 1, 0.5, 2 ),
        2,
        0.5);
lf2  =  LFNoise0.ar(
        dRead2.abs.linlin( 0, 1, 0.5, 2 ),
        2,
        0.5);

mod = Impulse.ar( arg1 );
mod2 = Impulse.ar( arg2 );
mod = DynKlank.ar(
    `[  [
        Rand( 100, 800 ),
        Rand( 100, 800 ),
        Rand( 100, 800 ),
        Rand( 100, 800 )
    ] * dRead * arg7 * lf, nil, [ 1, 1, 1, 1 ] * arg3 ],
    mod
);
mod2 = DynKlank.ar(
    `[  [
        Rand( 100, 800 ),
        Rand( 100, 800 ),
        Rand( 100, 800 ),
        Rand( 100, 800 )
    ] * dRead2 * arg8 * lf2, nil, [ 1, 1, 1, 1 ] * arg4 ],
    mod2
);

input = Dseq( [ 0.1, 2.1, 7,3, 5, 5 ] * ( rd )
    + mod fold2: 1, inf );
input2 = Dseq( [ 0.2, 2.1, 7, 3, 5, 5 ] * ( rd2 )
    + mod2 fold2: 1, inf );

wr = Dbufwr( input, buffer, 0 );
wr2 = Dbufwr( input2, buffer2, 0 );

signal = Duty.ar( 1 / SampleRate.ir , 0, wr )
    + Duty.ar( 1 / SampleRate.ir, 0, wr2 );
```

Figure 13: Synthesis example from *NeuroMedusae I*.

are read from and written to at a rate equal to the Sample Rate of the sound card. Injected into this are a series of Impulse generators, mod and mod2, which are fed into dynamic resonant filter banks. These resonant filter banks filter not only the incoming impulses but also the feedback as it emerges after being read from the buffers.

The frequency of these filter banks is modulated over time by both noise generating oscillators, and by manual controls given to the performer. This signal is then distorted using a distortion algorithm called folding which wraps any input above 1 and below -1 by mirroring it equal to the magnitude that crossed that threshold. The amount of this distortion changes each sample based on a step sequencer that changes each sample. The signal is finally then written back to the feedback bus. Each trip the feedback audio makes through the loop it becomes more distorted, and more resonantly filtered, often creating sustaining tones. These tones shift over time based on the shifting amounts of distortion, the incoming signals being written to the feedback buffer, and based on the modulations of the resonant filters by the noise generators and by the performer's manual input.

Sounds such as these often exhibit complex and emergent behaviour when played on their own. However, when paired with other sounds, being modulated by other performers, a kind of conglomerate whole is formed, and a fusion of two sound producing feedback circuits is created. These conglomerated feedback loops take on even more complex and emergent behaviours, and serves as a way of melding the different members of the ensemble into a group sound. Effects such as these are simply not possible given traditional ensembles or instruments.

No two performances of Neuromedusae I have been the same, and often the system exhibits emergent behaviour, producing synthesis results that were not manually programmed into the piece. The whole is greater than the sum of its parts. OSCthulhu showed to be quite valuable in constructing this type of Network Music System, greatly simplifying construction of the system, as well as making it more reliable (McKinney and McKinney, 2010).

4.4.2 Neuromedusae II

Neuromedusae II was the second piece created using OSCthulhu and the Medusa System. Like Neuromedusae I, this piece focuses on network based audio feedback loops. However, instead of using Single-Sample Feedback loops, the system investigates the sonic possibilities of real-time Convolution Reverberation in a feedback chain. Convolution Reverb is a technique that is often used to give the impression that a digital signal is playing in an emulated reverberant space. To achieve this effect an *Impulse Response*(IR) buffer is produced that is a recording of a single impulse, often times a single hand clamp or click of sticks, in a real-world space with desirable reverberant qualities. This sharp impulse provides

an approximation of all of the frequencies from zero to the Nyquist limit, and how these frequencies attenuate over time is provides the reaction of the acoustic space as a function of time. This information may then be used to calculate how any arbitrary input signal would react in the same environment, thus creating the illusion that the signal is playing in the space that the IR buffer was recorded in (Farina, 2000).

In Neuromedusae II there are two types of synthesised sounds symbolised by the Sound Ellipses. The first type of sounds are impulse-based with short bursts of energy. These sounds are intended to excite the virtual spaces and mostly do not contain any feedback loops themselves. The second type of sounds are modified Convolution Reverb effects that contain a custom IR buffer and an input and output which feedback into itself. By interlocking these sounds using the Buffer Ellipses, it is possible to create chains of simulated spaces that feedback into each other, creating an impossible virtual world where the soundscape of a forest is contained within that of a Violin.

In some instances IR buffers are taken from recordings of something other than impulses in a reverberant space. These sounds may range from a human voice, to the sound of pouring water. This produces a surreal effect wherein the overall timbre of the recorded sound imbues a characteristic frequency response and reverberation upon the impulse sound. By chaining these sounds together, a self-modulating morass of tones emanate, with each member of the Network Ensemble making alterations to different points along the feedback chain. An example of this type of convolution based feedback is found in the SynthDef code found in Figure 14

In this synthesis code there are two separate feedback inputs, one each for left and right channels. An LFO is created through the combination of a Phasor oscillator (similar to a Saw wave) and a noise generator. A signal is generated out of two Formant oscillators that may be modulated by performer input. These oscillators are then amplitude modulated by the previous LFO. These formant oscillators are then mixed with the feedback inputs and convoluted with a sample of the sound of a woman speaking. This convoluted signal is then mixed with a version of itself which is pitch shifted down an octave. This output is then written to the feedback output. The convolution tends to have the effect of placing over the incoming signal a kind of pseudo reverberation that gives the illusion that the sound is coming out of a woman's mouth.

Each trip through the feedback loop the signal convolutes itself and a portion of it shifted down another octave. Thus any high pitch material eventual become low tones. Furthermore the delay from the block time involved in processing the feedback loop introduces a kind of stuttering into the signal that propagates throughout the frequency spectrum. Thus first is hear a high tone, and then slightly later a lower version, then slightly later an octave lower than that, and so

```
buffer = bufNum;
buffer2 = bufNum2;

rd = InFeedback.ar( bufNum, 1 );
rd2 = InFeedback.ar( bufNum2, 1 );

lfo = Phasor.ar(rd,LFNoise1.ar(0.25, 1, 1),arg7,1,arg7);
lfo = lfo + arg8;

imp = Formant.ar(
    arg1.linexp( 0.0, 1.0, 20.0, 20000.0 ),
    arg3.linexp( 0.0, 1.0, 1.0, 20000.0 ),
    arg5.linexp( 0.0, 1.0, 1.0, 20000.0) * lfo
);
imp2 = Formant.ar(
    arg2.linexp( 0.0, 1.0, 20.0, 20000.0 ),
    arg4.linexp( 0.0, 1.0, 1.0, 20000.0 ),
    arg6.linexp( 0.0, 1.0, 1.0, 20000.0 ) * lfo
);

conv = PartConv.ar(
    imp + ( rd2 * 0.05 ) + ( rd2 * 0.05 ),
    4096,
    bufNumArray[10],
    0.05
);
conv2 = PartConv.ar(
    imp2 + ( rd * 0.05 ) + ( rd * 0.05 ),
    4096,
    bufNumArray[10],
    0.05
);
conv = PitchShift.ar( conv2, 1.0, 0.5 ) + conv;
conv2 = PitchShift.ar( conv, 1.0, 0.5 ) + conv2;

signal = [ conv , conv2 ]*env;

Out.ar( bufNum1, signal[ 0 ] );
Out.ar( bufNum2, signal[ 1 ] );
```

Figure 14: Synthesis example from *NeuroMedusae II*.

forth. Furthermore the signal crosses the left and right feedback inputs, so that the signal bounces back and forth from left side to right side. This synth pairs well with other synths when they are on the same feedback buffer.

When paired with other sounds the sounds fed into the feedback buffer are likes wise convoluted such that they sound as if they are emanating from the woman's mouth, and are also pitch shifted. The sounds tend to intermix and produce strange self-sustaining undulating and atmospheric soundscapes.

Like Neuromedusae I, this creates a system where the individuals melded their individual musical inputs into a single kaleidoscopic mass. An audio recording of Neuromedusae II is available in the attached DVD portfolio.

4.5 Renditions

Renditions is a collaborative multi-media piece that was created by Curtis McKinney and Alain Renaud for the Sonorities Festival symposium in Belfast, Northern Ireland (Sonorities Festival, 2010). The symposium focused on the musical possibilities of spontaneous improvisation over high-bandwidth research networks, and was the concluding event of a major project, funded by the European Union Culture programme, on network music performance. There were three sites in three different cities (Belfast, Graz, and Berlin) that performed music with each other remotely via Jacktrip for low-latency high quality audio transfers. The main research goal of *Renditions* was to investigate the musical and visual possibilities of creating a melding of a traditional graphical improvisation score and an audio visualiser as means of facilitating multi-person improvisation over a network. This score would provide a structural framework from which the improvisers could work, as well as a means to create visual cues for musical materials in a different manner than a simple video feed of a performer might give.

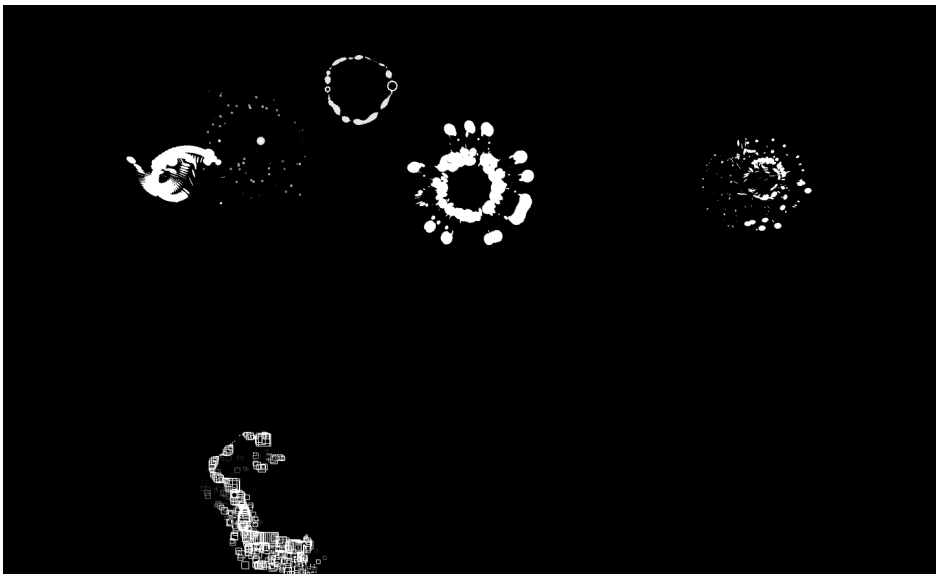

Figure 15: Sound visualisation during a performance of *Renditions*.

4.5.1 Technical Overview

There are four subsystems that comprise the technical portion of *Renditions*. The first subsystem is the audio transmission system. This system was comprised of a set of microphones at each of the three concert halls, the audio mixing consoles at each site, computers running the Jacktrip software for audio transmission, and JMess(a system for dynamically changing JACK audio server router configurations) to coordinate piece changes (Caceres, 2010). The final output of this first subsystem was fed into the second, an audio processing environment built by Alain Renaud that processed each of the individual audio streams, outputting six individual channels: three non-processed signals from each of the three sites, and three processed signals derived from them. These signals are passed to the third subsystem, a computer running SuperCollider code generated by Curtis McKinney that broadcast real-time audio analysis of these streams in the form of OSC to a graphics generating application at each of the three locations. This graphic score system, created in Java using the Processing visual programming library, comprised the fourth subsystem.

The visuals were produced by taking buffers of 512 samples of audio and creating stylised oscilloscopes from this information. The shapes produced by this method were intricate and reflected the audio put into it in an extremely vivid manner. Pure tones produced more round soft shapes, while noisier tones produced more complex and evolving shapes. There are three static elliptical shapes that visualised the three non-processed streams, and three floating shapes that included physics simulations, using Box2D, that represent the three processed streams (Catto, 2010).

4.5.2 Structure and Performance

The visuals serve essentially as both score and conductor, giving the performers individual musical cues, as well as ushering the performance on from one section to the next. There are three main structural points that occur through the course of the twelve minute piece. The initial structural point features a dark background with white shapes that react in a very direct way to the sound that is input into it. As the piece progresses eventually this gives way to a brighter scene with a white background and black shapes. In this second scene the shapes start reacting in more violent ways to the incoming signals. The fourth section starts fading back to black and the shapes grow less violent until eventually the three physical simulated shapes break apart, scattering the oscilloscope's individual components chaotically over the scene. The end is marked by a fade to black. Figure 15 shows a screen capture of the visual system.

This structure gave a very vivid and responsive framework within which the performers could improvise (Renaud and McKinney, 2010). The piece was very well received by the audience as well as by the performers themselves who found

Figure 16: Yig, a feedback based network music instrument.

playing with the system to be a unique experience. Future research into dynamic network based graphics scores include the possibility of using a similar system for a completely electronic performance for three co-located laptop performers sharing information over a local network.

4.6 Curse of Yig

A new piece entitled *Curse of Yig* was created as a reaction to some of the perceived strengths and weaknesses of the Medusa system and the experience of creating visuals for the network activity in *Renditions*. Like *Neuromedusae I/II*, the piece is based upon networked feedback sound producing engines that dynamically form circuits and are controlled through group manipulation, albeit with more emphasis on rhythmic and pulsating sonic material. Named after a story by the horror Author H.P. Lovecraft, the piece attempts to channel the concept of galactic horror and chaos into a pulsating sonic realm of seething energy. This piece runs on a new NMI, simply called Yig, that was developed by Glitch Lich member Chad McKinney with help from the author Curtis McKinney. Yig is much like Medusa, making use of ellipses that represent sound making engines that may be connected together by moving them around a two-dimensional space, however it has been updated with cleaner technology, making use of the Qt GUI framework and a version of SuperCollider embedded directly into C++. Having SuperCollider directly embedded into the application made it possible to create oscilloscopes for each of the ellipses, greatly improving clarity and control for the performers. As well, these updates made it possible to bundle the software as a standalone application that does not require any other installed software or dependencies. Figure 16 shows the Yig system in action

A major development with the composition of *Curse of Yig* was the creation

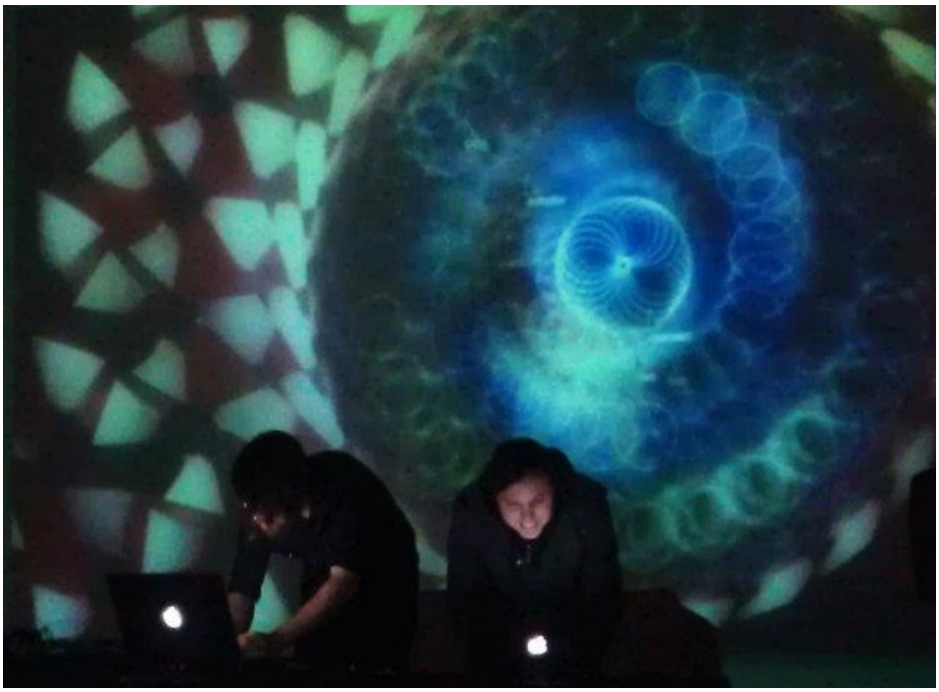

Figure 17: Yig in performance at the Network Music Festival 2012.

of a custom made visualisation system that was meant specifically for that piece. Previously Glitch Lich mainly relied upon showing the GUIs of the various software instruments we created used during performance. While this proved edifying for the audience, it was not particularly aesthetically pleasing. During the composition of *Curse of Yig*, there were experiments with creating a second version of the GUI, but instead of being used by the performer, it would be specifically made to be viewed by the audience. The visualisation would share many of the same characteristics of the main performance GUI, but would tailor it to both be more aesthetically pleasing, and to be more readily cognisable by an audience member sitting several feet away and watching it on a projected surface.

To accomplish this there are several clear differences between the "performance" GUI and the "audience" GUI. In the audience GUI there exists the same general elements as the performance GUI, such as ellipses representing sounds, sonic interactions, parameter changes, cursors representing each members cursor on the screen, and inter-band chat. However all of these elements have been modified to be more readable. Generally this means enlarging their size and increasing visual contrast, as well as removing all of the other elements deemed unnecessary (such as readout data from SuperCollider scserver process). As well, these elements were given visual styling and 3D visuals effects that fit more in-line with the aesthetics of the piece, instead of the utilitarian visual nature required of the performance GUI. To enforce the aesthetic of the piece even further, the visuals are themselves fed into their own visual feedback loop, providing the audience with two layers of recursion, one sonic and one visual, which themselves feed into each other. Figure 17 shows this feedback based audience "GUI" on display

during a Glitch Lich performance.

4.7 Leech

Part of the research conducted was to investigate the possibilities of using the underlying data structure of networks as a musical resource in an installation environment as opposed to the normal concert environment that has been used by Glitch Lich. Several network based sound art installation pieces have been created in this vein as part of this research, including *Leech*, *Flow*, and *Flow Redux*.

The goal was to sonify some form of network data in a musically meaningful way, as means to illuminate the very internal mechanisms that network performance relies upon. Eventually, it was decided that it would be musically, academically, and politically interesting, to investigate illicit data networks, specifically BitTorrent networks, used for the transfer of pirated music (Cohen, 2011). This was chosen as it had a meaning to people beyond being a simple exercise in translating data from one medium to another. Table 1 shows an overview of how the different data types in a BitTorrent download are sonically and visually mapped in *Leech*.

4.7.1 Technological Overview

Leech involves several interlocking open source technologies. The visuals and logical systems are developed with the Java programming language (Reges, 2010). The BitTorrent transfers are accomplished using the OSX application Transmission (Transmission Project, 2011). Analysis of transfer traffic is executed with the Java library Jpcap (Fujii, 2011). Geographic placement of peers is derived using the freely-distributed version of Max Mind's GeoLite City (MaxMind, 2011).

Visual representation and GUI elements are developed with the Processing programming language, used as a library from within Java (Reas et al., 2007). Sound is produced with the real-time sound synthesis programming language SuperCollider (Wilson et al., 2011). Communications between Processing and SuperCollider is accomplished with the Open Sound Control (OSC) protocol (Wright, 2002). LAME is used to convert partially completed MP3 downloads and load them into SuperCollider for audio processing (Lame Project, 2011).

The basis for all of the visual and musical content in *Leech* is derived from data-mining. Therefore it is the data-mining technologies that are the core engine of the whole system, driving the flow of the entire experience. There are three distinct modules that act in coordination to derive information about the BitTorrent transfer.

The first module is a Remote Procedure Calls (RPC) communication layer that controls and queries the Transmission BitTorrent Client. Through individ-

ual calls to Transmission the module can control the torrent download by starting and stopping the transfer, altering the number of peers to download from, and increasing or decreasing transfer speed. Data can be requested about the download, including IP addresses of peers, name and size of the torrent files, download rate, and progress of download.

The second module utilises the IP address database GeoLite City. This database contains the geographic location of most of the distributed IP addresses on the internet. Regularly updated, the freely distributed version is accurate to the city level in most cases, which is more than adequate for the purposes of this piece. By cross referencing this database with the IP addresses obtained from Transmission, it is possible to geographically place the peers that are transferring pirated audio.

The third data-mining module monitors Internet traffic on the local machine, capturing each packet of information that is being transferred to and from the local host. From these packets of information it is possible to derive the sending and receiving IP address and payload information. By cross referencing this module with the previous two modules it is possible to derive when a packet of pirated BitTorrent information is being transferred between the local host and particular peer. This information may then be depicted geographically, and sonically rendered.

Mined Data	*Module*
Torrent Progress(%)	Torrent Client
Download/Upload Rate(kB/s)	Torrent Client
File Names/Sizes(mB)	Torrent Client
Number of Peers(int)	Torrent Client
Leecher vs. Seeder(%)	Torrent Client
Peer Location(ϕ/λ)	Torrent Client/GeoLite
Packet Transfer(ϕ/λ)	GeoLite/JPCap
MP3	Torrent Client/Lame

Table 3: Mined data and the modules used to derive them

4.7.2 Mapping Data

Leech is a multi-media composition, and thus it is not merely enough to derive the characteristics of a torrent download. Mapping this information in a visually and musically meaningful way is the challenge of the entire composition. The basic visual backdrop is a vectorised world map, upon which all other mined information is depicted (which may be seen in Figure 18).

Using the three data-mining modules it is possible to derive several characteristics of a peer. A peer's geographic location, download progress, and when they are sharing pirated information can all be derived. Using Processing, the

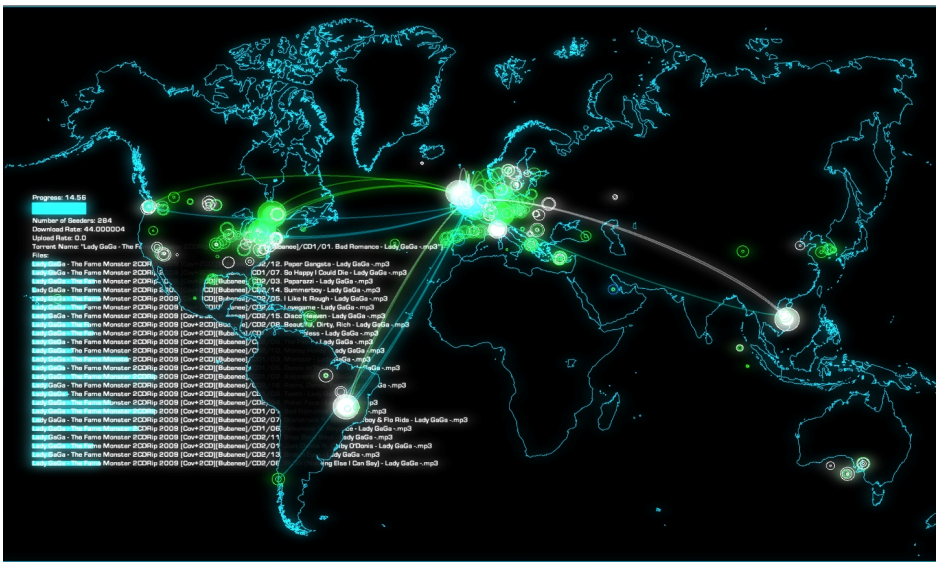

Figure 18: Visualisation of network data in *Leech*

geographic location of a peer is rendered visually as a pulsating ellipse placed geographically on a vectorised world map. The colour of the ellipse denotes the progress of the peer's own torrent download. A peer with less than 100% downloaded is represented with a white ellipse, and is referred to as a "leecher". A peer that has finished downloading and is currently only uploading data is represented with a green ellipse and is referred to as a "seeder". Currently there is no static sonification of a peer's geographic position, or when a peer is added to the system. Instead these parameters are sonified in conjunction with other mapping systems described later.

The overall progress of the BitTorrent download and the individual progress of each MP3 transfer are also mapped visually and sonically. On the left hand side of the screen a series of bright blue bars are shown extending horizontally towards the centre. As the transfer progresses to completion these bars extend further out. The names of each MP3 being downloaded is displayed over their respective bar to show their respective download progress. Sonically, these values are mapped much more directly than the packet transfer sounds, and it is much more easily cognisable to hear the effect of the download on these sounds.

One synth is produced for each individual MP3 being downloaded, usually in the range of 10 to 15 depending on the size of the album being downloaded. These sounds undulate as a kind of ambient background to the piece. As the download progresses from the beginning to completion several characteristics of the sound are modulated. High frequency content, undulation speed, feedback amount, and general timbral complexity all increase as the download progresses.

Figure 19 shows a snippet of SuperCollider code mapping file transfer data to synthesis parameters. Depicted is a Gendy stochastic oscillator, a concept conceived by composer Xenakis in his treatise *Formalized Music* (Xenakis, 2001).

Unlike periodic oscillators that oscillate linearly, this oscillates based upon a given distribution of probabilities. The oscillator is being deployed in sinus mode, which means that it is sampling an outside oscillator to provide a constantly shifting probabilistic distribution. The third and fourth inputs are the external oscillators being sampled, which are themselves Gendy oscillators(not depicted here). Inputs five and six determine the frequency of the oscillator. Very simply, as the download progresses, the pitch goes up. The final slot depicts the number of control points sampled during one period of oscillation. As the download progresses, the amount of control points sampled per period increases, thus increasing high frequency content and timbral complexity. This demonstrates a very direct influence of the download being exerted on the sound. The staggered progression of each file transmission produces a heterophonic texture that moves as a loosely connected cloud from relative timbral simplicity to more intense and complex tonal emissions. This is useful in giving an overall form and shape to the piece.

```
osc3 = Gendy.ar(
        6, //Sinus Mode
        6, //Sinus Mode
        osc1, //Sampled Oscillator
        osc1, //Sampled Oscillator
        fileProgress.linlin(0,1,520,47000), //MinFreq
        fileProgress.linlin(0,1,520,47000), //MaxFreq
        initCPs: 100, //Initialized Control Points
        knum: fileProgress.linlin(0,1,40,100).round(5)
);
```

Figure 19: File transfer sonification code in SuperCollider.

Each time a packet of information is identified as being part of the torrent download, the system identifies the parties sending and receiving the pirated information. This determines whether or not the local host is downloading or uploading information, and to whom they are uploading to or downloading from. Furthermore, by cross referencing against the attributes of the peer involved, it is possible to depict whether the transfer involves a seeder or a leecher. Using these parameters the system organizes packet transfers into four subtypes: downloads from leechers (DL), uploads to leechers, (UL) downloads from seeders (DS), and uploads to seeders (US).

Whenever a packet transfer is identified it is rendered visually as a coloured curve stretching from the local host to the peer involved. The orientation and colour of the curve depict what type of transfer it is. A DL transfer is a white line curving upwards. UL transfers are white and curve downwards. DS transfers are green and curve upwards. US transfers are blue and curve downwards. This information is also passed to SuperCollider via OSC to be rendered sonically.

In SuperCollider there are four types of synthesised sounds that are produced based upon the four packet transfers types. Characteristics of the packet transfer are also used to further modulate the characteristics of these sounds. Download rate, local transfer progress, peer transfer progress, and peer latitude and longitude are all characteristics that influence the synthesised sounds. Due to the large quantity of packet transfers throughout the course of a twenty minute performance, emphasis is placed more on variety of results rather than on simplistic sonifications of values. Thus it is difficult to briefly summarise how these values are mapped in each synthesised sound. Instead of attempting to dissect a large amount of sonification code, a small example of one line of code is provided to give some idea of the techniques used to sonify the packet data.

Figure 20 depicts a snippet of code near the end of a packet capture sonfying a synthesiser in SuperCollider. This code depicts a delay line that is processing an earlier synthesised audio signal. The input signal is being modified by two nested single pole band-pass filters. These filters' resonant frequencies are modulated by the geographic location of the peer that the packet of information is being transferred to or from. Thus the further west a peer is the more high frequencies in the first filter. This is fed into the second filter which filters out more low frequencies the further south the peer is located. The progress of the peer's download determines the delay time of the delay line. Peer progress ranges from 0.0 to 1.0, however here that value is being wrapped at a modulus of 0.5. Thus, as peer progress advances from 0.0 to 1.0, the delay time of the delay line will start at 0.0 seconds and reach a peak of 0.5 seconds at the mid point, then return to 0.0 and increase to another peak of 0.5 at the completion of the peer's download. Finally the original signal is summed with the delay line and fed into a feedback loop (not shown) to produce a recursively filtered echo effect.

This is one part of a much more complicated and interwoven whole, with each mapped parameter serving many purposes throughout the whole sound. This produces the desired effect: an intricate and constantly evolving sound with a wide array of variety to sonify the many different characteristics of the thousands of packets of pirated information that are transferred throughout the performance.

The final system manages the actual audio that is being pirated. This system is not so much mapping as it is resource collection. This system also addresses the real goal of pirating MP3s, which is to actually *listen* to them. Thus it seems technically and musically logical to provide a system for playing back these stolen sounds. By using the keyboard the performer may move a red rectangle between the MP3 progress bars. Pressing certain buttons will convert the selected MP3 into a WAV file and load it into SuperCollider. If the file is incompletely transferred, it creates a WAV file that skips missing audio data, providing a shorter audio file with sharp jump cuts. Then the system employs one of several playback synths that alter the audio in different manners. The goal with these synths is

```
BufDelay.ar(
        LocalBuf(SampleRate.ir*0.5),
        OnePole.ar(
                OnePole.ar(
                        synth,
                        lat.linexp(-150,150,-0.99,0.99)
                ),
                lon.linexp(-150,150,-0.99,0.99)
        ),
        (nodeProg%0.5),
        0.75,
        synth*0.75
);
```

Figure 20: Packet capture sonification code in SuperCollider.

to playback the audio in heavily altered yet still somewhat recognisable fashion.

Figure 21 shows an example of SuperCollider code that plays back pirated audio data. This system uses Fast Fourier Transformation (FFT) processes initialised with very large buffer sizes. Using a spectral buffer playback system, this plays the audio data at 3% of its original speed while maintaining the same pitch. Next the audio's spectral data is squeezed into half the space it normally fills. A brick wall filter is placed upon the signal to discard most of the high spectrum and leave the low end data. The low frequency data is then spectrally enhanced, placing three new harmonics above each frequency in the spectrum. Lastly it is once again squeezed into half of the spectral field. This produces a rich and slowly evolving low end drone sound that is heavily influenced by the bass drum and bass lines of a pirated song. It is thoroughly altered, however given familiarity with a song it is actually rather easy to detect a slow moving distorted version of the bass present in a song. While the other two systems are (more or less) tuned, this system consciously makes no effort to alter the tonality of the original song. A combination of these three distinct layers, the droning file transfer mapping, the percussive packet capture sounds, and the processed songs, produces a kaleidoscopic polytonal morass.

```
bufnum2 = LocalBuf.new(1024*16,1);
chain = PV_PlayBuf(bufnum2, recBuf, 0.03, 0, 1);
chain = PV_BinShift(chain, 0.5);
chain = PV_BinShift(chain, 0.5);
chain = PV_SpectralEnhance(chain,3,2,5);
chain = PV_BinShift(chain, 0.5);
```

Figure 21: Pirated MP3 playback code in SuperCollider.

4.7.3 Artistic Considerations

The network data being mapped in *Leech* may be categorised by the manner in which it traverses its range. Pseudo-linear data moves in one direction, never skipping forward or backwards. This includes the overall download progress, progress for each individual torrented file, and number of peers that have connected to the system. This data is not strictly linear however, as the time span it takes to traverse the range of this data is not predetermined and differs for each performance and for each datum. Other data traverses its range non-linearly, skipping forward and backward at differing rates of speed. This includes download/upload rate, peer locations and peer download progress.

Having these two different types of data present is quite useful for creating a musical composition. Linear data allows the piece to have an overall form and shape, and to create a sense of tension, much like a normal non-real-time precomposed piece. *Leech* will always start off with quiet drones in the beginning, with the download progress at zero. As the piece advances, the download progress reaches closer to 100%, the drones increase in amplitude and complexity, creating a long build in tension. However, the non-linear data serves to provide variety in the piece. While pseudo-linear data tends to have an effect on the top most scale of the piece, being the form, the non-linear data provides unpredictable embellishment at the note scale. Download and upload rate are in constant flux, and each peer that a packet is transferred to will have a different and unpredictable geographic coordinate and download completion. These constant variations on a smaller time scale produce different tonal and timbral figures and patterns and add unpredictability from moment to moment. In combination these two forces give the system a sense of direction and life.

Transparency in presenting the music being pirated is central to the piece. In compositions that focus on the concept of borrowed material, such as Luciano Berio's Sinfonia Mvt. 3, it can be at times difficult to identify exactly what is being borrowed and manipulated. *Leech* attempts to balance creative musical modification with transparency. Audio effects that maintain cognisable portions of the sonic material are purposefully employed.

One example of this is FFT based speed reduction, which create long evolving drones while maintaining identifiable pitch material. This audio transparency is accentuated by the use of visuals in the piece. The name of the artist, album, and each individual MP3 is clearly displayed in the visuals to inform the audience of exactly which material is being downloaded. Whenever a song is selected to be played back in modified form, it is hi-lighted on the screen to inform the audience exactly what song they are hearing being processed.

Collaboration is key to the mechanisms and philosophical underpinnings of *Leech*. The process of illegally obtaining music is in fact a social and communal activity. Peer-to-peer networks such as BitTorrent require that a group of users

proliferate information between each other in a mutually beneficial structure. The visuals in *Leech* attempt to demonstrate that the act of piracy brings together people from across the globe (though with less frequency in places such as Africa and China where free Internet usage is restricted or unavailable). These peers are from many different cultures and societies, setting aside any differences to collaboratively share music.

The sounds themselves are also a collaboration. Two composers are involved in the artistic production of the piece, Curtis McKinney and Chad McKinney, twin brothers who have been collaborating for years on musical compositions. These composers also collaborate with the artists whose works are being sonically manipulated. Furthermore, the actual choice of what to pirate for performance of the piece is determined by popularity on the BitTorrent search engine (Fung, 2010). Using this selection process popular artists such as Rhianna and Lady Gaga have been used for the piece in recent past.

4.8 Flow

In July 2010 SCAN commissioned Alain Renaud, Tom Davis, and Curtis McKinney to create a new piece for the Public domain arts festival held in the Bournemouth town centre gardens (Public Domain, 2010). One of the main attractions of the arts festival was the presence of a large LED outdoor screen that the artists were encouraged to use to create pieces that the public at large could appreciate.

Given the pastoral setting, and the large amounts of public exposure, it seemed a great opportunity to pursue research on multi-user instruments that may be played by members of the public. The design of the instrument was inspired by the children's game "Pooh Sticks", invented by *Winnie the Pooh* author A. A. Milne (Milne, 1928). The game involves children throwing sticks into a stream with the hope that their stick flows down stream faster to win a race to a designated endpoint. Flow takes this basic concept, and turns a stream into a water based sequencer for sonic and visual events. To accomplish this ten infra-red sensors were placed down the length of the stream. These sensors are triggered whenever the infra-red beam they emit are broken. This trigger information is fed into a computer via an Arduino sensor interface (Banzi, 2009).

There are ten different kinds of sounds that can be triggered, one for each sensor, though these sounds have random variables that change the way they sound each time they are triggered. There are also 10 coloured ellipses on the previously mentioned outdoor screen that grow in size, and change colour when the corresponding sensor is triggered. To play the instrument a person takes a beach ball and throws it into the stream. As the ball floats down stream it triggers the infra-red sensors one by one(though in some instances due to wind a sensor might be triggered more). A picture of this set-up in action can be seen in Figure 22. Multiple people were able to throw balls in at the same time, or in staggered time,

creating a polyphonic musical melody.

Figure 22: Public performance of *Flow* at the Public Domain arts festival.

The sounds that were created for *Flow* were specifically tweaked as to be sonically interesting, yet easily listenable, and were tuned to a C pentatonic scale. This was done so that the piece could be listened to for long periods of time(as the piece runs for hours at a time), and so that members of the general public would find it pleasant to interact with. To give the piece a site specific flair a background layer of sound is comprised of various samples of sounds that were collected from all over the Bournemouth town gardens. These sounds included the sound of the stream itself, birds in the trees around the stream, and the sounds of people chatting as they walked by. The sounds were processed using time stretching, pitch manipulation, reverberation, and amplitude modulation to produce a soft eerie atmosphere. All of these attributes gave *Flow* a calm ambient character that was easy on the ears. A video recording of Flow may be found on the accompanying DVD portfolio.

The piece was received very well by public, with approximately 300 individuals using the piece during the time the that it was running. Unexpectedly, *Flow* was very well received by children, many of whom would throw balls into the stream and listen to the sounds many times over. The one flaw with the system, as minor as it was, was the ball reclamation system that was utilised. Unfortunately there was no feasible method for automatically returning balls that had flowed to the end of the stream. Thus, for every ball that was thrown down stream an individual, would have to manually carry it back to the beginning. This turned out to be a rather inefficient and manpower intensive procedure. Should the piece

be done again, thought will be given for a different manner to recollect balls that flow to the end. Future research into this area may include re-appropriating *Flow* for different mediums than water to increase the venues that the piece may be performed at.

4.9 Mutagen

In the summer of 2011 research began on a new software system that focuses on utilising the unique capabilities that the newly developed OSCthulhu version 2.0 has to offer. This software, dubbed *Mutagen*, is informed by the research conducted on both Flow and Neuromedusae I and II, taking into account lessons learned from systems that enables multiple users to collaboratively play a single musical system. Mutagen is a multi-user network based sequencer that allows multiple individuals to simultaneously create and modify a time-based sequencer that may be used to freely drive any sonic or visual software system that accepts OSC messages. The project was inspired by the Iannix OSC based sequencer developed by the Iannix Team in France (Iannix Team, 2011). Iannix is a flexible OSC based sequencer that allows a user to build sequences of events that may drive any software that accepts OSC. Mutagen began as project aimed at taking a similar concept and opening it up so that multiple individuals may edit the same sequencer at the same time. Since then, Mutagen has taken on more of a character of its own and has evolved to the point that the two pieces of software are not so comparable. Whereas Iannix attempts to create innovative new sequencer designs of differing functionality, Mutagen instead focuses on taking tested traditional methods of sequencing and finding the unique possibilities that multiple users collaboration may bring out of it.

4.9.1 Networking a DAW

At first glance Mutagen appears similar to a MIDI sequencer you might see in commercial software such as Logic or Pro Tools (Nahmani, 2009) (Avid Audio Inc., 2011). By default there is a grid that organises the sequencer like a traditional MIDI sequencer, complete with beats, and 127 steps that are organised according to the keys on a piano. These values are floating point, and the range and scaling are customisable by the user. A currently unimplemented planned feature of Mutagen will be to allow users to dynamically change the way in which the sequencer organises and quantises musical material. A user may input notes as traditional note blocks whose placement and length may be adjusted. Mutagen also allows for the creation of free-form multi-breakpoint quadratic curves. These curves allow the user to sculpt sequences of events that are more about change over time than rhythm or melody. An image of the Mutagen interface with these control curves may be seen in Figure 23.

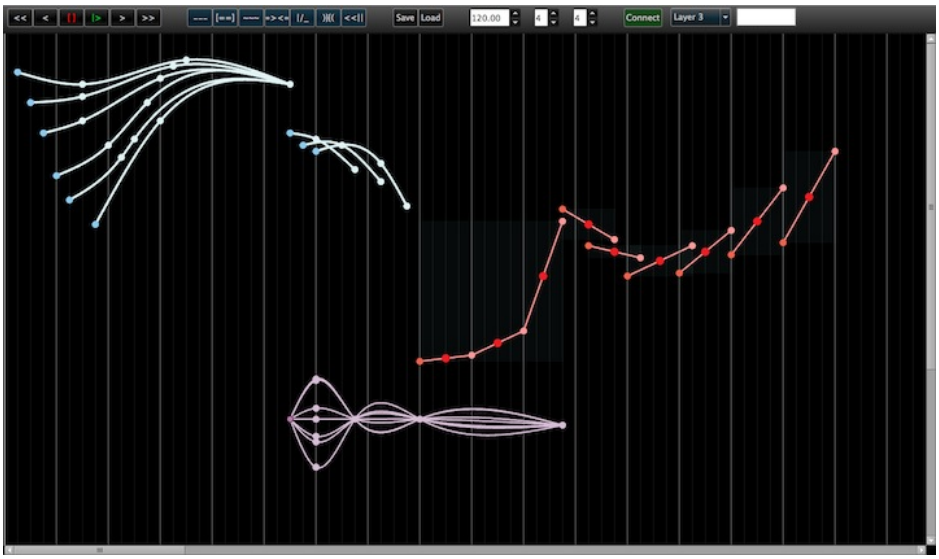

Figure 23: Screen capture of the *Mutagen* sequencer.

The major difference between Mutagen and commercial software systems currently available is the ability for the system to share the changes a user makes in real-time over the OSCthulhu network. Each user will have their own tracks that they may edit, which are stacked vertically. To provide more space to visibly see everyone's musical material each user's track is capable of having multiple layers of musical material represented on a single musical track. Currently up to 5 layers may be placed simultaneously for each track, though that may be increased in the future. These layers are also how the user organises their OSC message output. Each layer may have a different OSC address associated with it, so that each layer's messages may be interpreted differently from each other. A user may save and load sessions just like any sequencer software.

The multiple user interface allows for some interesting musical interactions that may be used in either performative or home-editing situations. Users can build repeating rhythmic and melodic riffs that change over time based on the edits that each user makes to their sequencer tracks. The multiple users may interact with each other by freely copying the musical material other users have created and pasting it into their own track, allowing to modify what other users have created in real-time. Furthermore, users may create structural meeting points by placing certain markers at certain points on the sequencer. This allows users to create simultaneous section changes or coordinated cues that may not be possible in a normal real-time improvisational situation.

4.9.2 Glitches....And Not The Good Kind

However, while initial testing showed the novelty of networking a DAW, results from attempts at using it in actual live performance settings proved to be less than desirable. Given the complexity of software such as DAWs, which are often given

large paid dedicated development teams, developing one as a lone research student resulted in numerous glitches and undersized behaviour. Adding networking, a wily and unpredictable beast in its own right, on top of these issues only compounded matters. Furthermore the project proved to be something of a time sink and large amounts of time were put into it simply getting it to function correctly, instead of using it to produce a new piece.

4.10 Simulacra

A new piece was being composed for Glitch Lich during the time that Mutagen was being developed, and this piece was used to drive much of Mutagen's later periods of development. This piece, entitled *Simulacra* was a continuation of the ideas and experiences gained from composing *Curse of Yig*, seeking to create visual systems for representing an NMIs GUI in an aesthetically pleasing manner. To represent the curved and flowing lines of Mutagen, a series of serpentine like figures would slither around a three dimensional field like a kind of bio-luminescent underwater digital organism. Figure 24 shows these visuals in motion. Thanks largely in part due to the problems with Mutagen, *Simulacra* spent a considerable amount of time in gestation, though most of that time was spent simply attempt to get the technology to work correctly. During much of this time *Simulacra* was more of a proof of concept than actual performable piece.

Figure 24: Visuals in *Simulacra*

4.10.1 Failure and Reboot

The issues with Mutagen came to a head when Glitch Lich performed an initial version of *Simulacra* using Mutagen at the *Live Interfaces* conference in Leeds in September 2012. By all accounts this performance was a failure, as there

were many technical issues, crashes, and glitches throughout the performance. Mutagen had been experiencing issues with performing remotely with the other members. Due to this it was decided that only local members Chad McKinney and Curtis McKinney would perform. However, even given this simplified set-up, the concert proved disastrous. By the end of the performance a total of twelve hard crashes of Mutagen had occurred.

The performance was semi-salvageable simply because for *Simulacra* the sound production engine was moved out of the sequencer and into the visualisation application so as to more easily create oscilloscopic effects. This had the added benefit of the sound system not crashing when Mutagen went down. However this made performing the piece extremely difficult. Even when Mutagen was behaving, it was found that the amount of micro-management required to perform with it made it particularly cumbersome. Even with all of this micro-management, enacting musical gestures seemed difficult and awkward, and at no point did the members of the band feel completely in control of the performance.

After the debacle in Leeds it was decided that something drastic should be done to improve the reliability and performability of the piece. Thus, despite the more than year long development, Mutagen was outright scrapped as the sequencer for *Simulacra* in favour of a new solution. Instead of a complex networked DAW used to sequence the piece, a much simpler and more elegant solution was chosen. The visual system is instead now both the aesthetic three dimensional visualisation as well as the GUI system, controlled directly by a MIDI controller. As well, given the long and arduous development of Mutagen it was decided that efforts should be put into more reusable solutions for Glitch Lich software. What emerged from this was a new framework for NMIs entitled *Azathoth* (covered in depth in section 4.11).

4.10.2 Composition and Sonic Infrastructure

Artistically, *Simulacra* is a departure from some of the previously more chaotic, unruly, and amorphous pieces composed by the ensemble. The piece was taken as an opportunity to extend further some of the rhythmic musical developments that Glitch Lich had begun to employ with *Curse of Yig*, but to bring these elements into even tighter synchronicity. Due in no small part to the lengthy gestation period, *Simulacra* developed into perhaps the most clearly composed pieces by the ensemble, featuring a strict structure and musical pacing over time. However, there is still much room made for performability and algorithmic complexity.

Simulacra relies upon some of the same techniques used in previous pieces, namely networked feedback loops, however it attempts to harvest the feedback audio in new ways for the ensemble. Unlike previous feedback instruments, which used the audio collected from feedback inputs in real-time, instruments in *Siumulacra* store this audio into audio buffers and then uses this buffered audio

Figure 25: Performance of *Simulacra* at the Network Music Festival 2013

in ways similar to sample processing. In this way, NMI is constant self-sampling itself from different regions of its output. Emphasis is placed on cold, glitchy, and digital sounds. This self-sampling technique helps create these kinds of sounds by constantly chopping and scanning through recordings of the different component parts, creating sharp and rhythmic sounds intermixed with washes of noise and delay-like effects.

Another new technique tried in *Simulacra* is division of responsibilities for synth control in the ensemble. In previous piece while a performer's synth might be connected to other performer's synths in some form of network, the modulation controls always belonged to one performer at a time. In *Simulacra*, all of the synths created have exactly two modulation parameters to control them, one of which is controlled by the performer who "owns" the synth, and another controlled by another random member of the ensemble. This was done to explore the effects of even deeper interconnectedness within the ensemble. An example of these types of networked self-sampling synths can be found in Figure 26.

This is code taken from a synthdef file in the piece *Simulacra*. This synthesis code samples audio that is written to the main output bus by other performers and synths in the NMI, mangles that audio, then writes it back to the same output bus for usage by the other synths in the network. At the top the signal on the main outs is input into the synth. Next, a series of rhythmic triggers are generated using an impulse generating oscillator and two probability gates. Noise oscillators generate modulating values used to modify the characteristics of the sound over time, stored in variables mod1 and mod2. Two buffers for audio are created and stored

```
signal = InFeedback.ar( 0, 2 ); //Input from mains
trig  = Impulse.ar( 10 );

trig1 = CoinGate.ar( 0.8, trig );
trig2 = CoinGate.ar( 0.8, trig );

mod1 = Latch.ar(
        LFNoise0.ar( 400 ).range( 0, buf1.numFrames ),
        trig1
);

mod2 = Latch.ar (
        LFNoise0.ar( 400 ).range (0, buf2.numFrames ),
        trig2
);

buf1 = LocalBuf( SampleRate.ir * 4 ).clear;
buf2 = LocalBuf( SampleRate.ir * 4 ).clear;

RecordBuf.ar(signal[ 0 ],buf1,mod1,1,0,trigger: trig1);
RecordBuf.ar(signal[ 1 ],buf2,mod2,1,0,trigger: trig2);

signal = ( signal * env * 1.25)+( [
        PlayBuf.ar( 1, buf1, 1, loop: 1, trigger: trig1 ),
        PlayBuf.ar( 1, buf2, 1, loop: 1, trigger: trig2 )
] * env2 );
```

Figure 26: Synthesis example from *Simulacra*.

in variables buf1 and buf2. Next, the incoming audio is recorded into the buffer at constantly shifting sample indices. These sample index changes occur every time a trigger is received, creating a characteristically rhythmic jittering/chattering to the sound. Next these buffers a series of oscillators play back the stuttered audio recorded into the buffer, and the outputs this audio in combination with the original feedback signal onto the output bus.

Using these techniques of networked self-sampling, shared control, and visualisation, *Simulacra* seeks to explore the concepts self-perception and identification in a society increasingly interconnected. *Simulacra* explores a world space where these interconnections are taken to their logical next step, tying members of the ensemble into a singular unit, and losing track of where one member begins and the next ends. This interconnection is taken even further during performance, by inviting the audience to connect with the ensemble in real-time during performance. Like other performance systems used by Glitch Lich, *Simulacra* has a system for displaying the chat of the ensemble during performance to give the audience a peak into the inner-workings of the band during performance. However, this chat window has been opened up to the members of the audience. During

performance any member of the public may participate in this shared dialogue by creating a tweet with Twitter and using the hash tag glitchlich (Twitter, 2013).

4.11 Azathoth

After developing, and helping develop, many NMIs over the course of this research it was deemed that a framework could be created that could significantly decrease the amount of repeated "boiler-plate" coding which was involved in the creation of an NMI. This would be directly applicable for members of Glitch Lich, but could also be of use to other developers seeking to create similar styled network instruments. This framework was imagined as being an analogue to a video game engine, but specifically geared for creating multi-media network pieces. This "Network Music Engine" is called *Azathoth*.

4.11.1 Features

Azathoth is jointly developed by Glitch Lich members Curtis McKinney and Chad McKinney, and embodies the lessons that have been learned over the course of developing NMIs as part of Glitch Lich for several years. Azathoth is design from the ground up to be a general purpose, plug-in and play library for developing NMIs in C++. Azathoth supports the creation of multi-media piece by bundling together all the components required for algorithmic audio and visual creation into one linkable library package, as well as many of the tools required for a network piece.

To accomplish this Azathoth ties together several technologies into a singular framework. Azathoth is divided into six technological modules that consist of C++ style namespaces and static methods that may be called at any point in a user's code. These modules provide the end-user with all of the capabilities necessary to easily construct a new multi-media network piece. The first module is the core module which is contained in the "az::" namespace. This modules serves as the main hub for controlling *Azathoth* and contains the methods necessary to start and stop a network piece. Furthermore, all of the standard method calls for each of the other modules are routed through the core namespace so that the user is only required to include a single header file and need only to reference a single namespace. Should the user require more capabilities outside of the general purpose methods they are free to include the other headers files and call lower-level method calls from there.

The second module is the "osc" module whose purpose is the handling of all the networking capabilities in *Azathoth*. For this networking the osc module uses an implementation of OSCthulhu that has been completely retooled for C++ use, and making extensive use of the Boost C++ library. The core module has been structured in such a way that for regular usage the end-user is no longer required

to write any network code themselves. To accomplish this the osc module encapsulates all of the necessary OSCthulhu osc based API calls inside of methods and call-back functions, most of which are handled automatically by the system without the need to be manually called by the user. The requisite network calls needed for joining and leaving an OSCthulhu session are called when the user calls the core start and stop methods. From the start method the user may optionally also specify the name of the piece that they are starting, and ports for the piece, OSCthulhu, and the server, should that choose not to use the defaults. A system of call-back functions is in place to handle the addition and subtraction of OSCthulhu SyncObjects. These call-back functions are implemented through usage of the Boost library's signal/sockets framework. To add an object on the OSCthulhu server the user calls the addSyncObject method. The user provides this method with a SyncObject container into which the user must stream a unique id (which optionally may be provided by OSCthulhu automatically) as well as the SyncObject's type and subtypes. The user may also stream a series of initial arguments for the SyncObject. This initialisation list will also determine the number of SyncArgs the SyncObject contains on the server. The arguments provided must be either of types string, int, or float. Any number of these arguments may be provided and they are indexed by order as opposed to a key style interface. To remove a SyncObject from the OSCthulhu server the user calls removeSyncObject and provides the string id of that SyncObject.

To handle SyncObjects arriving from the OSCthulhu server the user provides a boost style callback function which is stored in a container with a given type and subtype string. This function will be called whenever a SyncObject of the given type and subtype is received from the server. The user also provides a callback function to be called for removed SyncObjects. A similar interface is in place for SyncArgs. To set a SyncArg a user simply calls setSyncArg and provides the SyncObject id, the argument number, and the value it should be set to (which should be the same type it was initialised to, otherwise the message will be ignored). To handle receiving SyncArg changes from the server the user provides a callback function which will be called when a setSyncArg message is received matching that particular SyncObject id and argument number. Through extensive usage of OSCthulhu in the past it has been useful to preemptively set an argument locally while simultaneously sending a setSyncArg message to the server that has a flag set to prevent the message from bouncing back to the user. This is done to make usage of the networked system seem more immediate to the local user while keeping the OSCthulhuServer up-to-date. However accomplishing this was an overly complicated task before requiring the user to manually create their own call-back system. With the new call-back interface provided by Azathoth this method is greatly simplified and the user may accomplish the same task by simply calling the setSyncArgLocal method instead.

The module for creating and manipulating visuals in *Azathoth* is provided through the usage of the Cinder C++ openGL framework. This allows the user to create complex and interactive 3D graphics which will be networked by OSCthulhu. This visuals module is further extended by the "GUI" module, whose purpose is to provide the end-user with the capability to create extensible GUI systems for the multi-media pieces. A series of inheritable abstract class have been created to allow the user to easily create new GUI windows. Further some commonly used GUI features and menus are pre-baked for usage by user. GUI widgets for network chat, logged user info, and SuperCollider server management and monitoring all come pre-developed with Azathoth, while also allowing the user the capability to customise the look and feel of these widgets.

The fifth module handles sound synthesis and uses a newly developed implementation of Chad McKinney's *libsc++*. This library is a port of SuperCollider to the C++ programming language. Having the sound creation embedded into the same software as the rest of the application gives many benefits, including the ability bundle the application as stand-alone software, easier implementation of sound data visualisation, and the elimination of OSC networking between sound and visualisation software. This module behaves nearly identically to sclang in SuperCollider, with the capability to spawn and stop synths, modify synth arguments, and create and modify sound buffers. Notably however there is currently no pattern interface for creating rhythmic material in libsc++, forcing the user to implement rhythmic materially internally in SynthDefs with demand rate UGens. Future work will be done to further develop libsc++ and provide an easier convention for the creation of rhythmic material.

The final module in *Azathoth* "midi" module, which handles MIDI input to control network pieces. This module is still in initial development and currently only contains an interface for easily interacting with the Korg NanoKontrol MIDI controller (the MIDI controller of choice for Glitch Lich) (Korg, 2008). Further developments are planned to provide simple plug and play interfaces for other MIDI controllers. Currently other controller types, such as Wii-motes, or devices connected through Arduinos, may interface Azathoth simply by calling the requisite OSC API calls to OSCthulhu. In future work simple interfaces for these types of devices may be included in *Azathoth* as well.

Currently Azathoth is in its infancy and is changing every day as new features are added and current features streamlined and changed. For future work, once Azathoth has reached a mature enough stage, the author would like to release a stable binary to the network music community at large so that others might benefit from the hard work and lessons learned during this research.

In this chapter, aim #2 of this research, "the creation of new, and the refinement of old, tools and techniques for composing performing and designing NMIs", has been addressed in detail. In particular, the following objectives have

been achieved:

6. *Using the methodology established in Chapter 3, initialise a design space for creating new NMIs* - A design space predicated on the aesthetics of Glitch Lich was created.
7. *Use this design space to establish technical requirements for designing new NMIs* - From the design space that was created fifteen technical requirements were generated.
8. *Identify short-comings in previous technologies for accomplishing the technical requirements of the initialised design space* - In particular, short-comings were found in the state-of-the-art for musical networking software, as well as a distinct lack of well-established software-frameworks for the creation of NMIs.
9. *Create new tools, NMIs, compositions, and performances with the established methodology and initialised design space, taking into account the shortcoming of established technologies, and overcoming them by creating new technologies where necessary* - Guided by the fifteen technical requirements, and the short-comings found in current software, seven new NMIs, one software-framework, and one networking tool were created during the course of the research.

5 Analysis of Work

This chapter details aim #3 of this research: ”Determine the effectiveness of these newly created, or refined, tools and techniques”. To achieve this aim, several objectives (numbered ten through twelve in accordance with their order of appearance in the Introduction Chapter) have been fulfilled:

10. Quantitatively study the effectiveness of new tools created to overcome shortcomings of previous technologies for usage by NMIs.
11. Analyse a live performance of several NMIs, examining the quantitative and qualitative effectiveness of the techniques established in the research.
12. Use the taxonomy and analysis tools deployed in the survey to dissect the new NMIs designed in this research.

5.1 Divergence Test of OSCthulhu

Chapter 4 established the short-comings of current musical networking software and detailed the creation of a new tool to overcome this, OSCthulhu. In particular, OSCthulhu was created specifically to mitigate issues of divergence in network music performances. A test was conducted to demonstrate this effect. This test consisted of two nodes, one in London, England and the other in Boulder, Colorado, both using standard consumer level broad band networks, sending messages to each other. Standard broadband was chosen for this experiment as OSCthulhu has been designed specifically to facilitate network music performance in real world environments outside the confines of academic institutions with access to research networks. The results gathered in this experiment may differ on these academic research networks and future experiments are planned to investigate the differences this makes.

These two nodes created and altered various data sets on their own systems, while simultaneously sending messages to each other to coordinate those same changes on the other node. There were three different actions a node could make: create an array(with a random number of indices, each containing a random value), alter an index of an array, or delete an array. These actions were chosen randomly, with index alterations occurring twenty times more often than creating or removing an array, to reflect real world scenarios. The test was conducted with four different send rates at which changes would occur and messages would be sent: every 250, 100, 25, and 12.5 milliseconds. These messages were sent over a period of two minutes, using either OscGroups or OSCthulhu on subsequent run-throughs for comparison.

A value labelled as *divergence* was collected every 10 milliseconds for each run-through. This test defined divergence as a measurement of the difference between the two node’s states at any given moment in time. For example, if at

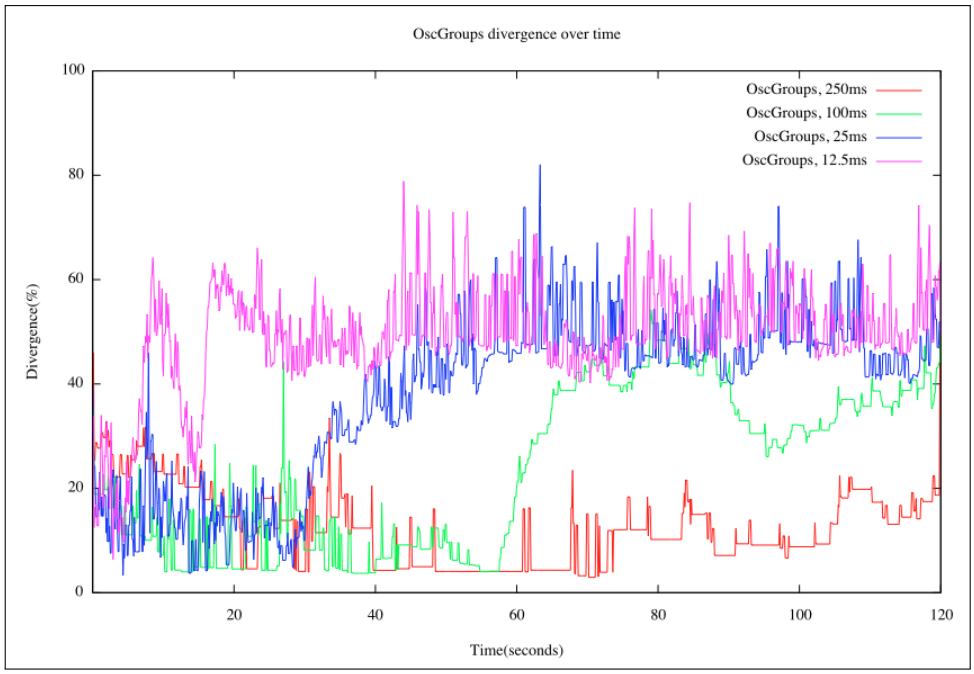

Figure 27: OscGroups divergence over time.

a given moment node one contained four arrays, and node two contained five arrays, but four of those arrays were identical to those contained in node 1, then the systems would be considered 20% divergent. Figure 27 shows the results produced by OscGroups.

5.1.1 Results

The results show a staggering amount of divergence, with the systems immediately beginning at approximately 20% divergence, and becoming more divergent over time, settling at approximately 50%. This divergence can be accounted for by packet loss, lag time, and the cascading nature of divergence(i.e. if an array is missing on one node, the other node is not aware of this and will continue to attempt to set values in it. They will not realign until the second node serendipitously removes the array). Glitch Lich has personally encountered this divergence in performance, wherein a member at one node is creating sounds with a certain unit generator they have created, but the other nodes do not contain this unit generator, therefore the first node's performance is effectively non-existent.

Figure 28 shows the results for OSCthulhu. The results show a stark difference as the amount of divergence is predominantly zero, with spikes up to 5-10%. There are two main reasons for this large difference in divergence between the two systems. Firstly, the effects of packet loss are drastically minimised, as the GCSM server synchronisation cycle ensures that every cycle period (1000 milliseconds used for this test) the two nodes locked back in step (unless the synchronisation packet itself is lost, which does happen on occasion). This prevents the cascading effects of divergence from taking hold, so differences do not pile upon each other

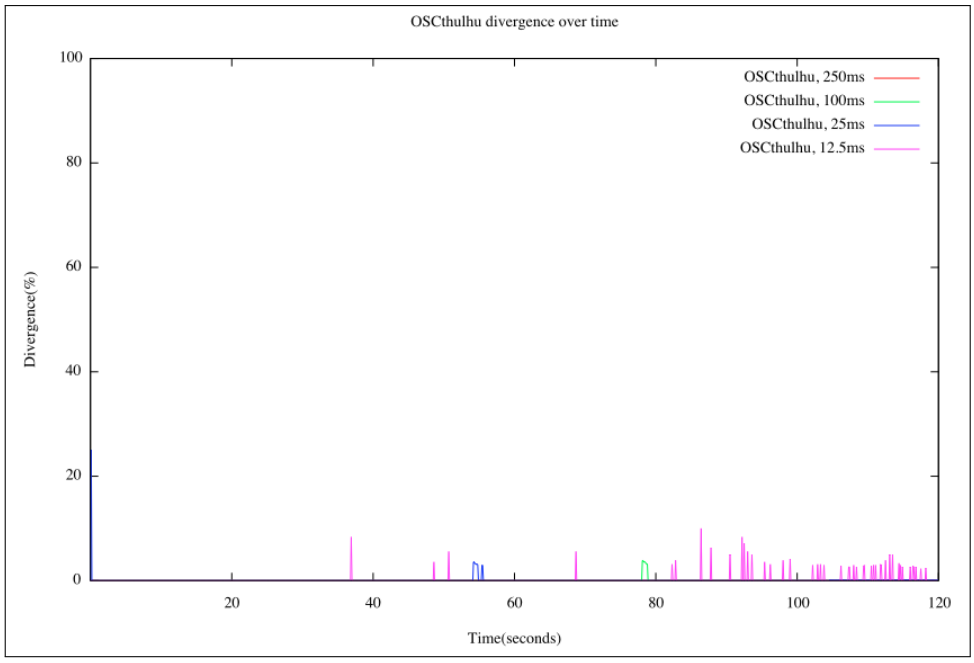

Figure 28: OSCthulhu divergence over time.

over time. Secondly, differences due to lag time are also minimised, as all the actions are first sent to the central server which then simultaneously broadcasts the effects to both nodes. These nodes then receive the message and act upon it in a very similar time scale.

5.1.2 Benefits of Convergence

One manner in which networked ensembles may take advantage of this capability is through the usage of what may be deemed *Remotely Rendered Synthesis*. In many network music bands, including most of the work conducted by The Hub, and PLOrk, network messages are transmitted among multiple participants to influence each other's behaviour. Each member then uses their our computer to output their own sounds. In comparison, in a Remotely Rendered Synthesis configuration, each of the participants share their sound synthesis descriptors with each other member beforehand (in the case of Glitch Lich, SuperCollider *SynthDefs* are used); then, a *Sound State* is constructed that is mirrored on each participants own computer. This Sound State is similar to a Game State in Sweeny's GCSM, except that the data being synchronised represents the state of a sonic world instead of a virtual game world. OSCthulhu keeps track of the Sound State present on each user's computer. Whenever a member makes a change to their particular version of the Sound State, this change is replicated on the server, and shared with the whole group. Then, each member's computer outputs audio that contains the full sound present in the piece, including audio that is being produced by other members. This is useful for network music performances wherein all the members are not geographically co-located.

Another benefit of OSCthulhu is that it does not require NAT traversal or UDP hole-punching, due to the multicasted traffic being forwarded through a centralised server. Hole-punching techniques, although mostly successful, have been shown to be ineffective in as much as 20% of routers in general use (Ford et al., 2005). Since OSCthulhu utilises a more traditional client-server model, firewalls and routers recognise the traffic passing through as legitimate outgoing and incoming traffic, similar to that you would see from any web based server.

5.2 Anatomy of a Performance

A series of analyses of a Glitch Lich performance have been conducted to quantitatively examine the tools and techniques for network based multi-user instruments that have been developed during the course of this research. This performance occurred on April 17th, 2013, at a launch party for Mute Magazine's spring issue, entitled *Slave To the Algorithm* (Slater, 2013). This performance made use of the latest version of the OSCthulhu networking system (the latest feature being network logging), the Azathoth network music engine, and the NMI Simulacra. A video screen capture of this performance, may be found on the included data CD.

This was an informal concert, staged in a dingy warehouse with an old network router of questionable quality. The audience attending the concert were largely non-academic made up of individuals who most likely have never been exposed to a "Network Music" concert. The projector was shoddy and a bit blurry, there weren't any power outlets available near the performance area, there were no sound checks, and the imbibing of beverages occurred. From this author's point of view, this made it both an aesthetically desirable gig to play, as well as a good real-world test run for the technical capabilities of the technology to produce an intercontinental network music performance in a technically lacking environment outside of the safe haven of academic concert halls with gigabit research networks. Unfortunately due to scheduling conflicts Ben O'Brien was unable to join this performance. The author and Chad McKinney performed on site, while member Cole Ingraham performed remotely from Boulder, Colorado.

A certain amount of lip service in this dissertation has been paid to the "democratic" and "egalitarian" nature of network based multi-user instruments. In an attempt to quantify and in some manner analyse this aspect of the technology the network data has been logged and examined from the Mute Magazine performance. This network data and the code used to analyse them may both be found on the included data CD. This network data includes all of the control signals produced and shared throughout the performance. The network and compositional infrastructure of *Simulacra* has been discussed in Section 4.10.2. To recap, the essential capabilities of each performer is the ability to add a synthesis engine, to remove a synthesis engine, and to control that synth via modulating control

parameters.

Of particular note in this infrastructures is the presence of a system for implicit group control of synthesis engines during the course of the piece. Each synthesis engine has two control signals that influence its sound behaviour. One of these control signals is reserved for the individual that spawned it. The second control signal is chained to the primary control signal of another synthesis engine that was previously spawned, chosen at random, which may in fact be controlled by a different performer. This creates a situation whereby a performer musically interacting with their own sounds, would automatically be thrown into a musical relationship with another performer. The thought behind this was to create a kind of socialised performance engagement between performers. However, an "out" of sorts was allowed into the system, in that sometimes the synthesis engine would be chained to another synth that was created by the same performer, thus giving them sole control of the sound (though now two of their sounds would now be interlocked). This would give the performers opportunities to break away from the ensemble and establish their own identities. This infrastructure would theoretically offer the performers a good mixture of interconnectivity and self-establishment, hopefully leading to an egalitarian performance practice (though not really a democratic one, instead it may be thought of as socialised instead).

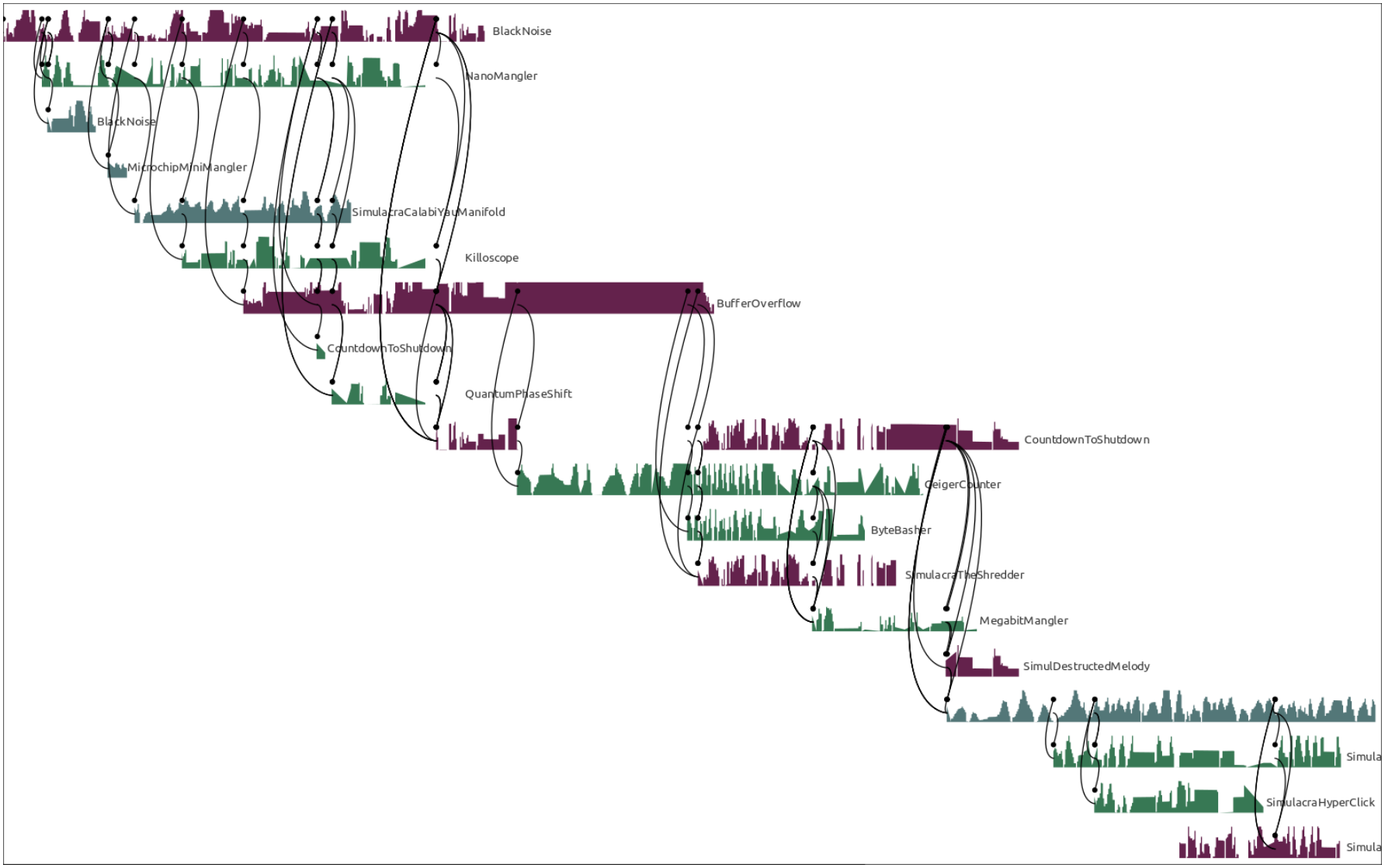

Figure 29: Timeline of a live performance of *Simulacra* in London, England.

The complete network data for this performance has been charted in Figure 29. The differently coloured horizontal bars represent the synthesis engines as they were added and removed over time. Their colour is determined by the performer who created them, purple for Curtis McKinney, green for Chad McKinney, and blue for Cole Ingraham. Their vertical height is determined by their order of entry, with the highest being first, and lowest last. The fluctuating peaks and valleys in these bars is determined by their primary control signal. The black curved lines represent the chained connectivity between synthesis engines. These are the points of forced interactivity in the ensemble. When two synths are chained together any parametric alteration of one of the synths alters them both. This can occur to two synths controlled by a single performer, or to two synths controlled by two different performers. Furthermore the chains form a stream of control. For example, synth A may be chained to synth B which is chained to synth C which is chained to synth D, etc. To the right of each bar is the name of each synthesis engine used. This may be cross-referenced with the video of the performance and the source code included with the data CD for deeper insight into the sound and construction of each synthesis engine. At first glance this seems fairly balanced, with no synthesis engine hanging around too long, and a healthy distribution of synths spawned by the three performers, and their subsequent interlocking. Also, it is clear that connectivity was well established throughout the performance, with no performer disconnections, and a steady stream of modulation signals. However, other views of this data paint a somewhat different picture of performer interactions.

Figure 30 shows the total control signal chaining between performers over the course of the piece viewed from the lens of player interconnection. This might be thought of as a "connectome" (akin to the interconnections of neurons in the brain) of the ensemble over the course of the performance (Hagmann, 2005). In this chart, the coloured circles represent the performers, and the colour curved lines represent the control signal interconnections (Note: during performance each player has a chat alias that they use, Curtis McKinney is casiosk1, Chad McKinney is octopian, and Cole Ingraham is 55hz). A curved line from one performer to another indicates that they established a chained control interaction. Control signal chaining that occurs between the same user's synths is represented by curved line that loops back to the same circle. The horizontal orientation of the lines in relation to their origin circle indicates when that connections occurred temporally during the performance.

From this view certain things become apparent. The system was successful in establishing connections between different users, and throughout the course of the piece all performers interacted with each other. Indeed, cross referencing Figures 29 and 30, establishes that these connections always performed a loop in the ensemble, and that many different configurations of interconnections

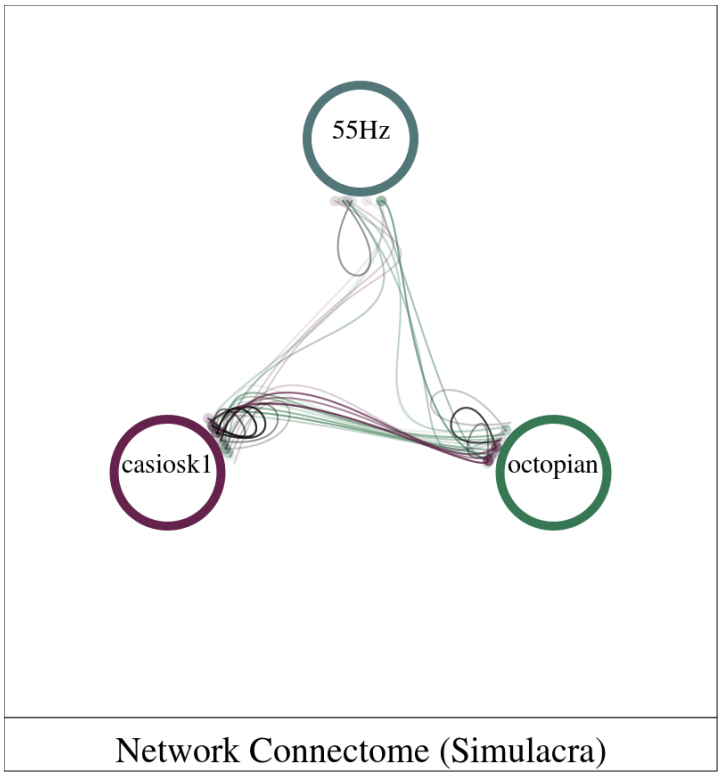

Figure 30: User connectome in *Simulacra*.

occurred, ranging from single performers connected to themselves, to performers exclusively connected to each other, with the third performer connected to themselves, and a full interconnection between the ensemble. This variety of performer relations was an established goal and network data confirms that these various interconnectivities occurred.

However, it is also apparent that despite the socialised connection system established, members Curtis McKinney and Chad McKinney were more active overall, and tended to connect to each other more than to Cole. How might two performers connect two each other more often if they have no actual choice in whom they connect to? Figure 31 shows the percentage of connectivity participation for each performer, divided by connections to others (outlined in red) and self-connections (no outline). These percentages confirm the intuitive reading of the previous figure and show that performers Curtis McKinney and Chad McKinney overall participated in more connections than Ingraham. This seems to somewhat conflict with the stated egalitarian goals of the software, and begs the question how it was that Ingraham participated less in these interactions given the presence of a unbiased connectivity engine.

In Figure 32 we see the bandwidth usage of each member of the ensemble, represented by percentage of the total control signal bandwidth used over the course of the performance. Bandwidth usage meaning, the percentage of control messages sent and received, including add, remove, and modulation messages. From this data it becomes obvious that performer Chad McKinney was simply

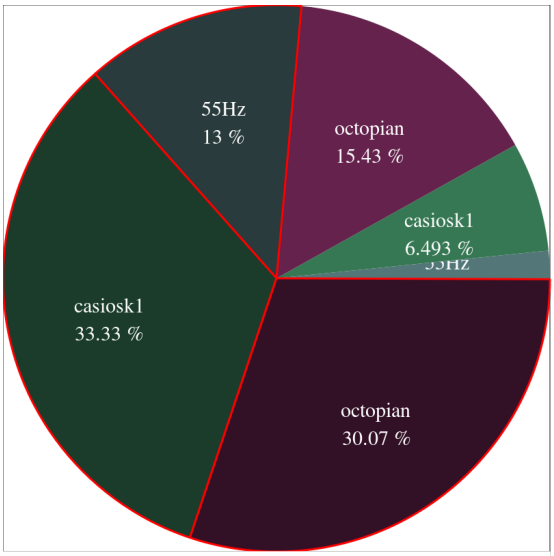

Figure 31: Percentage of connectivity participation in *Simulacra*. Connections to others (red outlines) vs self (no outline).

more active than the rest of the ensemble, and by a large margin as well. Chad McKinney constituted 54.75% of total network bandwidth, more than the other two performers combined. Ingraham's total activity only accounted for 11.46% of the total activity in the piece, with Curtis McKinney accounting for 33.78%. Given this, it is actually supportive of the egalitarian and socialised goals of the technology that both Cole Ingraham's and Curtis McKinney's share of interconnectivity (14.73% and 39.82%, respectively) were higher than their bandwidth usage percentages, perhaps showing an impact of the socialised interconnection system.

Figure 33 presents the percentage of synth engines added and removed by each performer during the performance. Like the previous graph, this also reveals an imbalance in the power structure of the ensemble during the performance. From this graph it is obvious that performer Curtis McKinney created and destroyed the majority of the synths during performance. Given this imbalance in synth engine real-estate, it is once again impressing that the interconnectivities metric showed more balance than the imbalance of either this metric or the metric of total control signal bandwidth. This seems to hint that the socialised interconnection system in place played a role in balancing out ensemble control imbalances.

The analysis of the network data has been useful in examining how the technology has fared against some of the established aesthetic demands and their resultant design specifications. Dislocative intercontinental network music performance was made possible with no detectable user disconnections or interruption. This was maintained in the face of wanting technical facilities, satisfying the ensembles desire to perform in more informal musical settings. Interperformer

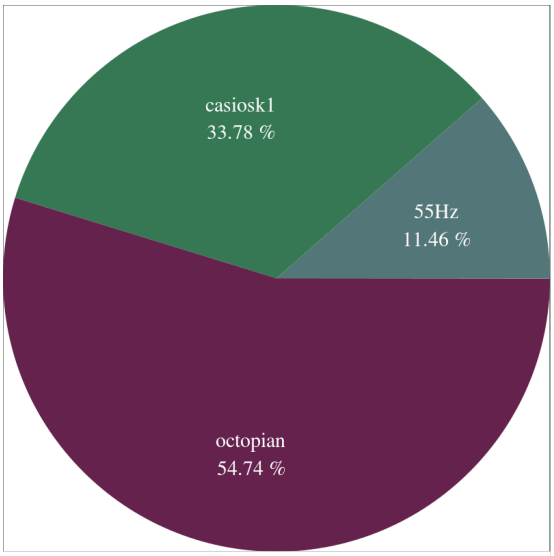

Figure 32: Percentage of bandwidth usage in *Simulacra*

interaction was not only achieved but guaranteed, even despite individual performers playing in a more selfish manner. And despite several in balances of ensemble control, a system was in place to enforce a more egalitarian and socialised musical control structure. However, only so much may be gained from looking at network data. To analyse how the technology satisfies the other design goals, another means of analysis is required.

One manner of analysing the performance is to examine the audio produced during the course of the performance, and to cross reference these analyses with the network data produced. This may help to elucidate more about how the developed NMI technology satisfies or dissatisfies the established design goals. Collecting and analysing musical information in this is often referred to as Music Information Retrieval (MIR).

Four different audio signals were gathered from the Mute Magazine performance. A discrete stereo audio recording was gathered for each of the performers, as well as the sum total audio output of the entire performance. These audio signals were each subjugated to several different music information retrieval processes. These different data sets were then cross-referenced to form a composite analysis. All of these metrics were gathered via transforming the audio using a standard Fast Fourier Transform (FFT) process, in which the audio is transformed from a time-domain to a frequency-domain (Brigham, 1973). These analyses were generated using the SuperCollider programming language, using the SCMIR quark created by Nick Collins (Collins, 2011). As with the previous network analysis, the source code for this analysis may be found with the attached data CD. The goal of all of these analyses is to determine the presence and amount of interactivity versus self-determinism in the ensemble during performance, a key design goal of the NMI technology techniques developed.

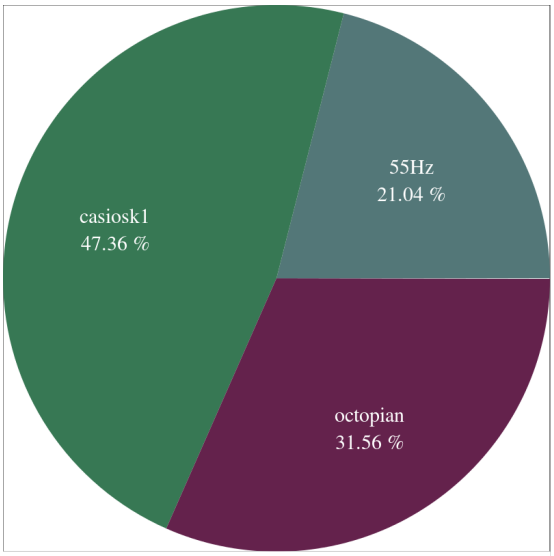

Figure 33: Object creation and destruction activity in *Simulacra*

The first metric gathered is perceptual "loudness". This loudness is not based on the objective amplitude of the signal, but instead intends to reflect the perception of how loud a sound is to a person. This is attained by mapping several frequency bands from the signal onto an Equivalent Rectangular Bandwidth (ERB) scale. This ERB scale defines the shape of frequency loudness perception as an "auditory filter", that may be thought of as filtering the objective amplitude power of the frequency spectrum (Krishnamoorthi et al., 2008). An approximation of this "auditory filter" has been developed in past research by subjecting individuals to tests in which the individual must listen and attempt to detect a certain frequency signal in the midst of the presence of a noise band with a notch-filter placed at the detection signal's frequency (Moore and Glasbers, 1995). By incrementally changing the amplitude of the detection signal until the point that the subject is able to perceive it amongst the noise, the researchers were able to create an approximate "auditory filter" that the human ear and mind places on incoming sounds.

The second metric gathered is a set of Mel-Frequency Cepstrum Coefficients (MFCC). A cepstrum is the Inverse Fast-Fourier Transform (IFFT) of the logarithm of the spectrum of a signal (gathered using a normal FFT process) (Roads, 1996). Thus, a cepstrum is a spectrum analysis of a spectrum analysis. However in a MFCC analysis, the signal's spectrum is first mapped to the Mel scale, which is a scale developed that attempts to map the frequency scale, measured in hertz, to a scale that reflects how the human ear perceives tones of equal distances (Stevens et al., 1937). This MFCC analysis is useful in comparing timbral characteristic of audio signals.

The third metric gathered is the spectral centroid of the audio signal. The spectral centroid is the weighted mean of a given audio signal's FFT spectrum.

This metric may be used to determine the perceptual "darkness" or "brightness" of a given audio signal's timbre (Grey and Gordon, 1978). The fourth metric gathered is spectral flatness. This is derived by dividing a given spectrum's geometric mean (the mean of the product of a data set as opposed to the mean of the sum of that data set) by it's arithmetic mean. This flatness metric ranges from completely flat (white noise), to completely sharp (a sinusoid).

The fifth metric gathered is spectral roll-off. This is found by determining the frequency at which the cumulative sum of a section of a spectrum occurs below a given percentage of that spectrum's total frequency span (Lerch, 2012). For these analyses two measures of spectral roll-off were gathered, one calculated at 95% and one calculated at 80% of the spectrum frequency.

The final metric collected is spectral crest, which may be thought of as the "peakiness" of an audio signal's spectrum. This spectral crest is derived by first creating a list of the squared magnitudes from an FFT spectrum. Then the highest value from these squared magnitudes is divided by the mean of the squared magnitudes. A spectral crest measurement was taken for three different spectral bands, one for the entire spectrum up to the Nyquist frequency (half of the current sampling rate), one for frequencies ranging from 0 to 2000 hertz, and one for frequencies from 2000 to 10000 hertz (Blackledge, 2006).

After these data sets were collected they were normalised and combined to create a composite analysis for each the four audio signals recorded from the performance. Lastly, several self-similarity matrices were generated for this conglomerated composite analysis of the six metrics just described. A similarity matrix is a manner of looking at the similarities of two sets of data. The similarity of any one point in a data set as compared to any point in the second data set is visually displayed as colour value in a cell of a two-dimensional grid. The value at the lower left-hand corner represents the similarity between the first point in the first data set with the first point in the second data set. Travelling up the grid from that point compares that value from the first data set to each subsequent value on the second data set. Travelling right from the original left-hand corner position compares the first value of the second data set with each subsequent value in the first data set. IF you travel diagonally from the lower-left hand corner up to the upper-right hand corner you sequentially compare the similarity of each value of the two data sets.

A self-similarity matrix uses the same comparison system as a similarity matrix, however it displays the similarities between a single data set plotted in a series against itself. Thus each point in the given data set is compared for similarity to each other point in the data set. This is useful for detecting patterns and boundaries in data sets. For musical analysis, than can be used to attempt to detect patterns, section changes, and comparisons of musical material. The self-similarity matrices used in these analyses plot the similarity of the composite

music information retrieval analyses of the Mute Magazine performance's audio signals as a function of time. The values range from black to blue to green to red, with black being completely dissimilar to red being completely similar. Furthermore, self-similarity matrices have been generated for the raw network control data streams for each of the three members. This network data will be cross-referenced with the composite MIR analyses. A comparison of self-similarity matrices was chosen as a method of analysing the performance in hopes of gleaning if the interactions found in the network data would manifest in the audio data of the performance as well. It is not enough that parametric changes occur in an interactive manner along the networked ensemble, it must in fact manifest audibly. By comparing the structures present in the self-similarity matrices it may be deduced if changes in the structure of one self-similarity matrix propagate out to changes in the structure of the other self-similarity. The presence of these points of mutual change would point to there being interactions in the audio that correspond to interactions in the network data.

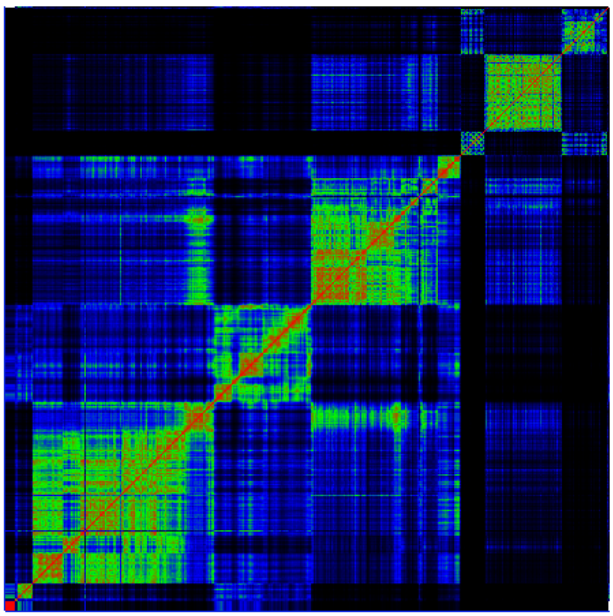

Figure 34: Self-similarity matrix of a live performance of *Simulacra*

Lastly, novelty curves are generated for each of the audio and control data sets. A novelty curve is a curve which plots the amount of change from cell to cell that occurs in a data set. The novelty curves generated here are calculated by scanning the data sets for the cells with the highest peaks of change, removing those peaks which occur within the sampling size from the beginning or end (Foote, 2000). The novelty curves for the control data were determined by looking at the parametric changes that occurred in the musical subsystems over time. Each new synth that was created was deemed to have introduced an amount of change into the system relative to the number of other synths already running. In

other words, if there is one synth running and one new synth is added a change of 100% is recorded. If two synths are playing and a third is introduced a change of 50% is recorded. Parametric changes are also relative. Each synth has a parameter associated with it that may change from 0-100%. If there are 5 synths playing and a synth's parameter is changed from 0% to 50%, then a 10% change is recorded. Furthermore there is also some filtering of this data to clump closely occurring peaks and to set a peak thresholds limit. Thus, there is some art in the science of the novelty curve results. For MIR data, this is useful for detecting section boundaries and moments of change or significance.

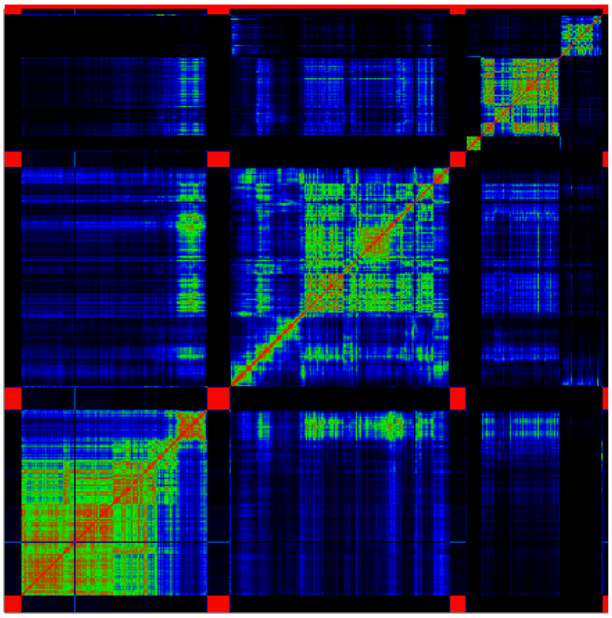

Figure 35: Self-similarity matrix for casiosk1's audio signal in *Simulacra*

The first portion of this analysis will examine of the of the self-similarity matrices and novelty curves generated by each performer in turn, and then a comparison between this data will be conducted. In Figure 34 is the self-similarity matrix of the composite analysis of the full-audio signal. Some structural properties of the piece may be noted. There is a short introduction followed by four distinct sections in the music, something which the author's own opinion of the performance of the composition corroborates. Furthermore, it can be deduced that the end of the first section shares some material with the third. As well the final section is further divided into three subsections, with the first and the last sharing musical material. Henceforth these shall be referred to as sections one through four.

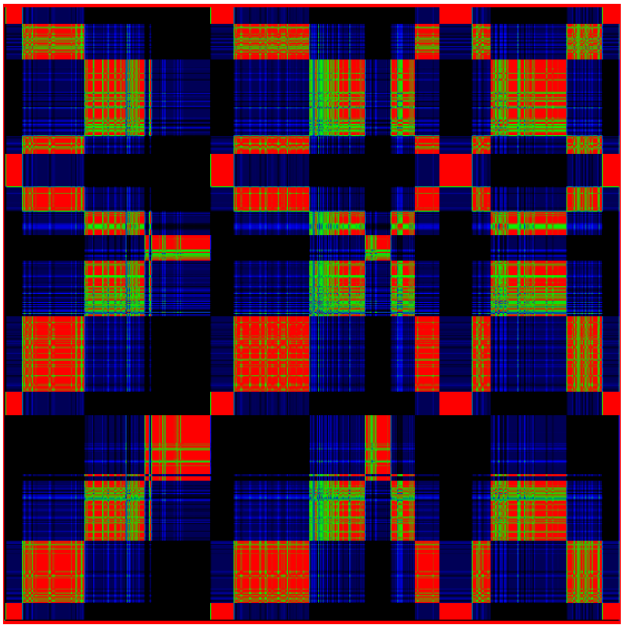

Figure 36: Self-similarity matrix for casiosk1's control signal in *Simulacra*

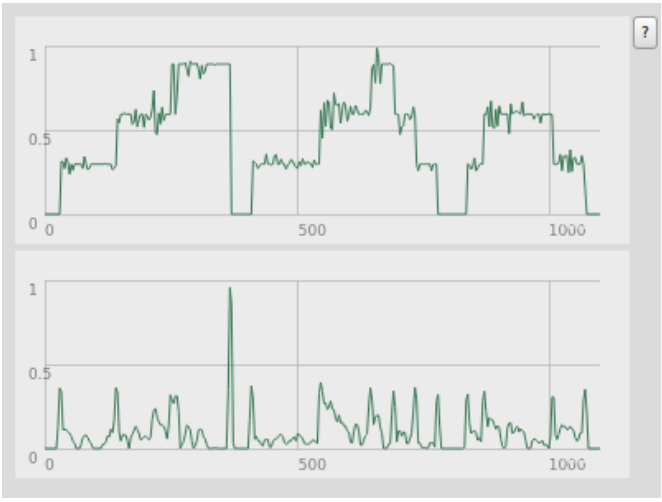

Figure 37: Comparison of casiosk1's raw control signal (top), and it's computed novelty(bottom) over time in *Simulacra*

Figure 35 displays the self-similarity matrix for Curtis McKinney's audio stream. It has a very similar structural make up as the full audio signal, however it displays more commonalities throughout the second and third sections. As well, there are several points of silence, which can be seen by the presence of the completely red blocks in between sections. In Figure 36 we see the self-similarity matrix for Curtis McKinney's control stream. Predictably, this shares the same structure with Curtis McKinney's audio signal, however the control signal reveals more deeply divided sections which are more drastically delineated. In Figure 37 we see Curtis McKinney's raw control signal data (summed and plotted to a

one-dimensional plane), and its computed novelty curve. This data once again reinforces the same structure as found in the previous matrices.

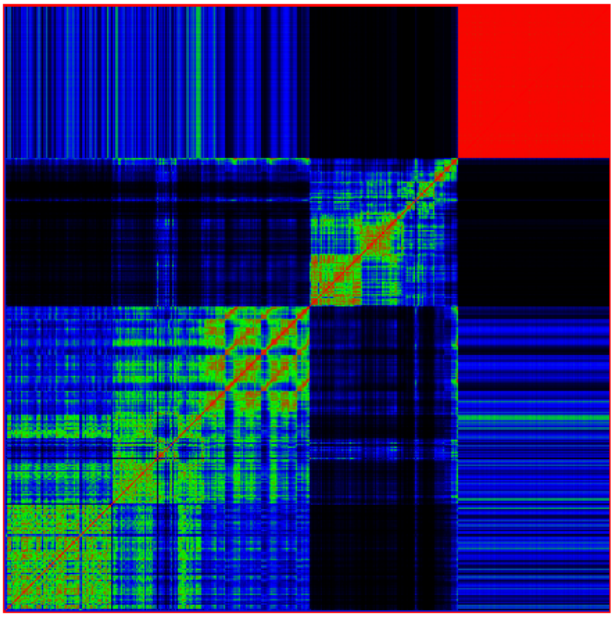

Figure 38: Self-similarity matrix for octopian's audio signal in *Simulacra*

Chad McKinney's audio signal, as seen in Figure 38, displays a similar structure to the previous matrices, however there is much more similarity between the first and second sections. Upon examining the third section of Chad McKinney's audio and comparing it to the third section from Curtis McKinney's audio, one may observe that a very similar pattern emerges in both of their matrices. Upon reviewing the network data, it can be concluded that at this point in the piece a certain synth was instantiated (by Chad McKinney) which had the effect of co-processing the two-performer's audio together.

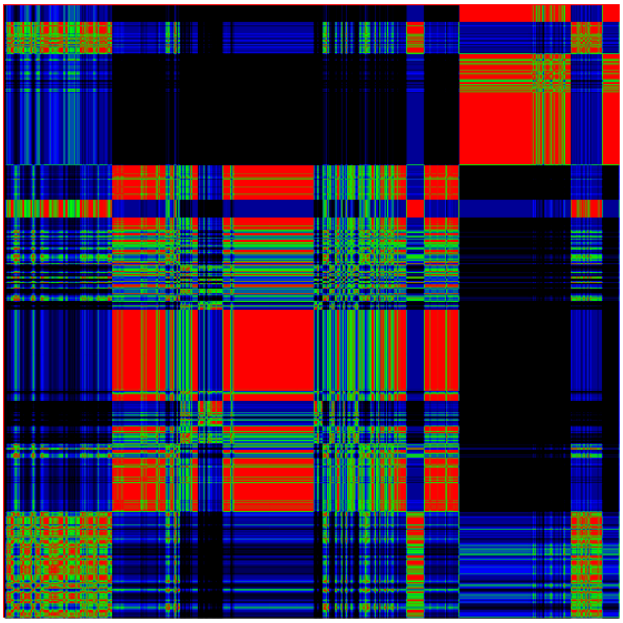

Figure 39: Self-similarity matrix for octopian's control signal in *Simulacra*

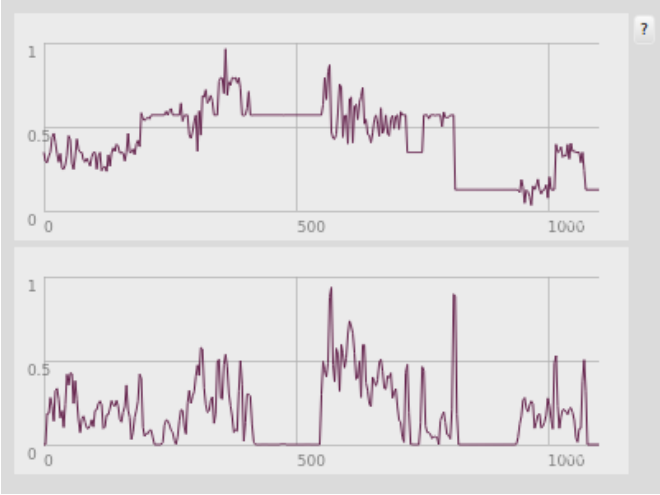

Figure 40: Comparison of octopian's raw control signal (top), and it's computed novelty(bottom) over time in *Simulacra*

Furthermore, it is clear that Chad McKinney's audio signal outputs silence in the final section of the piece, indicated by the large red block at the end. This is due to the fact that the audio feature that shares the most similarity with itself is in fact silence. The large section of absolute similarity indicates that no audio was being output on this channel. Notably however, when reviewing Chad McKinney's network self-similarity matrix, shown in Figure 38, Chad McKinney was still generating audio. How might this be possible? The answer is once again related to a specific synth instantiation, this time by Curtis McKinney, which had the effect of inputting chained signals from the other members and processing it, only this time this audio processing was not output to all of the channels, but

instead to Curtis McKinney's channel only. Figure 35 , displaying Chad McKinney's summed control signal and its novelty curve further reinforces the presence of network data during this final section.

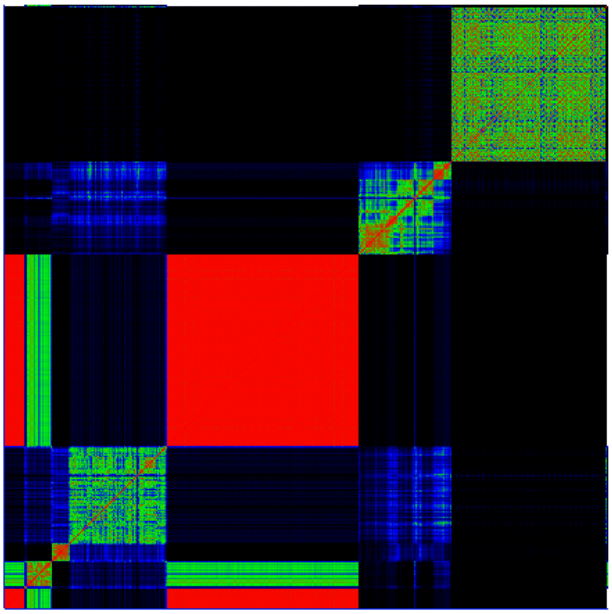

Figure 41: Self-similarity matrix for 55hz's audio signal in *Simulacra*

The self-similarity matrix for Cole Ingraham's audio signal may be seen in Figure 41. Like the previous matrices, this displays the same structure found throughout the performance. However, it is clear that Ingraham rested during the second section of the piece. The fourth section exhibits a very clear pattern signature indicating that cyclic material was being produced in this section. The third section displays the very same patterned signature as displayed by the previous matrices' third sections, thus Ingraham was also the beneficiary of the audio-coprocessing done via Chad McKinney's synth instantiation in that section.

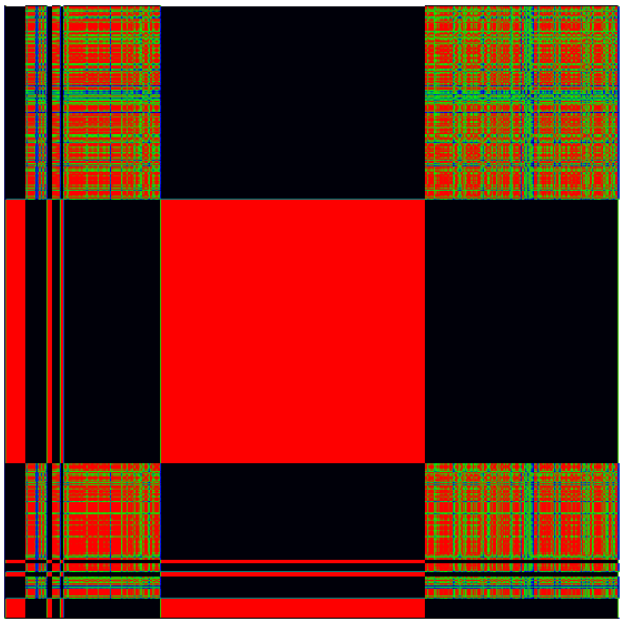

Figure 42: Self-similarity matrix for 55hz's control signal in *Simulacra*

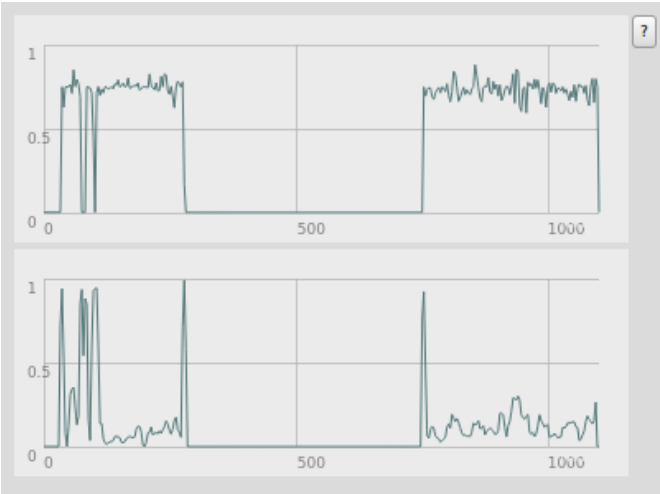

Figure 43: Comparison of 55hz's raw control signal (top), and it's computed novelty(bottom) over time in *Simulacra*

However, when reviewing Ingraham's control self-similarity matrix, shown in Figure 42, it is clear that Ingraham was in fact not creating network data during this section. So how is it possible the Ingraham produces sound, but no network data? The answer is that even in the absence of Ingraham's participation in the ensemble, the ensemble itself may still utilises him (or rather, the synthesis engines he creates and controls) without his assistance. Ingraham's raw control data, displayed in Figure 43 and its computed novelty curve confirm this lack of personal activity on Ingraham's behalf during the third section.

To compliment these self-similarity matrices, several comparison matrices were made that compared the novelty curves of the control signals of each of the

four performers against the novelty curves of each of the performers audio signals (these matrices may be found in Appendix A). This analysis was conducted as an attempt to analyse interconnections in the ensemble at a small level (i.e. the level of parameter changes as opposed to sections). Furthermore, the same novelty curves were compared to pseudo-randomly generated noise, to account for how random coincidental data would compare. In summary, the results revealed two interesting points. The first is that the interconnections between parameters and the various audio signals were more significant than noise, with noise computing to an approximate 50% similarity as a mean of the given data set. The similarity curves also displayed a distinctly noisy similarity curve over time. The control signals on the hand provided a much less noisy curve. The second item of interest that was revealed was that small parameter changes, even in large quantities, did not have any substantial similarity to the novelty of the audio, only exhibiting approximately 10%-20% similarity between signals, even when comparing a performer's own control signal to their own audio signal.

However, this was extrapolated further, by comparing peaks of novelty in the control signals and audio signals of each of the performers. If a peak in a novelty curve is shown at the same moment as a peak in the novelty curve of an audio signal being compared (or close to, as some synth sounds fade in once instantiated, for instance; however these lags in reaction may only occur after a peak in the control signal), it is considered to be an ensemble interaction. To filter for some coincidental activity, if a given performer's own control signal displays a peak at the same moment that his own audio signal registers a peak in it's novelty curve, then no other performer may be considered to have interacted with that peak, as it is assumed that the original performer is the actual progenitor of the peak in novelty.

Three charts depicting the highest of these peaks may be seen at the end of Appendix A . Comparing novelty signals in this way improves the similarity between the control signals and audio signals. For Curtis McKinney's audio signal, 60.00% of the novelty peaks in the audio are accounted for with peaks in the novelty curves of the control signals of one of the three performers, while 42.1% of Cole Ingraham's audio signal peaks, and 66.6%o of Chad McKinney's audio signal novelty peaks are accounted for. Outside of the direct impact from control signals, the rest of the peaks detected in the novelty of the audio signals are most likely derived from the various algorithmic constructs that are programmed into each of the synthesis engines, or from emergent behaviours associated with introducing interconnected feedback loops amongst the ensemble.

Table 4 displays this novelty curve interaction percentage data. The table also breaks down the interaction percentages of the audio signals for each of the control signals. It is logical that both Chad McKinney and Curtis McKinney's audio signals derive the majority of their interactions from their own control signals.

	Curtis Audio	Chad Audio	Cole Audio
Total accounted for	60.0%	66.6%	42.1%
Curtis Control	50.0%	33.3%	33.3%
Chad Control	33.3%	50%	44.4%
Cole Control	16.6%	16.6%	22.2%

Table 4: Percentage of interactions occurring between control signals and audio signals during the Mute Magazine performance of *Simulacra*.

However, it interesting to see that Ingraham's audio interactions do not derive mainly from his own control signal. Instead, his control signal accounts for the least interactions of the three control signals.

Two sets of logs have been collected from two recent Glitch Lich performances, in order to do an informal assessment of how the NMI technology and techniques satisfies the design and aesthetic goals that have been set out . The first is a log of tweets that were sent during Glitch Lich's performance at College Station, Texas, while in residency at Texas A & M. For this performance audience members were invited to create tweets (short 140 character messages sent using the social media website Twitter) with the hashtag (a word used by twitter to collate tweets of similar content) #glitchlich (Twitter, 2013). A program was created that responded to any tweets with the hashtag #glitchlich, and would display these tweets in real-time during the glitch lich performance. In this way the audience was able to communicate with the band on stage and with each other. A record of these tweets may be found in Appendix C.

The second set of logs is the band's inner chat logs stored from the Mute Magazine performance (unfortunately the chat logs for the Texas A & M performance were not recorded). These chats were shared with the audience live by projecting them over the accompanying visuals for the performance. This is a long-standing technique utilised by network bands, dating back to The Hub performances from the 1980s (though they have since abandoned its usage), which is useful for letting audience members into the inner-workings of the ensemble (Brown and Bishcoff, 2002). A record of the chat logs may be found in Appendix D.

The data for both of these logs is informal and small, thus there will not be an attempt to do some kind of formalised statistical linguistic analysis or qualitative sentiment analysis. However, it is still of some interest to note some social interactions that occur in both of these. The fostering of a rather informal performance atmosphere is clear, with the tweets and chats containing humorous messages, onomatopoeic mimicking of the music, spelling and grammatical errors, and references to Internet memes. Some of these messages spurned ongoing themes throughout the performance. For example, one audience member wrote:

Mason Morgan @Ramblings_Of: It tastes like a laser! #glitchlich

This was shortly followed by more messages with references to lasers:

> **Taylor Phillips** @tayphil8992: Frickin' laser beams #glitchlich
> **Kevin Hernandez** @Kevhernandez: #glitchlich ima FIRIN' MAH LAZARRRRR!!!!!
> **Clara Turk** @lifeonourtongue: In case anyone's wondering #glitchlich is the friendly neighborhood Spider-Man but transplanted into a shark's body with lasers for eyes!
> **glitch lich** @glitchlich: Thank you Texas people and lasers and things, it was definitely worth losing a Saved by the Bell lunch box #glitchlich

It is also possible to deduce from the tweets that there was some sense of presence of the individual members of the group for the audience, despite the fact that only one member was physically present (for this performance, Cole Ingraham was the local representative). For instance, for this performance member Ben O'Brien was not able to perform due to schedule conflicts, yet the audience was aware of his absence from the group:

> **crewxp2 @crewxp2**: Where is your fourth member? #glitchlich
> **Nicky McMurrer** @Nicksta_: #glitchlich Have y'all ever all performed together? Would that effect connect probs or clog up the bandwidth of the venue?

The members of the ensemble also freely communicate with each other and directly to the audience, often with a similar humorous style. For instance, while talking about what direction to take the music next, the following correspondence occurred:

> **casiosk1**: I think I'll mangle it up a bit
> **casiosk1**: fFSEFSpfs8efsf sEFfsefSefF2324@34243@3@3424@32
> **octopian**: asl;ckas;lckascl;kasc;l
> **casiosk1**: 34234234@44444334!!1dssdds
> **55hz**: hah

The chat capabilities of course also provides a means for the ensemble to conduct logistics and ensemble organization during performance. The fact that this is projected for the audience can even let them in on some of the internal drama that occurs during network music performances, which, due to the technical complexities involved, often exhibit some form of technical or user error. An example of this may be seen in the following correspondence in the chat logs, where Cole Ingraham accidentally activated the wrong synth sound during the performance:

> **casiosk1**: Cole throw in that manifold synth
> **octopian**: who wants to bring in the calabi

octopian: ok
casiosk1: that would be the incorrect synth
casiosk1: lol
octopian: fail
octopian: haha
casiosk1: not that
octopian: curtis do it
casiosk1: there we go
55hz: too similar of names...

An interesting piece of anecdotal evidence gathered from this performance came from the concert organizer, who informed the ensemble that this was the first electronic music performance they had encountered where after the concert began the audience members moved themselves, of their own volition, to the front of the concert hall.

5.3 Collaborative Dimension Spaces

Several dimension space charts have been created to help analyse the NMIs created during this research, and to cross reference them with the other multi-user instruments covered in Chapter 2.

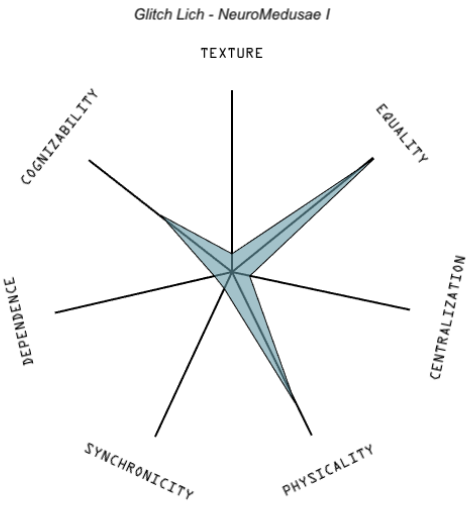

Figure 44: Collaborative dimension spaces for *NeuroMedusae I*

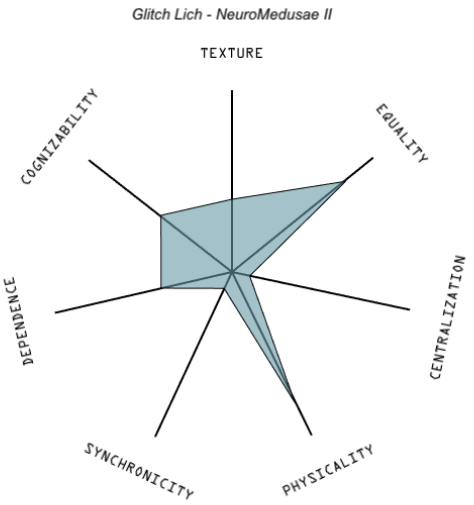

Figure 45: Collaborative dimension spaces for *NeuroMedusae II*

Dimension space charts for *NeuroMedusae* I and II may be found in Figure 44 and 45, respectively. These NMIs share some common characteristics (owing to their construction with the same software). Both pieces contain a large degree of ensemble equality. For both of these NMIs, all of the performers in the ensemble have the exact same capabilities, and may interact freely with each other. There is no overriding structural mechanism that defeats this. They are also both highly centralisation, due to the mandatory requirement for a centralised synchronisation server; they simply would not even function without the centralised synchronisation scheme. The physicality is fairly free, as the instrument is completely virtual, with the only gestural control required coming from mouse-pad controls. Both are also highly synchronised, due to the server, but also due to the fact that the piece is real-time in nature, with no sequential-ordering to events.

The cognisability of of *NeuroMedusae* I and II are middling. They are electronic instruments with no inherent physical nature or gestural control, however there are some primitive visualisations that help instruct the audience about what is occurring. These NMIs however differ somewhat in texture and dependence. *NeuroMedusae I* interweaves the constructs of the ensemble more closely though the use of audio feedback loops, which produces an instrument in which the individual members blend in together, thus making the instrument more homogeneous, as well creating a group dynamic where the ensemble members are more interdependent. *NeuroMedusae I*I on the other has syntheses engines that are less interconnected, with more personal identity and self-reliance (though still more interdependent than not) through the usage of convolution based feedback loops. Thus *NeuroMedusae II* is less homogeneous and more independent than *NeuroMedusae I*.

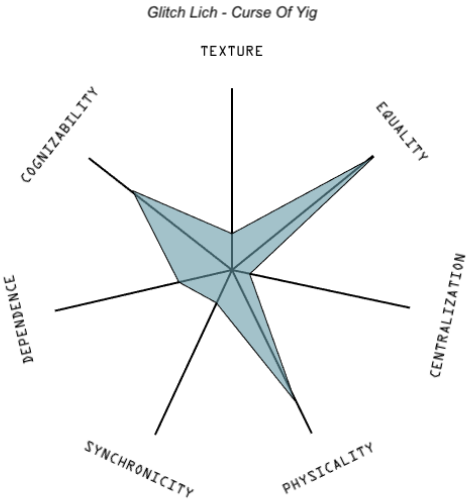

Figure 46: Collaborative dimension spaces for *Curse of Yig*

Curse of Yig has a very similar dimension space mapping as that of *NeuroMedusae* I and II, as it is essentially an evolution of the original Medusa system. However, an evolution in this instrument is the presence of a specifically made system for the production of visuals, as opposed to the primitive GUI-visuals in the Medusa system. Therefore Curse of Yig rates higher in cognisability.

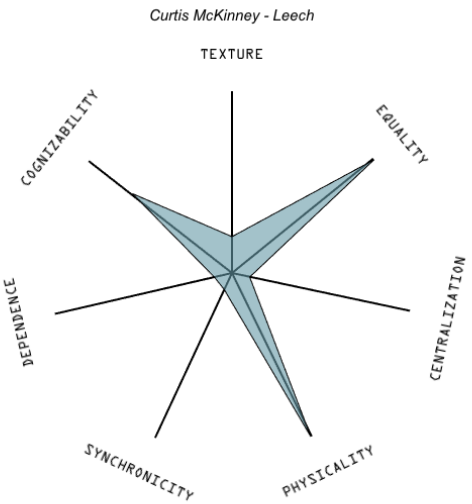

Figure 47: Collaborative dimension spaces for *Leech*

The construction of leech is very different to that of the Medusa or Curse of Yig NMIs. However, the dimension space chart, which may be found in Figure 47, once again offers a similar shape. *Leech* is an NMI with the possibility of hundreds of users, though most of their contributions are incredibly homoge-

neous, creating more of a tapestry than soloistic voices, therefore the texture is considered to be mainly homogeneous. All of the users in *Leech* have the same capabilities (which is essentially only to download and upload data), creating a system that is highly equal. The entire system depends on the single laptop running the Leech system to run, and all traffic is routed through that single laptop, making *Leech* highly centralised. Leech offers no physical embodiment and no actual controls (gestural or otherwise), making it extremely physically free. *Leech* happens all in real-time, and all performers play simultaneously, giving *Leech* a high degree of synchronicity. The users in the system are not only dependent on each other to make music, they are also completely dependent on each other to download the content they wished to download as well. This makes the system interdependent in nature. *Leech* offers a very detailed and explicit visual presentation which intends to create a high degree of cognisability.

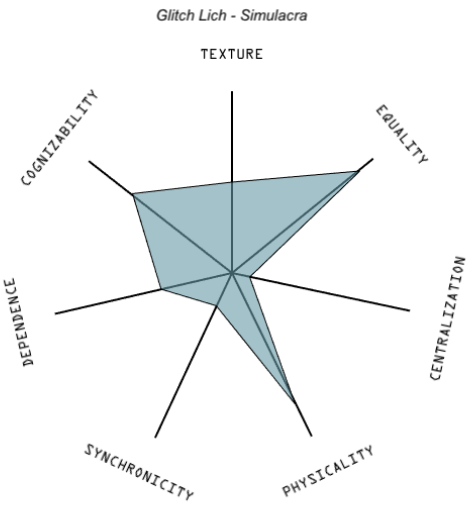

Figure 48: Collaborative dimension spaces for *Simulacra*

Figure 48 shows the networking dimension space chart for *Simulacra*. The piece is characterised by interconnecting both control and audio through a systematic mechanism for random ensemble signal chaining. Also, all of the users of the system have the same basic capabilities (add synth, remove synth, modulate synth). *Simulacra* uses the same OSCthulhu based centralised synchronisation server as the previous pieces, similarly scores highly in synchronisation. The NMI has no physical embodiment and offers only gestural control as far as MIDI knobs could be deemed gestural control, therefore is mainly physically free. While there will always be some lag times in network pieces, the main crux of performing with *Simulacra* is a real-time synchronous performance. Furthermore, the performers are encouraged to play whenever they chose, which makes the system much synchronous than sequential. Performers in *Simulacra* share

responsibility of control for every synthesis engine created. Furthermore, synths are often times chained together in buffer-based feedback loops and audio processing chains. This creates a highly interdependent system. *Simulacra* has only one interface, which is explicitly shown during performance. This leads to higher cognisability. However, it is a highly stylised interface that sometimes obscures what is happening for visual effect.

It is interesting to note that largely these NMIs demonstrate a similar shape in their dimension space. In comparison to other established multi-user instruments covered in Chapter 2, this shape is very similar to those found in the Interconnected Laptop Ensembles group, such as Renaud's *Frequencyliator*, Hajdu's *QuintetNet*, and the JacksOn4's *The Loop*. These seems natural as the NMIs developed during this research could also be identified as being an Interconnected Laptop Ensemble. Systems of this nature characteristically seem to exhibit a high degree of equality, textural homogeneity, and centralisation, as well as a lack of physical embodiment to the instrument. Though, due to their lack of a physical interface and their usage of bleeding-edge technology with no traditional counterpart to key audience members into how they work, these systems could be seen to offer a lower amount of immediate cognisability to the audience. The NMIs created during this research attempt to mitigate this through the usage of several key features, such as anaesthetised GUI's, real-time visuals, projected chat logs, and interactions with audience members through social media.

In this chapter aim #3 of this research, and the objectives fulfilled to reach that aim, have been discussed. This aim was to "Determine the effectiveness of these newly created, or refined, tools and techniques". To achieve this aim, several objectives (numbered ten through twelve in accordance with their order of appearance in the Introduction Chapter) were fulfilled:

10. *Quantitatively study the effectiveness of new tools created to overcome shortcomings of previous technologies for usage by NMIs* - A test was conducted which empirically demonstrated that OSCthulhu far out-performed OSCGroups in terms of mitigating divergence during a musical performance.

11. *Analyse a live performance of several NMIs, examining the quantitative and qualitative effectiveness of the techniques established in the research* - Both quantitative and qualitative analyses were conducted of two different performances using NMIs created in this research. In particular, it has been demonstrated that inter-ensemble influence has been quantitatively established to a satisfiable level. Furthermore, a method for mitigating the problems of network music performance has shown qualitative evidence of effectiveness.

12. *Use the taxonomy and analysis tools deployed in the survey to dissect the new NMIs designed in this research* - Dimension space analyses were generated for five of the NMIs created in this research. These dimension space

analyses were also compared and contrasted with those presented during the survey in Chapter 2.

6 Conclusion

This study set out to investigate the construction of a special breed of musical instruments called Network Based Multi-User Instruments (or NMIs), and the creation of several software tools and compositional and performative techniques to facilitate their construction and usage. The study also sought to improve upon previous techniques utilised by previous designers of NMIs and other electronic music instruments for performance and composition. NMIs are a new breed of musical instrument that, due to their unique capabilities as networked collaborative musical instruments, allow composers and performers to create new musics with techniques that were not possible before. Research into NMIs is fertile ground, as there are manifold possible design choices and technical problems involved in their construction and usage. In the general literature and previous design landscape there are several open questions in regards to how one creates, composes for, and performs with NMIs. The research conducted in this study sought to answer several questions:

1. What is a multi-user instrument, and how is it defined?
2. Are there distinct morphologies of multi-user instruments, and may a taxonomy be created to organise them?
3. Is there a manner in which to examine the collaborative capabilities of a multi-user instrument?
4. Is a methodology for designing NMIs based on the aesthetics of a specific ensemble able to generate NMIs which satisfy the design requirements generated by that ensemble?
5. May the issues of liveness and disembodiment inherent to NMIs be addressed in some manner?
6. Are there better techniques for overcoming the technical difficulties involved with networking geographically displaced ensembles?
7. Is there a way to streamline the development processes of creating NMIs?

The main findings found in this research are chapter specific, and may be found in the following chapters: Chapter 2, A Survey of Multi-User Musical Instruments, Chapter 3, Aesthetically Driven Iterative Design Methodology, Chapter 4, Design, Development, and Composition, and Chapter 5, Analysis of Work.

6.1 Empirical Findings

Here each research question is individually reviewed, followed by relevant conclusions.

6.1.1 What is a multi-user instrument, and how is it defined?

This question has been answered by the creation of a definition of a multi-user instrument as such: A multi-user instrument is a musical instrument, piece, or ensemble, wherein multiple individuals have shared performative control over a single sound-producing source or engine, or where the connections in a network of discrete sound producing sources or engines controlled by separate individuals achieve a sufficient level of interconnectivity that it is difficult to differentiate between those discrete sources and a group whole.

6.1.2 Are there distinct morphologies of multi-user instruments, and may a taxonomy be created to organise them?

a. Reviewing the literature on multi-user instruments reveals several physical characteristic of multi-user instruments: Upon examining instruments found in the literature, a set of physical characteristics of multi-user instruments has been created, based upon an extension of the characteristics of multi-user instruments as defined by Jordà (2005). These defining characteristics are as follows:

- *User-number and user-number flexibility*: The number of performers for the given instrument. This may be variable. Theoretically, the more performers on the instrument, the more simultaneous musical information may be manipulated, increasing "musical bandwidth".
- *User-roles and role flexibility*: Many multi-user instruments feature different roles for each performer. For example, one performer may determine the pitch of the instrument, while another determines the amplitude. Some multi-user instruments also allow for the performers to dynamically change what their role is during performance. Different roles in an instrument allow for multiple musically intense or demanding tasks to be executed simultaneously and with full attention.
- *Interdependencies and hierarchies*: The degree to which performers interact with and affect each other. Also, the manner in which influence is shared and exerted in the instrument(i.e. democratically, anarchically, dictatorially). The more inequality in the capabilities that each performer has, the more pronounced the hierarchy that emerges from the system.
- *Geographic group distribution*: Some multi-user instruments have the unique capability to allow performers to be distributed across different geographic locations.
- *Incidental versus coordinated group formation*: Some instruments are constructed so that they may be performed at any point in time by users that incidentally arrive at the instrument within the same time span. These users

may even have never been associated with each other before the performance. In some instances, these performers may not even be aware that they actively performing *at all*. Other instruments are comprised of groups of performers that coordinate when they perform on the instrument, such as a regularly practising ensemble.

- *Number of sound sources*: Some multi-user instruments are voiced through multiple articulated sound sources, such as the different laptops in a laptop ensemble. The key difference between an ensemble of laptops simply playing with each other, and a multi-user instrument comprised of a group of laptops is the issue of interconnectivity.
- *Medium*: Some instruments are inherently tied to a certain medium, such as acoustic, electro-acoustic, and digital instruments.

b. A taxonomy for multi-user instruments has been created based on grouping instruments by their defining characteristic as defined previously, and by their design methodology, and performance practices: This taxonomy creates several distinct categories for multi-user instruments:

- *Utilitarian Multi-User Instruments*: These are acoustics instruments that employ multiple performers due to some logistical reason, such as the unwieldy size of the instrument. The user number and user roles in these instruments are fixed, and the performers are located in a single space. Traditionally these instrument relied upon coordinated group formations. These instruments only have one sound source, and are acoustic in nature.
- *Extended Traditional Instruments*: Instruments that extend an acoustic instrument, often times through the use of electronic sound processing. The user numbers and roles tend to be more fixed, with performers assuming specific duties, such as exciting the acoustic instrument versus processing the instruments output. The performers of these instruments are locally located, and have coordinated group formations. There may be multiple sound sources, and are electro-acoustic in nature.
- *Surface Instruments*: Instruments that employ the use of a surface as communal medium for multiple performers to perform on, as well as to provide visual feedback. The user number and user roles on these instruments tend to be rather flexible and egalitarian. The groups that play on these instruments tend to be local, though incidental group formations are often possible as these instruments may be situated as an installation for public interaction. There is usually a single sound source, and they often use a digital medium.
- *Interconnected Laptop Ensembles*: Ensembles that are enmeshed to such a high degree, often through the use of digital networking technologies,

that they could be identified as a sort of meta-multi-user instrument. User-number and user-roles are very flexible, the groups that play these instruments may either be located in a single space, or geographically displaced. They are structured as an ensemble, and thus use coordinated group formations. There may be multiple sound source, and they use the digital medium.

- *Cloud Instruments*: These instruments live in "The Cloud", as it were, usually on the internet and accessible by any individual with a connection to that network. User numbers and roles are very flexible for these instruments as they are intended for use by the general public, thus they have highly incidental group formations. They may have many different sound sources and are of either an analogue or digital medium.
- *Kinetic Group Instruments*: These instruments map the movements and gestures of a group of individuals to control a singular instrument. User numbers and roles in these instruments are often well defined. These groups formed locally and are usually coordinated. There may be multiple sound sources, and are digital in nature.
- *Game Instruments*: These instruments map the actions of multiple individuals engaged in some kind of game to performative controls over a musical instrument. The user number and roles in these are fixed, and the groups are locally formed in a coordinated fashion. There may be multiple sound sources, and may be either analogue or digital.

6.1.3 Is there a manner in which to examine the collaborative capabilities of a multi-user instrument?

A dimension space has been created as part of this research to examine and compare the collaborative capabilities of multi-user instruments. This dimension space extends the dimension space as described by Hattwick and Wanderley (2012), and takes into consideration more capabilities which are especially pertinent to the performance of electronic multi-user instruments. The axes of this dimension space are as follows:

- *Texture* - Homogeneous to Heterogeneous: Are individual parts uniquely discernible, or do they blend together and/or sound similar?
- *Equality* - Unequal to Equal: Do performer have equal capabilities, or are there multiple roles with differing functions? Is there a hierarchy or uneven distribution of power over the instrument?
- *Centralisation* - Centralised to Decentralised: Is there are a single server or other source the player must use? Is there a conductor? Do performers have access to the same information/data or is it fire-walled between performers.

- *Physicality* - Fixed to Free: Is there a physical manifestation to the instrument, or is it virtualised? How important is physical gesture and communication?
- *Synchronicity* - Synchronous to Sequential: Do performers play simultaneously to each other, or is there a substantial lag-time to collaboration? Do performers take turns?
- *Dependency* - Interdependent to Independent: Do performers interact with and depend on each other, or are they independent of outside influence? Do performers rely on each other to produce sound at all?
- *Cognisability* - Obscure to Cognisable: Are the interactions that occur on the instrument easily cognisable to an audience member, or is the behaviour of the instrument obscured?

6.1.4 Is a methodology for designing NMI's based on the aesthetics of a specific ensemble able to generate NMI's which satisfy the design requirements generated by that ensemble?

To test this a new set of design requirements were created which were based upon the aesthetics of the band Glitch Lich. These aesthetics demands for NMIs were derived from the ensemble, which has been performing together for over five years, and was used to generate a set of design requirements. These aesthetics demands and their corresponding design requirements may be found in table 5.

Several iterations of NMIs were designed and developed over the course of this research in alignment with these design goals. The final iteration, entitled Simulacra was analysed in depth in Chapter 5. Several of these design goals relate to network capabilities. The networking capabilities of each of the NMIs developed in this research rely upon a new software system entitled OSCthulhu. Each of one of the non-networking related design goals, and how the NMIs developed in this research satisfy or dissatisfy the proposed design goals, are addressed individually in the following:

Composability and performability / Intricate and commandable controls - Several of the NMIs generated in the course of this research utilise a similar interface based upon the creation of synthesis engines and interconnecting them, as represented graphically by networks of ellipses situated on a two-dimensional plane. These NMIs (NeuroMedusae I, NeuroMedusae II, and Yig), despite their similarities in control, still give rise to vastly different sounding performances of music. Furthermore, these NMIs have been used in the creation of large amounts of musical content, over several hours of performance material. Given these facts one may surmise that the NMIs, despite their shared interface structure, are capable of being performed in musically unique ways as commanded by the performers in each different performance scenario.

Aesthetics Demands	*Design Requirements*
Dislocative collaborative interactions	Networking infrastructure
	Minimised and obfuscated lag time
Shared musical resources	Network Synchronisation
Informal performance settings	Low-bandwidth networking
	Disconnection recoverability
Composability and performability	Intricate and commandable controls
Virtuosic performability	Complex interface interaction
Long-standing ensemble	Less need for low learning curve
Improvisational capability	Variety and variability of controls
	Interperformer interaction
Algorithmic musical material	Support for algorithms
Sense of liveness in performance	Interperformer interaction
	Emergent sonic behaviours
	Visual projection
Egalitarian/Socialised distribution of power	Homogeneous/shared user capabilities

Table 5: NMI Design Requirements as dictated by the aesthetics of Glitch Lich.

Long-standing ensemble / Less need for low learning curve - The NMIs created during this research make no special efforts to attempt to be used by novices or non-musicians. The instruments require knowledge of sound-synthesis, audio-bussing, feedback networks, improvisation, and the specifics of the NMIs themselves. These instruments therefore are aimed at individuals with this knowledge and willing to spend the time to learn their idiosyncrasies.

Improvisational capability / Variety and variability of controls - The NMIs created in this research in general actually contain only a small number of performable actions at the performer's command. These actions are the abilities to create a synthesis-engine, to remove a synthesis-engine, to modulate that synthesis-engine in some way (though prescribed control parameters), and the ability for the structure of the network of these synthesis engines to change (manually in NeuroMedusae I/II and Yig and automatically in Simulacra). However, the feedback-nature of these synthesis engines makes them inherently combinatorial in nature.

This means that different combinations of synthesis engines, in different network arrangements, with different control parameter values will produce different results. The number of combinations of this kind can quickly be come exorbitant. In Simulacra, there are 39 possible synthesis engines to choose from, each of which contain 2 different control parameters. If you were to do a crude calculation, by saying that the minimum and maximum of each control parameter produced a different effect, then each new addition of a synthesis engine has 156

possibilities for each synthesis engine added. If one were to add just two synthesis engines, this would mean that there are already 24,336 possible combinations. If one were to add ten (the usual upper limit to the number of simultaneous synths in Simulacra), there would be over 10^{21} possible combinations. That is more possible combinations than the number of stars found in the Milky Way galaxy (Clark, 2011). Neuromedusae I provides for even more combinations. In NeuroMedusae II there are 40 synths, each with 8 different control parameters. Given the same crude calculation treatment, two synthesis engines allow for 409,600 possible combinations. Ten synthesis engines would provide for over 10^{28} possible combinations. This is more combinations that the number of stars in the entire universe (van Dokkum and Conroy, 2010). To say the least, this should provide more than enough variability for the performers to utilise.

Virtuosic performability / Complex interface interaction - Given the previously discussed number of possible instrument combinations of synthesis-engines states, it is logical that the interface provides the ability for the user to create complex interactions, and to facilitate the possibility of virtuosic performability, by encouraging the user to learn the possible areas in these large zones of combinatorial possibility that the instrument may be taken.

Algorithmic musical material / Support for algorithms & Sense of liveness in performance / Emergent sonic behaviours - Chapter 5, section 5.2 explains how roughly 40% of the changes that occur in a given performer's audio signal is not derived from either the performer directly controlling a specific synth, or from the other performers with which the performer is interacting. This left over change can most likely be attributed to one of two causes. One of these causes is on the small scale, and is related to algorithms. Each individual synth is designed with its own generative/algorithmic capabilities. The amount of variability in these generative structures found in the different synths differs greatly from one synth to another. One synth may be completely static in nature, while another may derive a lot of activity from programmed algorithmic behaviours. The next cause is likely derived from the more macro-cosmic effect of feedback networks.

As discussed in Chapter 4, NeuroMedusae I/II, Yig, and Simulacra all depend heavily on different kinds of feedback networks. NeuroMedusae I utilises single-sample feedback loop, NeuroMedusae II uses convolution networks, Yig uses rhythmically interacting feedback and distortion, and Simulacra uses buffer based feedback. These behaviour of these feedback networks can be quite unpredictable and depends largely on the combinations of synths that are present. The analysis conducted based on network data and music information retrieval makes the presence of these kinds of effects obvious, however separation the two out is very difficult. One reason for this is that the algorithms found in the synths also influence the feedback networks themselves, so the two are interconnected. Another reason is that without dissecting each synth directly and deriving its ran-

domisation seed at the exact moment it was used, it is very difficult to differentiate between emergent and algorithmic material. However for the purposes of the study, and with prior knowledge that both are programmed into the NMIs, it is enough to state that there is room in the analysis data for the presence of both.

Sense of liveness Improvisational capability/ in performance /Interperformer interaction - The in-depth analysis of the Mute Magazine performance of Simulacra, found in Chapter 5, section 5.2, reveals the complex interactions that occur during the performance. Network data and music information retrieval analysis shows that the individual outputs of each musician displays interactivity with each other musician within the ensemble, and that the shape of the network of this interaction constantly changes over the course of the piece. This allows performers to connect and interact with each other in many surprising ways, opening the door for musical improvisation.

Sense of liveness in performance / Visual projection - Visual projection plays a large role in several of the NMIs developed, including Leech, Flow, Yig, and Simulacra. These NMIs display several descriptive characteristics of the ensemble, the instruments, and of the music that is produced. These characteristics include, project virtual representations of the ensemble members, the chat log of said members, the different synths created and their orientation in a network, and visualisation of audio signals, all of which are aesthetically rendered as to cohere with the artistic considerations of the piece.

Egalitarian/Socialised distribution of power & Homogeneous/shared user capabilities - The analysis of Simulacra in Chapter 5, section 5.2, clearly shows that there are some imbalances in the structure of the performance. Performer Cole Ingraham accounts for only 11.46% of all the network activity in the performance, while performers Chad McKinney and Curtis McKinney account for 54.75% and 33.78% respectively. This includes any actions a performer may take, including adding synths, removing synths, and modulating synths. However, while this is true, the analysis shows that the socialised interconnection scheme created by Simulacra actually serves to mitigate this deficit of presence in the piece, and boosts the performer's effect upon the ensemble. The music information retrieval analysis shows that Ingraham was able to influence 16.6% of both Chad McKinney's and Curtis McKinney's audio signals. Ingraham also only accounted for influencing his own audio signal by 22.2%, meaning that both Chad McKinney and Curtis McKinney influenced his signal more than he did himself. This is a double-edged sword of socialised interconnectivity. With that said, while performer Chad McKinney exhibited by far the most network activity, accounting for a majority of *all* activity in the performance, it is interesting that his effect is actually somewhat mitigated in the audio analysis, which shows that he influenced each of the performer's audio signals about the same amount as Curtis McKinney, even though Curtis McKinney's network activity was much lower.

In conclusion, while there are some mixed results in the findings for the distribution of power in Simulacra, overall the implementations of the NMIs seem to satisfy the design requirements that were produced by the aesthetic demands established by Glitch Lich. Furthermore, these aesthetics demands and design goals gave clear direction to the design, development, and implementation of these NMIs, and can be seen as a beneficial and efficient manner of organising the processes of creating an NMI.

6.1.5 May the issues of liveness and disembodiment inherent to NMI's be addressed in some manner?

This question was partially addressed in the previous section, however here it will considered in full and synthesised further. The issue of liveness in performing with NMIs has been attacked in this research in several ways. One manner is by developing NMIs which make heavy usage of real-time techniques that bring about seemingly spontaneous or emergent characteristics to performances. Algorithmic and generative capabilities have been introduced into many of the synthesisers, and feedback networks serve as the main crux of several of the NMIs (NeuroMedusae I/II, Yig, Simulacra).

Another method for creating a deeper sense of liveness and mitigating the effects of disembodiment in NMIs is through the projection of visualisations as part of the construction of the developed NMIs. These projections show artistic rendering of several of the inherent qualities of the NMIs, including " virtual representations of the ensemble members, the chat log of said members, the different synths created and their orientation in a network, and visualisation of audio signals."

Experimental data, found in Chapter 5, section 5.2, shows the presence of the effects of some these techniques for liveness. This data shows that there are emergent behaviours that occurred during a performance of Simulacra, as discovered in an audio analysis cross-referenced with network analysis. This data shows that up to 40% of the behaviours of the instrument are accounted for by the emergent properties of the NMI itself, and not directly related to the actions of the performers.

Furthermore, feedback from audience members, dissected in Chapter 5, and found in full in Appendix C, shows that audience members displayed an awareness for the different ensemble members in their performance, despite the fact that they were not physically present. This data also shows that a relationship of sorts was created during the performance, and the audience members were engaged, despite the fact that only one member was there. Finally, the audience was even aware enough to notice one of the four members of the group was *not* performing that evening with the ensemble, showing an acute awareness for the embodiment of the NMI. This seems to show that the combination of visual projections of

NMIs characteristic, audience engagement through public chat, and the usage of emergent properties of the NMIs, creates more liveness and mitigates the disembodiment that is inherent to the performance of NMIs.

6.1.6 Are there better techniques for overcoming the technical difficulties involved with networking geographically displaced ensembles?

During the course of this research a new software system for networking geographically displaced NMIs was created, entitled OSCthulhu. This networking system is based upon the principles of the Generalised Client-Server Model as created by Tim Sweeny for the networking engine utilised in the video game *Unreal*. This method of networking seems to be much more appropriate for geographically displaced NMIs than the current standard for electronic music networking, based upon mesh networking and implemented in software systems such as OSCGroups, created by Ross Bencina. Chapter 5 shows a rather definitive comparison of OSCGroups and OSCthulhu for performing network actions at a distance. OSCGroups displaced approximately 50% divergence in tests that were conducted, while OSCthulhu exhibited mainly zero, with spikes of divergence up to 5-10% that were quickly mitigated by the synchronisation cycle that is the heart of the OSCthulhu system.

6.1.7 Is there a way to streamline the development processes of creating NMI's?

Chapter 4, section 4.11, discusses the creation of a new framework for creating NMIs called *Azathoth*. This framework is based upon the concept of a "video game engine", as collection of libraries and software tools that provide a complete package for developing a video game. Azathoth attempts to extend this concept into the field of network music by making a "network music engine." Azathoth is created specifically to create the kinds of NMIs as discussed in this research, and facilitates several tedious tasks involved with creating NMIs.

There several ways in which Azathoth attempts to simplify and shorten the process of creating NMIs. Azathoth attempts to greatly simplify the complex task of creating network systems, by mostly removing the problem from the user's hand. Instead, networking is transformed into a system that is more akin to traditional function calls. The software system will automatically propagate the appropriate changes amongst the ensemble and synchronise any differences that occur during performance due to packet loss or other effects. As well, Azathoth provides many stock capabilities a user would require in an NMI in performance, with the kinds of capabilities described in this research. Chat, GUI windows, sound synthesis, and 3D graphics capabilities are all part of the Azathoth engine.

Azathoth is still in progress, and there is no experimental data at this moment to show the effectiveness of utilising it over hand-programming boiler plate-code

from scratch. However, anecdotal evidence gained from the author and the members of Glitch Lich utilising Azathoth seems to indicate that the presence of a library and set of tools has reduced the complexity and time requirements involved with the creation of new NMIs.

6.2 Implications, Future Work, and Conclusion

The research here indicates that it may be useful for future luthiers of NMIs to investigate the possibility of creating visualisation systems for their instruments, as these systems mitigate a major concern with network music, being the lack of physicality. Furthermore, those who engage in network music based on the passing of control messages across geographically displaced ensembles should seriously consider a Generalised Client-Server Model for their networking scheme, as mesh networking has been demonstrated to be inappropriate and ineffective outside of local area networks. A GCSM, informed by technology used by multiplayer video games, has been demonstrated to work efficiently at minimising packet loss encountered on the open Internet while performing network music.

The benefits of an aesthetically driven iterative design methodologies have shown benefits to the author in the design, development, and analysis of these instruments, by providing a solid framework with which to work within. This methodology seems to be appropriate to software projects that are creatively focused and seems much more applicable than the traditional "Waterfall" method, which has been cultivated for enterprise software distributed to the general public instead of a small specialised target group.

The work conducted with Azathoth is being continually developed. This development has taken it much further than the small library of boiler-plate code that it was originally envisioned as. To reflect the change in scope there has been a change in name as well, now being called Necronomicon (the compendium of forbidden knowledge which figures heavily in Lovecraft's works). Instead of being written in C++ Necronomicon is written in the programming language Haskell. Haskell is a pure functional language which focuses on expressive power (via functional language features such as closures and lambdas) and safety (via a superb type system and ardent pursuit of purity without side-effects) (O'Sullivan, 2008). This focus on both expressiveness and safety have proven to be a good combination for working with network computer music, which requires both expressiveness during composition, and safety during performance. The scope has grown to include not only boiler-plate code, but the entirety of what is required to create NMIs. Necronomicon features its own UGen based Digital Signal Processing audio engine, a three-dimensional linear algebra library, a full fledged 3D graphics engine complete with shader support, a new networking engine and server which uses both UDP and TCP, a system for networked GUI widgets, and a parser for musical pattern generation. All of this is available as a single library

which is linkable to in Haskell. This emphasis on breadth of features, safety, and ease of use should hopefully attract more composers with the prospect of creating their own NMIs. The previous technologies developed for this research have seen less than ten users make use of it. However there is a large degree of complexity and long development time required for their usage and continued maintenance. It is hoped that Necronomicon, being an all-in-one package will attract a larger user base of composers and performers.

Lastly, the design space for Network Based Multi-Users Instruments is vast and fertile ground. The possibilities contained in just one of these instrument has been demonstrated to rival the number of stars in the universe. Networks in general are encroaching upon more and more of the daily lives of human beings. It seems only natural for people to want to connect to each other. This research has shown just a few of the ways for people to connect on a deep and meaningful level, as developers, musicians, audiences, and humans. The future of networks and music seems very bright.

References

Avid Audio Inc. (2011). *Pro Tools 101 Official Courseware Version X*, Delmar Cengage Learning.

Aylward, R. and Paradiso, J. A. (2006). Sensemble: A wireless, compact, multi-user sensor system for interactive dance, *NIME '06: Proceedings of the 2006 conference on New interfaces for musical expression*, pp. 134–139.

Banzi, M. (2009). *Getting Started with Arduino*, Make.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A. and Cunningham, W. (2001). Manifesto for Agile Software Development.

Bellona, J. and Schropp, J. (2012). Sound pong: An interactive exploration of sonic space, *In Proceedings of the1st Symposium on Laptop Ensembles and Orchestras*.

Bencina, R. (2013). OscGroups. Available from: `http://www.audiomulch.com/~rossb/code/oscgroups/` [Accessed 20 April 2015].

Birmingham Ensemble for Electroacoustic Research (2012). Available from: `http://www.birmingham.ac.uk/facilities/BEAST/research/Birmingham-Ensemble-for-Electroacoustic-Research.aspx` [Accessed 10 October 2012].

Blackledge, J. M. (2006). *Digital Signal Processing: Mathematical and Computational Methods, Software Development and Applications*, Woodhead Publishing.

Blaine, T. and Perkis, T. (2000). The jam-o-drum interactive music system: a study in interaction design, *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques*, ACM, New York, NY, USA, pp. 165–173.

Bongers, B. and Sensorband (1998). An Interview with Sensorband, *Computer Music Journal* **22**(1): 13–24.

Borgo, D. (2007). *Sync or swarm: improvising music in a complex age*, Continuum International Publishing Group Ltd.

Boulder Laptop Orchestra (2012). Available from: `http://cismat.org/blork.html` [Accessed 10 October 2012].

Bowen, N. (2012). 4quarters: Real-time collaborative music environment for mobile phones, *In Proceedings of the 1st Symposium on Laptop Ensembles and Orchestras*.

Bowers, J. (2008). *Improvising Machines: Ethnographically Informed Design For Improvised Electro-Acoustic Music*, PhD thesis, Princeton University.

Brauchli, B. (2005). *The Clavichord*, Cambridge University Press.

Brigham, E. O. (1973). *The Fast Fourier Transform*, first edn, Prentice Hall.

Brown, C. and Bishcoff, J. (2002). Indigenous to the Net: Early Network Music Bands in the San Francisco Bay Area. Available from: `http://crossfade.walkerart.org/brownbischoff/IndigenoustotheNetPrint.html` [Accessed 2 August 2010].

Burk, P. (2000). Jammin ' on the Web- a new Client/Server Architecture for Multi-User Musical Performance. `http://www.transjam.com/info/transjam2000.pdf`.

Bush, D. and Kassel, R. (eds) (2006). *The Organ, an Encyclopedia*, Routledge.

Caceres, J. P. (2010). Jmess. Available from: `http://code.google.com/p/jmess-jack/` [Accessed 24 May 2011].

Catto, E. (2010). Box2D. Available from: `http://box2d.org/` [Accessed 24 May 2011].

Chafe, C. and Leistkow, R. (2008). Levels of Temporal Resolution in Sonifcation of Network Performance, *Proceedings of International Computer Music Conference*.

Chew, E., Sawchuk, A., Tanoue, C. and Zimmermann, R. (2005). Segmental tempo analysis of performances in user-centered experiments in the distributed immersive performance project, *SMC '05: Proceedings of the 2005 Sound and Music Computing Conference.*

Clark, S. (2011). *Galaxy: Exploring The Milky Way*, Fall River Press.

Cockburn, A. (2008). Using both incremental and iterative development, *CrossTalk, The Journal of Defense Software Engineering* **23**(3): 27–30.

Cohen, B. (2011). Bittorrent. Available from: `http://www.bittorrent.com/` [Accessed 24 May 2011].

Collins, N. (2011). Scmire: A supercollider music information retrieval library, *Proceedings of the 2011 International Computer Music Conference*, pp. 499–502.

Cross, L. (1999). Reunion: John cage, marcel duchamp, electronic music and chess, *Leonardo Music Journal* **9**: 35–52.

Davis, T. (2012). The loop: A distributed instrument approach to networked performance, *Symposium for Performance of Electronic and Experimental Music*, pp. 6–7.

Farina, A. (2000). Simultaneous measurement of impulse response and distortion with a swept-sine technique, *Audio Engineering Society Convention 108*.

Fels, S. and Vogt, F. (2002). Tooka: Explorations of two person instruments, *NIME '02: Proceedings of the 2002 conference on New interfaces for musical expression*, National University of Singapore, pp. 1–6.

Foote, J. (2000). Automatic audio segmentation using a measure of audio novelty, *In Proceedings of International Conference on Multimedia and Expo*.

Ford, B., Srisuresh, P. and Kegel, D. (2005). Peer-to-Peer Communication Across Network Address Translators, *In Proceedings of the 2005 USENIX Annual Technical Conference*.

Fujii, K. (2011). Jpcap. Available from: `http://netresearch.ics.uci.edu/kfujii/Jpcap/doc/` [Accessed 24 May 2011].

Fung, G. (2010). Isohunt, bittorrent search engine. Available from: `http:/www.isohunt.com` [Accessed 10 March 2013].

Gong, N., Laibowitz, M. and Paradiso, J. A. (2009). Musicgrip: A writing instrument for music control, *NIME '09: Proceedings of the 2009 conference on New interfaces for musical expression*, pp. 77–74.

Grey, J. M. and Gordon, J. W. (1978). Perceptual effects of spectral modifications on musical timbres, *Journal of the Acoustical Society of America* **63**: 1493–1500.

Gurevich, M. (2006). Jamspace: a networked real-time collaborative music environment, *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, ACM, New York, NY, USA, pp. 821–826.

Gurevich, M. and Fyans, A. C. (2011). Digital Musical Interactions: Performer?system relationships and their perception by spectators, *Organised Sound* **16**: 166–175.

Hagen, S. (2006). *IPv6 Essentials*, second edn, O'Reilly Media.

Hagmann, P. (2005). *From Diffusion MRI to Brain Connectomics*, PhD thesis, École Polytechnique Fédérale de Lausanne.

Hajdu, G. (2004). Composition and improvisation on the net.

Harari, P. and Legge, K. (2001). *Psychology and Health*, Heinemann College.

Harold, E. R. (ed.) (2004). *Java Network Programming, Third Edition*, O'Reilly Media Incorporated.

Hattwick, I. and Wanderley, M. M. (2012). A dimension space for evaluating collaborative musical performance systems, *NIME '12: Proceedings of the 2012 conference on New interfaces for musical expression*, National University of Singapore.

Herrera, J. (2009). Mols: Multiperformer online synthesizer.

Hoftstadter, D. (1979). *Gödel, Escher, Bach: An Eternal Golden Braid*, Basic Books.

Iannix Team (2011). Iannix. Available from: `http://iannix.org/` [Accessed 16 August 2011].

Ingraham, C., McKinney, C., McKinney, C. and O'Brien, B. (2012). Glitch lich: Evolution of an intercontinental network band. 1st Symposium on Laptop Ensembles and Orchestras.

Ishii, R. (2005). *The Development of Extended Piano Techniques in Twentieth-Century American Music*, PhD thesis, The Flordia State College of Music.

Japan Publications (1979). *Hanafuda: The Flower Card Game*, Kodansha.

Jordà, S. (2005). Multi-user instruments: Models, examples and promises, *NIME '05: Proceedings of the 2005 conference on New interfaces for musical expression*, National University of Singapore, Singapore, Singapore, pp. 23–26.

Jordà, S. (2009). The reactable: Tabletop tangible interfaces for multithreaded musical performance, *Revista KEPES* **5**(14): 201–223.

Kahn, C. H. (1980). *The Art and Thought of Heraclitus: A New Arrangement and Translation of the Fragments with Literary and Philosophical Commentary*, Cambridge University Press.

Kim-Boyle, D. (2009). Network musics: Play, engagement and the democratization of performance, *Contemporary Music Review* **28**(4-5): 363–375.

Kleimola, J. (2006). Latency issues in distributed musical performance.

Korg, I. (2008). *NanoKontrol Owner's Manual*, Korg Inc.

Kramer, J. (1988). *The Time of Music: New Meanings, New Temporalities, New Listening Strategies*, Schirmer Books.

Krishnamoorthi, H., Berisha, V. and Spanias, A. (2008). A low-complexity loudness estimation algorithm, *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 361–364.

Lame Project (2011). LAME. Available from: http://lame.sourceforge.net/ [Accessed 24 May 2011].

Laptop Orchestra of Louisiana (2012). Available from: http://laptoporchestrala.wordpress.com/ [Accessed 10 October 2012].

Lerch, A. (2012). *An Introduction to Audio Content Analysis*, Wiley-IEEE Press.

Linux Laptop Orchestra (2012). Available from: http://l2ork.music.vt.edu/ [Accessed 10 October 2012].

Lovecraft, H. P. (2014). *The Complete Fiction of H.P. Lovecraft*, Race Point Publishing.

Manning, P. (2004). *Electronic and Computer Music*, Oxford University Press, Inc.

Manzo, V. J. (2011). *Max/Msp/Jitter for Music*, Oxford University Press.

MaxMind (2011). GeoLite. Available from: http://www.maxmind.com/app/geoip_country [Accessed 24 May 2011].

McConnell, S. (2006). *Software Estimation: Demystifying the Black Art*, Microsoft Press.

McKinney, C. (2009). *Metagaming: Experiments with art and games*, Master's thesis, Mills College.

McKinney, C. and Collins, N. (2012). Liveness in network music performance, *Proceedings of the 2012 International Conference on Live Interfaces*.

McKinney, C. and McKinney, C. (2010). Available from: http://curtismckinney.com/NeuroMedusae_I.wav. [Accessed 3 March 2013].

McShaffry, M. (2012). *Game Coding Complete*, Delmar Cengage Learning.

MIDI Manufacturers Association Inc. (1995). *The Complete MIDI 1.0 Specification*, MIDI Manufacturers Association Inc.

Milne, A. (1928). *The House at Pooh Corner*, Methuen.

Moore, B. J. and Glasbers, B. (1995). Comparison of Auditory Filter Shapes Obtained With Notched Noise and Noise-Tone Makers, *Journal of the Acoustical Society of America* **97**: 1175–1182.

Moscow Laptop Cyber Orchestra (2012). Available from: http://cyberorchestra.com/ [Accessed 10 October 2012].

Nahmani, D. (2009). *Apple Pro Training Series: Logic Pro 9 and Logic Express 9*, Peachpit Press.

Neuhaus, M. (2010). The Broadcast Works and Audium. Available from: `http://www.max-neuhaus.info/bibliography/` [Accessed 10 August 2010].

Neumann, E. (1954). *The Origins and History of Consciousness*, Princeton University Press.

Ostertag, B. (2002). Human Bodies, Computer Music, *Leonardo Music Journal* **12**: 11–14.

O'Sullivan, B. (2008). *Real World Haskell*, O'Reilly Media.

Pritchett, J. (1996). *The Music of John Cage*, Cambridge University Press.

Public Domain (2010). Public Domain Digital Arts Festival. Available from: `http://scansite.org/publicdomain.htm` [Accessed 16 July 2010].

Puente, J. P. B. (2011). *Gnu psychosynth: A framework for modular, interactive and collaborative sound synthesis and live music performance*, Master's thesis, University of Granada.

Ramakrishnan, C., Freeman, J. and Varnik, K. (2004). The architecture of auracle: a real-time, distributed, collaborative instrument, *NIME '04: Proceedings of the 2004 conference on New interfaces for musical expression*, National University of Singapore, Singapore, Singapore, pp. 100–104.

Reas, C., Fry, B. and Maeda, J. (2007). *Processing: A Programming Handbook for Visual Designers and Artists*, The MIT Press.

Rebelo, P. and King, R. (2010). Anticipation in networked musical performance, *Proceedings of the 2010 international conference on Electronic Visualisation and the Arts*, British Computer Society, pp. 31–36.

Rebelo, P. and Renaud, A. B. (2006). The frequencyliator: distributing structures for networked laptop improvisation, *Proceedings of the 2006 conference on New interfaces for musical expression*, NIME '06, IRCAM — Centre Pompidou, Paris, France, France, pp. 53–56.

Reges, S. (2010). *Building Java Programs: A Back to Basics Approach*, second edn, Addison Wesley.

Renaud, A. and Câeceres, J. P. (2001). Playing the network: the use of time delays as musical devices, *Proceedings of the 2001 International Conference on Auditory Display*.

Renaud, A. and McKinney, C. (2010). Available from: http://curtismckinney.com/Renditions.wav . [Accessed 3 March 2013].

Renaud, A., Rebelo, P. and A., C. (2007). Networked music performance: State of the art, *Proceedings of AES 30th Conference on Intelligent Audio Environments 2007*.

Reynolds, M., Schoner, B., Richards, J., Dobson, K. and Gershenfeld, N. (2001). An immersive, multi-user, musical stage environment, *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, pp. 553–560.

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

Salmoria, N. (1997). Mame. computer software.

Sanden, P. (2013). *Liveness in Modern Music: Musicians, Technology, and the Perception of Performance*, Routledge.

Sasso, L. (2002). *Native Instruments: Reaktor 3- The Ultimate Hands-on Guide for All Reaktor Fans*, Wizoo.

Sebestik, M. (1992). Listen.

Shim, K. J., Hsu, K.-W. and Srivastava, J. (2011). Privacy, security, risk and trust: An exploratory study of player performance, motivation, and enjoyment in massively multiplayer online role-playing games, *2011 IEEE Third International Conference on Social Computing*, pp. 135–140.

Siwiak, D. (2012). Laptop orchestra of arizona state: The student venture, *In Proceedings of the1st Symposium on Laptop Ensembles and Orchestras*.

Slater, J. B. (2013). Mute Magazine: Slave To The Algorithm, **3**(4).

SNK Playmore Corporation (1993). Samurai showdown. video game.

Sonorities Festival (2010). Available from: http://www.sonorities.org.uk/ [Accessed 24 May 2011].

Stelkens, J. (2003). peersynth - a p2p multi-user software synthesizer with new techniques for integrating latency in real time collaboration., *ICMC '03: Proceedings of the 2003 International Computer Music Conference*, National University of Singapore, Singapore, Singapore.

Stevens, S. S., Volkmann, J. and Newman, E. B. (1937). A Scale for the Measurement of the Psychological Magnitude Pitch, *Journal of the Acoustical Society of America* **8**: 185–190.

Stevens, W. R. (1994). *TCP/IP Illustrated*, Addison-Wesley Professional.

Stevens, W. R., Fenner, B. and Rudoff, A. M. (2004). *UNIX Network Programming: The sockets networking API*, Addison-Wesley Professional.

Summerfield, M. (2010). *Advanced Qt Programming: Creating Great Software with C++ and Qt 4*, Prentice Hall.

Sweeney, T. (1999). The unreal networking architecture: The server is the man. Available from: `http://udn.epicgames.com/Three/NetworkingOverview.html` [Accessed 16 May 2010].

Tanaka, A. and Bongers, B. (2001). Global striing: A musical instrument for hybrid space, *In Proceedings of cast01: Living in Mixed Realities*, Fraunhofer IMK, St. Augustin, Germany, pp. 177–181.

The Huddersfield Experimental Laptop Orchestra (2012). Available from: `http://helo.ablelemon.co.uk/doku.php/start` [Accessed 10 October 2012].

Transmission Project (2011). Transmission. Available from: `http://www.transmissionbt.com/` [Accessed 24 May 2011].

Trueman, D., Cook, P., Smallwood, S. and Wang, G. (2006). Plork: The princeton laptop orchestra, year 1, *ICMC '06: Proceedings of the 2006 International Computer Music Conference*.

Twitter (2013). Twitter, social networking site. Available from: `http:/www.twitter.com` [Accessed 10 March 2013].

van Dokkum, P. G. and Conroy, C. (2010). A substantial population of low-mass stars in luminous elliptical galaxies, *Nature* **468**: 940–942.

Wanderley, M. M. and Battier, M. (eds) (2000). *Trends in Gestural Control of Music*, IRCAM.

Wang, G. (2002). *The ChucK Audio Programming Language: An Strongly-timed and On-the-fly Environ/mentality*, PhD thesis, University of East Anglia.

Wang, G. (2010). Evolving the mobile phone orchestra, *Proceedings of the 2010 Conference on New Interfaces for Musical Expression (NIME 2010)*.

Wang, G., Misra, A., Davidson, P. and Cook, P. R. (2005). Coaudicle: A collaborative audio programming space, *In Proceedings of the International Computer Music Conference*.

Weinberg, G., Aimi, R. and Jennings, K. (2002). The beatbug network: A rhythmic system for interdependent group collaboration, *NIME '02: Proceedings of the 2002 conference on New interfaces for musical expression*, pp. 107–111.

Weinberg, G. and Gan, S.-L. (2001). The squeezables: Toward an expressive and interdependent multi-player musical instrument, *Computer Music Journal* **25**(2): 37–45.

Wilson, S., Cottle, D. and Collins, N. (eds) (2011). *The SuperCollider Book*, MIT Press, Cambridge, MA.

Wright, M. (2002). Open sound control 1.0 specification. Available from: `http://opensoundcontrol.org/spec-1_0` [Accessed 2 May 2010].

Xenakis, I. (2001). *Formalized Music: Thought and Mathematics in Composition (Harmonologia Series, No 6)*, Pendragon Pr.

Appendices

A *Simulacra* Control Signal Comparisons

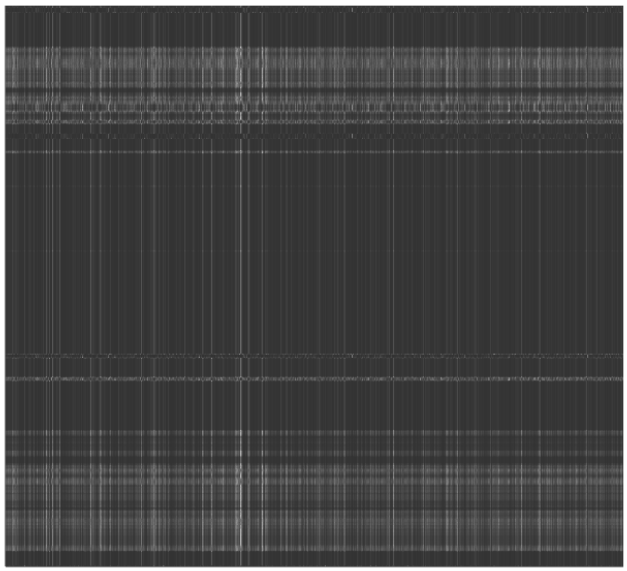

Figure 49: Similarity matrix comparing randomly generated noise with casiosk1's audio stream in *Simulacra*

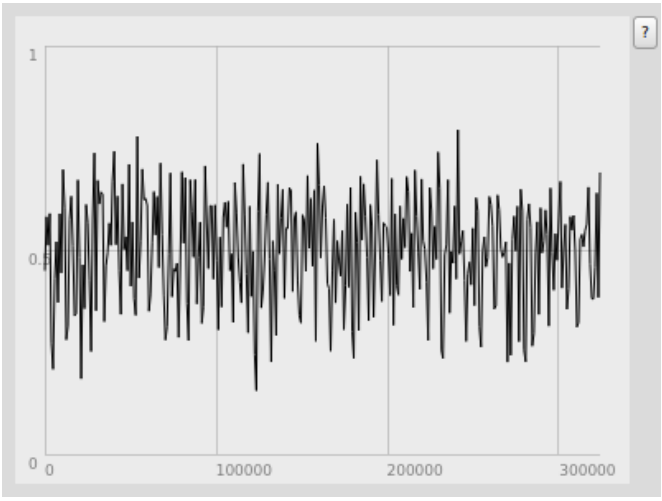

Figure 50: Similarity of randomly generated noise with casiosk1's audio stream in *Simulacra* plotted over time.

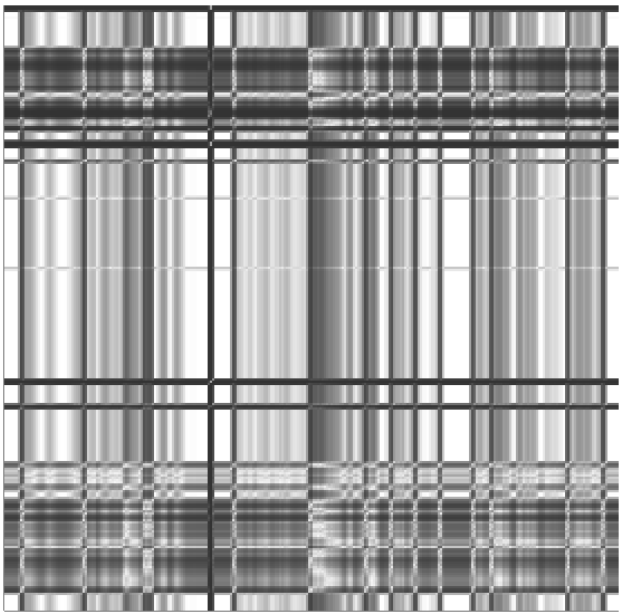

Figure 51: Similarity matrix comparing casiosk1's control stream with casiosk1's audio stream in *Simulacra*

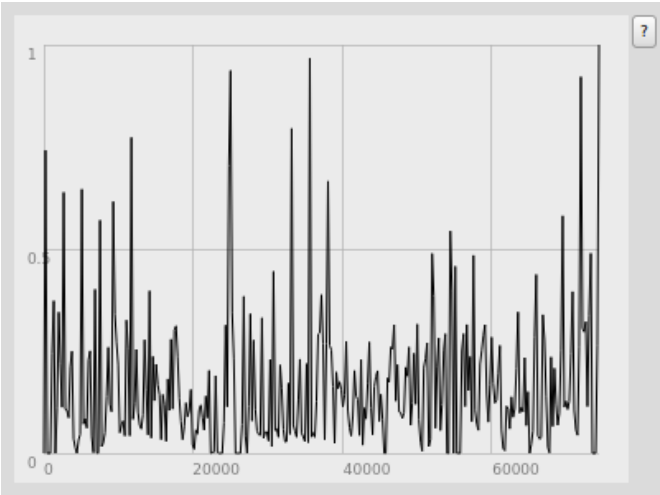

Figure 52: Similarity of casiosk1's control stream with casiosk1's audio stream in *Simulacra* plotted over time.

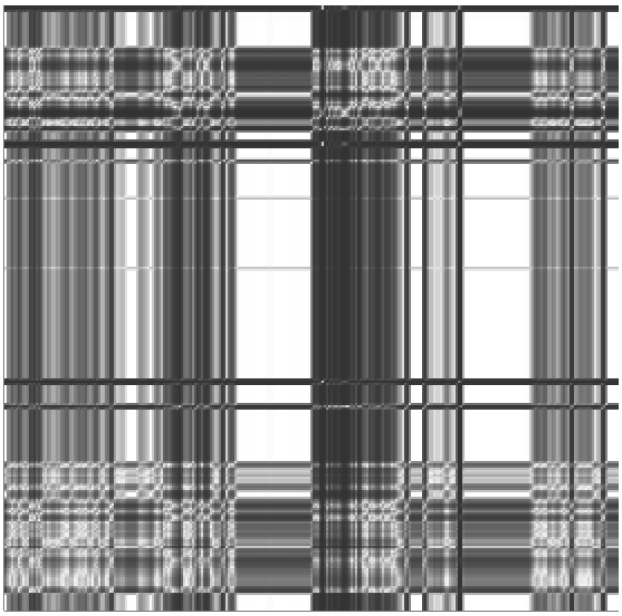

Figure 53: Similarity matrix comparing octopian's control stream with casiosk1's audio stream in *Simulacra*

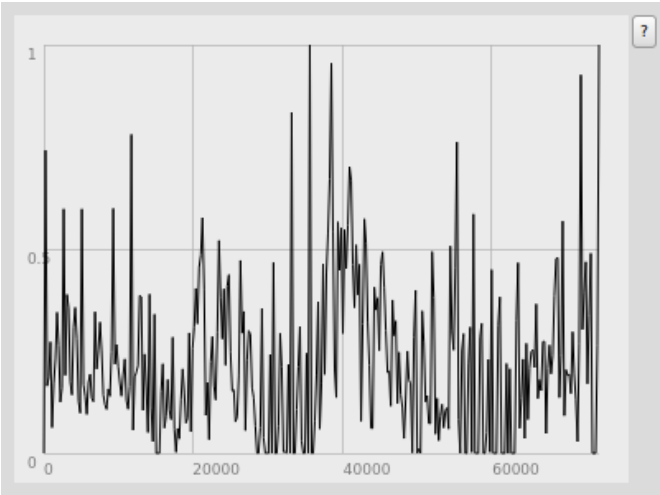

Figure 54: Similarity of octopian's control stream with casiosk1's audio stream in *Simulacra* plotted over time.

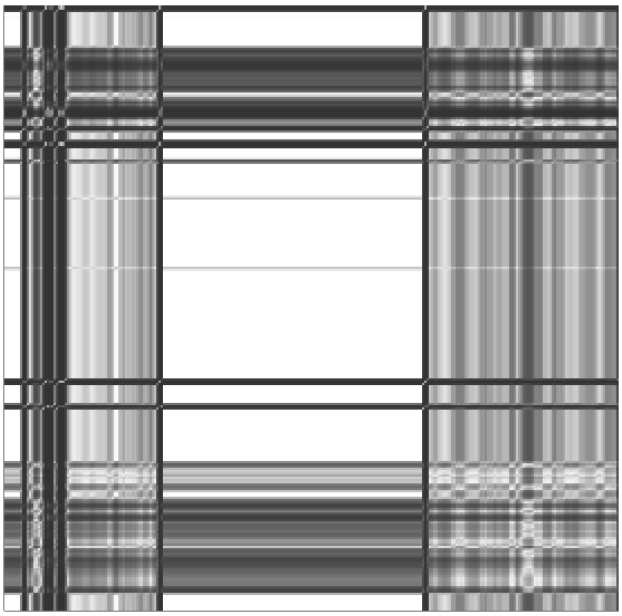

Figure 55: Similarity matrix comparing 55hz's control stream with casiosk1's audio stream in *Simulacra*

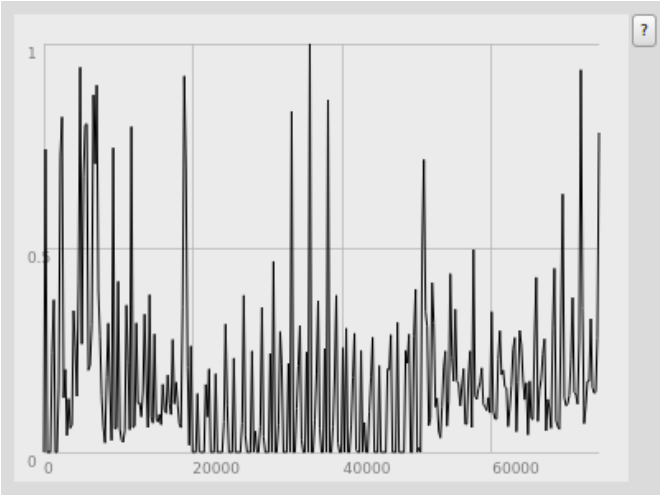

Figure 56: Similarity of 55hz's control stream with casiosk1's audio stream in *Simulacra* plotted over time.

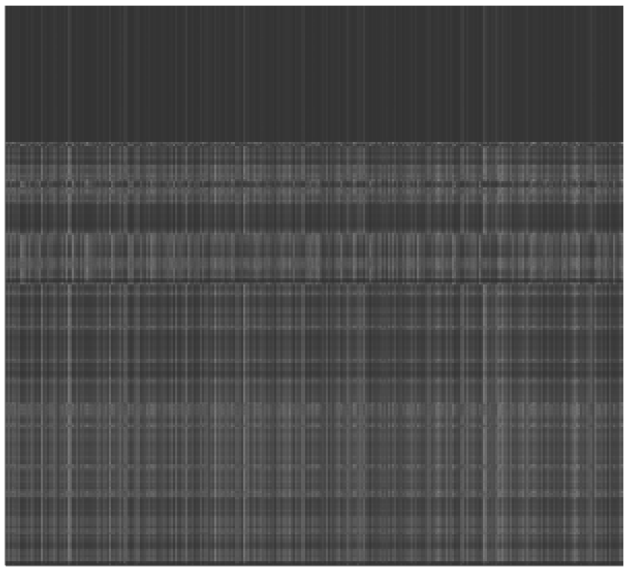

Figure 57: Similarity matrix comparing randomly generated noise with octopian's audio stream in *Simulacra*

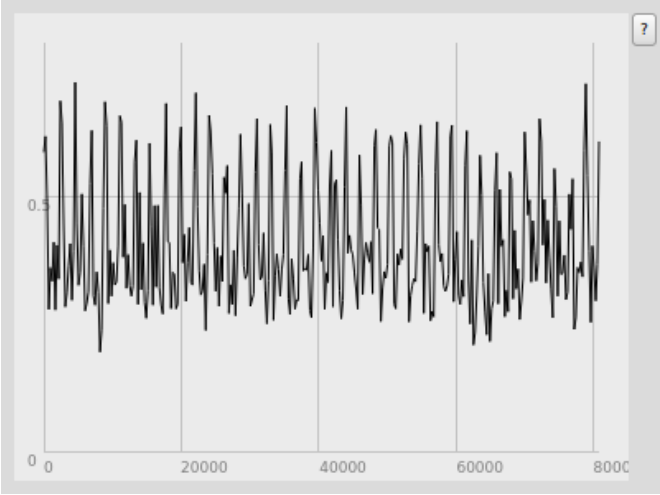

Figure 58: Similarity of randomly generated noise with octopian's audio stream in *Simulacra* plotted over time.

Figure 59: Similarity matrix comparing casiosk1's control stream with octopian's audio stream in *Simulacra*

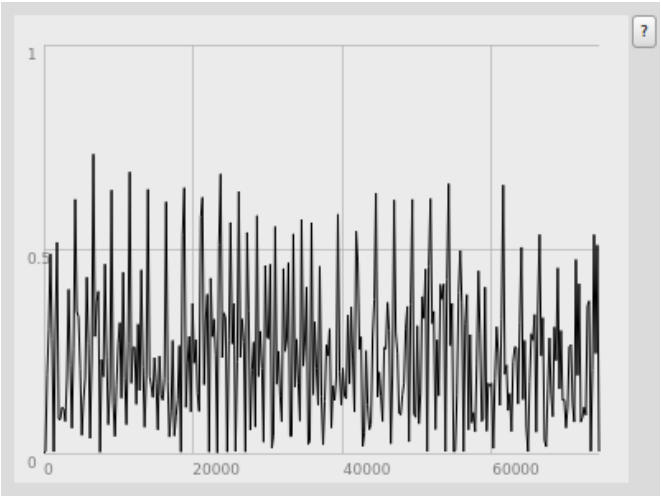

Figure 60: Similarity of casiosk1's control stream with octopian's audio stream in *Simulacra* plotted over time.

Figure 61: Similarity matrix comparing octopian's control stream with octopian's audio stream in *Simulacra*

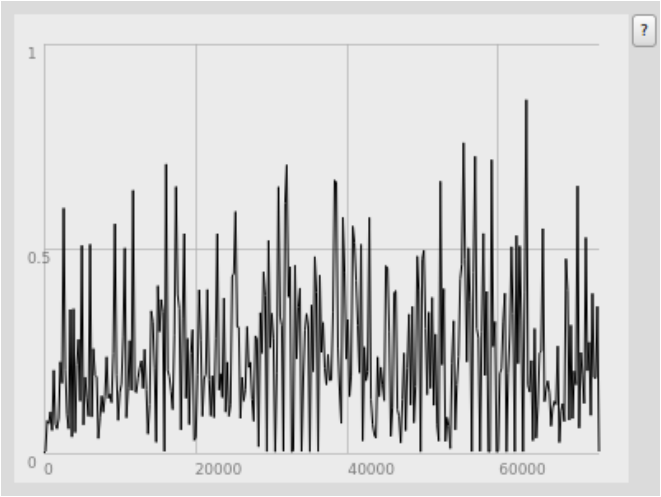

Figure 62: Similarity of octopian's control stream with octopian's audio stream in *Simulacra* plotted over time.

Figure 63: Similarity matrix comparing 55hz's control stream with octopian's audio stream in *Simulacra*

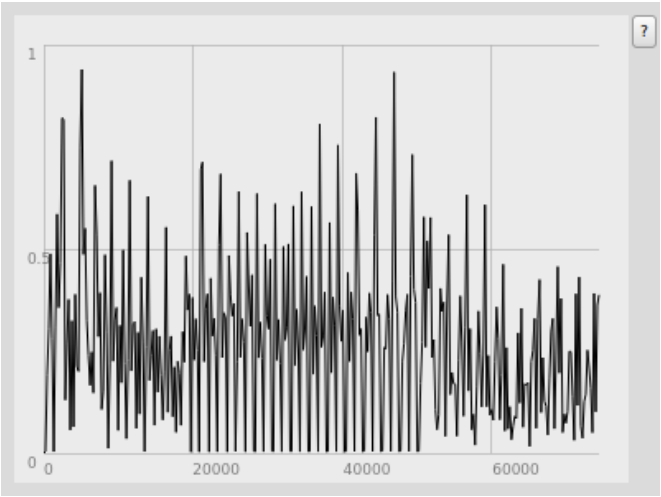

Figure 64: Similarity of 55hz's control stream with octopian's audio stream in *Simulacra* plotted over time.

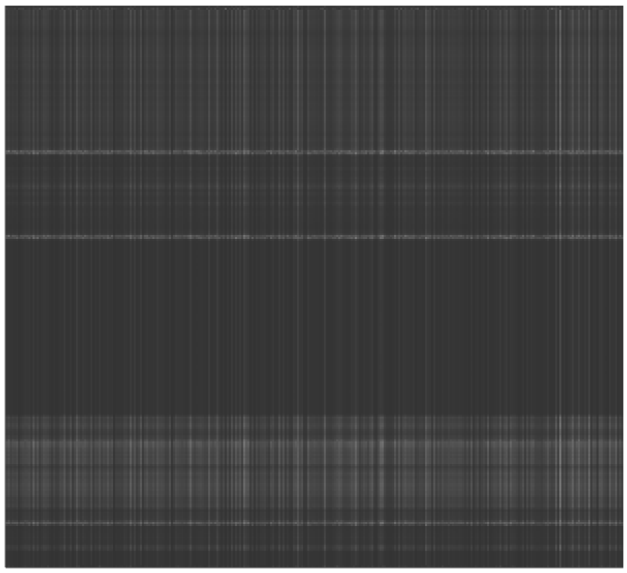

Figure 65: Similarity matrix comparing randomly generated noise with 55hz's audio stream in *Simulacra*

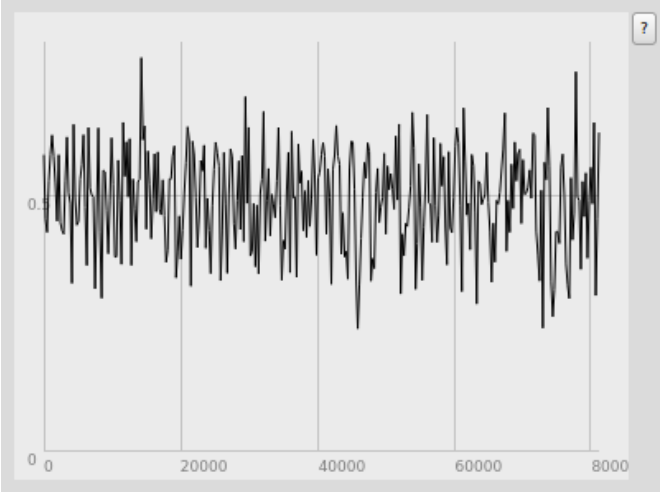

Figure 66: Similarity of randomly generated noise with 55hz's audio stream in *Simulacra* plotted over time.

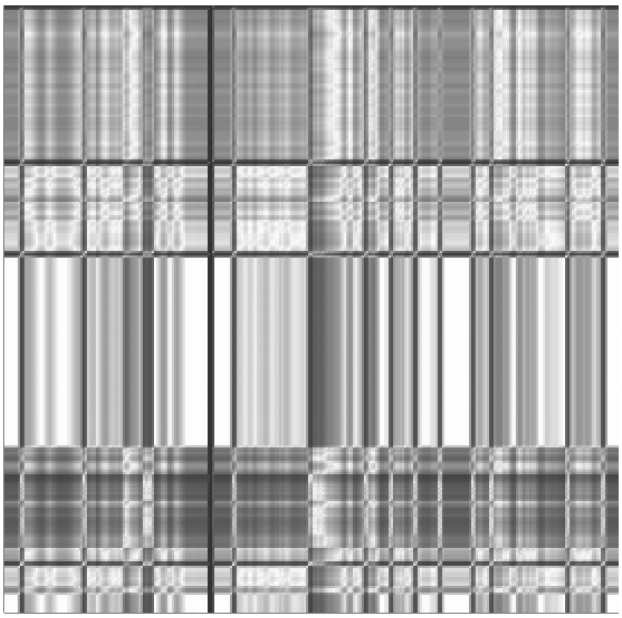

Figure 67: Similarity matrix comparing casiosk1's control stream with 55hz's audio stream in *Simulacra*

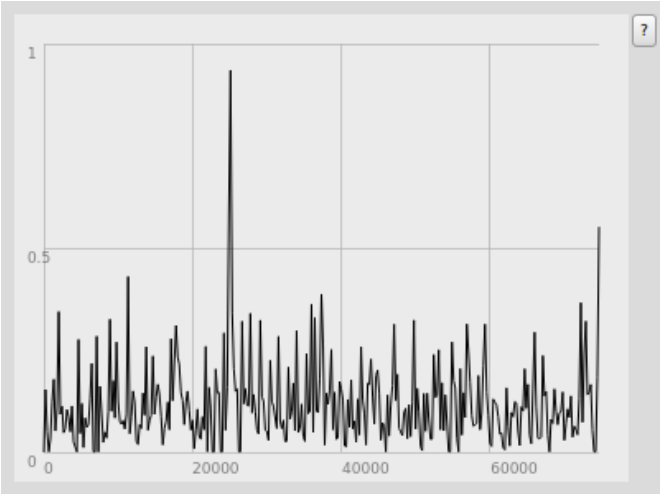

Figure 68: Similarity of casiosk1's control stream with 55hz's audio stream in *Simulacra* plotted over time.

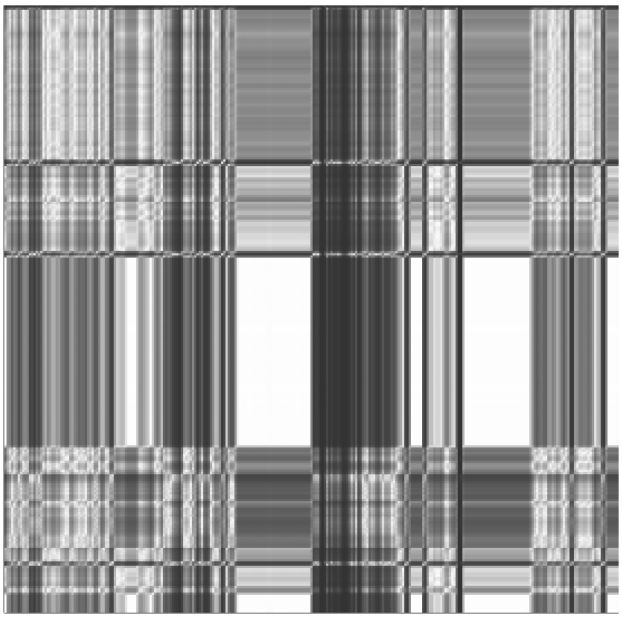

Figure 69: Similarity matrix comparing octopian's control stream with 55hz's audio stream in *Simulacra*

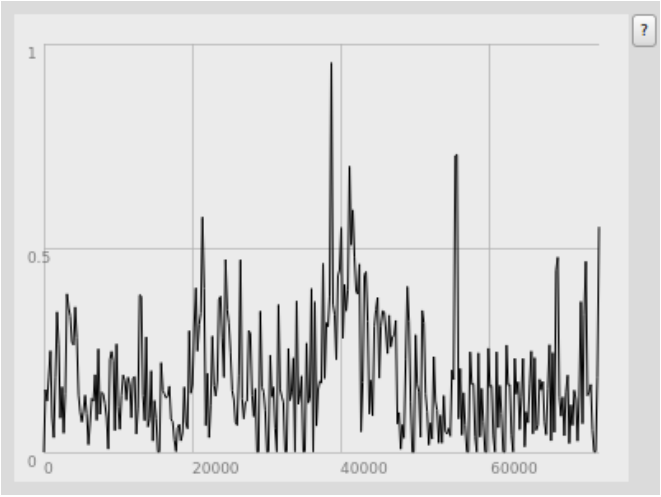

Figure 70: Similarity of octopian's control stream with 55hz's audio stream in *Simulacra* plotted over time.

Figure 71: Similarity matrix comparing 55hz's control stream with 55hz's audio stream in *Simulacra*

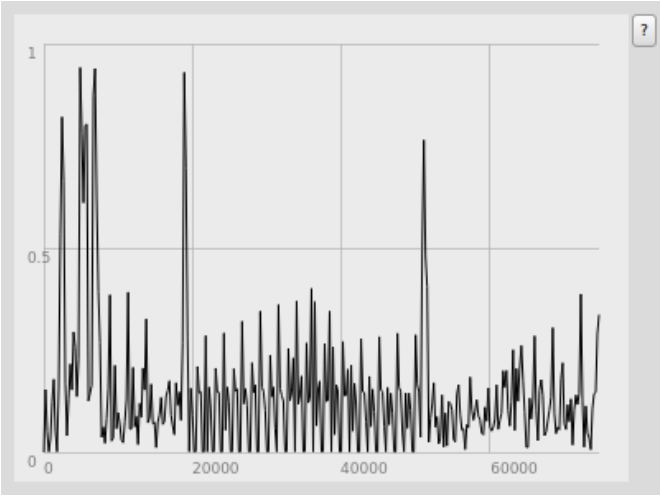

Figure 72: Similarity of 55hz's control stream with 55hz's audio stream in *Simulacra* plotted over time.

B Novelty Curve Peaks

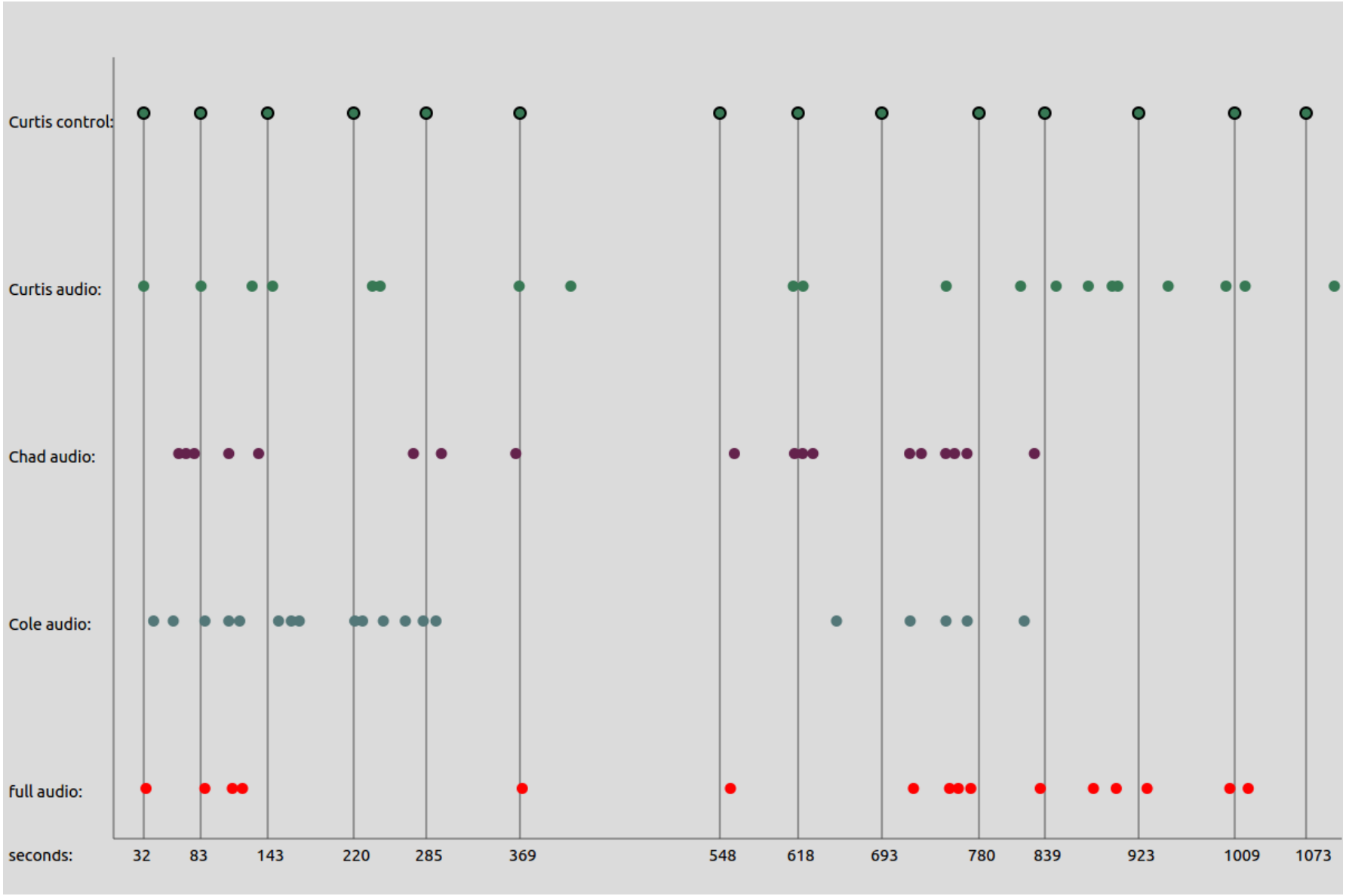

Figure 73: Comparisons of detected boundaries in Curtis McKinney's control signal, and boundaries for the four audio signals.

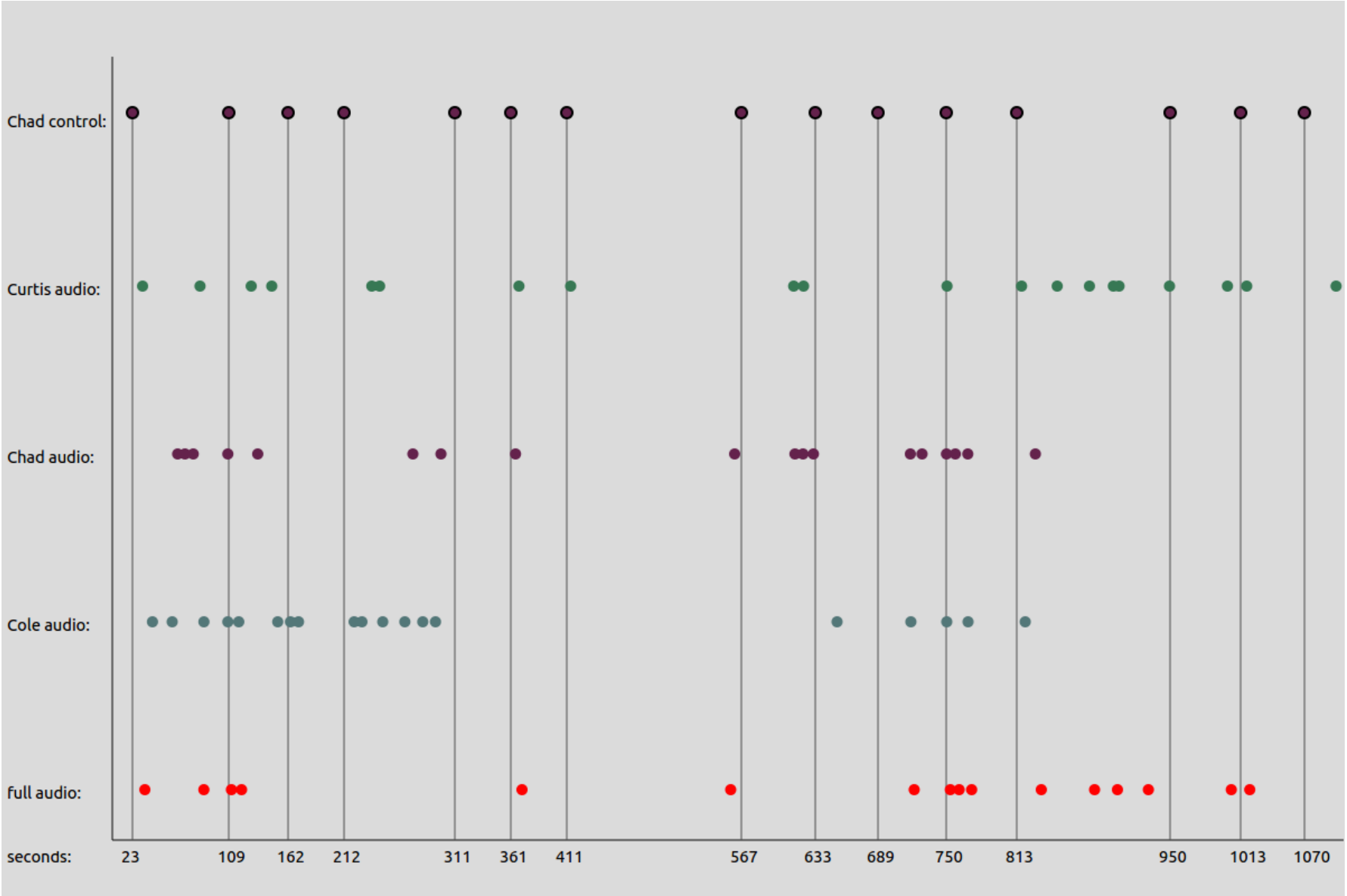

Figure 74: Comparisons of detected boundaries in Chad McKinney's control signal, and boundaries for the four audio signals.

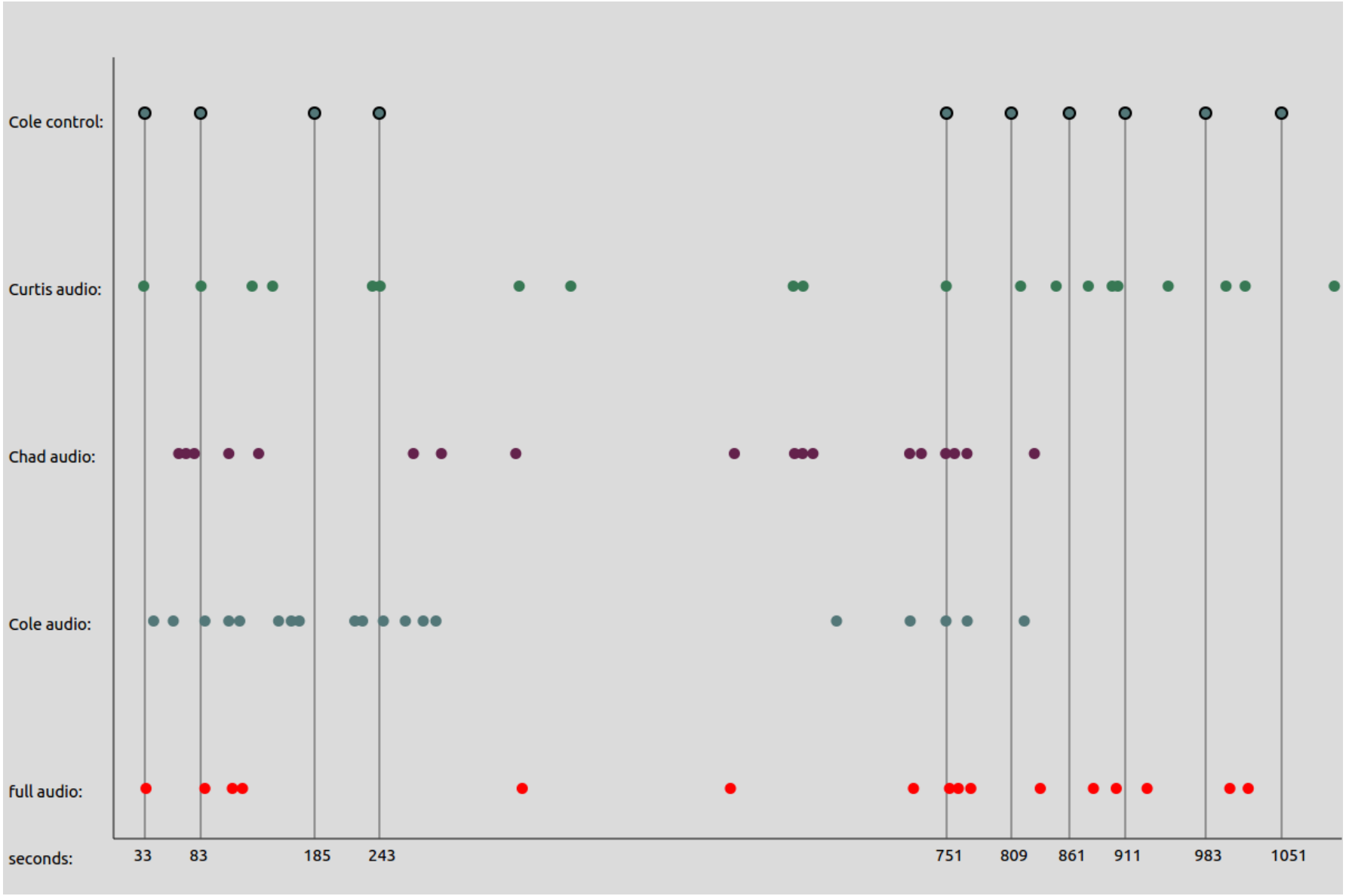

Figure 75: Comparisons of detected boundaries in Cole McKinney's control signal, and boundaries for the four audio signals.

C Texas A & M performance tweet logs

glitch lich @glitchlich: Texas Invasion Imminent. #glitchlich
Casey Gilbert @caseyhope53: 'MERICA > EUROPE #glitchlich
Marco Pisterzi @marcopisterzi: #glitchlich about to get things done right, here in Rudder Theatre. Come out and hang tho.
Taylor Phillips @tayphil8992: this concert is going to be like so not mainstream #glitchlich
Marco Pisterzi @marcopisterzi: No, only me is ready #glitchlich
Clara Turk @lifeonourtongue: Totally want to walk up to the MacBook Pro and pretend to be Cole. Also thanks guys now I want a shiner... #glitchlich
Casey Gilbert @caseyhope53: WHY DID IT LEAVE?! #glitchlich
Clara Turk @lifeonourtongue: #glitchlich awesome I happen to have just those items in my back pocket! Mary poppins pockets!
glitch lich @glitchlich: IS THIS REAL?? #glitchlich
crewxp2 @**crewxp2**: O.o #glitchlich
Clara Turk @lifeonourtongue: #glitchlich looks like I need to move countries!
Mason Morgan @Ramblings_Of: It tastes like a laser! #glitchlich
Easton Miller @SlothsAreDope: Interesting... #glitchlich
Justin @j_sizzle24: Eargasm #glitchlich
Taylor Phillips @tayphil8992: Frickin' laser beams #glitchlich
crewxp2 @**crewxp2**: Wubwubwub wub wub wubwub #glitchlich
Kevin Hernandez @Kevhernandez: #glitchlich I think we just had contact with alien life forms
Justin @j_sizzle24lich: #glitchlich you're a glitch lich
Clara Turk @lifeonourtongue: #glitchlich that's why you should have dogs. I'm not saying I'm just saying.
Casey Gilbert @caseyhope53: I don't know what's happening but I like it #glitchlich
Taylor Phillips @tayphil8992: This is making my reactors undulate #glitchlich
Kevin Hernandez @Kevhernandez: #glitchlich pikachu used thunder!
Mason Morgan @Ramblings_Of: #glitchlich is music from the future. The DYSTOPIAN FUTURE HAHAHA no but really it sounds like cats dying
Mason Morgan @Ramblings_Of: Whoa #glitchlich you got some music in my noise
Clara Turk @lifeonourtongue: #glitchlich only if you grab us all one!
Justin @j_sizzle24lich: #glitchlich I'm sweating and haven't even popped a Molly woo
Mason Morgan @Ramblings_Of: #glitchlich hey guys I heard a rumor that y'all like turtles that for real? Play some turtle music
Taylor Phillips @tayphil8992: Tron 2 soundtrack? #glitchlich
Kevin Hernandez @Kevhernandez: #glitchlich did we log in yet #dialup

Clara Turk @lifeonourtongue: #glitchlich TAKE US TO CANDY MOUNTQIN CHARLIE! YEAH! CANDY MOUNTAIN!

Justin @j_sizzle24lich: LVL UP #glitchlich

Marco Pisterzi @marcopisterzi: #glitchlich are turtley enough for the turtle club.

Clara Turk @lifeonourtongue: #glitchlich No disassemble number five! #short-circuit

Mason Morgan @Ramblings_Of: In the future, music will be dudes messin with computer trackpads and talking bout turtle lovin. Like, explicitly. Nasty stuff #glitchlich

Justin @j_sizzle24lich: EXPERT MODEEEEEEEE #glitchlich

Marco Pisterzi @marcopisterzi: I think megaton is taking out New York. #glitch-lich

Analicia @aggieana14: Should I tell you I have a pet turtle now or later? #glitch-lich

Kevin Hernandez @Kevhernandez: #glitchlich ima FIRIN' MAH LAZARRRRR!!!!!

Taylor Phillips @tayphil8992: Pretty sure y'all just broke the Matrix #glitchlich

crewxp2 @**crewxp2**: Its dying.... #glitchlich

Kevin Hernandez @Kevhernandez: #glitchlich awkward silence lol

Mason Morgan @Ramblings_Of: #glitchlich someone should start scatting to this

Clara Turk @lifeonourtongue: #glitchlich but I am le tired! Then take a nap. THEN FIRE THE MISSLES!

Justin @j_sizzle24lich: Wonderful weather we're having #glitchlich

Kevin Hernandez @Kevhernandez: #glitchlich it's sounds like contra on the NES

Mason Morgan @Ramblings_Of: #glitchlich I don't see it going over well if they played this for Super Bowl halftime. Now THAT'D be a power outage

Easton Miller @SlothsAreDope: I didn't know y'all did the screeches on the Prometheus trailer.#glitchlich

Taylor Phillips @tayphil8992: #glitchlich

Clara Turk @lifeonourtongue: #glitchlich crazy sample names ftw!

Taylor Phillips @tayphil8992: #glitchlich would you rather fight 100 horse-size ducks or a thousand duck-size horses?

Easton Miller @SlothsAreDope: Heh redditors #glitchlich

Justin @j_sizzle24lich: 2 CHAINZZZZZZZZ #glitchlich

Clara Turk @lifeonourtongue: #glitchlich campfire crackles make me want s'mores!

Clara Turk @lifeonourtongue: #glitchlich s'more sick beats that is!

Kevin Hernandez @Kevhernandez: #glitchlich Mr. Krabs: Bee-boo-boo-bop, boo-boo-bee-bop? Not bee-boo-boo-beep? Bop-bee-boo-boo-bop?

Taylor Phillips @tayphil8992: #glitchlich MAKE A MOLECULE! Or maybe a venn diagram of cats?

Justin @j_sizzle24lich: #glitchlich did y'all do the score for Inception too?
Clara Turk @lifeonourtongue: #glitchlich followers be like who the heck is glitch lich? #freePR#glitchlichtwitterstreetteam?
Brent Phelps @BrentPhelps: #glitchlich BUBBLES!!!!! my bubbles...
Clara Turk @lifeonourtongue: In case anyone's wondering #glitchlich is the friendly neighborhood Spider-Man but transplanted into a shark's body with lasers for eyes!
crewxp2 @**crewxp2**: Where is your fourth member? #glitchlich
Kevin Hernandez @Kevhernandez: #glitchlich is mayonnaise an instrument?
Clara Turk @lifeonourtongue: My wrong! GLORIOUS FRICKEN LAZRZ! #glitchlich
Justin @j_sizzle24lich: Go home music, you're drunk #glitchlich
Mason Morgan @Ramblings_Of: Hang on babe, lemme put on some mooood music. *puts on#glitchlich , the world explodes*
Analicia @aggieana14: #glitchlich beer time?
Nicky McMurrer @Nicksta_: #glitchlich Have y'all ever all performed together? Would that effect connect probs or clog up the bandwidth of the venue?
glitch lich @glitchlich: Thank you Texas people and lasers and things, it was definitely worth losing a Saved by the Bell lunch box #glitchlich

D Mute Magazine performance chat logs

2013-Apr-13:16:57:02.382341 : **casiosk1**: Logging into Simulacra
2013-Apr-13:16:58:24.670017 : **55hz**: Logging into Simulacra
2013-Apr-13:16:59:57.590103 : **octopian**: Guys there?
2013-Apr-13:16:59:57.750048 : **55hz**: I am
2013-Apr-13:17:00:04.761114 : **casiosk1**: in
2013-Apr-13:17:00:16.462263 : **octopian**: Ready?
2013-Apr-13:17:00:21.992353 : **55hz**: yeah
2013-Apr-13:17:00:24.769069 : **octopian**: ok going in
2013-Apr-13:17:00:30.946541 : **octopian**: Logging into Simulacra,
2013-Apr-13:17:00:36.109276 : **octopian**: in
2013-Apr-13:17:00:40.022928 : **55hz**: in
2013-Apr-13:17:00:47.859262 : **casiosk1**: just a sec
2013-Apr-13:17:00:54.927144 : **octopian**: ok
2013-Apr-13:17:01:12.672217 : **casiosk1**: ok, cool
2013-Apr-13:17:01:21.290302 : **casiosk1**: I set it up to where it's not drawing the graphics
2013-Apr-13:17:01:27.017600 : **casiosk1**: so I can at least hear shit correctly
2013-Apr-13:17:01:29.929245 : **octopian**: hahah ok
2013-Apr-13:17:01:33.890137 : **casiosk1**: I'm ready now
2013-Apr-13:17:01:37.193367 : **octopian**: chat is up, are you ready
2013-Apr-13:17:01:41.625601 : **casiosk1**: ready
2013-Apr-13:17:01:42.375271 : **55hz**: ready
2013-Apr-13:17:01:50.084044 : **octopian**: ok
2013-Apr-13:17:01:54.746151 : **octopian**: ill start
2013-Apr-13:17:02:00.503196 : **casiosk1**: ok
2013-Apr-13:17:02:26.129818 : **casiosk1**: mangling
2013-Apr-13:17:02:48.294086 : **55hz**: whadaya know, I'm working now
2013-Apr-13:17:02:49.902512 : **55hz**: =)
2013-Apr-13:17:02:57.857936 : **casiosk1**: cool
2013-Apr-13:17:03:10.984594 : **casiosk1**: Cole throw in that manifold synth
2013-Apr-13:17:03:13.823965 : **octopian**: who wants to bring in the calabi
2013-Apr-13:17:03:16.243350 : **octopian**: ok
2013-Apr-13:17:03:35.544290 : **casiosk1**: that would be the incorrect synth
2013-Apr-13:17:03:36.402441 : **casiosk1**: lol
2013-Apr-13:17:03:39.450160 : **octopian**: fail
2013-Apr-13:17:03:41.941728 : **octopian**: haha
2013-Apr-13:17:03:44.466787 : **casiosk1**: not that
2013-Apr-13:17:03:50.794936 : **octopian**: curtis do it
2013-Apr-13:17:04:02.610302 : **casiosk1**: there we go
2013-Apr-13:17:04:03.762422 : **55hz**: too similar of names...

2013-Apr-13:17:04:11.112802 : **casiosk1**: haha
2013-Apr-13:17:04:34.049274 : **casiosk1**: Chad, buffer overflow
2013-Apr-13:17:04:42.152230 : **casiosk1**: How is London?
2013-Apr-13:17:04:49.810225 : **casiosk1**: I may slightly miss that place
2013-Apr-13:17:04:51.226454 : **casiosk1**: haha
2013-Apr-13:17:04:54.416565 : **octopian**: going well how about you
2013-Apr-13:17:05:09.616275 : **casiosk1**: Drinking a beer, so good
2013-Apr-13:17:05:17.356890 : **55hz**: it's raining in Colorado
2013-Apr-13:17:05:24.783632 : **55hz**: I'll be getting beer after this
2013-Apr-13:17:05:37.519511 : **octopian**:
2013-Apr-13:17:05:39.846092 : **octopian**:
2013-Apr-13:17:05:50.553433 : **octopian**: hey crowd people hows it going out there
2013-Apr-13:17:06:22.480844 : **octopian**: bring in the countdow
2013-Apr-13:17:06:25.940379 : **octopian**: ok there it is
2013-Apr-13:17:06:29.760680 : **casiosk1**: bringing int he quantum
2013-Apr-13:17:06:31.352266 : **casiosk1**: lol
2013-Apr-13:17:06:57.333944 : **55hz**: wob wob wob wob wob
2013-Apr-13:17:07:32.456304 : **casiosk1**: ok chad, countdown
2013-Apr-13:17:07:36.996188 : **octopian**:
2013-Apr-13:17:07:53.250653 : **octopian**: done
2013-Apr-13:17:08:14.825463 : **octopian**: bring down the buffer
2013-Apr-13:17:08:15.025080 : **casiosk1**: ok, good
2013-Apr-13:17:08:16.446528 : **octopian**: yeah
2013-Apr-13:17:08:23.110535 : **casiosk1**: black noises out
2013-Apr-13:17:08:27.408750 : **casiosk1**: buffer over flow down
2013-Apr-13:17:08:33.085278 : **octopian**: done
2013-Apr-13:17:08:40.288700 : **casiosk1**: overflow is inverted remember
2013-Apr-13:17:08:58.895325 : **octopian**: ok
2013-Apr-13:17:09:10.202022 : **casiosk1**: so far so good
2013-Apr-13:17:09:32.097182 : **casiosk1**: cole ready up for a byte basher,but wait on starting it
2013-Apr-13:17:09:40.895552 : **55hz**: kk
2013-Apr-13:17:09:56.583903 : **casiosk1**: chad you're on shredding duty
2013-Apr-13:17:10:02.084910 : **octopian**: ok tell me when
2013-Apr-13:17:11:02.272993 : **casiosk1**: anticipation, haha
2013-Apr-13:17:11:03.768371 : **casiosk1**: ok
2013-Apr-13:17:11:04.559814 : **casiosk1**: now
2013-Apr-13:17:12:13.032058 : **casiosk1**: pretty epic, haha
2013-Apr-13:17:12:26.048029 : **casiosk1**: I think I'll mangle it up a bit
2013-Apr-13:17:13:12.823991 : **casiosk1**: fFSEFSpfs8efsf sEFfsefSefF2324@34243@3@3424@32

2013-Apr-13:17:13:17.997151 : **octopian**: asl;ckas;lckascl;kasc;l
2013-Apr-13:17:13:18.697383 : **casiosk1**: 34234234@44444334!!1dssdds
2013-Apr-13:17:13:21.392571 : **55hz**: hah
2013-Apr-13:17:13:38.192574 : **casiosk1**: kill the shredding
2013-Apr-13:17:13:57.399929 : **casiosk1**: and out with the byte bashing
2013-Apr-13:17:14:03.970948 : **55hz**: done
2013-Apr-13:17:14:26.689110 : **casiosk1**: cole bring in destructed melody
2013-Apr-13:17:14:43.650144 : **casiosk1**: out with the countdown
2013-Apr-13:17:14:54.417962 : **casiosk1**: we've got one too many destructed
2013-Apr-13:17:15:12.058483 : **55hz**: I only have one on me
2013-Apr-13:17:15:19.839614 : **casiosk1**: chad out with your destructed
2013-Apr-13:17:15:25.847563 : **casiosk1**: and countdown
2013-Apr-13:17:15:48.959561 : **casiosk1**: a little late on the dra there eh?
2013-Apr-13:17:15:50.256111 : **casiosk1**: hahah
2013-Apr-13:17:17:11.560161 : **casiosk1**: Is that melody changing pitches over there?
2013-Apr-13:17:17:15.842993 : **casiosk1**: also , the bass can come in now
2013-Apr-13:17:17:17.439479 : **casiosk1**: chad
2013-Apr-13:17:17:42.643999 : **octopian**: in
2013-Apr-13:17:17:58.695965 : **casiosk1**: I <3 Algorithms
2013-Apr-13:17:18:07.953348 : **casiosk1**: flocking = beats, right?
2013-Apr-13:17:18:13.617778 : **octopian**: or uh something
2013-Apr-13:17:18:20.432036 : **casiosk1**: Diamond Squared melodies?
2013-Apr-13:17:18:29.448348 : **casiosk1**: Genetic Algoraves?
2013-Apr-13:17:18:32.766101 : **octopian**: cellular groovamata#
2013-Apr-13:17:18:44.615833 : **casiosk1**: ok, out with the bass + drums
2013-Apr-13:17:20:19.510566 : **octopian**: dead things
2013-Apr-13:17:21:23.991813 : **casiosk1**: spectating
2013-Apr-13:17:22:53.143837 : **casiosk1**: Have we explained wtf was happening to the audience?
2013-Apr-13:17:22:56.399508 : **casiosk1**: haha
2013-Apr-13:17:25:17.607537 : **casiosk1**: While I was kidding before, this really is Diamond Square beats,
2013-Apr-13:17:25:18.559398 : **casiosk1**: haha
2013-Apr-13:17:30:59.153425 : **55hz**: lol
2013-Apr-13:17:31:48.951270 : **casiosk1**: I think our server really cares about me
2013-Apr-13:17:31:57.359145 : **casiosk1**: It keeps send me the same message over and over again:
2013-Apr-13:17:32:02.039173 : **casiosk1**: areYouAlive?
2013-Apr-13:17:32:25.834333 : **55hz**: it wants to know if it's killed you yet

2013-Apr-13:17:32:33.977770 : **octopian**:
2013-Apr-13:17:32:36.990022 : **octopian**: done
2013-Apr-13:17:32:46.362663 : **casiosk1**: cool.....
2013-Apr-13:17:32:49.255136 : **casiosk1**: how did it go?
2013-Apr-13:17:32:50.391045 : **casiosk1**: haha
2013-Apr-13:17:33:05.266346 : **octopian**: GOOD JOB GUYS!
2013-Apr-13:17:33:07.087210 : **casiosk1**: Simulacra went well
2013-Apr-13:17:33:08.733899 : **octopian**: THANK YOU EVERYONE!
2013-Apr-13:17:33:12.406920 : **casiosk1**: Shoggoth?
2013-Apr-13:17:33:14.926580 : **casiosk1**: haha
2013-Apr-13:17:33:17.358831 : **casiosk1**: Thanks for having
2013-Apr-13:17:33:18.574919 : **casiosk1**: us
2013-Apr-13:17:33:20.599030 : **casiosk1**: that's the show
2013-Apr-13:17:33:32.348201 : **55hz**: now time for bier!
2013-Apr-13:17:33:52.031112 : **casiosk1**: Always time for beer
2013-Apr-13:17:34:04.810484 : **55hz**: well, since I'm out
2013-Apr-13:17:34:08.986350 : **55hz**: now I can go get moar
2013-Apr-13:17:34:19.558938 : **casiosk1**: Get on that