Keele University

Declaration Part 1. To be bound in the thesis

SUBMISSION OF THESIS FOR A RESEARCH DEGREE

Part I. DECLARATION by the candidate for a research degree. To be bound in the thesis

Degree for which thesis being submitted Doctor of Philosophy in Music

Title of thesis An evaluation of digital interfaces for music composition and improvisation.

This thesis contains confidential information and is subject to the protocol set down for the submission and examination of such a thesis.

NO

Date of submission 23 Dec. 2015 Original registration date 26 Sep. 2011
(Date of submission must comply with Regulation 2D)

Name of candidate Konstantinos Vasilakos

Research Institute Humanities Name of Lead Supervisor Mike Vaughan

I certify that:

(a) The thesis being submitted for examination is my own account of my own research
(b) My research has been conducted ethically. Where relevant a letter from the approving body confirming that ethical approval has been given has been bound in the thesis as an Annex
(c) The data and results presented are the genuine data and results actually obtained by me during the conduct of the research
(d) Where I have drawn on the work, ideas and results of others this has been appropriately acknowledged in the thesis
(e) Where any collaboration has taken place with one or more other researchers, I have included within an 'Acknowledgments' section in the thesis a clear statement of their contributions, in line with the relevant statement in the Code of Practice (see Note overleaf).
(f) The greater portion of the work described in the thesis has been undertaken subsequent to my registration for the higher degree for which I am submitting for examination
(g) Where part of the work described in the thesis has previously been incorporated in another thesis submitted by me for a higher degree (if any), this has been identified and acknowledged in the thesis
(h) The thesis submitted is within the required word limit as specified in the Regulations

Total words in submitted thesis (including text and footnotes, but excluding references and appendices) …22.525……

Signature of candidate  Date 26 Jan. 16……

Note
Extract from Code of Practice: If the research degree is set within a broader programme of work involving a group of investigators – particularly if this programme of work predates the candidate's registration – the candidate should provide an explicit statement (in an 'Acknowledgments' section) of the respective roles of the candidate and these other individuals in relevant aspects of the work reported in the thesis. For example, it should make clear, where relevant, the candidate's role in designing the study, developing data collection instruments, collecting primary data, analysing such data, and formulating conclusions from the analysis. Others involved in these aspects of the research should be named, and their contributions relative to that of the candidate should be specified (*this does not apply to the ordinary supervision, only if the supervisor or supervisory team has had greater than usual involvement*).

An evaluation of digital interfaces for music composition and improvisation

Konstantinos Vasilakos

For the degree of Doctor of Philosophy in Music

March 2016

Keele University

Abstract

This PhD reports research on current representative performance paradigms using various interfaces for real time interaction with computer-based musical environments. Each device was selected to cover a particular range of interfaces. Research covers the following areas: hardware interfaces (tangible & game devices); live coding; optical devices, and hardware prototyping.

The projects highlight affordances, comparative strengths and weaknesses, and provide suggestions for further improvements for each paradigm. Particular focus is given to the importance of mapping. Each project comprises corresponding software that was developed to facilitate each performance paradigm.

The work is not intended to provide an exhaustive evaluation of the technology used in this research; instead, it aims to examine its feasibility for artistic and musical context. The outcomes of the examinations include a series of musical performances employing improvisation as the basis for composition. These paradigms are examined in a live context as well as fixed media that uses material originating in live performances.

Contents

List of figures

List of tables

Submitted media

Study II (08’26”)	2011
The Song Has Sung (07’32”)	2012
Glitchy (14’26”)	2013
Formations (08’14”)	2013
Blind date (12’26)	2013
It All Starts with Noise (09’00”)	2013
It All Ends with Noise (13’08”)	2013
Ataraxia (09’34”)	2014
Power of People (excerpt 02’50”)	2014

Submitted software

Glitchy (preamble code*)

Blind date (preamble code* and Fair Algo: projection utility)

Formations (preamble code*)

PoP

Stay On This Gesture

Greap

*Preamble code acts as a basis for live coding performance.

Acknowledgements

I would like to express my special appreciation to my supervisors, professors Mike Vaughan and Rajmil Fischman for their constructive criticism throughout my PhD. I would like to thank you for allowing me to grow as a researcher, an independent thinker.

I would like to thank Angeliki, her help and encouragement have been a strong wind in my musical sails…

A special thanks to Mark Summers, Kostas Kartasidis, and Kostas Theocharis for their invaluable support.

Most importantly, thank you to my mother. Words cannot express how grateful I am for all sacrifices that she has made on my behalf.

Overview

This thesis comprises a portfolio of original compositions created during my PhD project. The text that follows is a commentary on the compositions, elaborating on aesthetic and technical concerns that underpin their creation.

During my PhD research I investigated representative approaches of performance practices aided by diverse interfaces. The selected interfaces each represent a particular family of hitherto available devices. It is commonly known that appropriate mapping[1] is an integral part of performing with interfaces and computers (Winkler, 1995; Rovan, Wanderley, Dubnov and Depalle, 1997; Hunt, Wanderley and Kirk, 2000; Hunt and Wanderley, 2002; Hunt, Wanderley and Paradis, 2003), therefore particular attention was given to this topic. Mapping is the part that lies between the interface and the musical environment and connects the performer with the latter.

I carried out individual projects using available hardware suitable for musical creation integrated with custom made software that explored various mapping strategies. During these projects I investigated affordances of each interface as well as strengths and weaknesses for each performance practice. Although the projects often included the examination of technology, the objectives of my research are not oriented towards an exhaustive technical appraisal, but rather to evaluate the feasibility of each performance paradigm in terms of musical and artistic use.

Primary studies included the evaluation of live coding as a performance paradigm, where the performer interacts through the implementation of generative data in order to communicate with the sound synthesis software. One of the strengths of live coding is that the relationships between human agency and sound creation are established on the fly. Contrary to the norm of developing a system in advance of the performance, 'leaving only

[1] 'Mapping refers to the liaison or correspondence between *control parameters* (derived from *performer actions*) and *sound synthesis parameters*.' (Hunt, Wanderley and Kirk, 2000, p.209)

the finished program to “go live” ’ (Wang and Cook, 2004, p.138), live coding offers the ability to explore various levels of affordances with the musical environment where the relationships between the performer and sound are established in real time. In other words, ‘Code becomes a real time, expressive instrument’ (Wang and Cook, 2004, p.138). Wilson et al. (2014, p.63) state that:

> Live coding does not free us from the limitations of our “instruments” (these instruments are arguably just more flexible and less specified; thus, not necessarily a strength). It does afford us, however, the opportunity to avoid the narrowly conceived instrumentality that typifies much interface design for live electroacoustic performance, particularly within the context of laptop ensembles.

The next project investigated in the field of hardware prototyping. Using the Arduino[2] micro-controller board I developed an interactive installation using environmental sensors to control the parameters of a sound synthesis system. Objectives of this project included the participation of the audience to interact with the parameters of the sound synthesis environment, as opposed to my former investigation into single performer interaction: in other words I attempted to expand my focus on mapping and use data that is generated by more than one performer. In addition to the sensory capabilities, the system uses the internal microphone to analyse diverse information of the produced sound in real time, this information is then coupled to other control parameters of the system. The system also acts as a model for wider interpretation of sonification data.

Further studies included the investigation of performing with tangible interfaces, for example a Nintendo Wii Remote (Wiimote)[3] video game controller and joysticks. In recent years many artists, researchers, and digital musicians focused on the development of software components (plug-ins, programming objects, and third party applications) to

[2] See https://www.arduino.cc
[3] See http://wiibrew.org/wiki/Wiimote

integrate the Wiimote with many programming environments for musical creation (Paine, 2009, p.148).

Given the nature of the sensors that the Wiimote offers, such as accelerometers and orientation sensors to track physical gestures, the performer can create mapping strategies that enable physicality and a causal relationship with the sound. Hyper-instruments also fall in this category, as technological similarities can be seen between the sensory capabilities of these and of the Wiimote. Devices such as the Wiimote are made for game interaction, thus offering robust communication and stability of the gestural data that can be useful during a live performance.

Finally, the last chapter investigates the use of optical interfaces including wireless body sensing. For this project I used a Leap Motion[4] sensor. In considering expressivity as the primary aim of this project, mapping being the means to achieve this, particular interest was given to the mapping strategies that are informed by gestural metaphors to foster transparency of sound manipulation (Fels, Gadd and Mulder, 2002; Wessel, Wright and Schott, 2002; Fischman, 2013).

Findings and conclusions are presented at the end of each chapter and summary conclusions are presented at the end of this commentary.

All software was developed using the open source SuperCollider programming language (McCartney and Others, 2014).

[4] See Leap Motion https://www.leapmotion.com

Introduction

The most important question that a digital instrument tackles is interactivity. This interactive paradigm has undergone extensive examination over the past years in the field of interactive computer music.

> … an indeterministic instrument outputs a substantial amount of unpredictable information relative to a performer's controls. In working with such an instrument, a performer shares control of the music with algorithms as virtual co-performers such that the instrument generates unpredictable information to which the performer reacts, the performer generates control information to which the instrument reacts, and the performer and instrument seem to engage in a conversation. Interaction means “mutually influential”. Since the instrument is influenced by the performer's controls, and the performer is influenced by the instrument’s output, I have called such instruments “interactive instruments”. (Chadabe, 2002, p.2)

According to Chadabe, interactivity means a mutual intervention of performer and the machine. Here the performer is building a relation; communicating his or her ideas to the computer, in turn the computer transforms it in sound. To facilitate interactivity some important conditions need to be established. These are created when one devises the connections between the human gestures and the parameters of the synthesis engine known as mapping (Hunt, Wanderley and Kirk, 2000, p.209). At this point an important distinction should be made. The sound producing body is separate from the interaction interface in contrast with the traditional instrument in which the interaction device is embodied with the sounding object (Miranda and Wanderley, 2006, p.4). This disjunction creates the need for mapping, a vital part of a digital instrument. Through mapping we establish the relationships between the external device and the control inputs of the musical environment; hence our gestures are translated to sound via these notional

connections. Additionally, the appropriate mapping enhances expressivity of a performance environment (Winkler, 1995; Rovan, Wanderley, Dubnov and Depalle, 1997; Hunt, Wanderley and Kirk, 2000; Hunt, Wanderley and Paradis, 2002; Hunt and Wanderley, 2002).

Since the implementation of real time devices in electronic music it is possible to interact with technology in the same way that a musician interacts with his or her instrument and influence the output dynamically. Personally, I am interested in using improvisation and real time interactivity working with electronic media. Improvisation in electroacoustic music could be viewed as a process of constant evaluation by the performer of the resulting sound and responding back to it. A sort of cause and effect situation. Pressing (2001, p.130) states that:

> To begin with, improvisation (or any type of music performance) includes the following effects, roughly in the following order
>
> 1. complex electrochemical signals are passed between parts of the nervous system and on to endocrine and muscle systems
>
> 2. muscles, bones, and connective tissues execute a complex sequence of actions
>
> 3. rapid visual, tactile and proprioceptive monitoring of actions takes place
>
> 4. music is produced by the instrument or voice
>
> 5. self-produced sounds, and other auditory input, are sensed
>
> 6. sensed sounds are set into cognitive representations and evaluated as music
>
> 7. further cognitive processing in the central nervous system generates the design of the next action sequence and triggers it.
>
> - return to step 1 and repeat –

Pressing argues further that the differences between a fixed performance and improvisation are in step 6 and 7, with ‘important differences’ in step 3 (Pressing, 2001, p.130).

To support and maintain this interactive process an interface must be stable and provide accurate data generated by the performer. The environment then must be able to implement the mapping that will allow this interactivity to take place and will enable the performer to experiment with its affordances in real time.

1. Live Coding

1.1. Introduction

Live coding is the real time building or modification of the source code of software that produces either music or visuals. Live coding uses the act of programming in a live context. The performer writes algorithms in a programming environment whereby they interact with the produced sound by altering the code on the fly. This live process is usually projected onto a screen to let the audience follow the changes and the development of the code in conjunction with the musical outcome. Live coding is a paradigm of performance in computer music, but not limited to this genre's characteristics, as it extends to other electronic music examples which use the computer as their main medium such as electronica, glitch and rave.

In contrast with other contemporary electronic music genres such as the acousmatic tradition, live coding performance shows the audience the moment when the music is being created. Through the projection of the code, which is shown on the screen during a performance, the listener is able to watch the performer's manipulations and to presume how his or her actions are influencing the overall sound. From this perspective, the audience can understand the presence of the live performer through the code and its development as the musical discourse unfolds. This is something that in acousmatic music, for example, is absent, although this absence is an essential characteristic of the acousmatic tradition.

In the post digital era the computer is integral to the arts. Computers are at the core of contemporary electronic music and are fundamental in performance practice. Hugill (2012, p.5) states that:

> A digital musician is one who has embraced the possibilities opened by new technologies, in particular the potential of the computer for exploring, storing,

> manipulating and processing sound, and the development of numerous other digital tools and devices which enable musical invention and discovery. This is the starting point for creativity of a kind that is unlike previously established musical practice in certain respects, and requires a different attitude of mind.

Computer or digital musicians and laptop artists employ the computer as the main medium when creating and manipulating sonic material. Real time sound synthesis, signal processing and sound diffusion are realised through the computer. Digital musicians amalgamate various roles when creating music. Some of the tasks they need to fulfil are computer programming and software development as well as composition and performance of their works, blurring the borders and distinctions between technical, oriented practitioners and artists. One could argue that live coding leads to the collapse of the separation between programmers who designed the musical software and artists who for many years were considered as two distinct entities.

> The performers are almost exclusively the same people who have designed and written the software instruments in countless hours. The traditional separation into composer, instrumentalist and instrument maker is not valid for them anymore. And since these people spend most of their time at the design of their instruments (which, due to the power of general purpose machines, are not “just” instruments but can also hold scores and algorithms, which will eventually form the “composition”), it is only logical that this is the field where they gain the greatest skill and virtuosity: the design of algorithms and their implementation in source code. (Zmölnig and Eckel, 2007, p.295)

Live coding embodies live programming but its essential priority is music; this is illustrated by the fact that live coders are music practitioners and not necessarily programmers. The first known performance of live coding was realised by Ron Kuivila who was a sound artist

with a background in live electronics and circuit bending at the Studio for Electro Instrumental Music (STEIM) in Amsterdam in 1985 (Blackwell and Collins, 2005, p.122).

In live coding programming becomes the instrument of the performer and serves as an extension of his or her mind and body just as the bow of a violinist serves as an extension of his or her hand. Live coding is a post-digital art form, where the improvisation of algorithms and code is the core of its ethos. The following list is a selection of excerpts from the manifesto of Toplap[5] to illustrate this concept further:

> We demand:
>
> - Programs are instruments that can change themselves.
>
> - Live coding is not about tools. Algorithms are thoughts. Chainsaws are tools. That's why algorithms are sometimes harder to notice than chainsaws. (Toplap, 2010)

1.2. Routes

According to Nilson (2007, p.112) live coding performances can be viewed as a process of problem solving that is similar to the early days of mathematicians who were challenging other mathematicians to solve complicated and unsolved problems in public.

> Some Authors [...] argue that the tournament on cubic equations between the two Italian mathematicians Nicolo Fontana Tartaglia and Antonio Maria Fior about 1539 might be considered an early Live Coding performance (albeit it lasted for several weeks and is thus not directly comparable to today's short-lived performances). (Zmölnig and Eckel, 2007, p.295)

It can be argued that live coding emerges out of the tradition of circuit bending, as both follow the idea of hacking sound modules in real time. Circuit bending, the process of

[5] A website dedicated in live coding. See http://toplap.org

hacking an electronic component, is characterised by the act of someone trying to modify its original construction. This has a relationship to the modification and hacking of the source code of a running application. During this process there may be a sort of detachment with the sonic output, arising from the interaction of the running algorithms as a consequence of being tweaked. The relationship between circuit bending and live coding is strengthened when comparing the ethos and the procedural circumstances of live coding and circuit bending practices, where improvisation is the predominant mode of performer interaction. In both cases the performer is focused on the live modification of sound modules in pursuit of musical artistry.

Another parallel that can be drawn between live coding and other earlier musical paradigms is to the genre of live electronics, for example, the work of Musica Elettronica Viva who put an emphasis on the hacking of electromechanical devices and other non-musical objects on stage. (Manning, 2004, p.161)

A characteristic work of MEV is the realisation of Variations IV (1966), which was composed by John Cage, and it was scored for instruments and other unconventional sounding objects such as transistor radios, a Volkswagen bus, a garden hose and wooden chairs. MEV also used a range of more traditional electronic devices including tape-delay systems, contact microphones, Moog synthesiser modules, and alpha waves decoders. The idea of hacking hardware sound modules on stage, as demonstrated by these examples, is closely connected with the idea of modifying the software source code in real time, using improvisation as the main driving force. From this perspective, live coding can be viewed as a logical extension of the live electronics tradition.

The works of MEV focused more on the individual motivations of the players, rather than the interpretation of an overall plan (Manning, 2004, pp.157–162). MEV's performances illustrate the idea of a collective improvisation act rather than prioritising the performer as an individual. Similarities in approach can be seen in the methodology adopted by many of

the current laptop ensembles who make use of live coding as their performance practice, to name but a few, Benoit and the Mandelbrots, Cybernetic Orchestra, Birmingham Ensemble for Electroacoustic Research (BEER), and the PowerBooks Unplugged.

1.3. Environments for live coding

There are many environments used for live coding; some of them include the broader and well known environments for sound synthesis and algorithmic composition in real time, whereas others have a more idiomatic nature created by the artists themselves, and reflect their idiosyncratic preferences. Some of these environments are text oriented languages such as SuperCollider, and Chuck (Wang, 2008) as well as the graphical environments such as the open source environment Pure Data (Puckette, 1996), and its commercial version Max/MSP (Cycling74, 1998). Typical idiomatic languages include Impromptu (Sorensen, 2005), and IXI Lang (Magnusson, 2011). For an extensive list of programming environments as well as events and topics covering live coding see Toplap's website[6].

1.4. Performing with BEER

My personal work includes an affiliation with BEER. The ensemble researches collective improvisation using live coding and network performance. Personal experience with the ensemble over the past three years has proven very helpful in regard to developing skills in live coding within the context of an ensemble, as well as raising my awareness of collaborative and networked performance.

BEER repertoire includes original compositions, often with bespoke software developed for specific pieces. Each piece comprises a set of interaction affordances, which are defined in the development of the software, the performers learning to improvise within these constraints. During the performance the members of the ensemble contribute by

[6] http://toplap.org/category/software/

modifying synthesis code in real time following a set of instructions that come with each piece.

The ensemble embraces the ethos of free improvisation strategies ingrained in post-free jazz groups as well as the structural models found in John Zorn's musical piece named Cobra (1984) (Wilson et al., 2014, p.54).

Creative outcomes of this collaboration include a series of concerts[7] around the UK and rest of Europe, as well as the co-authorship of a paper published by the Computer Music Journal in a special edition on live coding (Wilson et al., 2014).

1.5. The many faces of code

1.5.1. Code as interface

The idea of the code as an interface is well demonstrated by the BEER ensemble. According to the ensemble 'live coding provides one fertile solution to the problem of interface design for musical performance, with rich implications for improvisational practice.' (Wilson et al., 2014, p.54)

The ensemble uses Julian Rohrhuber's Just In Time library (JITLib)[8] in SuperCollider. One of the fundamental ideas of live coding is that the performers can modify the source code of a program while it is running in order to manipulate or to intervene in the sonic outcome. With this in mind, the JITLib provides some additional enhancements (programming classes) for live coding that allow an intervention without the need to stop and re-evaluate after the modification of their content. This is vital for the improvisational discourse.

[7] A selection of concerts include: Network Music Festival, 2012, UK; Live Code Festival, 2013, Germany; Akou Festival, 2014, Greece; Brno, 2015, Czech Republic. A selection of recordings of live performances of BEER can be found at this link: https://soundcloud.com/beer-ensemble

[8] See http://doc.sccode.org/Overviews/JITLib.html

```
Pdef(\a,
Pbind(\instrument, \synth,
     \dur, Prand([4/4, 4/8, 4/16, 4/32], inf),
          \rate, Prand([0.2, 0.4, 0.6, 0.8, 1, 3], inf),
               \buf, ~sounds[0]
          );
     );
```

Figure 1.1 Manipulating synthesis parameters with patterns in SuperCollider.

Figure 1.1 shows a way to interact with a synthesiser in real time and intervene in the sound without interrupting the performance using patterns in SuperCollider. Particularly, the above example illustrates the coupling of synth parameters with patterns implementing a Prand that comes with SuperCollider. In this example the *rate* variable of the synth is controlled by some random values that are defined as an argument inside the Prand.

Using the patterns the performer may achieve interactivity with the sound by modifying the enclosed values of a pattern while the streams run. A selection of patterns is available in SuperCollider, from which the performer can select either random or sequential data streams. A combination of patterns is also possible by embedding patterns within other patterns.

Once the patterns are defined the performer is able to map these streaming data to any parameter and control the variables of a synthesiser dynamically. Consequently, it is hard to ignore that this is similar to the use of a hardware interface that one would employ to manipulate the parameters of a digital instrument(s) externally. De Campo et al. (2007, p.4) states that:

> While such a phrase evolves, one can listen to it, read its code, make changes to it, and replace it with a new variant at an appropriate moment. This can be seen as a continuation of motivic development as in western classical music, as a form of genetic algorithms with the performers' aesthetic preferences as fitness functions, or as multi-path looped version of the surrealist technique cadavre exquis.

1.5.2.Code as communication medium

Whether the code is the medium to describe the sound that a composer wants to design through a programming environment or acts as the interface for creating musical phrases and controlling musical extents, it can also act as a dynamic medium that enables the performers to communicate with each other. For example, using the code as a documented body for further introspection. De Campo et al. (2007, p.3) states that: 'The music we play is highly communicative group improvisation, based on a body of code created in rehearsals and concerts (collected later), rewritten on the fly, and communicated back and forth continuously.'

The code can be seen as the creative link between the performers of a laptop ensemble. In the case of BEER for example, the code is available through the communication platform called Utopia (Wilson and de Campo, 2013) that the ensemble uses to communicate with each other. The code is available to share, edit, and send back through this platform. The coherence of the group is strengthened by the ability of the system to devise dynamic connections between the performers implemented in each piece specifically. For example, triggering cues of synthesis nodes and controlling the general tempo of the ensemble from a master computer. The coders can copy, change or evolve the code of the others dynamically on their computers, enacting collaboration between each other, and giving consistency to the overall musical outcome. Through this communication the group is able to explore structural attributes and characteristics during network performance. Finally, the communication between the members of the group is aided by a chatting window provided by Utopia; through chatting the members are able to communicate whether they are going to start and stop or suggest further directions of the performance.

In this context the group not only enables the communication between the performers' actions during the performance but also enacts communication characteristics similar to the collective improvisation found in jazz ensembles, for example, using eye contact or

head nodding to let the others know that their solo part is ending or to return to the theme of the piece, but in a much more explicit manner.

1.6. Live coding and the audience

Live coding is a growing paradigm of computer music performance. Live coding performances are often witnessed at academic events (conferences, symposiums etc.) as well as in smoke-filled noisy bars and festivals. The audience usually consists of people who come from both programming and non-programming backgrounds, some of whom have little understanding in the process. 'It was the first live coding performance I ever seen, and that was live coding yeah? It is quite bewildering watching it. I can't say I understand in the slightest what was going on.' (McCallum and Smith, 2011)

In response to the above, I believe that while watching a live coding performance there is no need to understand or to acknowledge the programming language. For example, when people listen to a large ensemble performing a particular classical work, they are not expected to know the specifics of what is going on, but they can still thoroughly enjoy classical concerts. The following statement is included in the Toplap's manifesto: 'It is not necessary for a lay audience to understand the code to appreciate it, much as it is not necessary to know how to play guitar in order to appreciate watching a guitar performance.' (Toplap, 2010)

De Campo et al. (2007, p.3) state that:

> A pianist in classical setting makes decisions on details that bring out the structure and the subjective emotional meaning of a piece; the "text" of the composition itself is usually not touched. Even if the pianist's hands are not seen, an audience can follow and appreciate these aspects quite well.

Taking this into account, live coding is not aimed exclusively at audiences with knowledge of programming. As in every kind of musical performance, regardless of its genre, the

intention of the audience is not to examine the technicalities of what they hear and to analyse it, but to enjoy the quality of the music itself.

The live projection of the code reveals the theatricality created by the dynamic modification of the code while performing. The audience may relate the changes between the sound and the code accordingly. In addition to the projection of the code as a way to enhance theatricality while performing it is also crucial to consider the location of the performers in respect to their positioning and in respect to the audience.

To reflect on this topic I will provide an insight from a personal experience while giving a performance with the BEER ensemble in a club in Corfu[9]. For this concert we decided to place our laptops in the middle of the room where the audience would sit around us creating a circle and facing each other. The loudspeakers were placed behind the audience creating a circle. Each performer had two speakers placed behind him/her accordingly. As the performance evolved I looked around the room and thought: if a person entered the room in that particular time s/he would experience mystical scene comprised three persons who were focusing solely on their laptops creating sounds; the audience tried to decode (or not) the actions of the performers and relate to the music they were listening.

The drones and ambient sounds that the group was creating at that moment in conjunction with our location in the room and the way we were sitting enacted an atmosphere similar to a musical liturgy. As a result of the set up and the position of the audience in relation to the performers (i.e. immersing the latter) it enacted theatricality, and a sense of intuitive communication between the performers and the audience.

[9] The performance took place during the summer academy *Akou 2014* of the music department of the Ionian University in Greece.

1.7. Music composed with live coding

Working with live coding I have created a series of live performances and fixed media pieces that comprise material from live sessions. Material includes videos of the computer's screen recordings with code manipulation and its sound. Additionally some annotations were applied on the video in order to describe the process and explain how the modification of the code changes the state of the running software.

Glitchy[10] (2013) is a live performance created with SuperCollider using live coding. It was created to explore the idea of the real time modification of a running program. The software consists of the implementation of a sound sampler, which uses audio files stored in the computer's hard disk[11]. Instead of starting from a clean slate, I created some preamble code[12] through which I interact throughout the musical performance. This code sets out various strategies and includes the following:

1. *Rhythmic interaction based on a tempo clock*. This is achieved through the Implementation of a tempo-clock through which I can change the speed of the manipulation of the sound and vary the rhythm of the piece. Some other code implemented the allocation of the buffers for the audio samples.

2. *Manipulation of the sound in a higher level*. Implementation of post-production effects. In addition to the sampler some effects are added in order to manipulate the output of the sampler. The effects provide their own parameters that I can manipulate in real time.

3. *Real time mapping of generative processes with parameters*. Interaction with the sampler takes place through the implementation of patterns, and the performance

[10] A video with the full performance of the piece is included in the accompanying DVD and online at this link:
https://www.youtube.com/watch?v=jRC0kpEpPUc&list=PLMmfcbi0xjDlZXjTsmJ7jy5sAYYi6SCOG&index=2
[11] Included in the root folder of the project in the accompanying DVD.
[12] Included in the accompanying DVD.

mainly elaborates by tweaking them on the fly. The patterns are grouped in different sets of pattern definitions that contain the couplings between the parameters of the sampler and the corresponding patterns. During the performance I am able to start and stop these definitions as well as change their internal structure and control the form of the piece.

4. *Expansion of interaction by introducing new controls.* This is mainly achieved through the implementation of more couplings between the patterns and the parameters of the sound system; these parameters must exist and pre-defined to the synth before the performance using default values until they receive continuous control from the pattern definitions.

The performance evolves through the interaction with code in a fully improvisational way. There is no initial or pre-conceived idea of the piece or its structure. All decisions are made based upon the free manipulation of the sound and the attempt to keep a musical coherence within the improvisational discourse. Glitchy is compelling both from audience and performer point of view. The audience can visualise the implication of the code and its correspondence with the sound manipulation in real time. It also explores the idea of the code as interface, investigating the interaction affordances provided by this performance paradigm. My approach focuses in the mapping aspect and its musical implications during the performance showing the ability of live coding to create complex mappings without being constraint to fixed parameter mapping compared to other performance paradigms.

A screen and audio recording of the performance is provided. At the left of the screen is located the post window through which SuperCollider returns messages about the executed commands or errors that occur during the performance. Timings refer to the documented performance. At the beginning of the video the post window shows the

process of booting the server[13] of SuperCollider. At 00'50" I execute a line of code that loads the audio samples on the server, the paths and the names of the samples appear in the left of the post window. At 00'53" I run a tempo clock, through which I will be able to control the tempo of the piece. At 00'55" I run the code that is the implementation of the sampler and at 01'03" I load more samples. At 01'05 I run the first chunk of code that contains the patterns, which control the parameters of the sampler and its delay effect. At 01'09" I start the first pattern set, which causes the sound turn on. In the next seconds I evaluate and start the rest of the pattern sets. At 01'54" I introduce a new control parameter in the pattern definition. Note that this parameter already exists in the sampler. In this case this parameter is the *decay* argument of the delay effect, I set it initially to a value of 0.2. At 02'03" I substitute the value with a pattern that implements a random sequence of embedded values, thereby I experiment with this by inserting more values to be used by the random generator. At 03'07" I switch off one of the pattern definitions. In the next seconds I start it again. At 03'54" I restart them as well as I introduce new sounds and improvise substituting values that correspond to the parameters of the delay effect. As the performance evolves I am introducing new arguments by copy and paste instead of hard coding them all the time. At 05'29" I am slowing down the tempo by replacing its current value and I experiment with it until 06'08". The rest of the performance involves tweaking and replacing the values of the parameters of the sampler and the delay effect through the pattern definitions.

1.8. Mapping immutability and hybrid environments

Programming environments such as SuperCollider can allow the dynamic alteration of its running processes without the need to recompile or restart the software. Thus it is becoming increasingly difficult to ignore the desire for live tweaking of the mapping while performing with hardware interfaces. Following this performance strategy the performer is

[13] Server is the synthesis engine of SuperCollider, see:
http://doc.sccode.org/Guides/ClientVsServer.html

able to achieve greater possibilities of interaction and drift between various interaction possibilities when performing with computer-based musical environments. This is not to say that by changing the mapping one eliminates the limitations of a musical environment, instead the performer moves to different interactivity features and expressivity possibilities while improvising with the performance environment. Additionally, it could be hypothesised that changing the mapping while performing with external interfaces releases the performer from fixed decisions and raises the possibility for a wider improvisational discourse. The unpredictability of the environment and the non-deterministic possibilities that emerge by the hacking of its mapping will create unpredictable, yet potentially interesting musical results, and will lead the performer to paths otherwise unexplored. Particularly, changing the mapping will avoid the human gestures resulting in repetitive sound manipulations. On the light of these objectives I followed the following strategies outlined as follow:

1. *Implementation of boiler code, something to start-up with.* Before the performance, I create some initial mappings, which I can change them live, this eliminates the time that it might take in order to prepare the code live in front of the audience.

2. *New interaction possibilities with the performance environment.* This is achieved by the introduction of new control inputs on my performance environment. That is, although a performance may start with a set of parameters, I am able to introduce (or amend) new synthesis parameters and implement their mappings on the fly[14].

3. *Musical implications, expansion of musical articulation*. Modifying the performance environment to change the musical outcome, and adapt the environment to new musical requirements while improvising. This is achieved by coupling a control variable of a joystick with a control parameter of the synth, and experimenting its

[14] Examples of this strategy will be discussed in the next paragraphs.

range specification or change the source of a synth that I am controlling externally, e.g. replace sine oscillators with saws etc.

The combination of these two powerful performance paradigms seems fruitful, particularly in the context of musical articulation and real time exploration of interaction affordances as it releases the performer from previous decisions that were taken before the performance. Examples of this approach were explored in Formations and Blind date performance, as discussed in the next paragraph.

Formations[15] (2013) is a live piece created using the method of improvising the mapping in real time. Listening to the piece it becomes apparent that changing the mapping relationships between the hardware device and the synthesis parameters can lead to structural variations and musical articulation.

```
OSCdef(\dens, {|msg| // line 1.
    ~playbuf.set(\dens, msg[1].linexp(0.1, 1.0, 0.5, 20.0)); //line 1.1
    ~playbuf.set(\mod, msg[1].linexp(0.1, 1.0, 0.1, 20.0)); //line 1.2
}, '/max_y');

OSCdef(\scroll, {|msg| // line 2.
~playbuf.set(\start, msg[1].linlin(0.1, 1.0, 1.0, 0.1)); // line 2.1
~playbuf.set(\mod, msg[1].linlin(0.1, 1.0, 0.1, 40.0)); // line 2.2
}, '/max_x');
```

Figure 1.2 Live coding the mapping of hardware devices.

An example of the live modification of the mapping is illustrated in Figure 1.2, which shows a preamble snippet[16] that was developed to fluctuate the parameters of a granular synthesiser using a joystick.

[15] The piece is included in the accompanying DVD and online at this link: https://soundcloud.com/konstantinos_p_vasilakos/formations

[16] Preamble code of Formations is included in the accompanying DVD.

The granular synth used pre-recorded samples stored in the hard disk of the computer, and it was implemented in SuperCollider. Table 1.1 illustrates the parameters that were implemented in the granular synth.

Parameter	Control
Buf	Sample selection from folder.
Dens	Grain density.
Start	Start (reading) position of the grain.
Mod	Frequency.

Table 1.1 Parameters of the granulator implemented in Formations.

In order to map the joystick in SuperCollider I used a programming class named OSCdef, which is a higher-level implementation of the Open Sound Control[17] (OSC) communication protocol in SuperCollider.

The interaction takes place within the OSCdefs, through which I alter their mappings in real time. The incoming control signal of the hardware device is denoted by the word *msg*. During the improvisation I am able to change range specifications of the control signal using *linlin* for linear and *linexp* for exponential scaling that come with SuperCollider. The first two arguments of the scaling functions are the initial low and high values of the input signal; the next two are the lower and higher values of the desired range. Lines 1.1, 1.2 and 2.1, 2.2 of Figure 1.2 illustrate a way to scale continuous control signals in SuperCollider.

Given the nature of code manipulations and complex alterations of each bit of code it is impossible to describe in detail what is happening throughout the piece. However, a description of the performance regarding the changing of the mapping revolves around the following example illustrated in Figure 1.2.

[17] See http://opensoundcontrol.org/introduction-osc

I initially begin with line 1.1, which maps the Y-axis of the joystick with the density (*dens)* parameter of the granulator. Later I add line 1.2, which maps the Y-axis with the amplitude modulation (*mod)* parameter. As the performance evolves I add another OSCdef, line 2, which uses the X-axis of the joystick. In line 2.2 the *mod* parameter is controlled by the X-axis. Line 2.1 introduces a new parameter that is implemented on the fly, and controls the *start* position of the grain.

Additionally, I created various effects to process and enrich the sound output. These included the live implementation of a reverb processor, a comb-filter, a resonator, and pitch shifting. During the performance I improvise by changing the balance between dry and wet signals of the effects and I experiment by routing the signals to each other.

Blind date[18] (2013) is a live piece using the same strategy that is changing the mapping dynamically during the performance. It took place in Chisenhale Dance Space, curated by Agony Art in London. The performance was made jointly with another laptop artist Shelly Knotts, and a team of dancers. The plan of the performance was based on blind date idea, i.e. meet with the dancers only at the day of the performance without any prearranged structure or agreed parts, except sharing a common philosophical concept, that of causality; I approached this concept by creating links of the input data and the resulted sound, a sort of cause and effect relationship between the dancers and the music. Figure 1.3 illustrates the control names and trajectories of the Wiimote that were used in the Blind date performance.

[18] A video of the performance is included in the accompanying DVD and online at this link: https://www.youtube.com/watch?v=2Pk1nmIAoQs&list=PLMmfcbi0xjDlZXjTsmJ7jy5sAYYi6SCOG&index=1

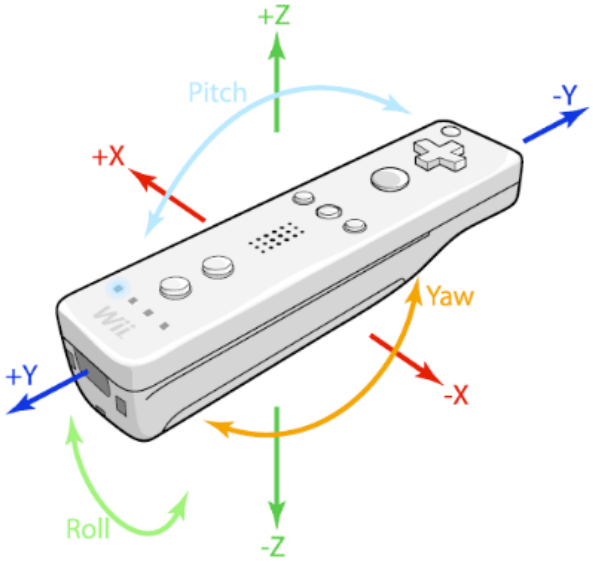


Figure 1.3 Control names and trajectories of Wiimote. Image taken from OSCulator user's manual version: 20120123 (Troillard, 2012, p.53). Copyright Wildora 2012. Reproduced with permission.

Figure 1.4 shows an example of the mapping that was implemented in order to map the Wiimote (used by the dancers) and the synthesis engines run on my computer. To receive the data from the device I used OSCulator (Troillard, 2011), a third party application to convert the data of the device to OSC. The control signals of the device were mapped to a granular synthesiser named *granular*[19]. The code uses the X and Y-axes of the device as well as its accelerator sensor and couples it to various synthesis parameters of the sound engine implemented during the performance.

Wiimote provides three control variables regarding its orientation angles; these are *pitch*, *roll* and *yaw* (see Figure 1.3). In line 2 of Figure 1.4 the *pitch* angle of the device given in line 1, controls the perceptual pitch (*rate*) of the grain. As the performance evolved I improvised with the mapping of the *pitch* and *roll* angles of the Wiimote and the parameters of a pitch shifter effect named *pitchShift*. This was implemented during the performance to process the output of the granular synthesiser. The pitch shifter included two parameters controlling the pitch deviation (*rateDev*) and the time deviation (*timeDev*) of the output, which were both controlled by the *pitch* angle of the Wiimote, lines 3 and 4.

[19] Preamble code is included in the Blind date folder in the accompanying DVD.

```
OSCdef(\wii_pitch, {|msg|
     ~pitch = msg[1]; // 1.
     Ndef(\granular).set(\rateDev, ~pitch.linlin(0.1, 1.0, 0.5,
1.0)); // 2. (line starts from Ndef to semi colon).
          Ndef(\pitchShift).set(\rateDev, ~pitch.linlin(0.1, 0.5,
1.0, 5.0)); // 3. (line starts from Ndef to semi colon).
               Ndef(\pitchShift).set(\rateDev, ~pitch.linlin(0.1, 1.0,
0.1, 2.0)); // 4. (line starts from Ndef to semi colon).
}, '/wii/1/accel/pry/0'); // 5.
```

Figure 1.4 Altering the mapping of synthesis parameters and Wiimote.

In addition to the pre-developed code I created a projection utility[20] in SuperCollider, which was able to pick the name of a performer and project it (see Appendix – Blind date/Fair Algo). The name appears on a projection screen that was placed in a corner of the dance floor. The order of the names as well as the handling timing was made based on an algorithm embedded in the projection utility.

This was used to apply structure and equality in respect of the time that each performer was able to use the device. Once the name of the next performer appeared on the projection the dancer had to hand it over. Finally, the utility uses the vibration of the Wiimote in order to provide tactile feedback to inform the performer to pass the device to the next dancer. Given the nature of code manipulations and complex alterations of each bit of code it is impossible to describe in detail what is happening throughout the video.

1.9. Live coding in studio: from code to tape

Live coding is often witnessed in performances of live electronics and computer music events. However, there is no reason to keep live coding practice limited only to a live context. It All Starts with Noise[21] and It All Ends with Noise (2013), are two fixed compositions that use material that originated from live coding performances. Once I created the material I manipulated it further and organised it in the studio, the

[20] Included in Blind date folder in the accompanying DVD.
[21] Included in the accompanying DVD.

manipulations included the transposition of the original pitch of the sounds as well as editing of their durations. Other processing included spectral manipulation, such as harmonisation and layering.

It All Ends with Noise[22] was premiered in SoundThought festival in 2014 organised by Glasgow University. The video shows the making of a tape piece using live coding in the studio. It begins with an explanation of various parts of the coding performance and proceeds with the music piece.

Timings refer to the documented video. The performance starts by implementing a sample player (0'21" – 0'40"), which uses sound samples that were stored in the hard disk of the computer. Once the samplers are implemented and running I add some postproduction effects, for example reverb, and pitch shifting processors (0'41" – 0'45"). As the performance evolves I improvise by fluctuating the values of the parameters of the samplers and the effects (1'31" – 2'09"). At 2'12" – 2'33" I improvise the mixing of the samplers with the effects. The performance lasted for approximately 40 minutes. The video continues with the final fixed version of the piece. It comprises the juxtaposed processed sound layers, for 2 and 8 speakers.

The strategy that was followed during the composition of these pieces was to create the material live, in line with structuring the composition. For example, while I was organising and editing the piece I wanted a specific passage or a layer of sonic gesture, thus I went back to compose it through live coding. The plan of the performance of this new layer was totally informed and (notionally) instructed by the structure of the piece. This way of composing resembles to the procedure of assembling a puzzle, in the case of the music however, its bits and pieces are created in the same time of forming it. This pre-conception of composition however, did not exclude improvisation. If I created a sound, which I did

[22] The piece was included in the journal's creative work section and can be found at this link http://www.soundthought.co.uk/journal2014/konstantinos-vasilakos-live-coding/

not plan I made it fit to the rest of my piece. To that end, although these pieces were created in an offline way, their initial structure was improvised and performed in studio in using similar rules and strategies I use to build my live pieces on stage.

1.10. Other aspects of live coding

Other forms of live coding include sonification design and the creation of sound synthesis for films. One example of live coding used for film sound is the experimental documentary Alles was wir haben (Alles was wir haben, 2004) where the sound is created in real time by performers interacting live with the visual footage.

> In the development of this soundtrack almost all real sounds were created in a process of interactive programming, where the two artists tried to find ways toward a certain sound impression from their memory. The collaborative process was only possible in this way because the textual description of this purely synthetic, algorithmic sound could be modified while active. (Rohrhuber, de Campo and Wieser, 2005, p.294)

This example of live coding as a real time sound for film is similar to the music making for the film Ascenseur pour l'échafaud (Elevator to the Gallows, 1958) where Miles Davis and his band improvised to the visual footage of the film to create the soundtrack.

Demonstrating the live process of this musical act by Davis and his ensemble, Boris Vian wrote about this session:

> Miles had hurt his lip, Boris claimed, and a piece of skin had come off and stuck to the mouthpiece, producing “a strange sonority” during the recording of “Diner au motel”: “And as some painters owe the plastic quality of their work to some accident, in the same way, Miles willingly greeted this “unheard of” element of music (“unheard of” in the literal sense of the word).” (Szwed, 2002)

With this extract I want to highlight the “unheard” elements that take place during improvisation. Regardless of the medium or the genre, these extra “unheard” musical aspects of the performance shape and form the overall outcome of the music. Similarly, live coding as a pure live act features some similar characteristics of live music making. Relating a live coding act to conventional or established and found forms of improvisation one could argue that a crash, a typo, or a misplaced and inappropriate hack of the active software while this runs could destroy the flow of the performance. Although a crash or a glitch caused by a fault might sound inappropriate during a performance, live coding ingrains them as essential parts of its main ethos. Additionally, these elements add tension to a performance, as the performer tries to overcome and solve these in front of the audience. A common misconception is that driving a computer to or beyond its limits is the sign of a great coder, however, crashing is not an indication of quality.

Finally, some live coding environments provide to the performer some artificial risk functions, as an attempt to create more tension while improvising. For example the IXI Lang’s *suicide function* (Magnusson, 2011) shows this extra element of risk during the performance.

1.11. Conclusion

Live coding is a growing paradigm of laptop performance. The initial idea is to hack the source code of running software or improvise a new one on the fly by tweaking algorithms. It provides multiple roles within the performance; it may serve the role not only of the interface where generic data is imposed, but also as the communication bridge through which the members of an ensemble communicate with each other.

The performer needs to integrate roles of programmer and performer at the same time, and to be competent musically as well. Improvisation is at the fore throughout the performance.

Using live coding to modify the source code of a running digital instrument broadens possibilities of interactivity while performing. Integrating live coding and gestural controllers helps extend the improvisational discourse. The idea of modifying and extending the mapping of an instrument on stage using live coding not only enacts broader interactivity and builds diverse affordances, but also creates wider compositional and structural variations leading to musical richness and greater articulation, as observed in paradigms of performing with hardware controllers.

During the process of building and modifying the synthesis software, glitches, crashes, and unintentional clicks seem to add to the musical outcome due to the real time tweaking of the source code of the running software shaping the musical outcome of the performance.

The form of the piece when improvising with code is open to the experimentation of the performer. They can start with a very simple idea and evolve, as seen in Formations, or they can prepare snippets of code that provide the initial platform to improvise and elaborate with a specific musical idea, seen in performances such as Blind date. However, most of the changes during the performance will require to type new code or amend the current one, consequently, this will lead to a sonic inertia. In other words, it will take some time to change radically textures and sounds or music moods if there are no snippets of code to create these changes, which were developed before the performance.

Moreover, contrary to other genres of electronic music, the live projection of the code shows the source from which the sound is created allowing the audience to establish a rough connection between the changes of the sound and the code. This enhances the aspect of theatricality in live coding.

In the case of performing with large ensembles, for example in BEER sharing the code with the rest of the performers through a network can sustain an immediate sense of collective improvisation by allowing the use of the same bits of code. By improvising common

snippets of code the group creates unified and consistent sound entities where each performer is able to intervene as well as to contribute.

In addition to the connection of the performers through sharing what each one creates in their computer by exchanging bits of code, the structure of the performance is aided by the communication that the performers have through their chatting windows. This allows them to align their actions during improvisation, e.g. decide how to end the performance. This way of performing brings to the fore the way of communication found in other improvisational paradigms such as jazz; for example, eye contact or head nodding as the sign to internal cues, but arguably in a more powerful way because it can be very specific.

2. Hardware Prototyping

2.1. Introduction

Developing interactive environments for musical creation has a long history as well as a significant number of on-going showcases, however the question of how to translate the physical to the digital domain still appears to perplex artists and software developers.

Power of People (PoP) is a sound installation created to explore new approaches in electroacoustic composition using representative interfaces for musical creation. The installation uses environmental data captured in real time such as temperature, light, and motion to control a sound synthesis system developed in the SuperCollider programming environment. More specifically, the project focuses on the mapping of more than a single performer, expanding the scope of my research into man/machine relationships. Work for this project included hardware prototyping as well as software development to build an interactive sound installation. Finally, the project also acts as a model for bigger installations able to implement sonification of various data.

Sonification is an interdisciplinary field in sonic art utilising all sorts of information into processing, information that ranges from statistical data to continuous signals (Walker and Kramer, 2005; de Campo, Rohrhuber, Bovermann and Frauenberger, 2011; Hermann, Hunt and Neuhoff, 2011).

2.2. Overview of PoP

As mentioned above PoP includes both hardware and software[23] development. Hardware includes a series of sensors that track environmental conditions such as light, temperature, and motion accumulated into an Arduino Uno micro-controller board. For a complete overview of the system including schematic diagram and other information see Appendix of PoP. The Arduino board communicates with the synthesis software through the

[23] Included in the accompanying DVD.

Firmata [24] protocol, which includes an implementation of the Firmata [25] protocol for SuperCollider that was used in this project. One of the benefits of following this approach is that it does not require the ability to program an Arduino. To establish the communication between the hardware and the computer the user has to upload the Firmata firmware onto the Arduino board. In order to read the pins of the board and the sensors' data in SuperCollider, the user must implement the classes that are provided from the Firmata implementation. These classes are used to read the digital and analog pins of the board using the protocol.

In the PoP system each sensor is represented by a variable. The system includes the following variables: *~light, ~light2, ~temp, ~motion*. In this way the user can create arbitrary mappings of these variables anywhere inside the program. These names illustrate the on-going values of the sensors attached to the board, except for the motion sensor, which has a binary state of 1 or 0 and works only as an On - Off switch.

For better readability of the code and ease of debugging, the system was developed in two separate parts, the interface and the synthesis engine. The interface includes the implementation of the mapping, and the treatment of the sensors' data (explained in section 2.3). The sound synthesis engine consists of a sine wave generator constrained by an envelope. The temperature controls the duration (release parameter) of the envelope. Depending on the temperature of the space that the installation is hosted in the sonic output varies from overlapping short grains to long drones. The frequency of the sine waves is influenced by a chaotic generator that creates variations on its fundamental pitch controlled by the *~light* sensor. Next in the synthesis chain is a feedback processor that uses the sine waves for input.

[24] See Firmata http://www.firmata.org/wiki/Main_Page
[25] See an implementation of Firmata in SuperCollider https://github.com/blacksound/SCFirmata

2.3. Calibration and scaling: sensors making sense

The digital and analog sensors that were used in the project have different ranges. The motion sensor is digital and appears in SuperCollider in binary range of 0 – 1. The analog sensors (light and temperature[26]) have a 0 - 5 voltage range, the micro-controller board converts it to a range of 0 - 1024 using an analog to digital converter[27] which is applied to the analog inputs of the board. Finally, a preamble calibration to a convenient range of 0.0 – 1.0 was applied to all sensors, excluding the motion sensor.

Having done all the calibration the sensor is ready to bind with the desired parameter. At this point some additional scaling may be necessary in order to adapt to the appropriate range of a synthesis parameter. Most parameters use diverse range, for example a frequency parameter would range between 220 – 1220.0 as opposed to amplitude, which usually ranges between 0.0 – 1.0 in most digital environments. This creates the need for adjustment i.e. constraining or expanding the control signal. Once the sensory input is converted into a convenient standard i.e. 0.0 – 1.0 it is ready to be adjusted according to the desired range of the corresponding parameter.

[26] The temperature sensor required additional conversion to degrees Celsius.
[27] This conversion is made via a circuit embedded in the Arduino board, and is applied when using the analogRead command in Arduino language, which is used to read the analog pins of the board. The Firmata implementation uses this command when reading an analog pin of the board and thus is applying this conversion automatically.

2.4. Mapping of PoP

Figure 2.1 shows the mapping that was implemented in PoP system.

```
Pdef(\x,
  Pbind(
    \instrument, \blip, // line 1.
     \nodes, Pfunc{~light2.linlin(0,1, 1,14)}, // line 2.
      \fund, Pfunc{~light.linlin(0,1, 120,1220)}, // line 3.
       \switch, Pfunc{~motion.asInteger}, // line 4.
        \granRelease, Pfunc{~temp/100 +0.1}, // line 5.
         \delta, 1); // line 6.
).play;
```

Figure 2.1 Mapping snippet in PoP.

Each line begins with the name of the parameter coupled with the sensor, followed by the configuration of the desired range, except lines 1 and 6. Line 1 shows the name of the synth, which the pattern definition controls. In line 2, *nodes* is the number of the generated sine waves controlled by the second *~light2* sensor. In line 3, *fund* is the fundamental pitch of the sine wave controlled by the first *~light* sensor. In line 4, *switch* is the on/off switch controlled by the motion sensor. In line 5, *granRelease* manipulates the release argument of the envelope, using the *~temp* sensor. The *delta* value in line 6, schedules the reiteration of the mechanism, which in this case this is set to every second. The mapping of the environment is implemented through this mechanism that couples the sensor variables with each parameter of the synthesis engine. This mechanism can be seen as a communication bridge between the sensors and the synthesis where some additional treatment is taking place. At the same time the mechanism is re-evaluating its content every second and assigns the current value of the sensors with the parameters. It is possible to manipulate the timing of this evaluation in order to make the system more or less responsive. Additionally, it is also possible to re-arrange all the mappings and tweak the scaling of the sensors or apply mathematical expressions on the fly in order to setup or tune the whole system.

2.5. Modes of interaction of PoP

PoP offers two modes of interactivity: Serene and Agitate. Serene is the primary state of the system. It senses light conditions of the room to control the pitch, and the temperature to manipulate the duration of the grains. The system switches to Agitate as a result of lack of mobility inside the interaction area leading to the sonic distortion of Serene.

The sound is influenced by the environmental conditions, however it is the mobility of the people that create the tranquillity or distortion of the sound output, switching incrementally between the two modes. People are invited to move freely inside the interaction area of the system. The participants are also able to interfere with the sensors and shade over the light sensors in order to alter the pitch of the sound. Most importantly, people are strongly encouraged to collaborate in order to alter the states of the system i.e. move or stay idle. Some observations regarding theatricality in PoP: people entering the installation's area try to interact with the system by moving their hands and twisting their bodies. Thus, the audience is not only experiencing the installation as a fixed musical spectacle but they are also able to intervene in the musical outcome.

2.6. Control and structure

The motion detector stimulates the system whenever it senses movement, swapping between the unprocessed and the feedback signal, if no motion occurs the system defaults to the Serene state. However, the system provides the option to merge the two states into one mode and deactivate the motion sensor. When the system is in Agitate mode an attenuator following the microphone's input signal manipulates its amplitude. This signal picks all the sounds in the space including the output of the system. Moreover, the system implements a pitch detector that analyses the signal of the microphone and uses this value to control the frequency parameter of a frequency modulator embedded in the feedback synth.

Adopting this 'ecosystemic approach' (Di Scipio, 2003, 2011, 2014; Anderson, 2005; Waters, 2007) it became apparent that the sonic result is a self-organised composition comprising wet and dry signals, although the sensors control the sound, it is finally structured by the environmental occurrences. The microphone picks the assembly of the sound coming from the speakers and the concurrent environmental sounds, and the response of the space as well as the sounds caused by the people. This approach is well documented in the work of Di Scipio and his *Audible Ecosystemics* pieces (Di Scipio, 2003, 2011, 2014; Anderson, 2005; Waters, 2007).

2.7. Music composed with PoP

PoP is created to explore collective interactions using various control inputs, such as human and space where the installation is hosted. The space is explored through a set of sensors that track environmental data (light and temperature). PoP shares similarities with the 'ecosystemic' approaches of Agostino Di Scipio, in that it uses attributes of its signal in conjunction with the responses and influences of the ambience to manipulate some parameters of the synthesis engine. The strategies that were followed are outlined below:

1. *Collective interactions, bringing people together*. Main rule of the project was that the participants have to engage in collective interactions in order to stimulate the sound installation. For example, to align their movements inside the field of view of the installation.

2. *Individual interactions.* The participants are also encouraged to play with the light sensor, for example shade it with their hands or cover it. The result of the installation was that it was creating a playground where people could joyfully collaborate with each other. The morphology of the sound was constantly changing according to the input of the space and the people's movement.

3. *Mapping more than one performer*. The system investigates the interaction possibilities that are offered by the multiple interaction, that is beyond the single performer/machine interaction paradigm, and it imposes the live interaction of the audience as active agents[28] instead of spectators during the musical event, as opposed with other paradigms examined in this research.

Although the system uses a small amount of sensors the sound manipulation was strong in terms of timbre variation due to the feedback manipulation of the signal and the influence of the real time analysis of the sound and the mapping of some of its attributes to the parameters of the system, creating evolving musical articulation. There is no preconceived plan or form since the system reacts to the actions of the people (and space) it is hard to predict the structure of the musical outcome.

This section includes a video[29] that demonstrates the interactivity between the system and environmental conditions, and how the former responds to the changes that occur in the space, for example the brightness of the room is translated sonically into the pitch of the sound and fluctuates based on the changes of the light conditions where the system is hosted. The temperature in contrast, doesn't change too rapid or radically as in real life, resulting to more stable fluctuations of the duration of the sine waves.

Timings refer to the documented video. The video starts in complete darkness, the system generates low frequency sine waves; after a while it culminates in feedback. At 0'32" I switch on one of the two lights, turning the pitch higher. At 0'37" the system tracks my movement and switches to feedback mode. At 0'51" I switch on the second light, turning the pitch higher. While I am moving inside the interaction area of the system the motion sensor turns on and off according to the movement. At 1'23" - 1'43" I experiment using a

[28] This explains the reason why the project was named Power of People.
[29] The video is included in the accompanying DVD and online at this link: https://www.dropbox.com/s/c340binrkaui3xt/videoRec1.avi?dl=0

mini torch to project some light over the sensors, turning the system's pitch to its peak. At 1'48” and onwards the sound fades out due to the lack of movement.

What becomes obvious to the spectator is that the pitch of the sound is linked with the light, for example the brighter the space is the higher the pitch of the sound will be. This link becomes apparent making clear the mapping between the light sensors and the way they are connected with the parameter that controls the pitch of the sine waves.

2.8. Remarks on PoP

A video of the system describing its functioning as an installation is provided at the following link[30]. The following description refers to the corresponding video. In the beginning of the video the system creates low frequency drones due to the dark conditions of the room stimulated by the appearance of the participants. At 0’09” the pitch of the sound is higher caused by the increase of the light. At 0’13” - 0’38” the system switches to Agitate (distortion state) caused by the participants. At 0’30”- 0’41” the participants engage with the system by shading its light sensor to change the pitch of the sound. At 0’45” - 0‘56” the system culminates in silence due to lack of mobility inside its interaction area. At 0’58” the system is stimulated again switching to Agitate. At 1’10” the participants engage in collective interactions by moving and staying idle inside the interaction area of the system as well as interacting with the light sensor. The system continues to swap between the two states of interaction caused by the mobility of the people 2’30” - 2’45”.

Observing the installation at work led me to the following idea, I am able to perform the installation in a similar way to a performer interacting with a computer-based environment. Di Scipio (2014, p.51) states that:

[30] The video is included in the accompanying DVD and online at this link: https://www.dropbox.com/s/7t2wjwz5mgx18gk/VID_20151102_180406.mp4?dl=0

> This work was born as an installation project. However, I eventually devised ways to use it in performative contexts. Indeed, a performer can look for places or surfaces in the total infrastructure that lend themselves to be efficiently acted upon, searching the affordances that allow for possible gestures and for actions enabling her/him to enter the sonic process and to affect it, to some extent. That turns the "installation" into a kind of "instrument", or better a sound generating device that includes the environment as a part of it — the same environment where the performer acts as part of the sound generation process. The form of presentation becomes uncertain: is it installation or performance? Or is it an instrument that one can play with?

In the same paper Di Scipio states that 'the task of composition becomes not so much one of interactive composing, but one of composing the interactions.' It is worth highlighting: 'composing the interactions' is based on mapping. In this context mapping could be seen as a framework that encloses all the interaction ideas implemented at this exact moment of development. One could argue that this defines the context of the interaction as well as forming the affordances of the system; in turn, through this the composer is able to instruct people on how to 'play' the installation. Furthermore, the music that results from this interaction forms the showcase of the idiomatic decisions and preferences of the composer that were established during the design of the mapping.

Inside this framework the participant understands how the work perceives his or her actions and creates a sort of performance through listening and experimentation. This is similar to what a performer does when s/he tries to learn a computer-based environment, for example performing with open air interfaces e.g., Leap Motion, Kinect, and other optical devices, where there are no visual or physical cues. It is through hearing that a performer understands how the system interprets his or her gestures or whatsoever one uses to interact with the environment. This is also observed in the Leap Motion project (see chapter 4).

However, one distinction from the single user interaction model where the performer is most of the time also the developer of the system, is the unpredictability that the system provides. This is because the participant has no previous knowledge of how his or her actions will be perceived and interpreted, compared to the former case where usually the designer/performer has an intrinsic idea about the behaviour of the system. This quest of the unknown is something that the designer needs to be aware of while developing the system and provide some generic concepts about the system explaining easily the behaviour of the system to novel users.

One way to enhance the accessibility and explain the affordances of a performing environment is with the use of metaphors. Through the implementation of metaphors one is able to explain the mapping as well as make it transparent to the audience. Metaphors are also examined in my Greap project (see chapter 4).

Evidently, metaphors are used widely when implementing mappings for auditory display and sonification projects. Walker and Kramer (2005, p.409) state that 'Mappings that are based on stronger or more natural metaphors should result in faster and more accurate control reactions. They should also be learned faster, which would lead to a greater improvement in performance across the blocks of the experiment.'

Some of PoP's control parameters were conceived to provide some metaphorical meaning[31]. The pitch of the sine waves is controlled by the brightness of the space i.e. the darker the room the lower the frequency of the sine waves, whereas brighter spaces will result in higher pitch, this will correlate to higher frequencies, thus brighter sounds. The current version of the system implements only this metaphor.

Another finding that arose while working with PoP was the problem of directing people inside the field of view of the system in order to stimulate the installation. When someone

[31] Also noted and demonstrated in the attached video.

enters a site-specific installation that requires some input by the participant, s/he is not aware of any physical constraints or borders (unless these are defined by the creator of the installation). One way to limit the interaction area and guide the people inside the active area of the system is through the use of bespoke immersive physical platforms, which will facilitate the installation and will define the space of interaction of system and people. Some examples include Flowspace (Bisig, Schacher and Neukom, 2011), the installation employs a platform in the shape of a dodecahedron where the participant can enter and experience the installation.

2.9. Conclusion

The music that is created with the PoP system is a result of an on-going process of trial and error and experimentation through hearing, where the participants facilitate their actions within a given range of affordances formed by the composer.

In the case of PoP this trial and error process does not only include someone interacting with it and trying to do something musically meaningful within a given range of interaction affordances, but the system is constantly trying to tune itself through its listening abilities adopting 'ecosystemic' approaches.

To provide a more refined way and direct the participants towards a proper interaction with the system the composer may use a physical platform.

This trial and error and functional glitches create an unpredictable musical outcome generated by the random responses of the people while interacting with the system. Therefore, it is difficult to define the structure of the musical outcome. However, the system revolves around long drones and short sounds depending on the temperature of the space whose the installation is hosted whereas the pitch of the sound is influenced by the brightness of the space, consequently providing an aural link between the brightness of the room and the pitch of the sound.

Finally, the installation appeared to provide theatricality, by people trying to interact with the installation and playing it by moving their hands and bodies within the interaction area, leading to an impromptu musical staging.

3. Tangible Interfaces

3.1. Introduction

'Humanising technology' refers to an anecdotal statement by Simon Emmerson in a recent interview (Studer, 2010). In the video he mentions his 'turning view on live music' generated by computers and how technology is approached nowadays stressing 'the importance of the human factor within the process of live generated computer music.' He also describes the creative artefacts of live coding and circuit bending in the musical discourse, which, according to him 'may be seen as an unpredictable process.'

There is a wide range of hardware controllers available for the design, transformation and triggering of sound for a variety of musical applications, e.g., composition, sound design, virtual-reality environments etc. Controllers that follow human gesture and transform physical energy to input data are generally categorised as gestural controllers. Of these, there is a sub-class of interfaces that involve the use of physical objects to interact with software environments known as tangible controllers. This chapter provides an evaluation of tangible controllers in an improvisational context. I describe the advantages and disadvantages of various approaches followed through my compositional process and experimentation, with particular reference to their use in performance and their potential for expressivity both in a live context and as a compositional tool in the studio.

Work presented here includes custom-built performance environments using a bank of unedited and heavily edited sound samples, and using a Wiimote to perform and improvise their manipulation live. This chapter will also describe the implementation of SuperCollider patterns (see section 1.5) to control the parameters of a synthesis engine and live fluctuation of their internal arguments via a Wiimote.

When performing with tangible interfaces the musician does not have to learn to play this instrument and go through a learning curve in a traditional manner; instead s/he may

focus on a wider range of performance affordances and avoid physical constraints that an acoustic instrument may present such as the difficulty of making a traditional instrument produce tones that are outside of its capabilities[32]. This view appears to be well supported since the very beginning of the advent of computers in musical discourse:

> Even a beautiful and cleverly designed instrument is constrained by inescapable mechanical limitations. Simply obtaining a good basic sonority on many instruments requires a long period of practice and expert counsel. Some instruments are more physically difficult to play than others. For example, the large instruments of the lower registers (bass and baritone saxophones, double bass, and tuba) require more strength to play and may necessitate stretching to achieve the proper note selection. (Roads, 1996, p.619)

3.2. Digital musical instruments (DMI)

These bespoke performance mediums that I am going to elaborate in the following chapters are often described as Digital Musical Instruments (DMI): 'An instrument that uses computer generated sound can be called a digital musical instrument (DMI) and consists of a control surface or gestural controller, which drives the musical parameters of a sound synthesizer in real time.' (Miranda and Wanderley, 2006, p.1)

The choice of a controller is usually a result of a performer's personal ergonomic and idiosyncratic preferences. A wide variety of gestural input devices and controllers can be found in order to serve various body movements; a classification has been proposed by Miranda and Wanderley (2006, pp.20–21) consisting of 'instrument-like' (controllers that resemble the shape of traditional instruments), 'augmented instruments' (instruments with attached extra sensory capabilities), and 'alternate controllers'. According to the same

[32] It is worth mentioning at this point that this distinction does not aim to undermine the unsurpassed beauty of the traditional instruments, rather to illustrate one of the key concepts of computer-based instruments.

authors an 'alternate gestural' controller may be roughly classified as one that does not share its characteristics with the previous two classes; thus a Wiimote could fall in the class of the 'alternate' controllers. Furthermore, 'alternate controllers allow the use of other gestural vocabularies in contrast to those of acoustic instrument manipulation' (Wanderley, 2001, p.638).

Improvisation benefits from the use of devices that follow the rapid alterations of human movement[33], providing high level of resolution of tracking in fulfilling some basic criteria of ergonomics for musical use; for example being easy to handle and manipulate its buttons. For this purpose the Wiimote appears sufficient to cover these needs, due to the variety of sensor technology included in one single device. Its features include:

- Multi directional gesture tracking.
- Satisfactory level of tracking precision.
- Ergonomic suitability for musical performance.
- Economic affordability.

This provides the following interaction affordances:

- Real time manipulation of multiple synthesis parameters and totally independent.
- Accurate musical expression.

To expand the tracking capabilities of the device I used its additional accessory called Nunchuck[34]. This extra attachment offers the same gestural tracking abilities as well as a small joystick placed on the top of it. In addition, the device offers the tracking of its spatial position if combined with an external infrared sensor, called Wiibar. This was not

[33] See improvisation steps (Pressing, 2001, p.130) in the introduction of this thesis.
[34] See http://wiibrew.org/wiki/Wiimote/Extension_Controllers/Nunchuck

used in the projects to avoid dependency on the spatial location of the performance. Instead, the mappings were focussed only on the orientation sensors and the accelerometer of the device. To examine the device in terms of its performance and compositional feasibility I developed two environments that allow manipulating sonic material in real time, named BiGrain, and Stay On This Gesture (2011).

3.3. Overview of BiGrain

BiGrain implements granular synthesis using pre-recorded audio samples stored in the hard disk of the computer. It comprises four granular processors through which the system is able to create simultaneous manipulation of four separate samples. The granulated signals are mixed using a morphing processor that sums the output of the signals into two channels. Although the system supports separate processing of the material, the mapping of the parameters was summed into two groups[35]. The variable names of the Wiimote refer to Figure 1.3, section 1.8. The *pitch* angle of the Wiimote controls the perceptual pitch of the grain of the first two granulators in the same time. The same variable of the Nunchuck controls the perceptual pitch of the other two. Table 3.1 illustrates the parameters that were implemented in BiGrain.

Group A	Group B	Control
Rate	bRate	Pitch of the grain.
Dur	bDur	Duration of the grain.
RateDev	bRateDev	Deviation of the pitch.
Dens	bDens	Amount of grains.
Stretcher		Stretching rate of the grain.
Offset		Offset of the grain.
X_morph		Mix of the first two

[35] There are two groups, which include the granular processors; each group contains two granular processors.

	granulated signals.
Y_morph	Mix of the second two granulated signals.

Table 3.1 Parameters in BiGrain.

3.3.1.Mapping of BiGrain

The communication of the device with the environment was made via OSCulator. A one-to-one mapping strategy was implemented to connect the device's variables with the control parameters. Table 3.2 shows the Wiimote's control variables[36], and their mapping to the parameters of the environment. The implementations of the granular parameters are identical for both granulators, thus the names of the parameters refer to both pairs of granulators. Both the Nunchuck and Wiimote devices implement the same mappings i.e. same control signals manipulate the same synthesis parameters. All the parameters that are connected with the Nunchuk prefix the letter b, e.g. *bDensity*. The following description of the mapping refers to both groups.

Wiimote	Nunchuck	BiGrain parameters
Pitch	Pitch	Pitch of the grain(s)
Roll	Roll	Duration of the grain(s)
Yaw	Yaw	Pitch deviation of the grain(s)
Accelerometer	Accelerometer	Density of the grain(s)
Accel. X-axis		Stretching rate of the grain(s)
Accel. Y-axis		Offset of the stretch effect
	Joystick X-axis	Morphing balance

[36] Illustrated in Figure 1.3, section 1.8.

	Joystick Y-axis	Morphing balance

Table 3.2 Mapping Wiimote to BiGrain.

The *pitch* angle of the device controls the perceptual pitch (*rate*) of the grain while *roll* angle controls the duration (*dur*); *yaw* angle of the device controls the deviation (*rateDev*) of the pitch parameter whereas the accelerometer controls the density (*dens*); increasing effort to shake the device increases the number of the grains. X and Y data of the accelerometer are coupled with a post-production effect that implements a time stretching effect to the granulator. Specifically, the X-axis controls the rate of stretching and the Y-axis controls the offset parameter of the effect. When the joystick is at its default position (midpoint) the system mixes the four signals equally balanced. Finally, the joystick of the Nunchuck controls the mix of the morphing effect between the granulated signals.

The decisions of the mapping were made to enable lucid connections between gestural input and the control parameters of the system. For example, when the performer lowers the device downward the perceptual pitch of the grains falls and vice versa, whereas rolling the device horizontally affects the duration of the grains, sloping the device to the left decreases whereas sloping right increases the duration of the grains.

Accordingly, the number of the grains was controlled by the accelerometer sensor enacting a physical relation between the effort of the performer and the density of the grains. Moreover, the accelerometer influenced the amount of the stretch of the grains increasing the physical connection with the sound. In addition to the continuous controls, I have also used some of the buttons embedded in the device to switch or select various momentary parameters. Buttons A, B, C, and Z of the device were used to toggle the selection of the samples used by the first two granulators incrementally. Each time the button is pressed the granulator picks the next sample in queue. Buttons minus and plus were used as simple switch to turn on and off the synthesiser.

3.4. Overview of Stay On This Gesture

Stay On This Gesture[37] implements granular synthesis using pre-recorded audio samples stored in the hard disk of the computer. Instead of using a one-to-one mapping strategy, this time the development of the environment focussed on the elaboration of the interaction through the implementation of patterns in order to expand sonic manipulation and to advance the improvisation discourse.

3.4.1. Mapping of Stay On This Gesture

The implementation of the mapping employs patterns instead of creating a one-to-one mapping strategy. The Wiimote was used to fluctuate the parameters of the patterns, which are grouped in separate blocks. The performer is able to use diverse kinds of patterns, thus the manipulations of the sound material occur largely due to the structure of the patterns, implementing Brownian motion[38] and probabilistic methods. In terms of interaction with the patterns, the performer can influence their continuous progression, for example, while the patterns run the performer is able to fluctuate the step size argument of the Brownian motion. Figure 3.1 shows an example of a block in Stay On This Gesture.

The environment comprised four blocks. In line 1, the pitch of the grain (*rate*) is controlled by a pattern that implements Brownian motion. The first argument of this pattern is the lower value of the range of the stream, in this case 0.1; the second argument is the higher value, which is controlled by the *pitch* variable of the Wiimote. In line 2, the position of the grain (starting point that is read from the buffer/sound sample) is controlled by the *roll* variable of the Wiimote. In line 3, the release arguments of the envelope are controlled by another Brownian motion pattern, which is multiplied by the *yaw* variable of the device. Finally, in line 4, the duration of the re-evaluation of the block[39] is controlled by a

[37] Software is included in the accompanying DVD.
[38] http://www.wolframalpha.com/input/?i=Brownian+motion
[39] Evaluation of the block's contents. This process is described in Mapping of PoP chapter.

Brownian motion pattern; its step size argument is controlled by the *yaw* variable of the Wiimote.

```
Pbind(\instrument, \grains,
 \rate, Pgbrown(0.1, Pfunc{~specs[\rate].map(~pitch)}), // line 1.
  \startPos, Pfunc{~specs[\start].map(~roll)}, // line 2.
   \release, Pgbrown(0.2, 2.0,
0.1)*Pfunc{~specs[\release].map(~yaw)}, // 3.
    \dur, Pgbrown(0.01, 0.1, Pfunc{~yaw}) // line 4.
).asStream;
```

Figure 3.1 A mapping block implemented in Stay On This Gesture.

Using the cross button of the Wiimote the performer is able to advance to the next pattern set, and plus and minus buttons of the device are used to start and stop the performance. Moreover, s/he is able to precompile any arrangement of pattern definitions or mappings in advance of a performance. Therefore the performance evolves by using these sets of patterns and move between them dynamically while interacting with the continuous progression of the patterns.

3.5. Music Composed with the Wiimote

To explore the compositional viability of BiGrain I created a piece named Study II[40] (2011). The musical characteristics of the piece depict the physical energy of the gestures in the sound. At 0'48" – 1'30" the spectrum content of the sound is radically transformed due to the fast fluctuations of the perceptual pitch (*rate*) and *density* parameters, which are controlled by the *pitch* angle and the data of the accelerometer of the device respectively. Instead of rapid movements that change the sound radically, the performer achieves some refined manipulations of the sound by performing small and accurate gestures. This yields slow evolving long drones that are controlled by the micro movements, affecting the microstructure of the sound: 1'29" – 2'31". These lesser movements could be viewed as similar to the slight movements while performing with a fretless instrument where the

[40] Recordings can be found at the accompanying DVD and online at this link: https://soundcloud.com/konstantinos_p_vasilakos/study-2-byegrain-v-1

performer slides its finger around the note in order to cultivate its sound. The character of the piece ranges between noise bursts, short grains, and long drones according to the movements of the performer.

The reasoning behind Study II was to make a piece that illustrates the gestural characteristics of a performance. The strategies that were followed are outlined below:

1. *Enhancing expressivity*. The performer enacts a transparent relationship between his or her gestures and the sound. This is achieved by the one-to-one mapping strategy that was followed.

2. *Virtuosity, built sonic manifestations of the gestures*. During the performance I am constantly monitoring and refining my gestures according to the sonic output. Similar to the performance with an acoustic instrument.

3. *Use of specific sounds, defining an order of sound manipulation*. The performance includes the use of specific sounds provided in the root folder of the project, which were used to compose the piece and build the context of the composition. These were placed in the folder in a specific order, thus following a specific cue of the sound manipulations that took place during the performance; this helped to build the structure of the piece and organising the material.

Both from the performer and audience point of view, the piece was successful to create links between my movements and the resulted sound, which was the primary aim of the project. The physical relation between my gestures and sound helped to provide a transparent mapping enacting expressivity during the performance. However, the one-to-one mapping strategy that was followed appeared to yield to repetitive sounds and iterated gestures, leading to a sort of predictability throughout the performance.

Song has Sung (2012) is composed using Stay On This Gesture. I wanted to explore the real time performance with an autonomous system, which it would be able to respond to

my input and influence my decisions. The performer acts as a humanising mediator between the sound and the computer's processes, sharing equally the stage with the computer, enacting mutual intervention (also described by Chadabe). The performer engages in an infinite loop between his or her input and the response of the environment, which in turn it shapes the sound. Some strategies included the following:

1. *Versatile manipulation of the material live.* This is achieved by using patterns in SuperCollider, which implement random processes to control and create a complex network of versatile manipulation of the samples in real time, instead of following the one-to-one mapping strategy described in section 3.3.1. Additionally, the performer is able to control their internal parameters in real time. One reason I became interested in using patterns was to create multi-timbre organisms that evolve in respect of the continuous progression of arbitrary calculations creating complex sound-spectra, not in the comfort of a studio but in a live and real time situation. Therefore, the music consists of juxtaposed and independent layers of sound, which create complex textures and contrasting rhythms.

2. *Mutual intervention on stage, dialog with a semi-autonomous system*. While the system is partly autonomous, the interaction affordances of the system allow the performer to give a direction rather than have total control of the musical outcome. Thus the performer is influencing the overall sound rather than controlling (thoroughly) the output of the system (as opposed to BiGrain, section: 3.3.1*)*.

3. *Organising the patterns, structuring the form of the piece.* The music created with this environment is an illustration of the complex sequences of data generated by the patterns enclosed in individual blocks, therefore the composition relies on the structure and the scheduling of these. Once the blocks are created, the performer triggers them during the performance constructing the overall form of the piece emerged from the scheduling of the blocks within a given timeline.

4. *Use of specific sounds, defining an order of sound manipulation*. The performance includes the use of specific sounds provided in the root folder of the project, which were used to compose the piece and build the context of the composition. These were placed in the folder in a specific order, thus following a specific cue of the sound manipulations that took place during the performance; this helped to build the structure of the piece and organising the material.

My intention was not to create an instrument, which it would translate all my actions to sound depending to the effort that was employed, but to create a system that the performer would be able to influence, and built a sort of dialog on stage. The performer is not manipulating the sound directly, but s/he intervenes with the patterns while they unfold in time influencing the sonic outcome in a similar way as Xenakis described in Formalised Music:

> With the aid of electronic computers the composer becomes a sort of pilot: he presses the buttons, introduces co-ordinates, and supervises the controls of a cosmic vessel sailing in the space of sound, across sonic constellations... now he can explore them at his ease… (Xenakis, 1992, p.144)

Due to the arbitrary manipulations of the sound and the vast number of controlling sequences it is impossible to describe the piece by explaining each movement. Instead, a representative recording[41] shows how a typical performance might progress.

3.6. Conclusion

Tangible interfaces provide stable communication for interacting with performance environments for real time manipulation of sonic material. Specifically, the Wiimote

[41] A recording of the piece is included in the accompanying DVD and online at this link: https://soundcloud.com/konstantinos_p_vasilakos/the-song-has-sung

(including its additional accessory), designed for game interaction, provides a fruitful solution to interact with computer-based environments.

Comparing the musical pieces Study II and The Song has Sung created using Wiimote it became apparent that mapping has a great impact not only to the interaction affordances of a system but also to the musical characteristics of each piece. In both pieces, the Wiimote was used as an extension of my body, which in conjunction with its accelerometer and the appropriate mapping to the synthesis engine avoided counter intuitive relationships between the gestures and the sound, enhancing the musical tension and theatricality throughout the performance.

Specifically, the one-to-one mapping strategy allowed me to achieve refined manipulation of the sound using my gestures in an accurate manner, however there is a trade off between accuracy and the variety of interaction possibilities; the same gestures resulted in the same sonic manipulations.

To investigate this I developed another environment emphasising on mapping strategy and employed patterns in SuperCollider.

Mediating mapping with patterns I created a wider range of sonic manipulations due to the complex sequences implementing random generators that offered complex and versatile sound processing influenced by the performer. Using Stay On This Gesture I was able to compose a piece by organising the patterns into separate blocks. Therefore, the microform of the piece derived from the continuous progression of the patterns, whereas the overall form of the piece emerged from the scheduling of the blocks within a given timeline.

This appeared to offer more flexibility in terms of musical variation, providing possibilities to orchestrate a piece by using the idea of blocks containing diverse sonic manipulation, as

well as offering the means to develop a sort of discussion connection between me and the system or a mutual intervention.

4. Optical Interfaces

4.1. Introduction

Performance practices with interfaces that use gestural movement to interact with a computer-based musical environment are integral to the investigation of present representative approaches to digital musical interfaces. A significant amount of investigation has been pursued over that past years by dedicated organisations in this field, such as the Studio for Electro-Instrumental Music (STEIM), and the New Interfaces for Musical Expression (NIME) community or individual artists and researchers who created pioneering work in this area such as Michael Waisvisz' The Hands (1984), and Radio Baton (1985) by Max Mathews and Robert Boie (Manning, 2004, pp.379 – 381).

This investigates the field of optical interfaces using a Leap Motion (LM) device. In considering expressivity as the primary aim of this project, particular attention was given to mapping strategies that are informed by gestural metaphors in order to foster *transparency*[42] (Fels, Gadd and Mulder, 2002; Gadd and Fels, 2002; Wessel, Wright and Schott, 2002; Fischman, 2013).

It is not long since LM became available to the public and some projects have already shown its potential for musical applications, showcasing the device as an interface to facilitate intuitive, expressive performances. Some use it as their main interface whilst others combine it with other controllers. In some other projects the device is used to trigger pre-developed sound material or to control effects of post-production software, to name a few: Touchless (Ma, 2013), and Human Electro (Fujimoto, 2013). In some other cases the device is used as an interface to emulate traditional instruments such as the piano – e.g. Crystal Piano (Silva et al., n.d.), Drumactica 2.0 (Bertelli, 2013), and Gesture Control Jam (Hoenig, 2014).

[42] Transparency of mapping refers to the ability of the instrument to create clear links between the actions of the performer and the resulting sound.

Although all these projects are successful in exploiting LM's ergonomic possibilities for intuitive musical interactions, they do not demonstrate the potential of the device to be used to shape sounds, not only from a purely aural point of view, but also in the creation of a causal connection between gesture and sound. To do this, it is necessary to design an efficient mechanism that will facilitate this functionality through mapping. Therefore, significant care and consideration over this issue was given throughout the development of Greap, a real time music environment built for the manipulation and improvisation with sonic material.

Greap -, Gr(ain) + (L)eap (Vasilakos, 2014) was designed to create computer music that exhibits audible transparency of real time gestural manipulation of sonic material. It consists of a software environment[43] integrated with the LM hardware device. The performer is able to pre-design the mapping blocks before a performance according to the interaction s/he wants to achieve. The mapping can then be changed dynamically, allowing the performer to shift between different sets of gestures and sonic manipulations, and to explore diverse interaction affordances offered by Greap.

From the perspective of both audience and performer it is hard not to see a resemblance to the Thérémin (1924) performance paradigm. This pioneering electronic sound device, consisted of 'two capacitor-based detectors, one a vertical rod, the other a horizontal loop. These controlled pitch and amplitude, respectively, by generating electrical fields that altered according to the proximity of the hands of the performer' (Manning, 2004, p.5).

In the case of Greap however, although the performer uses his or her hands in similar manner, controls the spectrum of the sound (i.e. the gestures manipulate the timbre of the sound). Therefore, this resemblance is only relevant to the kinesiology of the performance and not to the sonic outcome, since in the case of the Thérémin 'the morphology (the relationship between pitch, timbre and time) – remains fixed' (Paine, 2009, p.143).

[43] Included in the accompanying DVD.

Furthermore, the affordances of Greap are significantly different from the Thérémin due to the multidimensional tracking possibilities it provides, and thus it allows controlling many synthesis parameters simultaneously and totally independently, which can help to manipulate sonic material in a more intuitive way.

4.2. Overview of Greap

The main sound generator in Greap is a granular synthesiser that uses audio samples stored in the hard disk of the computer. It supports standard granular synthesis parameters including: transposition, duration, amplitude, and panning (stereo) position of each grain. In addition, the user is able to set the start and the end point of each grain, as well as the reading speed of the grains. Table 4.1 provides a summary of the main parameters of the environment.

Parameter	Control
Pos	Initial position of the grain
PosRate (posRateM, posRateE)	Reading speed
Rate	Pitch of the grain
Bufnum	Sample index to manipulate
Amp	Volume of each grain
GranDur	Duration of the grain
PanMax	Panning position of each grain (stereo)
TrigRate	Trigger of new grain
PosHi	End position of the grain

Table 4.1 Parameters implemented in Greap.

Although the above parameters are very powerful on their own, some auxiliary parameters are implemented. These are used to affect the main controls by creating slight fluctuations

to current values, and are controlled by the user via the LM. Table 4.2 provides a summary of the deviation parameters.

Parameter	Deviate
PosDev	Position of the grain
PitchDev	Pitch of the grain
DurDev	Duration of the grain
AmpDev	Volume of the grain

Table 4.2 Deviation parameters in Greap.

LM communicates the synthesis parameters via GECO (Bevin, 2014), a third party application that communicates data from the device to any application able to receive MIDI or OSC data. It provides a fixed set of control signals, including up and down, and vertical positioning of the hands, as well as the inclination values of both palms separately. Then, for example, the user can map the vertical position of the left hand to the duration of an event and the pitch to the upward/downward position of the right hand. GECO also provides a visual representation of the values for each control signal.

4.2.1.Scene handling/snapshots

In Greap, specific configurations of parameters within the system that can be planned in advance by the performer are called scenes. A scene may include information about mapping, audio sample, and parameter initialisation values. The user is able to shift dynamically between various scenes by using an external interface (for example, a MIDI foot switch) or change them natively via a selection menu implemented in the graphical interface of Greap. There is no limit to the number of the scenes.

When the performer switches to a scene, s/he has continuous control over a group of parameters included in the mapping. The parameters that are left out of the scope of the mapping will jump to a given value that the user decides not to alter. In this way, s/he

may set highly contrasting scenes and switch between them either instantly or in a gradual manner by means of a fading function implemented to enable smoothness of changes.

```
~presetMenu.addItem(\Scroller, { // line 1.
 args a, rate=1.0,rateDev=0.0,posLo=0.01,posHi=0.99, //line 2.
trigRate=100,bufnum=0,posRateM=1,posRateE=0,granDur=0.3;
        ~i=0; // line 3.
  x.set(\rate, rate, \rateDev, rateDev, // line 4.
        \bufnum, bufnum, \posLo, posLo, \posHi,
         posHi, \trigRate, trigRate,\posRateM,
          posRateM, \posRateE, posRateE,
          \granDur, granDur
    );
sl[\bufnum].value_(bufnum).doAction; // line 5.
sl[\rateDev].value_(rateDev).doAction;
sl[\rate].value_(rate).doAction;
sl[\posLo].value_(posLo).doAction;
sl[\granDur].value_(granDur);
sl[\posHi].value_(posHi).doAction;
sl[\trigRate].value_(trigRate).doAction;
sl[\posRateM].value_(posRateM).doAction;
sl[\posRateE].value_(posRateE).doAction
                  });
```

Figure 4.1 Example of a scene implemented in Greap.

Figure 4.1 shows an example of a scene. In line 1, is the name of the scene. Line 2, shows the values of parameters that the user has no access for continuous control; Line 3, is the index of the mapping that acts as a pointer to an array of mappings (in this case, this, points to the first one in the order of the cueing mappings). Line 4, is the synthesiser parameter values (taken from line 2 above). Line 5 sets the faders of the graphical interface (values again taken from line 2).

4.3. Mapping variability

The prime motivation of this project was to create mapping strategies that foster transparency between the gestural action of the performer and the resulting sound in

order to enhance expressivity while performing with a computer-based musical environment. Instead of using single mapping Greap employs an embedding mechanism that hosts various mappings that change dynamically while performing, without interrupting the musical flow of the performance. While in other systems this principle is implemented as an external library[44], in Greap it is implemented as an internal mechanism, avoiding dependency on third party software, which might affect the maintainability of the system in the long run.

Greap allows the user to pre-configure the mapping relationships before the performance according to the interactions s/he wants to achieve. Changing the mapping during the performance offers multiple interaction possibilities and allows exploration of the system's inherent affordances. The most important aspect of this feature is that the same gestures can control different parameters throughout the performance. Therefore, the performer can build blocks of interactions, which will result in diverse sonic outcomes, and most importantly without having to stop the flow of the piece. This type of functionality has also been explored in MAES (Fischman, 2013, p.334). In the current version of Greap, the variable mapping is implemented in a textual format. The user needs to couple the control signals coming from GECO with the synthesis parameters of Greap. For example, if the performer wants to control the duration of a grain with the horizontal position of the left hand then a MIDI number is assigned and coupled with the duration parameter of the grain. Greap allows an unlimited number of mappings, making it easy to create versatile combinations.

Figure 4.2 shows the code for two individual groups of mappings that the performer can switch on incrementally. Each group includes the parameters that the user will have access. Each group is enclosed in parentheses; the numerical values are the MIDI numbers that correspond to GECO's variables. In the first group, the number 0 couples the

[44] For instance, Libmapper (Malloch, Sinclair and Wanderley, 2013).

performer's left hand to the *gate* argument, 1 and 2 respectively couple the horizontal trajectory of each hand to the *low* & *high* read position in the grain, 3 couples the vertical trajectory of the left hand to the *deviation* of the read position of the grain, and finally, 9 couples the horizontal position of the right hand to the panning (*panMax*) parameter. In this version, a MIDI foot pedal controls the amplitude using number 7.

```
~cc = ([
     (0:\gate, 1:\posLo, 2:\posHi, 3:\posDev, 9:\panMax, 7:\amp),
          (0:\gate, 3:\rateDev, 2:\posRateE, 1:\posLo, 9:\panMax)
       ]);
```

Figure 4.2 Groups of mapping implemented in Greap.

4.4. Interaction affordances of Greap

Optical interfaces such as LM can provide a high degree of expressivity, letting the performer move his or her hands in any direction freely and effortlessly, yet with no visual cues and tactile feedback or restrictions relative to its tracking area. However, this openness does have limitations since the performer must always consider the appropriate position of the hands within the tracking range of the device. In Greap, this is partly solved by visualising the values of each parameter in its graphical user interface, using graphic faders and number boxes. However, monitoring range through the computer screen during performance might lead to the isolation of the performer from the audience, affecting eye contact, and leaving less room for theatrical and musical expression.

A way to avoid this, which was followed while learning Greap, suggests that by ignoring the visual display and relying merely on the sound outcome, the performer can learn and become skilful. Following this practice a gradual increase of gestural dexterity and virtuosity became apparent.

Some other issues regarding environmental conditions seem to influence the performance of the device. Reflective surfaces and external infrared light can reduce the precision of

tracking of the device. In addition, long sleeves may also detract from its accuracy to isolate hands.

Compared to other performance paradigms outlined in previous chapters, Greap offers the ability to shape sounds in a more immediate way. The ability to set mappings and state the values of selected parameters provides faster transitions between diverse states of sound manipulation. This is almost impossible to achieve with other performance paradigms. For example, in live coding these transitions tend to be time consuming due to the time taken to type in the code and change the state of the running program. In addition, Greap provides a very real sense of the shaping of sounds, giving the sonic medium an almost tangible physicality.

4.5. Effort and visualisation of musical tension

Devices like LM require no contact or physical effort. The user can interact within the tracking area of the interface without having to change or manipulate any physical state or mechanism of the controller. Although Greap does not require any physical effort or mechanical manipulation during performance, the bodily motion employed to shape sounds expresses 'effort' (Vertegaal, Ungvary and Kieslinger, 1996, Fischman, 2013, p.330) and musical tension.

4.6. Tangible sound

Although optical interfaces are intangible, Greap creates a potentially tangible connection with the sound – the hands move in such a way that the audience may experience a direct shaping of the sonic material correlated with the gestural movements, in a similar manner to pottery where the potter gives shape to clay with manual dexterity. To explore further this concept, future directions for Greap's development will include real time manipulation of a virtual object that the performer will be able to shape with his or her hands. Similarities can be seen with 'sound sculpting' (Mulder and Fels, 1998, pp.15–16).

4.7. Metaphors

Metaphors, in the context of computer music performance, are acts of gestural mimesis of everyday movements. 'A good mapping metaphor will help performers and the audience understand the effects of gesture on sound' (Sapir, 2000, p.3).

A metaphor is an effective way to enhance expressivity and mapping transparency while performing with a gestural controller. The system supports a series of gestural metaphors that are implemented through mapping. Table 4.3 provides an explanation of the metaphors implemented for the work Ataraxia, presented below.

Metaphor	Description
Scroll	Scroll within the range of the sound file.
Bend	Bending the pitch of the grain.
Stretch	Stretch the sample. Imitate stretching by moving hands in opposite directions.
Supress	Step in and out when sound occurs, supress the grain with the left hand.

Table 4.3 Metaphors in Greap.

4.8. Music composed with Greap

Ataraxia[45] (2014) is the first musical work composed with Greap. It is a semi-improvised composition structured in five scenes, each consisting of five separate metaphors. Although this structure is fixed, the actual interpretation of the indicated gestures within a scene depends on the performer's approach. Ataraxia enacts the proliferation of magic or augmenting reality. The performer appears to be a conjurer that shapes the sound, as it was an immutable object while this object is connected to the characteristics of the resulting sound. It is worth mentioning however, that even following accurately the

[45] A video with the full recording of the piece is included in the accompanying DVD and online at this link: https://vimeo.com/87510975

instructions that are provided in the score, the performer must keep in mind that in order to perform magic efficiently depends on the ability to sustain theatricality/dramaturgy throughout the performance, and thus this is the main priority and rule that the performer must comply with. Some strategies that appeared to be helpful for the successful realisation of Ataraxia are outlined below:

1. *Technological discretion*. The hardware must be hidden, for example cables, audio interfaces, and computers must not be apparent to the audience. From the audience's point of view people witness a clear/lucid connection between the hand movements and the resulting sound without being able to see the medium that creates it implying that the sound is being created by the bear hands of the performer, augmenting mysticism. This however, is also depended on the visual aspect of the performance. For example, the performer must not only be focussed on the control of the parameters of the environment, but s/he must be also able to employ the appropriate body language that demonstrates tension and effort of the whole body, this is totally depended on the next strategy.

2. *Listening to the sound, trust your ears*. Providing that there is no computer on stage that provides visual feedback from the responses of the system, the performer learns to adapt his/her movements through listening to the outcome of the sound.

3. *The ability of the performer to be theatrically vigorous*. Assessing numerous performances of Ataraxia show that in order the piece to be compelling and the performer is able to perform magic successfully, s/he needs to be able to impose some acting ability. For example, employing the whole body and create tension, as well as keeping eye contact with the audience. This is crucial to maintain theatricality and expressivity throughout the performance. In case the performer fails to sustain this there is a risk of spoiling the appearance of magic that is intended for the piece.

4. *Preservation of the guidelines of the score.* To keep the form of the piece intact the performer needs to preserve the instructions of the score, including the sound material that was selected for the piece. The performance instructions are provided in a graphical score that serves as a guide to the performer (attached in the Appendix – Ataraxia, Score of Ataraxia). Each movement contains instructions about durations of each scene, rests, hand gestures and their trajectories, names of the audio samples and mappings. However, within these constraints, the performer has the freedom to improvise. The instructions given by the score ensure that the piece is repeatable and recognisable.

5. *Performance through metaphors.* The piece implements a series of metaphors that the performer has to perform. Complete details about the metaphors implemented in Ataraxia are discussed later in this section.

6. *Scenes, grouped interaction affordances and structured improvisation.* As already noted Ataraxia is divided in separate scenes, providing specific directions and information that help to build the context of the piece. These include trajectories of the hands, specific sounds that are used in scenes, durations of the scenes, and the execution of rests during the performance. This information helps to guide the performer in certain paths throughout the performance rather than block his/her imagination. The scenes including all information that comes with them create diverse sets of interaction affordances; the performer is practising them, which helps them to display a sort of virtuosity while playing with the environment. The implementation of scenes allows building the context of the piece using directed improvisation following the guidelines that were devised for Ataraxia.

According to some confessions of the audience the performance of Ataraxia enacts a sense of conjuring. On the light of these confessions, it allows me to suggest that this

composition is successful according to my personal ambitions and research objectives, including expressivity and theatricality.

In order to enhance expressivity and create transparency of mapping, the following metaphors were implemented. Timings refer to the documented performance.

Scroll (0'10" - 1'30") the performer places the hands over the LM and moves them in a vertical trajectory controlling the lower and higher read positions of the grain. This corresponds to a visual metaphor of the reading position within the granulated sample.

Bend (1'30" - 2'54") uses the left hand to manipulate the pitch deviation of the grains. The higher the position of the hand the greater the deviation of the rate will be.

Stretch (2'55" - 4'25") the performer moves the hands in a vertical trajectory where both hands move in opposite directions in order to control time stretching: the wider the distance between the two hands the greater the stretch factor.

Suppress provides a dramatic scenario. The performer acts as if scared of the sound grains and trying to reach them (6'18" - 6'25"). Once s/he becomes familiar with this reaction, s/he tries to interact with the grains by increasing and decreasing their density as well as their pitch deviation and duration using the horizontal position of the hands (6'31").

4.9. Conclusion

For many years there has been a trade-off between complexity and timbre versatility, and the possibility of manipulating sound on the fly. Greap addresses this problem by switching to different mappings without interrupting a performance. Furthermore, the performer can develop and manipulate versatile timbre structures similar to those that are made in the studio: with dynamic changes of multiple settings and configurations, the composer/performer is able to access a wider range of sonic manipulation. Therefore, while Greap was mainly developed for live use, complexity is not sacrificed.

The musical result is the creation of idiomatic pieces that are consistent with the interface and the medium that was used to create it. Thus, the music that is created using an interface such as LM is highly gestural – the movements of the performer are reflected in the sounds, leading to a causal relationship between the former and the latter. Moreover, sound morphology is depicted by the fast changes that the performer may achieve due to the ability to make rapid manipulations of the main synthesis parameters (as well as of the auxiliary controls), as opposed to other performance paradigms examined in this research. Additionally, by using scenes the user may store multi-parameter functions such as mappings and other pertinent information, in order to create more complex pieces.

Using optical interfaces provides a large degree of freedom in regard to the performer's gestures and movements. However, this is possible only when the performer keeps the hands within the appropriate range of the device. There is neither a specific framework nor physical constraints that the user is aware of during the performance, thus the only way to make sure that the system is responding properly is through the produced sound. A strategy that relies solely on the sonic output of the system while performing was followed in this project.

Although there is no requirement for physical effort, mapping was developed to enhance expressivity by implementing a series of metaphors.

While Greap is fully functional, there is still some room for development, and this has become apparent through composition and performance, requiring a constantly evolving process of metaphor development and adaptation to new musical requirements. Some future refinements of the system will include its modification of the system to facilitate a more flexible mechanism for mapping which will be more accessible to novel users. This will be possible through the implementation of a matrix where the user will be able to bind LM's variables with the synthesis parameters of Greap. It will provide a visual

representation of the mapping and will help to build it without the need to deal with code. A similar approach is implemented in MAES (Fischman, 2013, p.334).

Figure 4.3 shows an under development implementation of the mapping in future version of Greap environment.

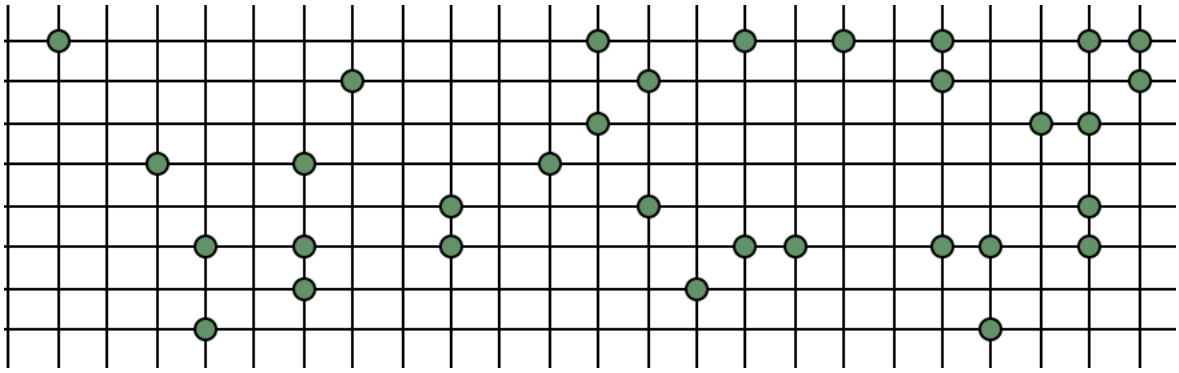


Figure 4.3 Matrix of the mapping in future version of Greap. Image produced using ixiViews[46] quark in SuperCollider.

[46] See https://github.com/supercollider-quarks/ixiViews

5. Conclusions – discussion of musical implications

In order to shed light on the background and processes that underpin the music systems in this thesis, this commentary has reported on research on current representative performance paradigms using various interfaces for real time interaction with computer-based musical environments. A number of studies were carried out to investigate the musical feasibility of readily available interfaces, using these to develop performance environments that employ improvisation as the basis for real time composition, with an emphasis on interactive features.

Each environment was optimised for different forms or variants of musical practice. Looking at the musical outcomes of each performance paradigm it becomes apparent that the medium that is used to drive or interact with the environment has a great effect on the characteristics of composition. This view is also supported by Vaughan (1994).

The music I have composed during my PhD is created by the need to express human spontaneity through computer-based environments that are able to provide intuitive communication with sound. Whether this is through the employment of physical gestures or via text-based interfaces such as code, my vision was to create the means to employ spontaneity, imagination and dexterity while improvising with computer environments. Similarities can be seen with the paradigm of a dexterous improviser performing on his or her acoustic instrument.

The musical environments I developed throughout my PhD helped me to investigate improvisation in various ways including free improvisation using the Wiimote as extension to my body, or structured improvisation following a set of instructions and the organisation of gestural affordances. Creating instruments and musical environments from scratch or redefine their structure as a performance strategy; conjoining various control inputs such as environmental conditions, and human agencies to interact with computer-based environments optimised for real time composition and improvisation. Further, my

research concerned a lot about how to interact with the sound in an accurate manner. Micro gestures corresponding to fine grains of sound that create a link between the minor movements and the sound outcome.

In one of his lectures, Varese stated that "I dream of instruments obedient to my thought and which with their contribution of a whole new world of unsuspected sounds, will lend themselves to the exigencies of my inner rhythm" (Wen-chung, 1966, p.11).

To achieve this the performer has to create the means to translate imagination to the digital domain, a sort of humanising technology. Thus, whilst the current exploration concerned technology, however my purpose was not to create state of the art computer environments, but rather to examine readily available means and use them to fulfil my artistic and musical needs.

Through the use of various interfaces and musical environments I was able to express myself according to the various strengths and competences that each environment offered. Therefore, the systems provided me with the means to stream my performance and manifest my musical imagination in different ways[47] i.e. using the expression of my gestures to interact with the music media. The most suitable path for me to achieve this is through improvisation, using it as a means to express myself in a direct way similar to an instrumental improviser. While I improvise I build a sort of conversational relation with the digital environment, for example, using patterns in SuperCollider I can implement generative processes that evolve on their own which then I am able to intervene and change their structure while I am introducing new elements of sound gradually building a climax. The occurrences during this conversation influence my next movements. When something unexpected occurs, which potentially distracts coherence, I use it and make it fit in the general outcome, embracing the unexpected.

[47] According to their idiosyncratic features.

Thus, my approach in improvisation deals solely with sound through which I explore myself and expose my thoughts to the audience at the same time, a sort of mirroring of my inner world through the digital environment I am using. I have explored improvisation in various ways through which I composed a series of pieces that were created live or composed later in a studio using material that originated from live performances.

I personally believe that a performance environment must be compelling and engage its user musically and make them like to *play*[48] it. When I was starting my PhD (and before), as a performer, I did not feel musically stimulated by a performing environment that required merely waving my hands and making sounds freely, while nodding my head in front of the audience. I felt this would put me in a situation of demonstrating the musical environment rather than expressing my artistic idiosyncrasies. As a performer I believe that improvising without a basic structure can lead to the blocking of imagination of the improviser during the performance. On the contrary, by following some trivial rules while improvising, not only I felt more comfortable on stage, my performance could benefit much more establishing a territory of specific goals and musical qualities or criteria that need to be fulfilled. Such clear intention will help the audience to judge if the performance was successful or not.

Therefore, I started importing into my pieces some basic performance scenarios and help to organise my improvisation in order to go a step beyond simply wandering on stage. An example of this approach was used first in Stay On This Gesture, followed by a more refined realisation of this later in Greap. It allowed me to structure improvisation in separate blocks consisting of separate improvised parts to build the composition in real time. Each part allowed a certain degree of free interpretation of the guidelines, which I

[48] Used with the literal meaning of the word, i.e. engaging, interesting to explore or investigate various aspects of something. In the context of a musical environment this could be translated to something that inspires the user with its sound qualities while interacting with it, but also unpredictable, for example, some nice occurrences of textures that the performer had not preconfigured while designing it, but it happened through the interrelation/correlation of the parameters that comprise the musical environment, or even an interesting glitch.

was able to interpolate in a given timeline. While I did not want to constrain myself with certain fixed rules, I needed something to guide me throughout the performance. Boden states that constraints are “far from being the antithesis of creativity, constraints on thinking are what make it possible. Constraints map out a territory of structural possibilities which can then be explored, and perhaps transformed to give another one” (Magnusson, 2010, p.63). I find this true of my own performances.

Although the majority of the current work was created live, I have also created a series of fixed media pieces composed in the studio. These pieces helped me to reflect on my live practice through a detailed examination of the material afterwards in the studio. This retrospection helped me to take a step back, gain control over the material and inform my live performance. For example, while composing a piece in the studio and needing a specific musical passage or texture, I created it using live coding, sometimes combined with a gestural controller if I needed to convey a specific *gesture*[49] in the morphology of the sound. Once the material was created I processed it further, using filtering, pitch-shifting or edit duration of the sounds in order to fit in the composition. This approach resembled the assembly of a puzzle, which specific pieces shape the general image, but in the case of the music these pieces had to be created.

As stated previously diverse paradigms of performance formed the ground to investigate various aspects of improvisation and instigate various modes of musical practice. PoP allowed me to create a real time composition based on the response to the spontaneity of the multiple participants in conjunction with the influence of the environment. Using PoP allowed me to investigate affordances that were not possible in the other paradigms examined in this research, these included multiple control agencies rather than single performance interaction. Through PoP I was also able to approach the composition in a

[49] In fixed media composition this word is used to express a sound that is characterised by gestural quality.

more relaxed way rather than having to perform in a formal manner in front of an audience.

Alternatively, using the Wiimote and the Nunchuk enacted a causal relationship between my physical gestures and the resulting sound. The devices were used as prosthesis to my body rather than a separate device, similar to the bow of a violinist. BiGrain allowed me to build an interaction relationship similar to the performance with a traditional instrument e.g. hearing the responses of the system caused by my actions, and constantly trying to adjust them while improvising. Thus, every movement of my hands created a sonic manifestation emerging from my imagination.

The Song has Sung was composed following a different mapping strategy than the one followed in BiGrain. Instead of translating my gestures directly to sound using one-to-one mapping with the synthesis parameters, I engaged in an interactive loop enacted by the complex calculations of the system and the human input. System and performer are equally influenced by the responses of each other, creating a mutual intervention on stage. The question that arose was who is in control of the performance, is it the system that encourages the performer with its complex sound layers that conjoin to each other, or is it the performer trying to influence the system and inject the human input in this endless continuum. Whichever way, performing with Stay On This Gesture led me to paths otherwise unexplored. The patterns extended my compositional imagination; by employing their complexity within my live performance I broadened my improvisational exploration.

Evidently, 'live coding provides one fertile solution to the problem of interface design…with rich implications for improvisational practice' (Wilson et al., 2014, p.54). It allows synthesising sounds from scratch or improvising source code on stage. In my personal practice I have used and demonstrated live coding as a tool to define and create interaction affordances that are detached from previous decisions by creating hybrid

environments employing hardware devices, which then I improvised their mapping live. This allowed me to explore a wider range of interaction possibilities and led me to compose pieces liberated from previous constraints and limitations. Most importantly, live coding allowed me to engage in collective improvisation with others. Through sharing code to the rest of an ensemble I was able to interact and express my musical imagination while contributing to a generic sonic outcome in a very coherent and specific manner, something that in other laptop performances is absent.

However, one of the main limitations that I felt during live coding was the inability to translate all the sounds that I had in my mind during improvisation, and the relatively slow speed reaching the kinds of the sound that I was imagining. Thus, while live coding is an unlimited interface that potentially lets me implement every possible sound live, it requires a sort of expertise in programming. I approached this by exercising every day. I experimented with various implementations of synthesis code and other generative processes, and I created an arsenal of tricks[50] that I could vary live according to the needs of the performance. This could be viewed as similar with the paradigm of the practice of an acoustic instrumentalist.

For many years there has been a trade-off between complexity and timbre versatility, and the possibility of manipulating sound on the fly. In Greap, using scenes I am able to pre-compose the interaction affordances and plan my improvisation in advance of the concert without sacrificing complexity on stage. Ataraxia, composed with Greap, illustrates the accurate representation of my decisions taken prior to the performance, which the environment allows me to store them and use repeatedly. Most importantly, while performing with Greap I was able to translate same gestures to result in diverse sound manipulations and organise the piece, something that in other performance paradigms examined in this research was not possible. Finally, using a specific score that I created for

[50] Some are available online at this link: https://github.com/KonVas/Ionio-liveCode-workshop new comers in live coding might find those interesting to experiment and adapt according to their needs.

the piece I managed to keep the form of the piece intact, maintaining the repeatability and recognisability of the composition even after many performances. Thus, the piece can be distributed and performed by other laptop artists.

Another aspect of the projects was theatricality. Each performance paradigm provided a degree of spectacle that emerged from the nature and the characteristics exhibited by each interface. For example, theatricality in live coding is enacted by the projection of the code. The audience can follow the implication of the code and how this affects the sonic outcome. In the case of tangible interfaces the theatricality is apparent by the appearance of effort using physical energy to manipulate the sensors of the interface and change of the mechanical state of the device.

Performing with optical interfaces, I achieved a high degree of expressivity enacted by the lucid connection between the gestures and the resulted sound. Adopting the use of metaphors (i.e. gestures that convey universal meaning, such as rub, spin, and twist) translating these into sound advanced the dramatic spectacle and theatricality. Ataraxia highlights this by adopting two approaches: avoiding any technological and hardware devices (including the computer) around the interaction area creating a magical/mystical representation of gestures to sound, and keeping an air of mystery. This is partly achieved by my decision to rely only to the sound outcome instead of looking at graphical representations on a screen, thus to monitor the systems' responses through the sound. This also allows eye contact with the audience, which is paramount to enhance the dramatic aspect during performance.

In the case of PoP, the movements and the actions of the people to stimulate and interact with the installation are the theatre. To interact with the installation the people wander around in order and play with the system. During this interaction some cumbersome elements were apparent since the affordances of the system were unknown to the participant appear to create an impact on the sonic outcome. This theatrical scenery

appeared to offer musical unpredictability due to the awkward interaction of the people and the system. Observing the installation it came to me that people approached the sound installation as a sort of a playground, where people stimulated the system and interacted with the sound with a playful but exploratory mood in the same time. Judging the joyful engagement of the people with the system, I consider the installation successful.

5.1. Encore

To conclude, during one of his lectures Varese stated that:

> "We should also remember that no machine is a wizard, as we are beginning to think, and we must not expect our electronic devices to compose for us. Good music and bad music will be composed by electronic means, just as good and bad music have been composed for instruments. The computing machine is a marvellous invention and seems almost superhuman. But, in reality, it is as limited as the mind of the individual who feeds it material. Like the computer, the machines we use for making music can only give back what we put into them." (Wen-chung, 1966, pp.18-19)

Embracing this idea, in my performance practice during this PhD, I did not expect the tools to compose the music on their own, by using their cerebral aptitudes without a sort of human touch. I wanted to exploit and take advantage of the modern media in order to explore new leads in improvisation and music composition using interactive media.

6. Further research

I will continue to refine the digital environments by adapting them to the constant compositional challenges as I expand my musical practice. Further research may elaborate in the investigation of digital environments that will provide intelligent approaches optimised for improvisation and real time composition. For example, systems that will be able to take decisions and adapt their behaviour according to the input data using gesture

recognition or feature extraction of control signals is worth exploring. Finally, I will refine the current digital environments in order to be used by other musical practitioners with diverse technical background.

References

Alles was wir haben. 2004. [video] Directed by V. Kamensky. Available at: <http://swiki.hfbk-hamburg.de/MusicTechnology/491> [Accessed 2 Aug. 2015].

Anderson, C., 2005. Dynamic Networks of Sonic Interactions: An Interview with Agostino Di Scipio. Computer Music Journal, [online] 29(3), pp.11–28. Available at: <http://www.mitpressjournals.org/doi/abs/10.1162/0148926054798142> [Accessed 2 Aug. 2015].

Szwed, J.F., 2002. *So what: the life of Miles Davis*. New York: Simon & Schuster.

Bertelli, E., 2013. Drumactica 2.0. [online] Available at: <https://www.youtube.com/watch?v=zMkoQMTWUeY> [Accessed 8 Aug. 2015].

Bevin, G., 2014. GECO. [MacOSX] UWYN. Available at: <http://uwyn.com/geco/>.

Bisig, D.,, Schacher, J., and Neukom, M., 2011. Flowspace – A Hybrid Ecosystem. In: Proceedings of the International Conference on New Interfaces for Musical Expression. [online] NIME2011. Oslo, Norway, pp.260–263. Available at: <https://www.zhdk.ch/fileadmin/data_subsites/data_icst/Publikationen/2011_NIME11_Hybrid_Ecosystem_Bisig.pdf> [Accessed 2 Aug. 2015].

Blackwell, A., and Collins, N., 2005. The programming language as a musical instrument. In: P. Romero, J. Good, A. Chaparro and S. Bryant, eds., Proceedings of Psychology of Programming Interest Group. [online] PPIG2005. pp.120–130. Available at: <http://www.ppig.org/papers/17th-blackwell.pdf> [Accessed 2 Aug. 2015].

Chadabe, J., 2002. The Limitations of Mapping as a Structural Descriptive in Electronic Instruments. In: Proceedings of the International Conference on New Interfaces for Musical Expression. [online] NIME2002. Dublin, Ireland, pp.38–42. Available at: <http://www.nime.org/proceedings/2002/nime2002_038.pdf> [Accessed 2 Aug. 2015].

Cycling74, 1998. Max/MSP. [MacOSX] California, USA: Cycling74. Available at: <https://cycling74.com/> [Accessed 2 Aug. 2015].

De Campo, A., Rohrhuber, J., Bovermann, T., and Frauenberger, C., 2011. Sonification and Auditory Display in SuperCollider. In: S. Wilson, D. Cottle and N. Collins, eds., The SuperCollider Book. Cambridge: MA: MIT Press.

De Campo, A.,, Vacca, A.,, Hoelzl, H.,, Ho, E.,, Rohrhuber, J., and Wieser, R., 2007. Code As Performance Interface - A Case Study. In: *Proceedings of the International Conference on New Interfaces for Musical Expression*. NIME. New York: Unpublished.

Elevator to the Gallows. 1958. Directed by L. Malle. Lux Compagnie Cinématographique de France. Available at: <http://rialtopictures.com/elevator.html> [Accessed 11 Aug. 2015].

Fels, S., Gadd, A., and Mulder, A., 2002. Mapping transparency through metaphor: towards more expressive musical instruments. Organised Sound, [online] 7(02). Available at: <http://www.journals.cambridge.org/abstract_S1355771802002042> [Accessed 2 Aug. 2015].

Fischman, R., 2013. A Manual Actions Expressive System (MAES). Organised Sound, [online] 18(03), pp.328–345. Available at: <http://www.journals.cambridge.org/abstract_S1355771813000307> [Accessed 2 Aug. 2015].

Fujimoto, R., 2013. Humanelectro × Leap Motion. [video online] Available at: <https://www.youtube.com/watch?v=-W_NYbPpkPQ&feature=youtube_gdata_player> [Accessed 8 Aug. 2015].

Gadd, A., and Fels, S., 2002. MetaMuse: metaphors for expressive instruments. In: Proceedings of the 2002 conference on New interfaces for musical expression. [online]

NIME2002. Dublin, Ireland: National University of Singapore, pp.1–6. Available at: <http://dl.acm.org/citation.cfm?id=1085206> [Accessed 2 Aug. 2015].

Hermann, T., Hunt, A. and Neuhoff, J.G. eds., 2011. The sonification handbook. First ed. [online] Berlin: Logos Verlag. Available at: <http://sonification.de/handbook/index.php/downloads/> [Accessed 2 Aug. 2015].

Hoenig, U., 2014. Leap Motion gesture control jam (Geco, Reaktor, Live). [video online] Available at: <https://www.youtube.com/watch?v=Q8AxhbCL-rM> [Accessed 8 Aug. 2015].

Hugill, A., 2012. The digital musician. Second edition ed. New York ; London: Routledge.

Hunt, A.,, Wanderley, M., and Kirk, R., 2000. Towards a model for instrumental mapping in expert musical interaction. In: Proceedings of the 2000 International Computer Music Conference. [online] ICMC2000. Berlin, Germany, pp.209–212. Available at: <http://www.ircam.fr/equipes/analysesynthese/wanderle/Gestes/Externe/Hunt_Towards.pdf> [Accessed 2 Aug. 2015].

Hunt, A., and Wanderley, M.M., 2002. Mapping performer parameters to synthesis engines. Organised Sound, [online] 7(02). Available at: <http://www.journals.cambridge.org/abstract_S1355771802002030> [Accessed 2 Aug. 2015].

Hunt, A., Wanderley, M.M., and Paradis, M., 2003. The importance of parameter mapping in electronic instrument design. Journal of New Music Research, [online] 32(4), pp.429–440. Available at:<http://www.tandfonline.com/doi/abs/10.1076/jnmr.32.4.429.18853> [Accessed 2 Aug. 2015].

Ma, 2013. Touchless at NIME show 2013 Live. [video online] Available at: <https://vimeo.com/81973975> [Accessed 3 Aug. 2015].

Magnusson, T., 2010. Designing constraints: Composing and performing with digital musical systems. *Computer Music Journal*, [online] 34(4), pp.62–73. Available at: <http://www.mitpressjournals.org/doi/pdf/10.1162/COMJ_a_00026> [Accessed 13 Aug. 2015].

Magnusson, T., 2011. ixi lang: a SuperCollider parasite for live coding. In: Proceedings of International Computer Music Conference. [online] ICMC2011. Huddersfield, UK: University of Huddersfield, pp.503–506. Available at: <http://sro.sussex.ac.uk/46869/> [Accessed 2 Aug. 2015].

Malloch, J.,, Sinclair, S., and Wanderley, M.M., 2013. Libmapper: (a library for connecting things). In: Extended Abstracts on Human Factors in Computing Systems. [online] CHI. Paris, France: ACM Press, p.3087. Available at: <http://dl.acm.org/citation.cfm?doid=2468356.2479617> [Accessed 2 Aug. 2015].

Manning, P., 2004. Electronic and computer music. Rev. and expanded ed. Oxford ; New York: Oxford University Press.

McCartney, J., and others, 2014. SuperCollider. [MacOSX] Available at: <http://supercollider.github.io> [Accessed 2 Aug. 2015].

Miranda, E.R., and Wanderley, M.M., 2006. New digital musical instruments: control and interaction beyond the keyboard. The computer music and digital audio series. Middleton, Wis: A-R Editions.

Mulder, A., and Fels, S., 1998. Sound sculpting: Manipulating sound through virtual sculpting. In: Proceedings of the 1998 Western Computer Graphics Symposium. [online] WCGS1998. pp.15–23. Available at: <http://www.xspasm.com/x/sfu/vmi/ss/WCGS98-p.pdf> [Accessed 2 Aug. 2015].

Nilson, C., 2007. Live coding practice. In: Proceedings of the 7th international conference on New interfaces for musical expression. [online] NIME2007. ACM Press, p.112. Available at: <http://portal.acm.org/citation.cfm?doid=1279740.1279760> [Accessed 2 Aug. 2015].

Paine, G., 2009. Towards Unified Design Guidelines for New Interfaces for Musical Expression. Organised Sound, [online] 14(02), p.142. Available at: <http://www.journals.cambridge.org/abstract_S1355771809000259> [Accessed 2 Aug. 2015].

Pressing, J., 2001. Improvisation: methods and models. In: J. Sloboda, ed., *Generative Processes in Music. The Psychology of Performance, Improvisation, and Composition*. [online] Oxford ; New York: Oxford University Press, pp.129–178. Available at: <http://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780198508465.001.0001/acprof-9780198508465> [Accessed 14 Aug. 2015].

Puckette, M., 1996. Pure Data. [online] Available at: <http://puredata.info/> [Accessed 2 Aug. 2015].

Roads, C., 1996. The computer music tutorial. Cambridge, Mass: MIT Press.

Rohrhuber, J., de Campo, A., and Wieser, R., 2005. Algorithms today notes on language design for just in time programming. In: *Proceedings of the ICMC 2005*. [online] ICMC 2005. Barcelona, Spain, pp.291-294. Available at: <http://web.cecs.pdx.edu/~dreeder/site/nysc/doc/rohrhuber,etal--jit.pdf> [Accessed 11 Aug. 2015].

Rovan, J.B.,, Wanderley, M.M.,, Dubnov, S., and Depalle, P., 1997. Instrumental gestural mapping strategies as expressivity determinants in computer music performance. In: Kansei, The Technology of Emotion. Proceedings of the AIMI International Workshop. [online] Genoa, Italy, pp.68–73. Available at:

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.4788&rep=rep1&type=pdf> [Accessed 2 Aug. 2015].

Sapir, S., 2000. Interactive digital audio environments: gesture as a musical parameter. In: Proceedings COST-G6 Conference on Digital Audio Effects. [online] DAFx'00. Verona, Italy, pp.25–30. Available at: < http://www.researchgate.net/publication/246498426_INTERACTIVE_DIGITAL_AUDIO_ENVIRONMENTS_GESTURE_AS_A_MUSICAL_PARAMETER > [Accessed 2 Aug. 2015].

Studer, A., 2010. *Simon Emmerson on live computer music*. [online video] 28 Apr. Available at: <https://www.youtube.com/watch?v=78z1_8J8oVE> [Accessed 2 Aug. 2015].

Di Scipio, A., 2003. 'Sound is the interface': from interactive to ecosystemic signal processing. Organised Sound, [online] 8(03). Available at: <http://www.journals.cambridge.org/abstract_S1355771803000244> [Accessed 2 Aug. 2015].

Di Scipio, A., 2011. Listening to Yourself through the Otherself: On Background Noise Study and other works. Organised Sound, [online] 16(02), pp.97–108. Available at: <http://www.journals.cambridge.org/abstract_S1355771811000033> [Accessed 2 Aug. 2015].

Di Scipio, A., 2014. The place and meaning of computing in a sound relationship of man, machines, and environment. In: Proceedings of ICMC|SMC|2014. [online] ICMC2014. Athens, Greece, pp.47–53. Available at: <http://www.smc-conference.net/smc-icmc-2014/papers/images/VOL_1/0047.pdf> [Accessed 2 Aug. 2015].

McCallum, L., and Smith, D., *Show Us Your Screens*. 2011. [video] Available at: <https://vimeo.com/20241649>.

Silva, E.S., de Abreu, J.A.O., de Almeida, J.H.P., Teichrieb, V., and Ramalho, G.L., 2013. A preliminary evaluation of the leap motion sensor as controller of new digital musical instruments. [online] Centro de Informática – Universidade Federal de Pernambuco (UFPE). Available at: <http://compmus.ime.usp.br/sbcm/2013/pt/docs/art_tec_1.pdf> [Accessed 3 Aug. 2015].

Sorensen, A., 2005. Impromptu: An interactive programming environment for composition and performance. In: Proceedings of the Australasian Computer Music Conference 2009. [online] ACMC 2005. Brisbane, Australia, pp.149–153. Available at: <http://eprints.qut.edu.au/31056/> [Accessed 3 Aug. 2015].

Toplap, 2010. *ManifestoDraft*. [Wikipedia] Toplap. Available at: <http://toplap.org/wiki/ManifestoDraft> [Accessed 19 Aug. 2015].

Troillard, C., 2011. OSCulator. [computer program] Wildora. Available at: <http://www.osculator.net/> [Accessed 3 Aug. 2015].

Troillard, C., 2012. *OSCulator user's manual*. [online] OSCulator's User's Manual. Available at: <http://s3.amazonaws.com/osculator/doc/OSCulator+2.12+Manual.pdf> [Accessed 9 Aug. 2015].

Varese, E., and Wen-chung, C., 1966. The Liberation of Sound. *Perspectives of New Music*, [online] 5(1), p.11. Available at: <http://www.jstor.org/stable/832385?origin=crossref> [Accessed 13 Aug. 2015].

Vasilakos, K., 2014. Sculpting Sound in Real Time with SuperCollider. [blog] *Leap Motion Blog*. Available at: <http://blog.leapmotion.com/sculpting-sound-real-time-supercollider/> [Accessed 5 Aug. 2015].

Vaughan, M., 1994. The human-machine interface in electroacoustic music composition. Contemporary Music Review, 10(2), pp.111–127.

Vertegaal, R.,, Ungvary, T., and Kieslinger, M., 1996. Towards a musician's cockpit: Transducers, feedback and musical function. In: *Proceedings of the International Computer Music Conference*. [online], pp.308–311. Available at: <http://www.cs.queensu.ca/~roel/publications/ICMC96/paper.html> [Accessed 11 Aug. 2015].

Walker, B.N., and Kramer, G., 2005. Mappings and metaphors in auditory displays: An experimental assessment. *ACM Transactions on Applied Perception*, [online] 2(4), pp.407–412. Available at: <http://sonify.psych.gatech.edu/publications/pdfs/2005TAP-WalkerKramer-ICAD1996paper.pdf> [Accessed 3 Aug. 2015].

Wanderley, M.M., 2001. Gestural control of music. In: International Workshop Human Supervision and Control in Engineering and Music. [online] Kassel, Germany, pp.632–644. Available at: <http://kkothman.iweb.bsu.edu/oldTeaching/mumet440/papers/wanderly-gestcontrol.pdf> [Accessed 3 Aug. 2015].

Wang, G., 2008. A Strongly-Timed, On-the-fly, Environ/mentality. [online] Princeton University. Available at: <http://www.researchgate.net/publication/259326122_The_ChucK_Programming_Language_A_Strongly-Timed_On-the-fly_Environmentality> [Accessed 3 Aug. 2015].

Wang, G., and Cook, P.R., 2004. On-the-fly programming: using code as an expressive musical instrument. In: Proceedings of the 2004 conference on New interfaces for musical expression. [online] NIME 2004. Singapore: National University of Singapore, pp.138–143. Available at: <http://dl.acm.org/citation.cfm?id=1085915> [Accessed 3 Aug. 2015].

Waters, S., 2007. Performance Ecosystems: Ecological approaches to musical interaction. EMS: Electroacoustic Music Studies Network, [online] pp.1–20. Available at:

<http://webuser.fh-furtwangen.de/~friedm/PerformEcosystems.pdf> [Accessed 3 Aug. 2015].

Wessel, D., Wright, M., and Schott, J., 2002. Intimate musical control of computers with a variety of controllers and gesture mapping metaphors. In: Proceedings of the 2002 conference on New interfaces for musical expression. [online] NIME 2002. Singapore: National University of Singapore, pp.1–3. Available at: <http://dl.acm.org/citation.cfm?id=1085213> [Accessed 3 Aug. 2015].

Wilson, S., Lorway, N., Coull, R.,, Vasilakos, K., and Moyers, T., 2014. Free as in BEER: Some Explorations into Structured Improvisation Using Networked Live-Coding Systems. Computer Music Journal, 38(1), pp.54–64.

Wilson, S., and de Campo, A., 2013. *Utopia*. [computer program] Available at: <https://github.com/muellmusik/Utopia> [Accessed 3 Aug. 2015].

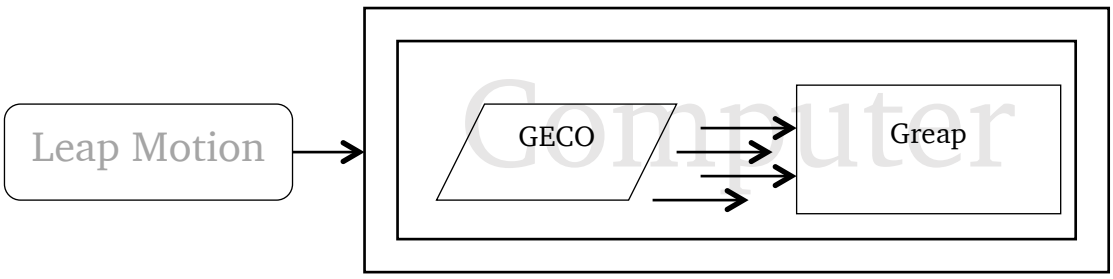
Winkler, T., 1995. Making motion musical: Gesture mapping strategies for interactive computer music. In: ICMC Proceedings. [online] International Computer Music Conference. Banff Centre for the Arts, Canada, pp.261–264. Available at: <http://www.brown.edu/Departments/Music/faculty/winkler/papers/Making_Motion_Musical_1995.pdf> [Accessed 2 Aug. 2015].

Xenakis, I., 1992. Formalized music: thought and mathematics in composition. Rev. ed. Harmonologia series. Stuyvesant, NY: Pendragon Press.

Zmölnig, Io., and Eckel, G., 2007. LIVE CODING: AN OVERVIEW. In: ICMC Proceedings. [online] International Computer Music Conference. Copenhagen, Denmark, pp.295–298. Available at: <http://quod.lib.umich.edu/i/icmc/bbp2372.2007.063/1> [Accessed 3 Aug. 2015].

Appendix – Ataraxia

Software	SuperCollider 3.6 (or above)
	GECO
Files	Greap.scd
Audio files	BasClar.aiff
	bell.aiff
	bidding.aiff
	isThatYou.aiff
	seaRoaring.aiff
	triangle.aiff
Resources	Subduct.scd
	GECOMapSC.geco
	BEERfers.scd
Hardware	Leap Motion
	MIDI foot pedal (optional).



Instructions and technical requirements

Instructions about how to set up Greap environment are as follows. In order to connect the Leap Motion device to the computer you require the software of the device, which is provided by the manufacturer, and can be installed when purchasing the device. To run the environment you will need SuperCollider and a third party application called GECO, which is used to tap the Leap Motion data in SuperCollider.

Current version of Greap has a stereo output. The system may be connected directly to a pair of self-amplified speakers using a mini jack (3.5mm) cable via the line output of the computer's sound card. GECO communicates the data of Leap Motion using MIDI protocol; future versions of Greap will use Open Sound Control (OSC) protocol as current version of GECO supports it.

To perform Ataraxia you will need a selection of sounds that were used for the piece, which are placed in a folder called 'sounds' in the root folder of the project. Additional files of the environment can be found in a folder named 'resources' inside the Greap[51] folder. Move both files Subduct.sc and BEERfers.sc from the resources folder to the SuperCollider extensions[52] folder. On a Mac, it is in the following path name:

```
Username/Library/Application Support/SuperCollider/Extensions
```

How to launch Greap

Open Greap.scd with SuperCollider. To run the environment press (Mac) ⌘ + A, and then ⌘ + Enter. Greap will launch GECO by loading the GECOMapSC.geco file automatically, which contains configuration for the mapping of Leap Motion to SuperCollider. If everything has gone as expected SuperCollider must be running the environment and you

[51] Included in the accompanying DVD.
[52] In case this file does not exist you have to create it.

are looking to a graphical user interface (GUI). It provides some faders and buttons, which can be used for testing purposes. Providing that you are in the first scene, movement along the x-axis of the left hand is controlling the start of the reading position of the sound sample and movement along the x–axis of the left hand is controlling the end of the reading position of the sound. The rest of the parameters of the synth remain fixed until you switch to the next scene, which provides interaction with other parameters. For complete details about the mapping of the environment see score below.

Troubleshooting

In the event that SuperCollider fails to start GECO you may launch the application and load the GECOMapSC.geco file manually. It is recommended to start GECO before SuperCollider.

For accurate realisation of Ataraxia you must use the sounds that were selected for the piece. Should you want to create your own version and use other sounds, replace the current ones with yours. If SuperCollider fails to produce any sound, make sure that these are monophonic sounds and are placed in the correct location, that is inside the root folder of the project.

If Leap Motion is functioning erratically, for example it fails to track your hands accurately, it is worth calibrating the device, to do so follow the instructions of the Leap Motion software in your computer or consult the official website[53].

Instructions for the performance of Ataraxia

Ataraxia is composed in five scenes or movements; each scene uses a different sound, and the environment will select it automatically by using the name of the audio file, denoted in each scene. The performance involves the use of gestural metaphors, which need to be

[53] It is worth visiting this page especially if this is the first time you are using the device: http://blog.leapmotion.com/troubleshooting-guide-vr-tracking/

performed as instructed in the score. This is crucial for the accurate realisation of the piece. For details about the implementation of the metaphors see Table 4.3 and section 4.8.

Configuring the performer: this score is to assist the performer to perform the piece Ataraxia. It contains figures and instructions as well as information about the mappings, audio samples, duration of the scenes, and hand trajectories. The performer is advised to make rests in each scene. This can be achieved by keeping Leap's interaction area[54] clear.

The notated rests given in the score are crucial and help to preserve the structure of the piece intact after various performances. To maintain the structure of the composition the performer has to sustain the connection between the scenes: when a rest sign is not notated in the score the performer has to keep the flow of the sound without interrupting the sound. It is important that the transitions between the scenes are performed without interruption where required.

To enable versatile interactivity, each scene enables different mappings i.e. the same gesture(s) may be coupled to other parameters; therefore the same gestures result in various sonic manipulations. However, some mappings remain fixed during the performance. These include gate, which is activated by the left hand. Panning: in the stereo image, which is coupled with the vertical trajectory of the left hand, and volume: that is controlled via an external foot switch, or with any other viable means (i.e. the mouse of the computer). Although the duration of the scenes is fixed, the performer is free to improvise, however, the duration of each scene must not be less than 2 seconds or exceed 2.5 minutes, and rests must not exceed 6 seconds.

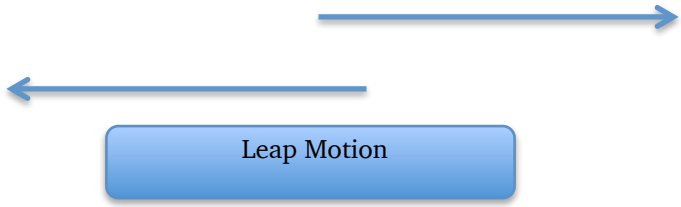
[54] According to the official website of Leap Motion the field of view of the device is two feet above the controller, by two feet wide on each side.

Score of Ataraxia

Scene: 1

Sound: *is that you*

Duration: approx. 2 minutes.

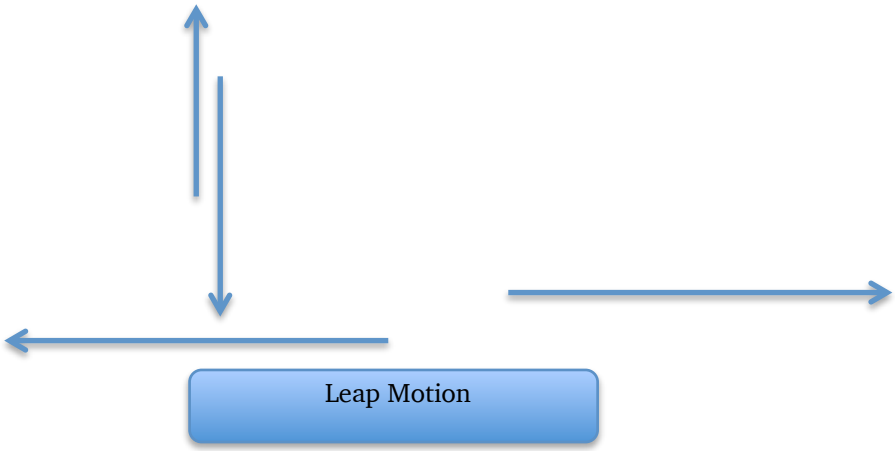


Scroll inside the sound file: both hands move on the vertical position over the Leap's interaction area. Left hand controls the lower position (grain's reading position); right hand controls the end position of the grain. Create rests where appropriate. Move to the next scene without interrupting the sound.

Scene: 2

Sound: *sea roaring*

Duration: approx. 2 minutes.

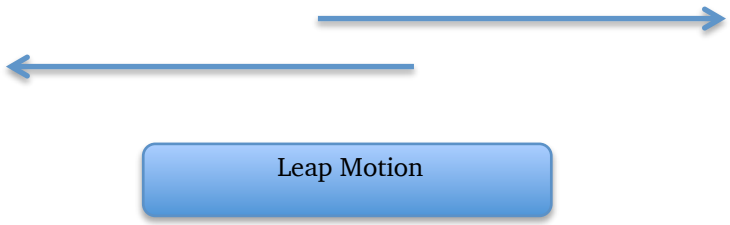


This scene includes the movement of both hands in the vertical position. In addition, it uses the left hand to create an upward & downward motion in order to deviate from the pitch of the grain. If the left hand is down and close to the device, the grain will have its original pitch, moving the hand upward the pitch starts to fluctuate from its original position. Create rests where appropriate. Move to the next scene without interrupting the sound.

Scene: 3

Sound: *bell*

Duration: approx. 2 minutes.



This scene implements a stretching metaphor. The performer has to open and close his hands over the horizontal axis of the Leap, (like s/he tries to stretch the sound with hands). This scene maps the left hand with the number of the grains and the density of the sound. The right hand controls the duration of the grain. Fade out by removing hands slowly.

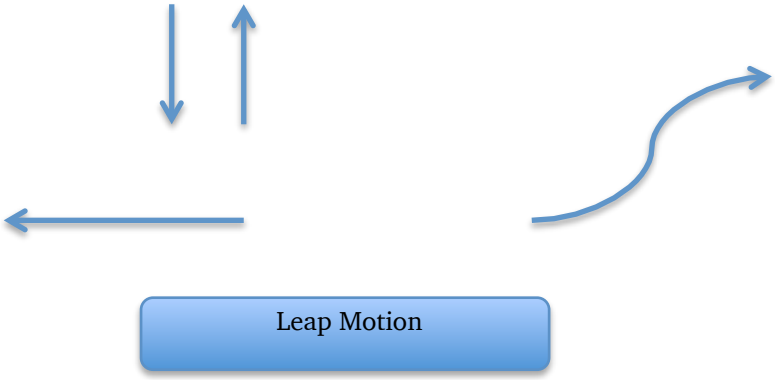
Duration: 5 seconds.

Rest. Keep Leap's interaction area clear.

Scene: 4

Sound: *triangle*

Duration: approx. 2 minutes.



Place the left hand over Leap and trigger the sound; causing the triangle to play for 4 times. Moving the left hand towards the left direction deviates from the current position of the pitch. Moving the right hand diagonally towards the right direction manipulates the grain duration. Move up and down to control the pitch. Remove hands rapidly and finish this scene with tenacity, move to next one after the following rest.

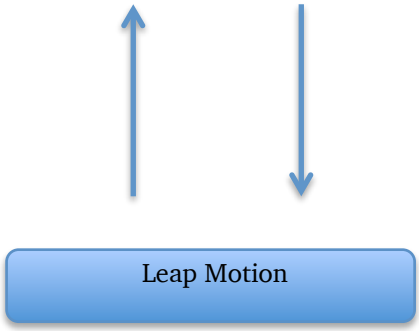
Duration: 5 seconds.

Rest. Keep LM's interaction area clear.

Scene: 5

Sound: *BasClar*

Duration: approx. 2 minutes.



This scene uses only the up and down positions of hands. Pretend that you are afraid of the sound; place slowly left hand in the interaction area, remove it when the sound is triggered and repeat this for couple of times but not more than three times. Then try to suppress the grain by moving the left hand downward. The left hand controls the density of the grains, moving it upward increase the number of the grains. Focus on the thickness of the sound by experimenting with the number of the grains for a while. The right hand controls the pitch deviation of the duration of the grain. Create rests when desired.

Appendix – Blind date

Software	SuperCollider 3.6 (or above)
	OSCulator
Files	Blind date.scd
	Blind date.oscd
	Fair Algo [projection util].scd
Hardware	Wiimote (connectivity: Bluetooth).
	Joystick (Logitech 3D Pro, connectivity: USB)
	Projector

Instructions and technical requirements

Blind date requires a Wiimote and a joystick game device. The piece is a live coding performance for which I have created some preamble code [55] including basic implementation of the mappings of the devices, and sound synthesis engines in SuperCollider.

For the realisation of the piece you will need a third party application called OSCulator. It is used to tap the data from Wiimote in SuperCollider via OSCulator. The device is connected to the computer via Bluetooth, and OSCulator converts it in Open Sound Control protocol (OSC), which can be mapped easily to any software that supports OSC communication.

[55] Included in the accompanying DVD.

If you are using the device for the first time you must pair it with the computer following the instructions provided by OSCulator. Once you have completed the pairing procedure successfully you must be able to see some controls at the OSCulator window illustrating the control variables of the device.

The system has a stereo output. Ideally, the computer will be connected to a mixing board via an audio interface using its main stereo out and two speakers. However, in case there is no audio interface the computer may be connected directly to a pair of self-amplified speakers using a mini jack (3.5mm) cable via the line out of the computer's sound card.

Additional files can be found in the Blind date folder[56]. Open the Blind date.oscd and follow the instructions provided by OSCulator to pair the device with your computer. Open the Blind date.scd file with SuperCollider and press (Mac) ⌘ + A, and then ⌘ + Enter, and wait until SuperCollider loads the environment. If everything has gone as expected, you must be now receiving the data from the Wiimote in SuperCollider. The Blind date.oscd contains boiler code sufficient to start the performance, which creates some connections between the Wiimote and the synthesis engines to start the performance. The performance elaborates in the alteration of the mappings of the device and parameters of the synthesis engines, and the modification of the sound engines themselves. An example of this includes the introduction of new parameters and its mapping with the device in real time and replacing the sound source of a synth (i.e. amend a sine oscillator with a saw oscillator etc.).

Fair Algo

The project uses a projection utility[57], which is used to visualise the names of the dancers that handle the Wiimote during the performance. It is developed in SuperCollider, thus no third party software is needed. The code of the utility must be executed before the

[56] Included in the accompanying DVD.
[57] see Fair Algo code at the end of this appendix.

performance. Open the file named Fair Algo [projection util].scd[58] in SuperCollider, on a Mac press ⌘ + A, and then ⌘ + Enter to launch it.

[58] Found in the Blind date folder included in the accompanying DVD.

Troubleshooting

It is recommended to launch OSCulator before SuperCollider to make sure that the communication between the two software is successful. Sometimes there might be an OSC port mismatch. In case you encounter this problem type the “NetAddr.langPort” command in SuperCollider, the port number that appears in the post window must be the same with the port number that OSCulator is sending the data. For complete details about how to set this in OSCulator consult the user’s manual of the software.

Instructions for the performance of Blind date

As mentioned already at the beginning of this appendix the SuperCollider file (Blind date.scd) included in the folder of the project provides the initial implementation of synthesis engines and mappings between the Wiimote and the parameters of the synths. The performance focuses in the live modification of these mappings but also in the alteration of the synthesis engines. Therefore, the plan is the hacking of the environment in real time as a way to explore its interaction affordances and its musical implications.

Fair Algo code

```
(
v = NetAddr("KV.local", 8000);
v.sendMsg("/vibrate");
v.sendMsg("/blink");

var w, r, a;
w = Window.new("Pass the Wii to:", Rect(100, 100, 400,
200)).front;
a = StaticText(w, Rect(60, 60, 300, 60));
//w.alpha = 0.8;
a.font_(Font( "Monaco", 50.0 ));
r = Routine({|time|
loop({
        var vary = 8.linrand;
        /*if ( vary > 0)
        {
            (
            v.sendMsg("/vibrate");
            0.2.wait;
            v.sendMsg("/vibrate");
            v.sendMsg("/blink");
            );
        };
        */
        (
        if ( vary ==1 ) {a.string = "Antonio"; a.stringColor =
Color.red; };
        if ( vary ==2 ) {a.string = "Petra"; a.stringColor =
Color.black; };
        if ( vary ==3 ) {a.string = "Manou"; a.stringColor =
Color.green;};
        if ( vary ==4 ) {a.string = "Evangelia"; a.stringColor =
Color.blue;};
        if ( vary ==5 ) {a.string = "Mariana"; a.stringColor =
Color.yellow;};
        if ( vary ==6 ) {a.string = "Martine"; a.stringColor =
Color.magenta};
        if ( vary ==7 ) {a.string = "Annelie"; a.stringColor =
Color.cyan;};
        );

        time = 0.5.wait;
    });
    a.string = "done,"; 5.wait; a.string = "Thank You,"; 3.wait;
a.string = "..Goodbye!";
    4.wait;
    w.close;
    r.yieldAndReset(reset:true);
});
AppClock.play(r);
)
//v.disconnect;
```

Appendix – PoP

Software	SuperCollider 3.6 (or above)
Files:	Interface.scd synth.scd
Resources	Firmata.sc Notification.sc
Hardware	See assembly list at the end of this appendix

Instructions and technical requirements

Power of People (PoP) is an interactive sound installation that uses an Arduino Uno board and a set of sensors to track motion, light and temperature. For a detailed list of the hardware that is required to implement the system see the assembly list at the end of this appendix.

The system has a stereo output. The computer may be connected directly to a pair of self-amplified speakers using a mini jack (3.5mm) cable via the line out of the computer's sound card. In addition, the system uses the embedded microphone of the computer to analyse its output, in case there is no microphone on the computer you must provide one and change the code according to your configuration, for example, plug a microphone in an audio interface and connect it to your computer.

All the necessary files and additional resources of the project are inside the PoP folder, which is included in the accompanying memory stick. Before launching the system you need to move the additional files named Firmata.sc and Notification.sc in the

Extensions folder, on a Mac computer the folder[59] is in the following path name.

```
Username/Library/Application Support/SuperCollider/Extensions
```

Once you have made all the connections (see hardware overview and assembly list below) plug the Arduino board to the computer via USB. To run the system you will need SuperCollider. Open the interface.scd file with SuperCollider and select the entire document, on a Mac press ⌘ + A, and then ⌘ + Enter, and wait until SuperCollider loads the environment. The sound engine of the system will be loaded automatically. If everything has gone as expected you must now see a graphical user interface with faders moving according to the input of the sensors. The first fader represents a light sensor, which is mapped to the pitch of the sound, the lower the position of the fader (dark) is the lower the fundamental pitch of the sine oscillators will be. The second fader represents another light sensor, which is controlling the number of sine tones produced by the system; the amount of light controls the density of the sine tones. The next fader (third from left) is representing temperature and controls the duration of the envelope of the sine oscillators. For example, low temperature creates grain sounds. When the motion detector (last fader from left) tracks movement it will jump at the highest position 1 of the fader whereas no motion occurs the fader will return to its lowest position 0. Finally, closing the fader's window will turn the system off.

Audience guidelines

The following paragraph is the explanation of the system to the participants describing what is PoP and how to interact with it.

Power of People (PoP), is an interactive sound installation using a set of sensors to capture light and temperature in order to control the parameters of a computer-based sound engine.

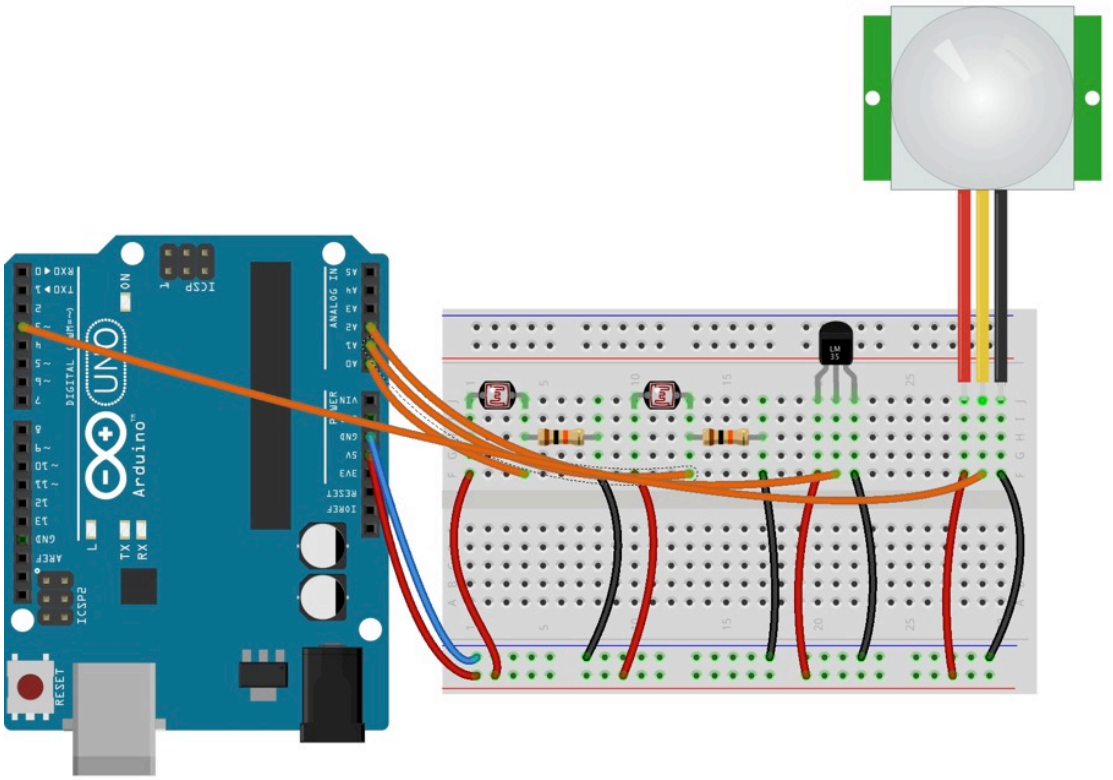
[59] In case this folder does not exist you have to create it.

A motion detector, which is capturing movement is used it to change the states of interaction called Agitate and Serene. Although the sound is influenced by the environmental conditions, it is the people who control which state will be executed by the system:

1. *Serene* is capturing the light of the space to control the pitch of the sound, and the temperature to control its duration.

2. *Agitate* is caused by stasis, or the lack of mobility inside the field of view of the system causing *Serene'* to be distorted.

If there is no movement for a long time, for example no audience or a passer-by to activate the system, the sound will fade out. People are advised to move freely inside the tracking area of the installation, and they are strongly encouraged to engage in collaborative interactions in order to change the states of the system and play with it by moving or staying idle.



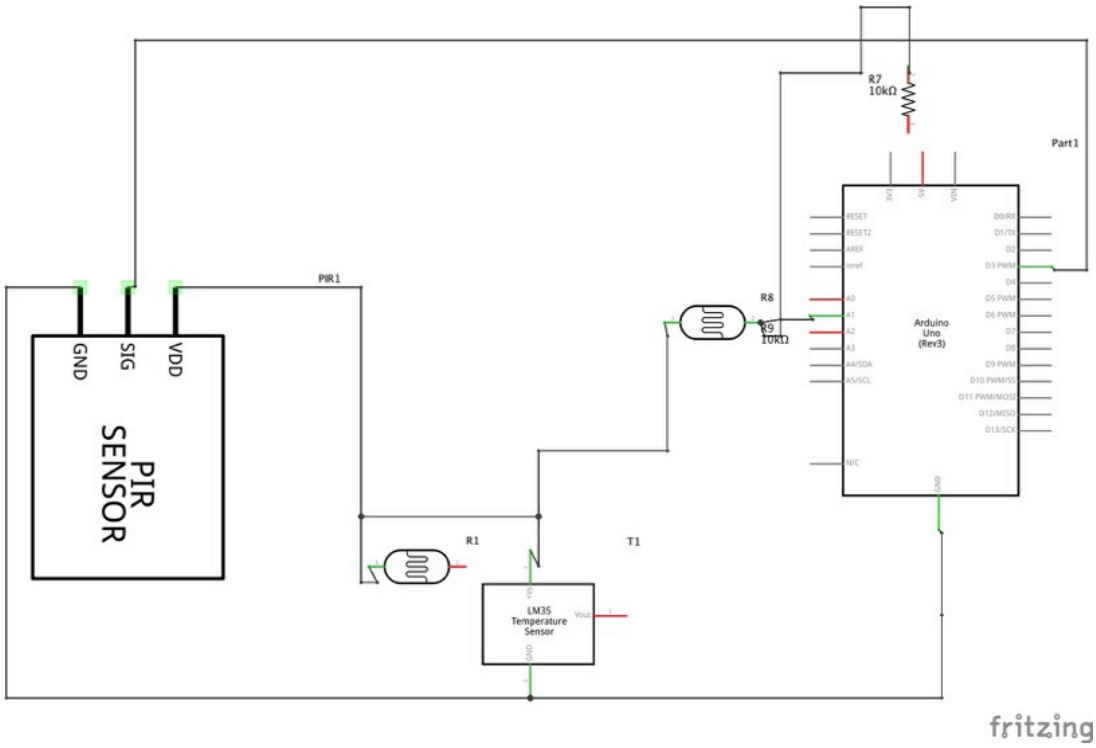
This image was created with Fritzing.[60]

.

[60] See Fritzing http://fritzing.org/home/

PoP schematic diagram and assembly list

The schematic diagram below illustrates the connections of the hardware. For a complete list of the hardware that was used in this project see table below.



Micro-controller board	Arduino Uno	Type Arduino Uno (Rev3)
PIR1 PIR sensor		
R1	Photocell	Variant pth; package photocell
R7	10kΩ Resistor	Tolerance ±5%; resistance 10kΩ; package THT; bands 4; pin spacing 400 mil
R8	Photocell	Variant pth; package photocell
R9	10kΩ Resistor	Tolerance ±5%; resistance 10kΩ; package THT; bands 4; pin spacing 400 mil
T1	LM35 Temperature sensor	Type LM35; package TO92 [THT]

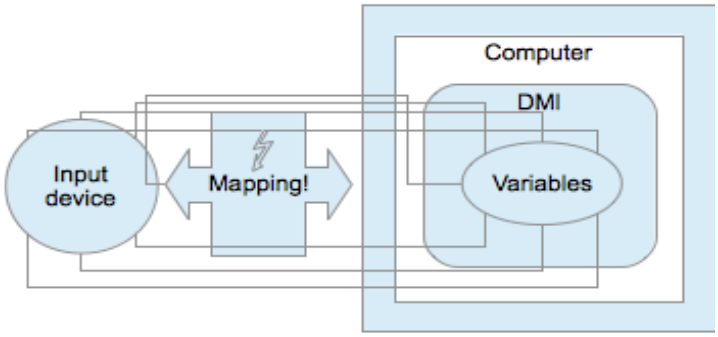
Troubleshooting

In case Arduino fails to connect with SuperCollider restart and run again the interface.scd file.

Appendix – Study II

Software	SuperCollider 3.6 (or above)
	OSCulator
Files	BiGrain.scd, BiGrain.oscd
Audio files	bakersegment(een).aiff
	BrechtSegmentMusic.aiff
	BrechtSegmentSong.aiff
	choirseqment(ohh).aiff
	clickseq.aiff
	crazyboy.aiff
	door.aiff
	fm4segment.aiff, goatbell.aiff
	AdolfSegment.aiff
	mboxmono.aiff,Leo.aiff
Hardware	Wiimote (connectivity: Bluetooth)
	Nunchuck

Wiimote is connected to the computer via Bluetooth. OSCulator converts Bluetooth in OSC and maps the data into SuperCollider. Wiimote variables are directly connected with the control inputs of the digital musical instrument (DMI).



Instructions and technical requirements

Instructions to set up the BiGrain environment and perform Study II are as follow. To run the environment you will need the latest stable version of SuperCollider and OSCulator. It is used to tap the data from Wiimote in SuperCollider via OSCulator. The device is connected to the computer via Bluetooth, and OSCulator converts it in Open Sound Control protocol (OSC), which can be mapped easily to any software that supports OSC communication.

If you are using the device for the first time you must pair it with the computer following the instructions provided by OSCulator. Once you have completed the pairing procedure successfully you must be able to see some controls at the OSCulator window illustrating the control variables of the device.

The system has a stereo output. Ideally, the computer will be connected to a mixing board via an audio interface using its main stereo out. However, in case there is no audio interface the computer may be connected directly to a pair of self-amplified speakers using a mini jack (3.5mm) cable connected to the line out of the computer's sound card.

Additional resources of the environment including the OSCulator configuration file can be found in a folder named BiGrain[61].

[61] Included in the accompanying DVD.

Providing that you have already opened and running BiGrain.oscd file in OSCulator open BiGrain.scd with SuperCollider and run it by selecting the entire file, on a Mac press ⌘ + A, and then ⌘ + Enter, and wait until SuperCollider loads the environment. To begin the performance simply press the plus (+) button, if you press minus (-) the environment turns off. If everything has gone according to the plan you must now hear sound, and moving the devices you must be able to manipulate the parameters of the sound synthesis environment. Movement along the y-axis of the Wiimote and Nunchuck manipulates the pitch of the grains and rolling along the x-axis of the devices manipulates the duration of the grains. Shaking the devices increases the number of the grains.

Instructions for the performance of Study II

To perform the piece you will need a selection of sounds[62] that were used to compose the piece. The sounds will be loaded automatically as long as the folder exists in the root folder of the project called BiGrain. There are no specific instructions to perform the piece. It is at the performers' liberty to improvise and synthesise the sound creating musically meaningful and compelling sounds using his/her intuition and imagination.

However, it might be worth mentioning that the system is able to create noise bursts or drone sounds depending on the physical effort of the gestures. For example, if the performer is moving the device abruptly the system will create noisy sounds, whereas smooth movements will create finer manipulations of the audio material.

Troubleshooting

It is recommended to launch OSCulator before SuperCollider to make sure that the communication between the two programs is successful. Sometimes there might be an OSC port mismatch. In case you encounter this problem type the "NetAddr.langPort" command in SuperCollider, the port number that appears in the post window must be the

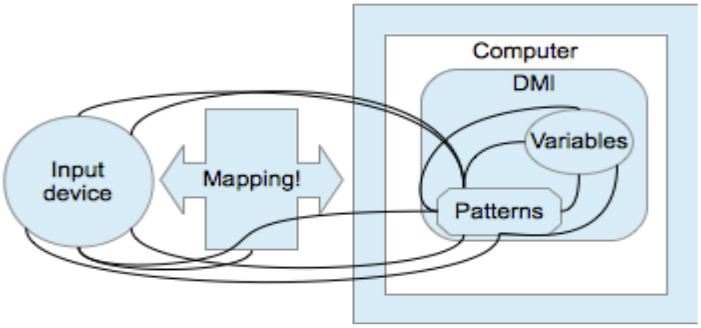
[62] Included in the folder of BiGrain (sounds folder) in the accompanying DVD.

same with the port number that OSCulator is sending the data. For complete details about how to set this in OSCulator consult the user's manual of the software.

Appendix – The Song has Sung

Software	SuperCollider 3.6 (or above)
	OSCulator
Files	Stay On This Gesture.scd
	Stay On This Gesture.oscd
Audio files	choir.aiff
	compreser.aiff
	crazydude.aiff
	door.aiff
	goatbell.aiff
	Adolf.aiff
	lcodeWaveformExamp.aiff
	musbox.aiff
	Leo.aiff
Hardware	Wiimote (connectivity: Bluetooth)
	Nunchuck

Wiimote is connected to the computer via Bluetooth. OSCulator converts Bluetooth in OSC and maps the data into SuperCollider. Wiimote variables are connected with patterns, fluctuating the control inputs of the digital musical instrument (DMI).



Instructions and technical requirements

Instructions to set up the Stay On This Gesture environment and perform Song has Sung are as follows. To run the environment you will need the latest stable version of SuperCollider and OSCulator. It is used to tap the data from Wiimote in SuperCollider. OSCulator uses Bluetooth to establish the communication between the Wiimote and the software, and converts it in Open Sound Control protocol (OSC), which can be mapped easily to any software that supports OSC communication. For details regarding the connections between the device and the environment see section 3.4.1.

If you are using the device for the first time you must pair it with the computer following the instructions provided by OSCulator. Once you have completed the pairing procedure successfully you must be able to see some controls at the OSCulator window illustrating the control variables of the device.

The system has a stereo output. Ideally, the computer will be connected to a mixing board via an audio interface using its main stereo out. However, in case there is no audio interface the computer may be connected directly to a pair of self-amplified speakers using a mini jack (3.5mm) cable via the line out of the computers' sound card.

Additional resources of the environment including the OSCulator configuration file can be found in a folder named Stay On This Gesture[63]. Providing that you have already opened

[63] Included in the accompanying DVD.

and running the Stay On This Gesture.oscd file in OSCulator open Stay On This Gesture.scd with SuperCollider and run it by selecting the entire file, on a Mac press ⌘ + A, and then ⌘ + Enter and wait until SuperCollider loads the environment. If everything has gone according to the plan you must now be able to control the environment. Movement along the y-axis of the Wiimote controls the pitch of the grains, for example moving the device upward increases the pitch, and rolling the device to the right increases the duration of the grains. Shaking the device increases the volume of the grains.

Instructions for the performance of The Song has Sung

To perform Song has Sung you will need a selection of sounds[64] that were used to compose the piece. The sounds will be loaded automatically as long as the folder exists in the root folder of the project called Stay On This Gesture. Stay On This Gesture, an autonomous digital environment, which was used to perform the Song Has Sung was developed to foster a mutual intervention between the performer and system employing patterns in SuperCollider. To this end, the performer must let him (her)self at the liberty of the system influence his or her imagination.

The piece is using four sets of diverse patterns; to initiate each pattern use the cross button on the top of the device. Pressing the button up will begin the second pattern, given that the first pattern set is initiated when starting the environment; pressing the button to the left will begin the third pattern, the last pattern set is initiated by pressing the cross button down. To begin the performance simply press the plus (+) button, pressing minus (-) will turn the environment off.

Troubleshooting

It is recommended to launch OSCulator before SuperCollider to make sure that the communication between the two programs is successful. Sometimes there might be an

[64] Included in the folder of BiGrain in the accompanying DVD.

OSC port mismatch. In case you encounter this problem type the “NetAddr.langPort” command in SuperCollider, the port number that appears in the post window must be the same with the port number that OSCulator is sending the data. For complete details about how to set this in OSCulator consult the user’s manual of the software.

Page left blank intentionally.