Commentary on the Portfolio of Compositions submitted for the degree of PhD in Composition: Heart of Light, Light of Heart, Spanish Ladies, Down Among the Dead Men, Pavanne, Christ ist Erstanden, Rezoplucker, Cliqbuz, Circle Theory.

KELCEY ROBERT SWAIN

Department of Music

Commentary on the Portfolio of Compositions submitted for the degree of PhD in Composition

Candidate:

Kelcey Swain,

St. Chad's College

Supervisors:

Prof. Peter Manning

Dr. Martyn Harry

20th April 2010

Abstract

Portfolio Contents:

- *Heart of Light* - Industrial Electroacoustics. First performed in Durham in November 2008.
- *Light of Heart* - Physical Modeling Electrominimalism. First Performed in November 2009.
- *Spanish Ladies* - Vocal Electroacoustics using folk song. Commissioned by the Durham New Music Marathon and first performed in Durham in June 2009.
- *Down Among the Dead Men* - Vocal Electroacoustics using folk song. First Performed in November 2009.
- *Pavanne* - Vocal Electroacoustics using Loré Lixenberg's voice. Commissioned by Loré Lixenberg and first performed at the CoMA Summer School, July 2008.
- *Christ ist Erstanden* - Vocal Electroacoutics using Jason Walsh's voice. Commissioned by Loré Lixenberg and first performed in Durham in January 2008.
- *Rezoplucker* - Live performance using the accelerometers from the Nintendo Wii. Commissioned by Culture Lab and first performed in Newcastle in January 2008.
- *Cliqbuz* - Live performance using IRCAM's Ethersense. Commissioned by Culture Lab and first performed in Newcastle in January 2008.
- *Circle Theory* - Live performance using circle packing algorithms and ChucK. Commissioned by Maebh Long and first performed in Durham April 2009.

Audio CD Track List:

1. *Heart of Light* - 15'34"
2. *Light of Heart* - 11'39"

3. *Spanish Ladies* - 11'08"

4. *Down Among the Dead Men* - 10'08"

5. *Pavanne* - 3'45"

6. *Christ ist Erstanden* - 5'14"

7. *Rezoplucker* - 3'58"

8. *Cliqbuz* - 6'07"

DVD Video Contents:

1. *Cliqbuz #1* - 4'37"

2. *Cliqbuz #2* - 8'20"

3. *Rezoplucker #1* - 4'27"

4. *Rezoplucker #2* - 2'29"

5. *Circle Theory* - 4'45"

Abstract:

The portfolio contains various works which explore the interplay between live performance of electroacoustic music and performance of pre-written work. Control of real-time parameters in computer generated music is now so prevalent that it can be used in the compositional process. This shift in possibilities is the focus of this commentary. It is not the intention of this commentary to either tell the reader how to interpret the music or to discuss æsthetic issues. This commentary should be used more like a construction manual to aid the listener where they might not be familiar with the concepts and techniques employed in the music. The introduction sets out the music aims, the conclusion explains how these aims are achieved in various ways and each chapter in between focuses on a different piece in the portfolio.

Acknowledgements[†]

There are many people who deserve thanks for all their help and support but I would like to single out a few for their crimes in allowing me to create such noises. Thanks go to Ron Berry and Martin Allison, the most helpful technicians an electroacoustic composer could have hoped for, not only was their technical expertise invaluable but they have always been there to juggle ideas and hair-brained schemes with me; My family for believing that what I do is of value and not merely frivolous noise making, even when it might be; Dr. Martyn Harry and Prof. Peter Manning for prodding me in the right directions when my imagination might have got the better of me; all my fellow postgraduates and friends with whom I have enjoyed many performances and concerts over the years; to Culture Lab, Newcastle University for letting me use their workshops to create instruments; and finally Eric Egan for being easy competition.

The completion of this PhD was made possible by the generous support of the Arts & Humanities Research Council (AHRC), who have funded the composer's study at Durham University 2007–2009.

Contents

Abstract iii

Acknowledgements vi

List of Figures and Listings ix

1 Introduction 1

2 Heart of Light 5

3 Light of Heart 9

4 Spanish Ladies 13

5 Down Among the Dead Men 18

6 Pavanne 22

7 Christ ist Erstanden 25

8 Rezoplucker 29

9 Cliqbuz 37

10 Circle Theory 42

11 Conclusions **50**

Bibliography **54**

List of Figures

2.1 Signal flow of Audiobulb's *SophiaABV4* 7

3.1 Signal flow of the 'Erratic Triangles' in *RezoPluckerSynth* 11

3.2 Signal flow of *RezoPluckerSynth* 12

4.1 The source material for *Spanish Ladies* 14

4.2 Simplified signal flow of the Reaktor patch for *Spanish Ladies* 16

4.3 Channel layout for the eight channel version of *Spanish Ladies* 17

7.1 Illustration of *Christ ist Erstanden* from Codex Admont 323 26

8.1 The Nintendo Wii controller (right) and Nunchuk (left) 30

8.2 The on-screen display of *RezoPluckerSynth* 31

9.1 Infrared LED and receiver attached to a prototype board 38

9.2 A prototype of the hardware of the *Cliqbuz* instrument 39

9.3 The composer working on *Cliqbuz* at Culture Lab 39

9.4 The on-screen display of the *Cliqbuz* synthesis engine 41

List of Listings

5.1 SuperCollider code for spectral sine waves 19

5.2 SuperCollider code for pedal notes 20

5.3 SuperCollider code for the “asynchronous” pattern 20

8.1 Python programme for generating the GlovePIE file 32

8.2 An example GlovePIE file generated by the Python file 8.1 33

10.1 JavaScript code for creating SVG circles and send the OSC signal out via Lily . . . 43

10.2 ChucK code for receiving the OSC signal and processing the sine waves 47

Chapter 1

Introduction

> 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.
>
> Edgard Varèse, *391*

My research interests lie predominantly in electroacoustic improvisation and performance techniques specifically focusing on the challenges inherent in the interaction between human and machine. There are two main areas of focus involved with this, firstly in the creation of performable electroacoustic instruments and secondly in their performance practice. It is the aim of my compositional research to go beyond the simplistic and restrictive mouse and keyboard combination by building non-logical and non-intuitive systems for the creation of electronic music. These systems or instruments should be performable by musicians and non-musicians alike with very little rehearsal time as they combine control and creation coherently. It is my intention that these "instruments" are viewed as either sonic artefacts or, even better, pieces of music in their own right in as much as a musical score is often seen as a piece of music. As such they would hold a comparable relationship with graphical scores, as they would both require a certain amount of improvisation to be realised musically whilst imposing limitations and restrictions on the possible outcomes. Parallels in instrumental music can be drawn with La Monte Young's *The Well Tuned Piano* where there are

different potential performances but the nature of the instrument shapes the possible outcomes As Elvin Jones said 'There's no such thing as freedom without some kind of control, at least self-control or self-discipline'[1], however the imposed discipline in many of my works occurs at the instrument level. Seeing as the performer could not possibly be experienced with the instrument the usual self-discipline does not apply readily so the physical and structural aspects of instrument constrain and restrict the performer. Comparisons can also be drawn with the ideas of Boulez and Lachenmann who have created "instruments" from extended techniques and systems by which the music is shaped but they approach this idea from the opposite side.

Physical performance restrictions can be harnessed such that the performer is guided though the possibilities of the instrument by the constrained methods of interaction, this means that one should not be able to take the instrument in a direction that will produce "bad" results. To aid this playability I am also interested in transgressing the boundaries between the macro level and the micro level, the structural level and the sonic level of music. My instruments, pieces and performances are aimed at a coherence of audio creation where numerical elements controlling the spectral qualities of timbre are just as capable of controlling the global music structure of the piece and vice versa. This can be realised by employing elements such as fractals, chaotic algorithms, neural networks and other non-linear functions in the synthesis of the sound. As Di Scipio writes, 'of primary interest was the merging of them, i.e. the blurring of the clear-cut distinction between the macro-level articulation of musical structure and the micro-level, timbral properties of sounds', and he goes on to say that, 'algorithmic composing was to result not so much in a music of notes (the "lattice" structure of quantised pitch, duration and intensity values) as in sound textures and complex sonic gestures defined compositionally by their timbre and internal development'[2]. It is this 'blurring' that I have attempted to encourage in my own work.

The real-time performance and audience interaction aspects of this music are also convenient

[1] Taylor, 228

[2] Di Scipio, 249

ways of avoiding the problems of acousmatic listening in electroacoustic music, a problem I face with my secondary research interest, that of creating electroacoustic composition in a non-real-time and non-improvised environment. The challenges in sustaining interest in electroacoustic music over a large scale work where there is no performer to enhance the legibility of sonic results are difficult to overcome especially with abstract sounds that cannot be traced to any real world events or actions. In this commentary I hope to show how I have addressed these challenges in my music.

My research into creating electronic musical instruments is influenced by the ongoing work in collaboration at STEIM (Studio for Electro-Instrumental Music) which has produced some remarkable examples of organo-technological synergies specifically the work done by Sensorband which is comprised of Edwin van der Heide, Zbigniew Karkowski, and Atau Tanaka. In Sensorband the three performers play their own electronic instrument; van der Heide plays the *MIDI-conductor*, Karkowski plays a cage of sensors and Tanaka plays *Biomuse*[3]. Of these three instruments the first is most similar to my own *Rezoplucker* in its technology if not its sound and the second is most like my *Cliqbuz*. *Biomuse* is possibly the most interesting from a construction perspective as it can read its data from electromyograms (EMGs), electroencephalograms (EEGs), electrooculograms (EOGs) and electrocardiograms (EKGs). All this means that the instrument is being controlled simultaneously by the conscious and unconscious movements of the body and its organs[4].

The chapters of this commentary are divided into detailed examinations and explanations of each piece in the portfolio. In each case the piece is taken as a discrete unit unrelated to the others and explored for its own æsthetics and techniques. The sum total of the portfolio will be brought together in the concluding chapter. It is also worth noting that whilst it might not normally be a good idea to refer to so many online sources in an academic commentary, the nature of this folio of compositions is such that the technical issues in computer-aided and generated composition take precedence over the theory and practice of electroacoustics. In the case of technical information on

[3] Bongers, 13

[4] Knapp, 43

electroacoustic systems the relevant information is best taken from the internet, especially the web sites of software developers. This information would often be out of date by the time it is published, if that ever even happens.

Chapter 2

Heart of Light

> Ancient life was all silence. In the 19th century, with the invention of machines, Noise was born. Today, Noise is triumphant and reigns sovereign over the sensibilities of men.
>
> Luigi Russolo, *The Art of Noises*

Heart of Light is a study of modulated synthesis in its extreme. The piece explores the levels of interaction between the micro scale and the macro scale by transcending the arbitrary boundaries between note and form. In the discipline of electroacoustic composition it is possible for a single note to transform into a full melody of sounds in an organic manner just as it is possible for a whole phrase to be reduced into a single utterance. As humans we perceive this boundary due to our brain interpreting anything with a rate of repetition of over around 20Hz as a single sound with a perceived frequency of around 20Hz. Reduce this rate and the sounds become repetitions of the same thing. If the sound can be slowed down without losing its core make up then we can start to hear inside the sound on a different level. In the real-world the sounds that would be capable of this interplay and exploration would be sounds that consist of discrete repetitions, not for example the sinusoidal periodicity of most orchestral instruments. In the electroacoustic domain this effect can be readily achievable using amplitude modulation, gating and modulating low-frequency oscillators. If we were to start with a single click or pulse and control the trigger rate

with a variable control then we could speed the rate of clicks to above 20Hz to create a pulse wave which would sound as a single note. It would be similar if a gated sine wave were used. Now we control the amplitude of that pulse wave by another low frequency oscillator and we get a double level of modulation in our sound. With a large network of similar modulators we can very easily produce some very complicated sounds. An example of a large complicated system of modulators is Audiobulb's synthesiser, *SophiaABV4*; fig. 2.1 shows its signal flow diagram.

Sophia is undoubtedly a very complicated design of synthesiser, so much so that the designer decided that the atmosphere invoked by its sounds should take precedence over its legibility and usability, asking instead that the user plays with it and learns how it reacts to the controls. x|k, the creator of the synthesiser, writes in its manual, '*Sophia* may be confusing to grasp at first, because the GUI (Graphical User Interface) is based on "setting the vibe" as opposed to usability. This was purely an artistic decision, rather than a practical one'[1], and goes on to say that the best way to understand it is by studying the signal flow diagram shown in fig. 2.1, but the diagram is very unclear.

Heart of Light was composed entirely using sounds generated by *Sophia* in four 10 minute real-time recordings. This was only possible after extensive experimentation and practice on the instrument in order to produce even the slightest levels of predictability. The resultant sound world is mechanistic and industrial due to the comparability of discrete periodicity in *Sophia* and in real-world machines. As with *Spanish Ladies* (Chapter 4) the source material was sequenced and composed into the piece using Cakewalk's Sonar.

The form of the piece is relatively free due to the experimentation of the interplay between the micro sound-world and the macro sound-world. Therefore the composition of the piece had to be approached holistically. Having said this the piece starts and ends in a similar sound-world so as to lend a sense of coherence to a piece that would otherwise sound alien to most listeners. The middle section of the piece which starts at around 7'40", is again comparable to the middle of its analogue

[1] http://www.audiobulb.com/create/sophia/Sophia1_1.zip

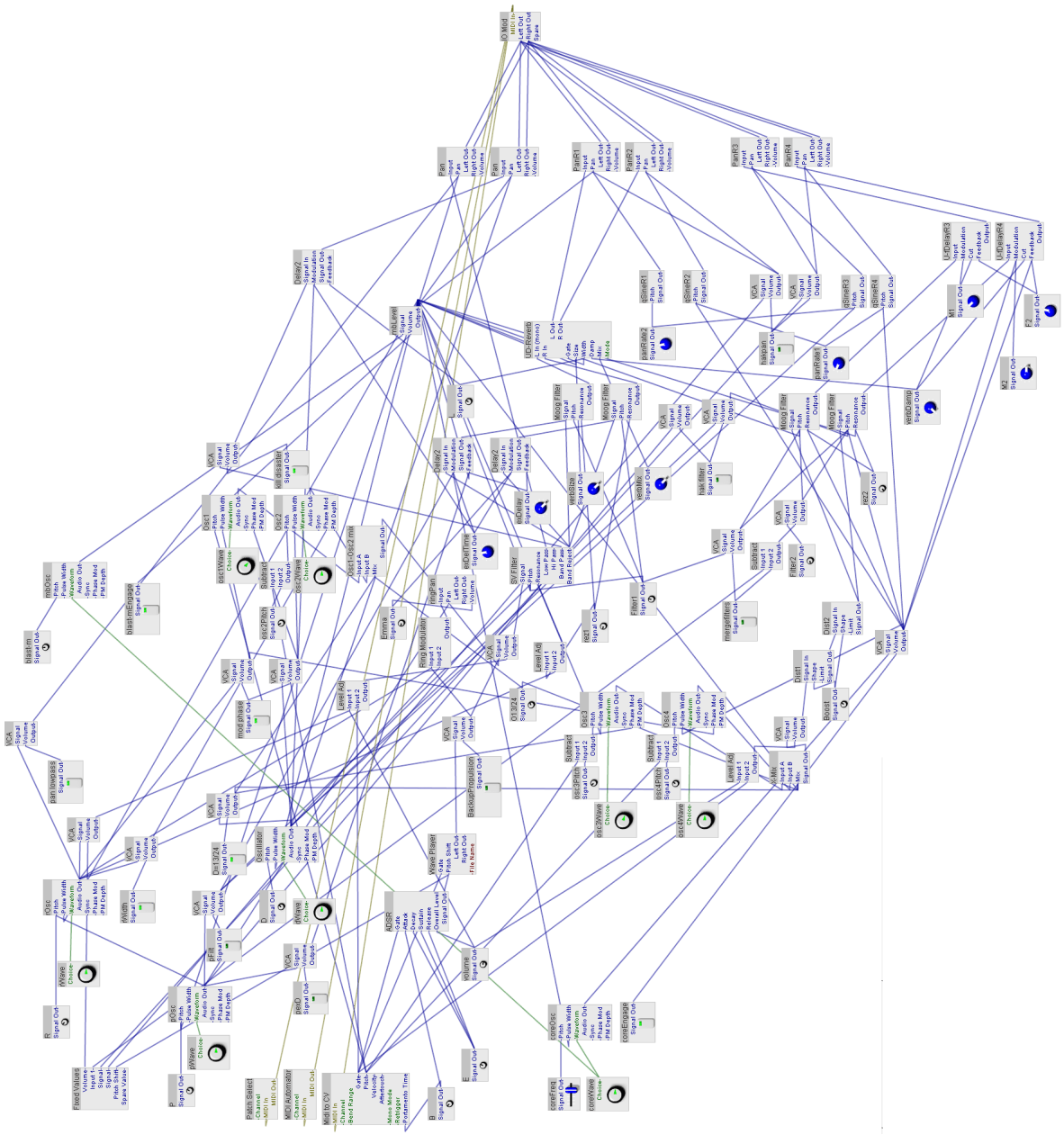

Figure 2.1: Signal flow of Audiobulb's *SophiaABV4*

Spanish Ladies by featuring a large scale stasis which acts as a mirror through with the piece reflects itself.

The overall structure of the piece after the introduction is that of a journey though levels of modulation and back round to itself. In this way it can be considered like the harmonic cycle of fifths which lead inexorably back to the starting point but take in all possibilities en route or like a rotating tesseract which appears in three dimensions to turn itself inside out only to end up as it started.

Chapter 3

Light of Heart

> How soft the music of those village bells
> Falling at intervals upon the ear
> In cadence sweet, now dying all away,
> Now pealing loud again, and louder still,
> Clear and sonorous as the gale comes on.
>
> William Cowper, *The Task*

Light of Heart, as the name suggests, was written as a piece to contrast with *Heart of Light* whilst also being entirely synthetically produced. On one level the piece is minimalist in its harmonies to the point of being one single extended cadence which gets close to resolving itself but is ultimately thwarted on the micro-level. However, the sound material is taken from a highly chaotic synthesis engine.

The source material for this piece is divided into two distinct sound worlds. One is the drone that produces the C and B along with their full array of overtones. The other is the physical modelling engine which produces the plucked sounds. The drones are produced by a built-in Reaktor patch called *SpaceDrone* by Martijn Zwartjes. *SpaceDrone* works by assigning an array of tones or pitches, in this case built up from the harmonic series, to independent but stochastically controlled amplitude envelopes, panning and resonance feedback loops and then giving the user global controls for the density of notes at anyone time, and the global parameters for the independent envelopes, plannings and feedback loops. As such, each "note" has autonomy but follows a global trend. The

outcome of this method of synthesis is the ability to produce very realistic real-world sounds, in this case a metallic and resonant "wind" sound.

The plucked sounds are made from a synthesiser which I built specially for *Light of Heart* called *RezoPluckerSynth*. Each channel of this synthesiser is constructed from a triangle LFO (Low Frequency Oscillator) with a small random amount of deviation which controls the frequency of a second triangle LFO which in turn controls the pitch of a triangle wave oscillator, as shows in fig. 3.1. This erratic waveform is low-pass filtered and then fed through a feedback resonator which resonates the pitches played on a MIDI Keyboard. The signal flow for the resonators is shown in fig. 3.2. The final sound is produced as the oscillators pass the resonant frequencies on their random walk coupled with a very short triangle wave played at the resonant pitch in order to guarantee a sound as soon as the key is depressed.

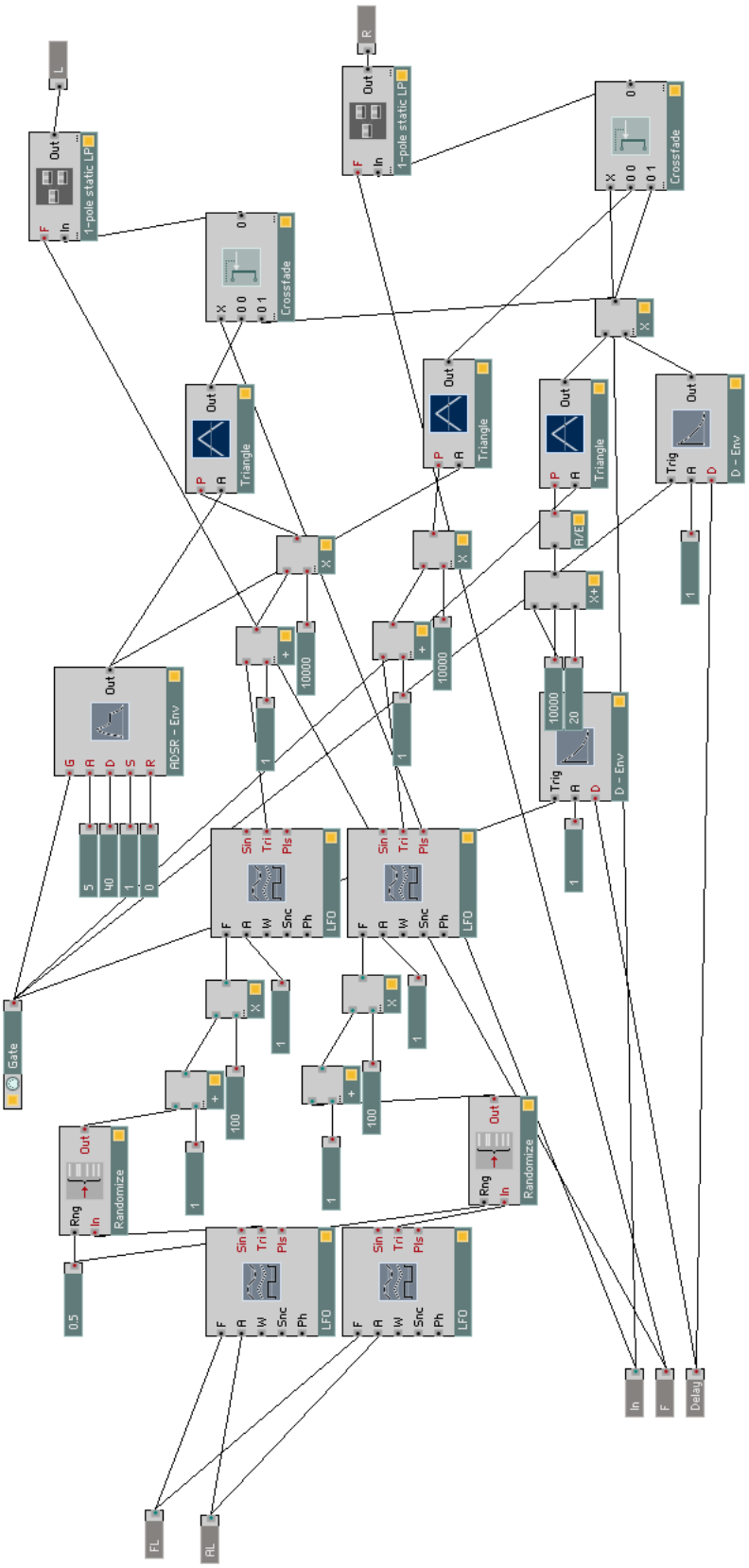

Figure 3.1: Signal flow of the 'Erratic Triangles' in *RezoPluckerSynth*

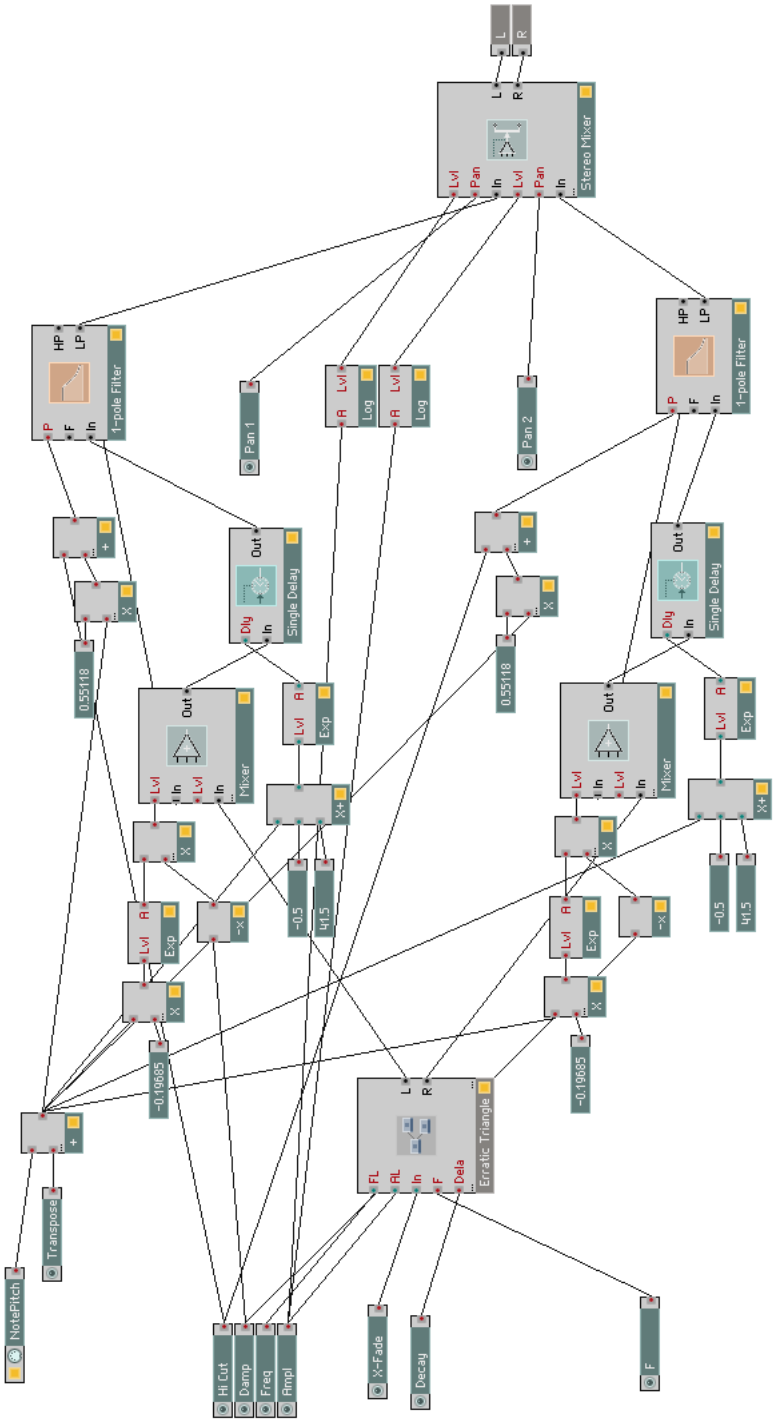

Figure 3.2: Signal flow of *RezoPluckerSynth*

Chapter 4

Spanish Ladies

We will rant and we'll roar
like true British sailors,
We'll rant and we'll roar
all on the salt sea.

Chorus of *Spanish Ladies*

The traditional song *Spanish Ladies* is generally considered to be a working song as opposed to a recreational song despite the claims of some earlier scholars of folk music[1]. The former is known as a sea shanty and the latter is a usually referred to as a sea song. The arguments for and against are beyond the scope of this commentary but I decided for the purposes of my composition to assume that it is a sea shanty. Purists claim that a sea shanty, being a work song used for its rhythmic devices, should neither be accompanied nor arranged[2] but merely sung unaccompanied and in unison by a "choir" of men. To this end I chose to create an electroacoustic work using a single recording of the first verse of the shanty as my only source material. By definition sea songs and shanties are strophic and given that the lyrics of the song play an ancillary rôle in my piece it was not necessary to record any more verses.

Seeing as the melody of the song has many variants I wrote out a version that incorporated aspects of the song I wished to include in my final work. My main concern was that the method

[1] Bullen, 32–3

[2] Saunders, 344–5

of treatment would require a specific and unchanging mode that would allow any one part of the melody to harmonise with any other part. Fig. 4.1 shows the song with the lyrics of the first verse as it was recorded, albeit not at pitch. The range at which it was recorded was decided by the timbre of the voice at its various registers.

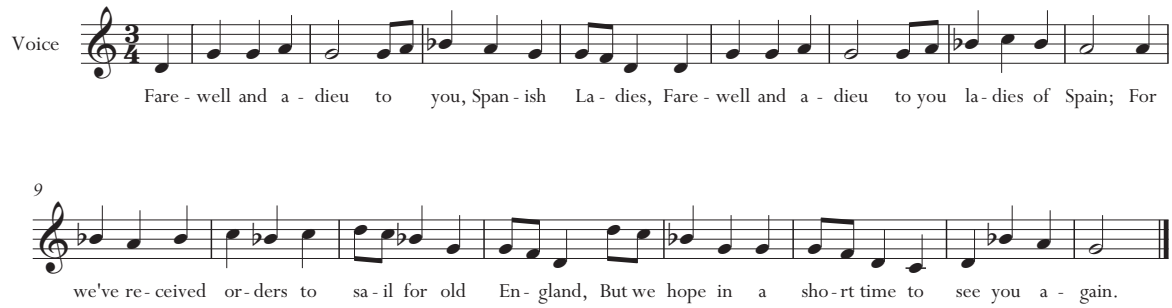

Figure 4.1: The source material for *Spanish Ladies*

The structure of this new version of *Spanish Ladies* is loosely based on a Sonata form but it is modified to accommodate the source material in its untransposed iterations. The first theme is the rhythm-less wash of drawn-out ethereal sounds produced from the original song lasting for about a minute and a half. With no traditional transition this leads into the second theme, which contrasts with the first by being unnaturally and stutteringly rhythmic to the point of being uncomfortable. The long development section is introduced not by a codetta but by a "reveal" of the untreated source material which invites listeners to draw a point of musical reference for the piece. In the development section we first hear a 4 part variation of the second theme which decays into a stasis of long notes reminiscent of the first theme but with no connection to the melody of the original song. This stasis is punctuated by a single note, again with the rhythmic stutterings which introduces a chattering of voices. This chattering can be compared to the previous stasis point; the chattering is to the stasis as the second theme is to the first. Furthermore the chattering is an agent to bring about the recapitulation at about nine minutes through the piece. The recapitulation cannot be treated as a normal recapitulation mostly because at no point have we modulated to a different key

as it was not my intention to treat the source material in this harmonic manner. To create a sense of resolution between the two themes they are allowed to coexist and play off each other. While the first theme finds its way slowly towards the tonic the second theme reaches its conclusion in a short coda of forceful repetition on a single note with no ambiance to hide its unnatural inhuman sound.

Spanish Ladies is believed to have been a shanty for use whilst turning the capstan of the ship, therefore its rhythm and more importantly its pulse was arguably the only relevant aspect of the song, all else being secondary. Fortunately during the performance of my version of this shanty no one is expected to undertake hard labour so the pulse no longer hold any value beyond its æsthetic qualities. However, the primacy of the rhythm and pulse of the shanty has not been lost in this piece. With the obvious exception of the "reveal" section, the rhythm of the original shanty is utterly fragmented to the point of it no longer being an audible consideration. On top of the original song a secondary pulsing layer has been added which is of perfect regularity but too fast to accompany any ship work.

The effect of the piece is a dialogue between natural organic and unnatural mechanical sounds; between song and pulse. The pulse is repetitive and all too perfect electronically performed micro-sounds whilst the song is ethereal and expansive. The piece also plays with the origins of granular processing but by separating out the grains and removing all stochastic variance it ultimately fails to produce the organic sounds of granular techniques. Even the organic sounds heard in the first theme are not created using granular processing as will be described.

The piece was created using a patch which I made in Native Instrument's Reaktor graphical programming environment. The patch is built from four sample buffers each containing identical versions of the original source material in its untreated form, the buffers are designed so that they do not simply play the sample through but repeat a minute section of the sample defined by the position of the control faders as shown in fig. 4.2. The first two buffers are sent straight to the mixer with no effects or treatment but the second pair of buffers are sent through a reverb unit

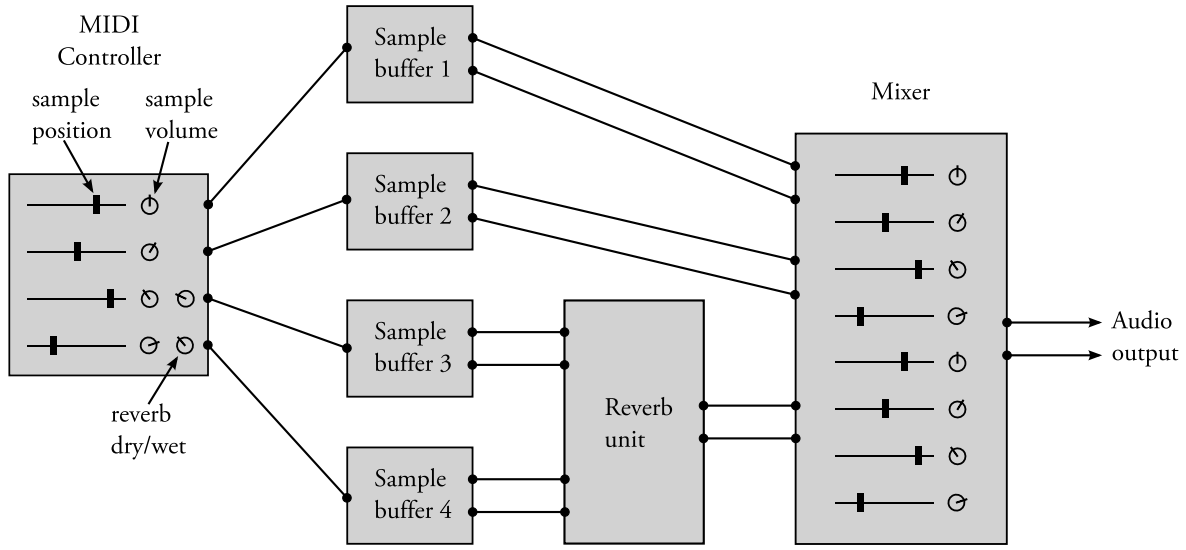

Figure 4.2: Simplified signal flow of the Reaktor patch for *Spanish Ladies*

which is set with a very long diffusion time. The mix of the reverb unit is also determined by the controller which means that if the mix is set to dry then this pair of buffers acts the same as the first pair. However if the mix is set to wet then these buffers no longer sound like the stuttering first pair, instead they take on a faux granular appearance. Importantly, they do not actually receive any actual granular treatment but the reverb blends the gaps between the fast stuttering repetitions. By changing the mix on the reverb unit the sound can blend seamlessly between the sounds heard in the first theme and the sounds heard in the second theme, this is most prevalent at the very end of the piece where the reverb is slowly removed exposing the repetitions without any ambient reverberations.

The patch was used to create short sections of the final composition which were then sequenced and arranged in the DAW Sonar. Both a two channel and an eight channel mix were written for CD and for concert performance respectively. The eight channel mix should be performed in the speaker configuration shown in fig. 4.3. For the most part buffers 1 & 3 are sent to the left channel and buffers 2 & 4 are sent to the right channel despite the fact that the source recording is in stereo. This allows the samples to move abruptly in and out of synchronisation with the samples of their

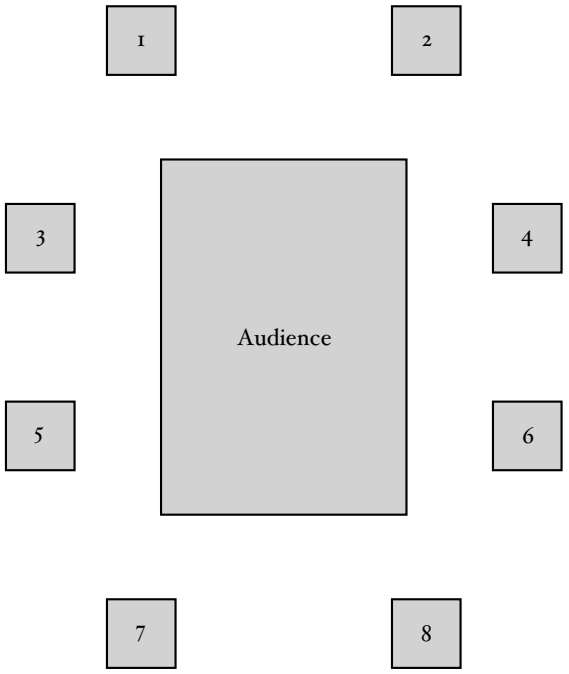

Figure 4.3: Channel layout for the eight channel version of *Spanish Ladies*

stereo partner buffer. This is mostly only used in the stutterings of the first two buffers but it does produce a rather unsettling panning experience for the listener as the sounds become very difficult if not impossible to place across a stereo field. This effect is only usefully audible in the two channel mix of the piece but the eight channel version has the compensation of being spacialised around the audience which allows "voices" to appear from many direction.

Chapter 5

Down Among the Dead Men

> Here's a health to the King and a lasting peace:
> to faction an end, to wealth increase!
> Come, let us drink it while we have breath,
> for there's no drinking after death;
>
> Down Among the Dead Men

Down Among the Dead Men was written as a companion to *Spanish Ladies* (Chapter 4) but was designed both as a contrasting piece and an exercise in learning to write SuperCollider[1] programmes. SuperCollider is an environment and language for programming real-time audio synthesis and processing which is comprised of both an interpreted language and a synthesis server using OpenSoundControl or OSC[2] to communicate. It is a robust tool which harnesses the power of object-oriented structures combined with built in unit generators.

The traditional English song *Down Among the Dead Men* is generally attributed to Robert Dyer although there is an earlier version of the song with lyrics in memory of Queen Anne[3]. The song was chosen as both a point of comparison and contrast with *Spanish Ladies*. They are similar in that the subject matter of both songs is worldly pleasure, in *Spanish Ladies* the object of interest is the eponymous ladies and in *Down Among the Dead Men* the main interest is having a drink while

[1] SuperCollider can be downloaded freely from `http://supercollider.sourceforge.net/`

[2] For more information on OSC see `http://opensoundcontrol.org/`

[3] Chappell, 643–4

you are still able to. The contrast between the two songs is the outlook taken by each, one holds optimism for seeing the ladies again whilst the other is more concerned with being able to '...drink it while we have breath, for there's no drinking after death'[4].

The first verse and chorus of the song were sung recorded and processed to create a whispering and distorted version of the song which can be heard at 4'05" and 8'35". The rest of the piece is built up from synthesis alone using SuperCollider and processing such as auto-panning and granulation. The spectral clusters were reproduced unprocessed and recorded directly from SuperCollider using the code in listing 5.1. The code creates a continuously cyclical cluster of 32 sine waves, each one at a random pitch between 67Hz and 2000Hz and then applies an envelope to each of the individual sine waves. The resultant sound is decidedly eerie and continues through two-thirds of the piece.

Listing 5.1: SuperCollider code for spectral sine waves

```
s.boot;
(
{
loop({
        {
        var n = 32;
        Mix.arFill(
                n,
                // 32 oscillators with random pitch and
                // random amplitude envelopes
                {
                        SinOsc.ar(
                                [67.0.rrand(2000), 67.0.rrand(2000)],
                                0,
                                n.reciprocal
                        )
                        *
                        EnvGen.kr(
                                Env.sine(2.0.rrand(17)),
                                doneAction: 0
                        )
                }
        )
        *
        EnvGen.kr(
                Env.perc(11, 6),
                doneAction: 2,
                levelScale: 0.5
        )
```

[4]ibid, 644

```
30          }.play;
31          10.wait;
32  })
33  }.fork;
34  )
```

Also through most of the piece are various pedal notes. These were created using the code in listing 5.2 but were enhanced with an auto-panning filter to create a sense of continuous movement within the sound.

Listing 5.2: SuperCollider code for pedal notes

```
1   s.boot;
2   (
3   SynthDef(\dynklank2, { arg out=0, freq;
4           // A synth definition using the built in DynKlank algorithm
5           var klank, n, harm, amp, ring;
6           n = 9;
7           harm = Control.names(\harm).kr(Array.series(4,1,1));
8           amp = Control.names(\amp).kr(Array.fill(4,0.05));
9           ring = Control.names(\ring).kr(Array.fill(4,1));
10          klank = DynKlank.ar(`[harm,amp,ring],
11                  {ClipNoise.ar(0.003)}.dup, freq);
12          Out.ar(out, klank);
13  }).send(s);
14  )
15
16  // Evaluate these lines to play a note
17  a = Synth("dynklank2", [\freq, rrand(30,440)]);
18  b = Synth("dynklank2", [\freq, rrand(30,440)]);
```

The final pattern created in SuperCollider were the unusual metallic sounds towards the end of the piece, from around 6'30" onwards. This was a polyrhythmic asynchronous pattern, coded in listing 5.3, and then granulated to an uncomfortably slow speed without changing the pitch. The code uses the instrument defined in the `SynthDef` (lines 3–16), which is, in simplistic terms, a resonated formant and plays them according to the patterns `a` (lines 21–25) and `b` (lines 26–30). Both patterns are polyrhythmic, in that they have non-equal length lists for durations and pitches, but also the length of the two patterns are not equal. This means that the resulting pattern has a very long repeat length which is only made longer by granular stretching.

Listing 5.3: SuperCollider code for the "asynchronous" pattern

```
s.boot;
(
SynthDef(\dynklank, { arg out=0, freq, dur=1;
        // A synth definition using the DynKlank algorithm
        var klank, n, harm, amp, ring, env;
        n = 9;
        harm = Control.names(\harm).kr([1.0, 1.2, 2.4, 3.6]);
        amp = Control.names(\amp).kr(Array.fill(4,0.05));
        env = EnvGen.kr(Env.perc(0.001, dur), doneAction: 2);
        ring = Control.names(\ring).kr(Array.fill(4,1));
        klank = DynKlank.ar(`[harm,amp,ring],
                {Formant.ar(XLine.kr(58, 1000, 2),
                XLine.kr(2000, 1500, 2), 800, 0.125)}.dup, freq);
        klank = klank*env;
        Out.ar(out, klank);
}).send(s);
)

(
(       // Two polyrhythmic patterns of different lengths
        a = Pdef.new(\example1, Pbind(\instrument, \dynklank,
                \freq, Pseq([58, 400, 440, 120, 90], inf),
                \dur, Pseq([0.25, 0.5, 0.25, 0.75], inf);
                )
        );
        b = Pdef.new(\example2, Pbind(\instrument, \dynklank,
                \freq, Pseq([220, 440, 230], inf), // freq arg
                \dur, Pseq([0.25, 0.5, 0.25, 0.25], inf);
                )
        );

);
(
        a.play;
        b.play;
)
)

(
        a.stop;
        b.stop;
)
```

The piece finished with a transposed version of the chorus which is not otherwise processed, the final words "let him lie" are heard on their own. Because the words are slowed down so much you can hear the movements of every part of the mouth as the final words trails off.

Chapter 6

Pavanne

> Order is repetition of units. Chaos is multiplicity without rhythm.
>
> M. C. Escher

Pavanne was commissioned by Loré Lixenberg as a piece to lead into Trevor Wishart's *Anticredos* from a performance of a polyphonic mass setting. The piece was first played in a dark concert hall on four channels surrounding the audience.

Anticredos, composed in 1980, 'gradually deconstructs the word 'Credos' by gradual transformations of its sonic constituents'[1], the word chosen specifically for its wide range of sonic constituents. *Pavanne* does not deconstruct anything but instead destroys a song sung by Loré Lixenberg, replacing it gradually with dirty noise created using a telephone wired into a resonator built in Reaktor and a VST plugin called Glitch[2]. The vocals are distorted as the glitchy sounds take over until finally the single voice is heard reverberating at the ending.

The two vocal recordings used for the base of this piece were provided by Loré Lixenberg but the source music is unknown to the composer. The two recordings were blended together using the Reaktor patch *Grainstates* to create an ambient wash over which the development takes place. As the piece starts the vocal lines are presented and then removed behind a layer of processing in

[1] Wishart, taken from personal correspondence with the composer.

[2] `http://illformed.org/plugins/glitch/`

order to create a calm and hospitable aural environment for the audience. The intention is to take the audience into a simple and safe sound-world. Once this has been achieved the piece slowly introduces the unusual electronic sounds that would not normally be associated with the kind of sound-world that has been explored using the vocal recordings. In the original performance of *Pavanne* the electronic sounds came from behind the audience in an attempt to further disturb and alienate the listeners. As the glitchy sounds start to take over the piece the vocal sounds also begin to break down, not in a structural way but by being compressed and distorted. As the whole piece reaches its climax the sounds start to become unbearably loud and overpowering, very much in contrast to the serene mood created at the opening. The crescendo continues with no obvious sign of ending until it abruptly stops leaving only an almost unmodified playing of the ending of one of the two original recordings. This last ten seconds or so are there to remind the audience that although they have been taken through a dark and hostile experience there was always a strand of beauty and simplicity to lend a sense of safety.

There were problems to overcome when faced with planning this piece, firstly I had only read about *Anticredos* and had not heard it before the performance and secondly I was asked to use specific source material. It was made clear to me that the intention of the piece was to allow the performers time to make it from the rear of the performance space, where they had been singing, to the front of the room for their performance of *Anticredos*. At this stage it seemed obvious to play with the two opposing ends of the performance space given that the concert itself was to take place at both ends. The fact that *Pavanne* reverses the chronological and stylistic orientation of the space is not accidental, *Pavanne* swaps the room around to better prepare the audience for the change in genre that they are about to encounter and created a cultural rotational symmetry for the concert.

The theme of *Pavanne*, as with much of my music, is the exploration of the interplay between order and chaos as if it were possible to have a fader and change the level of entropy in a given system. In this piece it is not that the vocal lines themselves are ripped apart but that something unpredictable and disturbing creeps in to obscure the designed beauty of the human voice singing

highly structured melodies. It is as if the purity of its design is infected with something it cannot control or incorporate.

The reasons for returning to the simplicity of the voice at the very end of the piece are purely musical and not conceptual. In ending *Pavanne* at the climax of the crescendo the piece would have left listeners confused but by allowing order to exist at the closing, even if very quietly, it draws the listener back to what might be called a comfort-zone.

Chapter 7

Christ ist Erstanden

Christ ist Erstanden
Von der Marter alle;
Des solln wir alle froh sein,
Christ will unser Trost sein.
Kyrieleis.

Medieval Easter Hymn - *Christ ist erstanden*

Christ ist Erstanden was written during and after recording sessions with Jason Walsh and Loré Lixenberg. The recordings were, originally, for an Easter Passion Play with electronic interplay to be performed in a concert called *Osterspielfragmente* in the monastery at Klosterneuburg, Vienna. During these recording sessions the medieval hymn *Christ ist Erstanden* was also recorded in multi-track with the aim that a piece could be constructed from it. The original setting was taken from manuscripts found in Klosterneuburg Stiftsbibliothek and was transcribed by the British-Austrian composer Robert Crow.

Christ ist Erstanden is one of the oldest Easter hymns if not in fact the oldest[1]. Many variations exist in manuscript form across Southern Germany and Austria, but this version is taken from the Library of the monastery at Klosterneuburg. Crow then transcribed the chant into a six part setting. The setting was divided between three mezzo-soprano and three baritone parts. During recording Loré Lixenberg sang the three mezzo-soprano parts and Jason Walsh sang the three baritone parts.

[1] Schweitzer

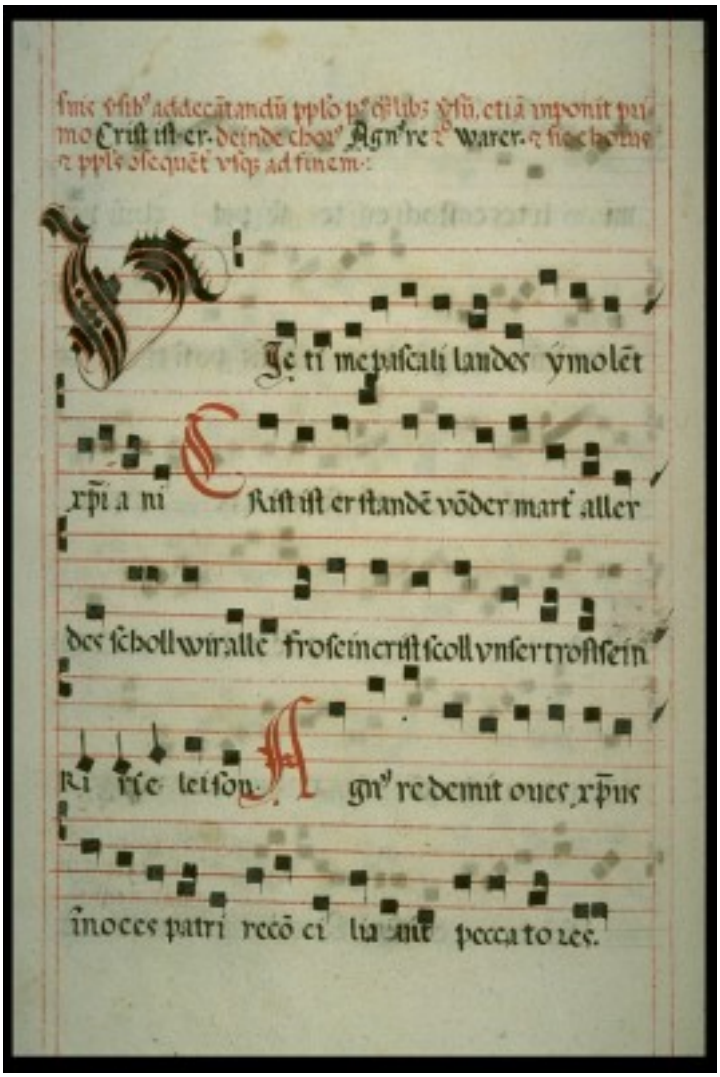

Figure 7.1: Illustration of *Christ ist Erstanden* from Codex Admont 323

The recordings were multi-track recorded and the result was used as source material for this piece. The facsimile manuscript shown in fig. 7.1 was originally from the Benediktinerstift Admont, about 125 miles west of Klosterneuburg. On the second line of this illustration there is a passage beginning with an large red 'C' which is the start of the passage used in the piece *Christ ist Erstanden*.

The first verse, sung by Jason Walsh, forms the majority of the source material. The words of the first verse are:

> **German:**
> Christ ist Erstanden
> Von den Marter alle;
> Des solln wir alle froh sein,
> Christ will unser Trost sein,
> Kyrie eleison.
>
> **English:**
> Christ is risen
> From all tortures;
> Therefore let us rejoice,
> Christ shall be our solace,
> Kyrie eleison.

Christ ist Erstanden opens with an unmodified presentation of the verse. On the '-son' of 'eleison' a stretched granulated version of the verse plays which is further processed by very rapid repetition of samples, on top of this is an ethereal extraction of part of the verse in reverse. At this point, not only are samples of Jason Walsh's voice made to sound an open fifth throughout, but there is also a layer of incomprehensible speech. The text of the speech is taken from a letter often attributed to Chief Seattle (1780–1866) concerning the ownership of land and the impact of modernity dated 1854. The letter was actually written by the screenwriter Ted Perry for *Home*, a film about ecology[2]. The text was recorded in a half-whispered voice and then chopped into three equal sections, these sections were layered on top of each other to further hide the actual words. The only words that were intended to be comprehensible were the last sentence of the letter, 'The end of living and the beginning of survival'. It was common in Medieval music for sacred and secular texts to be

[2] For the full text of the letter see: `http://www.essentia.com/book/history/chiefseattle.htm`

combined as different musical lines, a practice that persisted into the early Renaissance and into sub-genres such as the *L'homme Armé* Mass.

The text of the letter and the text of the song are in contrast with each other in mood, the song celebrates Jesus' resurrection on Easter Sunday and the letter laments the destruction of natural environment by irresponsible attitudes towards the earth. This opposition is expanded by the ominous and disturbing sound-world that is created by the electronic processing. If there were to be a moral to *Christ ist Erstanden* it would be that the world is not saved, and neither should we take 'solace' in anyone nor rejoice. This meaning might not be apparent from listening to the piece but it could be considered esoteric.

The multi-track recordings are then placed out of time in juxtaposition with each other in an attempt to disjoin the harmony of the original song. The samples are further broken by glitchy artifacts. These disjointed lines end with a warning 'beep', indicating a break in the sound-world, and a second 'beep' which announces the end of the break. At this point the song is played as originally written over the continuing background growls. The final ending of the piece is the same as the opening, it is as if the piece has gone nowhere and learnt nothing. It pieces together the song from its constituent parts, but ultimately only remembers the first verse.

Chapter 8

Rezoplucker

> The gesture and the tone, The entrance and the exit.
>
> Sathya Sai Baba

Despite the short amount of time that the Nintendo Wii has existed, there is nothing new about using its remote controller (fig. 8.1) as an interface for artistic expression. The fact that it can connect to an average computer via the Bluetooth protocol allows for its use by almost anyone for any purpose that the user can imagine. However, unless one uses the Max/MSP externals to gather the raw data one must script the data using a versatile programme called GlovePIE[1] which was originally written for manipulation of the P5 glove.

GlovePIE can take any number of inputs and map them to various outputs, for example, one can map a joystick to the controls of a mouse. When used in conjunction with MIDI Yoke one can take the raw data from the Wii remote and apply it to various MIDI control or note values.

To generate the script for GlovePIE I wrote a small Python programme (see listing 8.1) which would randomly assign note values to each of the three axes such that depending on which button is held down the accelerometers would change the velocity values of the relevant note.

This MIDI data was routed through the virtual patch bay, MIDI Yoke[2], and the data then

[1] http://carl.kenner.googlepages.com/glovepie

[2] http://www.midiox.com/myoke.htm

appears as a MIDI input in Reaktor.

I built a synthesiser within the Reaktor environment based upon the principles of physical modelling synthesis. In effect the synthesiser was a modified Karplus-Strong algorithm. However, instead of filtering a noise burst, an LFO controlling the frequency of a second LFO which in turn controlled the frequency of a triangle oscillator was used as the base tone. So, instead of noise an unpredictable collection of sweeping tones would be picked up as they passed the resonated frequencies. However, because this method could not guarantee an initial sound, a pitch sweeping triangle oscillator, from high to low, is struck as the sound is played. For more information on the *RezoPluckerSynth* see Chapter 3.

Figure 8.1: The Nintendo Wii controller (right) and Nunchuk (left)

The whole effect is that of a plucked sound somewhere between a harp and a guitar. To complete the instrument, the joystick of the Nunchuk (the second part of the Wii controller shown on the left of fig. 8.1) controls the brightness of the resonance on the Y-axis and pitch bend on the X-axis.

Before performance the player must run the Python file (listing 8.1) until a set of pitches is generated that is pleasing and appropriate for the specific performance. The Python file may be

Figure 8.2: The on-screen display of *RezoPluckerSynth*

edited if a subset of pitches is required, for example, if one wants to restrict the possibilities to the octatonic scale. When a GlovePIE script (listing 8.2) has been generated the performer must practise the precise set of movements required to create an interesting piece.

Due to the nature of the instrument, it is preferable that one attempts to create an introspective and meditative piece with enough silence between gestures so the audience can fully appreciate each set of sounds. The option of adding reverberation to the synthesiser is to be left to the particular performance and its context. If reverberation is added then the performance should allow time for the sound to dissipate between gestural combinations.

The instrument itself was designed to be a piece of music in its own right is as much as a score of a piece can be considered to be music. It exists as a set of possible sounds that can be created from its use. It is possible to think of it as such due to the inherent restrictions in its construction. Both gesturally and formally it is confined to a bounded space of possibility. Obviously, one cannot perform physically impossible gestures, nor can one produce sounds that the synthesiser is not

designed to produce. As such the instrument can only create a certain global musical sound despite possible variation in the form or structure of the piece. However, for the purposes of this portfolio, it can be considered an instrument with a text score, and as such it is seen as an improvisation piece.

Listing 8.1: Python programme for generating the GlovePIE file

```
#!/usr/bin/env python

###############################################
##   GlovePIE generator                      ##
##   by Kelcey Swain, 2008                   ##
##   Creates a randomised GlovePIE script    ##
###############################################

import random, time

# Create/open the GlovePIE file to work with.
f = open("wii.pie", "w")

# Names of the buttons on the Nintendo Wii controller as
# understood by GlovePIE.
buttons = ['A', 'B', 'Up', 'Right', 'Down', 'Left', 'One', 'Two', \
                'Minus', 'Plus', 'Nunchuk.CButton', 'Nunchuk.ZButton']
# Octaves we wish to confine our notes to.
octave = ['3', '4', '5']
# Note names as understood by GlovePIE.
note = ['c', 'csharp', 'd', 'dsharp', 'e', 'f', 'fsharp', 'g', 'gsharp', \
                'a', 'asharp', 'b']

def button(b, n1, n2, n3):
    '''Creates a block of code for GlovePIE that links button b
        to notes n1 on the x-axis, n2 on the y-axis and n3 on
        the z-axis. And makes the MIDI velocity of those notes
        proportional to the acceleration of the controler on
        that axis.'''
    a =  """if KeepDown(Wiimote.""" + b + """, 10 ms) then {
        midi2.""" + n1 + """Velocity = MapRange(Wiimote.RelAccX, \
                0 m per s per s,20 m per s per s, -2,1);
        midi2.""" + n2 + """Velocity = MapRange(Wiimote.RelAccY, \
                0 m per s per s,20 m per s per s, -2,1);
        midi2.""" + n3 + """Velocity = MapRange(Wiimote.RelAccZ, \
                0 m per s per s,20 m per s per s, -2,1);
        else
        midi2.""" + n1 + """Velocity = 0;
        midi2.""" + n2 + """Velocity = 0;
        midi2.""" + n3 + """Velocity = 0;
        };"""
    return a

def buttonN(b, n1, n2, n3):
    '''Does the same as the button() function but for the nunchuk.'''
    a =  """if KeepDown(Wiimote.""" + b + """, 10 ms) then {
```

```
        midi2.""" + n1 + """Velocity = MapRange(Wiimote.Nunchuk.RawAccX, \
                0 m per s per s,20 m per s per s, -2,1);
        midi2.""" + n2 + """Velocity = MapRange(Wiimote.Nunchuk.RawAccY, \
                0 m per s per s,20 m per s per s, -2,1);
        midi2.""" + n3 + """Velocity = MapRange(Wiimote.Nunchuk.RawAccZ, \
                0 m per s per s,20 m per s per s, -2,1);
        else
        midi2.""" + n1 + """Velocity = 0;
        midi2.""" + n2 + """Velocity = 0;
        midi2.""" + n3 + """Velocity = 0;
        };"""
    return a

def concatNote():
    '''Returns a note/octave string.'''
    return note[(random.randint(0, len(note)  1))] + \
        octave[(random.randint(0, len(octave)  1))]

# Perform the above functions for all buttons.
for i in range(len(buttons)):
    if buttons[i] == 'Nunchuk.CButton':
        f.write(buttonN((buttons[i]), str(concatNote()), \
                str(concatNote()), str(concatNote())))
        f.write('\n\n')
    elif buttons[i] == 'Nunchuk.ZButton':
        f.write(buttonN((buttons[i]), str(concatNote()), \
                str(concatNote()), str(concatNote())))
        f.write('\n\n')
    else:
        f.write(button((buttons[i]), str(concatNote()), \
                str(concatNote()), str(concatNote())))
        f.write('\n\n')

# Write the string to the GlovePIE file.
f.write("""midi2.Control31 = MapRange(Wiimote.Nunchuk.JoyX, -1,1, 0,1)
+ 0.01 \n midi2.Control32 = MapRange(Wiimote.Nunchuk.JoyY, -1,1, 0,1)""")

# Write an author comment at the bottom of the file to keep
# track of changes and when they were made.
f.write("\n\n//Generated by a python script by Kelcey Swain on ")
f.write(str(time.ctime()))

# Close the file.
f.close()
```

Listing 8.2: An example GlovePIE file generated by the Python file 8.1

```
if KeepDown(Wiimote.A, 10 ms) then {
midi2.gsharp4Velocity = MapRange(Wiimote.RelAccX,
        0 m per s per s,20 m per s per s,   2,1);
midi2.c3Velocity = MapRange(Wiimote.RelAccY,
        0 m per s per s,20 m per s per s,   2,1);
midi2.f5Velocity = MapRange(Wiimote.RelAccZ,
        0 m per s per s,20 m per s per s,   2,1);
```

```
8  else
9  midi2.gsharp4Velocity = 0;
10 midi2.c3Velocity = 0;
11 midi2.f5Velocity = 0;
12 };
13
14 if KeepDown(Wiimote.B, 10 ms) then {
15 midi2.b3Velocity = MapRange(Wiimote.RelAccX,
16         0 m per s per s,20 m per s per s,   2,1);
17 midi2.b3Velocity = MapRange(Wiimote.RelAccY,
18         0 m per s per s,20 m per s per s,   2,1);
19 midi2.d5Velocity = MapRange(Wiimote.RelAccZ,
20         0 m per s per s,20 m per s per s,   2,1);
21 else
22 midi2.b3Velocity = 0;
23 midi2.b3Velocity = 0;
24 midi2.d5Velocity = 0;
25 };
26
27 if KeepDown(Wiimote.Up, 10 ms) then {
28 midi2.b4Velocity = MapRange(Wiimote.RelAccX,
29         0 m per s per s,20 m per s per s,   2,1);
30 midi2.fsharp4Velocity = MapRange(Wiimote.RelAccY,
31         0 m per s per s,20 m per s per s,   2,1);
32 midi2.csharp5Velocity = MapRange(Wiimote.RelAccZ,
33         0 m per s per s,20 m per s per s,   2,1);
34 else
35 midi2.b4Velocity = 0;
36 midi2.fsharp4Velocity = 0;
37 midi2.csharp5Velocity = 0;
38 };
39
40 if KeepDown(Wiimote.Right, 10 ms) then {
41 midi2.gsharp3Velocity = MapRange(Wiimote.RelAccX,
42         0 m per s per s,20 m per s per s,   2,1);
43 midi2.fsharp3Velocity = MapRange(Wiimote.RelAccY,
44         0 m per s per s,20 m per s per s,   2,1);
45 midi2.a5Velocity = MapRange(Wiimote.RelAccZ,
46         0 m per s per s,20 m per s per s,   2,1);
47 else
48 midi2.gsharp3Velocity = 0;
49 midi2.fsharp3Velocity = 0;
50 midi2.a5Velocity = 0;
51 };
52
53 if KeepDown(Wiimote.Down, 10 ms) then {
54 midi2.c5Velocity = MapRange(Wiimote.RelAccX,
55         0 m per s per s,20 m per s per s,   2,1);
56 midi2.a4Velocity = MapRange(Wiimote.RelAccY,
57         0 m per s per s,20 m per s per s,   2,1);
58 midi2.gsharp3Velocity = MapRange(Wiimote.RelAccZ,
59         0 m per s per s,20 m per s per s,   2,1);
60 else
61 midi2.c5Velocity = 0;
62 midi2.a4Velocity = 0;
```

```
63 midi2.gsharp3Velocity = 0;
64 };
65
66 if KeepDown(Wiimote.Left, 10 ms) then {
67 midi2.fsharp3Velocity = MapRange(Wiimote.RelAccX,
68         0 m per s per s,20 m per s per s,   2,1);
69 midi2.f4Velocity = MapRange(Wiimote.RelAccY,
70         0 m per s per s,20 m per s per s,   2,1);
71 midi2.b5Velocity = MapRange(Wiimote.RelAccZ,
72         0 m per s per s,20 m per s per s,   2,1);
73 else
74 midi2.fsharp3Velocity = 0;
75 midi2.f4Velocity = 0;
76 midi2.b5Velocity = 0;
77 };
78
79 if KeepDown(Wiimote.One, 10 ms) then {
80 midi2.d4Velocity = MapRange(Wiimote.RelAccX,
81         0 m per s per s,20 m per s per s,   2,1);
82 midi2.fsharp3Velocity = MapRange(Wiimote.RelAccY,
83         0 m per s per s,20 m per s per s,   2,1);
84 midi2.a4Velocity = MapRange(Wiimote.RelAccZ,
85         0 m per s per s,20 m per s per s,   2,1);
86 else
87 midi2.d4Velocity = 0;
88 midi2.fsharp3Velocity = 0;
89 midi2.a4Velocity = 0;
90 };
91
92 if KeepDown(Wiimote.Two, 10 ms) then {
93 midi2.dsharp5Velocity = MapRange(Wiimote.RelAccX,
94         0 m per s per s,20 m per s per s,   2,1);
95 midi2.gsharp4Velocity = MapRange(Wiimote.RelAccY,
96         0 m per s per s,20 m per s per s,   2,1);
97 midi2.e5Velocity = MapRange(Wiimote.RelAccZ,
98         0 m per s per s,20 m per s per s,   2,1);
99 else
100 midi2.dsharp5Velocity = 0;
101 midi2.gsharp4Velocity = 0;
102 midi2.e5Velocity = 0;
103 };
104
105 if KeepDown(Wiimote.Minus, 10 ms) then {
106 midi2.gsharp5Velocity = MapRange(Wiimote.RelAccX,
107         0 m per s per s,20 m per s per s,   2,1);
108 midi2.csharp3Velocity = MapRange(Wiimote.RelAccY,
109         0 m per s per s,20 m per s per s,   2,1);
110 midi2.a5Velocity = MapRange(Wiimote.RelAccZ,
111         0 m per s per s,20 m per s per s,   2,1);
112 else
113 midi2.gsharp5Velocity = 0;
114 midi2.csharp3Velocity = 0;
115 midi2.a5Velocity = 0;
116 };
117
```

```
118 if KeepDown(Wiimote.Plus, 10 ms) then {
119 midi2.fsharp5Velocity = MapRange(Wiimote.RelAccX,
120         0 m per s per s,20 m per s per s,   2,1);
121 midi2.a5Velocity = MapRange(Wiimote.RelAccY,
122         0 m per s per s,20 m per s per s,   2,1);
123 midi2.d4Velocity = MapRange(Wiimote.RelAccZ,
124         0 m per s per s,20 m per s per s,   2,1);
125 else
126 midi2.fsharp5Velocity = 0;
127 midi2.a5Velocity = 0;
128 midi2.d4Velocity = 0;
129 };
130
131 if KeepDown(Wiimote.Nunchuk.CButton, 10 ms) then {
132 midi2.fsharp4Velocity = MapRange(Wiimote.Nunchuk.RawAccX,
133         0 m per s per s,20 m per s per s,   2,1);
134 midi2.a3Velocity = MapRange(Wiimote.Nunchuk.RawAccY,
135         0 m per s per s,20 m per s per s,   2,1);
136 midi2.a4Velocity = MapRange(Wiimote.Nunchuk.RawAccZ,
137         0 m per s per s,20 m per s per s,   2,1);
138 else
139 midi2.fsharp4Velocity = 0;
140 midi2.a3Velocity = 0;
141 midi2.a4Velocity = 0;
142 };
143
144 if KeepDown(Wiimote.Nunchuk.ZButton, 10 ms) then {
145 midi2.c4Velocity = MapRange(Wiimote.Nunchuk.RawAccX,
146         0 m per s per s,20 m per s per s,   2,1);
147 midi2.b4Velocity = MapRange(Wiimote.Nunchuk.RawAccY,
148         0 m per s per s,20 m per s per s,   2,1);
149 midi2.dsharp4Velocity = MapRange(Wiimote.Nunchuk.RawAccZ,
150         0 m per s per s,20 m per s per s,   2,1);
151 else
152 midi2.c4Velocity = 0;
153 midi2.b4Velocity = 0;
154 midi2.dsharp4Velocity = 0;
155 };
156
157 midi2.Control31 = MapRange(Wiimote.Nunchuk.JoyX,   1,1, 0,1) + 0.01
158 midi2.Control32 = MapRange(Wiimote.Nunchuk.JoyY,   1,1, 0,1)
159
160 //Generated by a python script by Kelcey Swain on Thu Feb 28 14:16:38 2008
```

Chapter 9

Cliqbuz

> A gesture cannot be regarded as the expression of an individual, as his creation, nor can it even be regarded as that person's instrument; on the contrary, it is gestures that use us as their instruments, as their bearers and incarnations.
>
> Milan Kundera

Cliqbuz, like *Rezoplucker* (Chapter 8), is an instrument designed to be a piece in its own right. Also, like *Rezoplucker* it can be viewed as an improvisation piece for the purposes of this portfolio.

This piece was originally inspired by the pioneering early 20th century instrument the theremin, created by the Russian inventor Léon Theremin. The theremin has two antennæ that sense the distance to the left and right hands of the performer using two capacitor based detectors[1]. Usually the right hand controls the frequency (pitch) and the left hand controls the amplitude (volume) of the resulting wave form. Whilst the principle of the instrument is very simple, the performance and practice of it is not so straightforward.

Cliqbuz grew from experimentation in creating a virtual theremin using Interface-Z's MIDI capture devices[2] and IRCAM's Ethersense[3] and a Reaktor back-end synthesis engine. Instead of using

[1] Manning, 5

[2] http://www.interface-z.com/

[3] http://recherche.ircam.fr/equipes/temps-reel/movement/hardware/index.htm

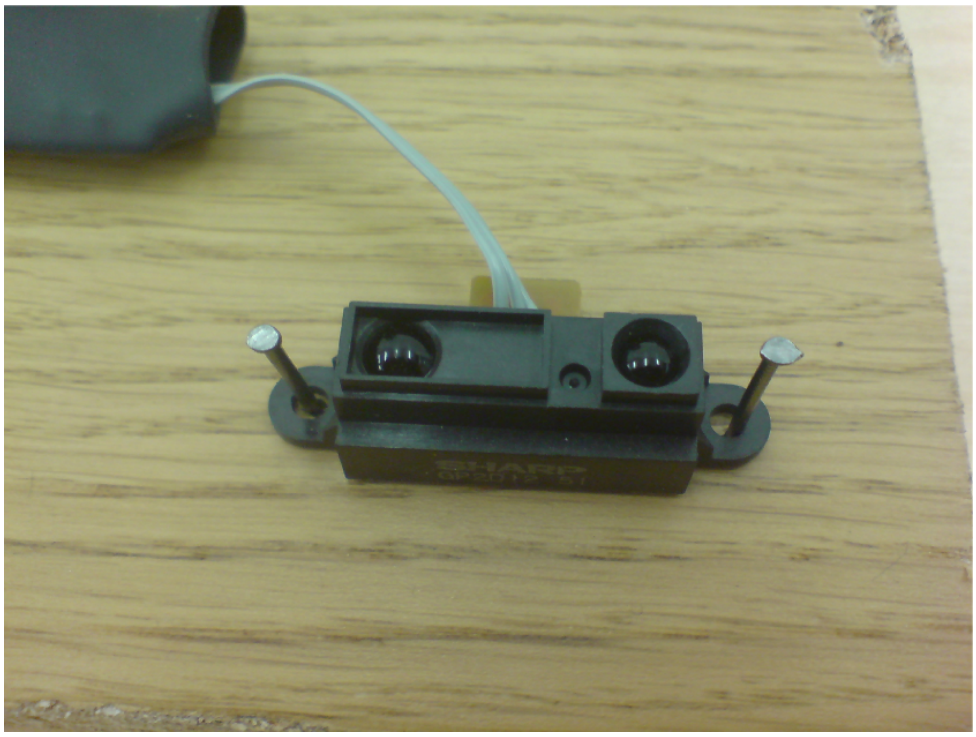

Figure 9.1: Infrared LED and receiver attached to a prototype board

electrical interference to sense the distance of the hands I used two infrared LEDs that could detect distances of up to around 50cm, as shown in fig. 9.1. The LEDs were connected into the input of the Ethersense interface which was in turn connected to the ethernet input of a computer running the OSC[4] compatible Reaktor modular environment. The right IR LED controlled the frequency and the left controlled the amplitude, as with the theremin.

At this point my continually adapting design of *Cliqbuz* departs from Léon Theremin's model and becomes an instrument in its own right. Two more IR LEDs were attached to the instrument in such a way that each hand could control two of the sensors, one at the fingertips and one at the wrist. Doubling the controls allows a doubling of waveforms, so the left-hand fingertips control the frequency of the first waveform and the right-hand fingertips control the frequency of the second waveform; the further away from the sensor the higher the pitch. To make the instrument more interesting without the need for too many controls the left-hand fingertips also control the rate of

[4]For more information on OSC see `http://opensoundcontrol.org/`

Figure 9.2: A prototype of the hardware of the *Cliqbuz* instrument

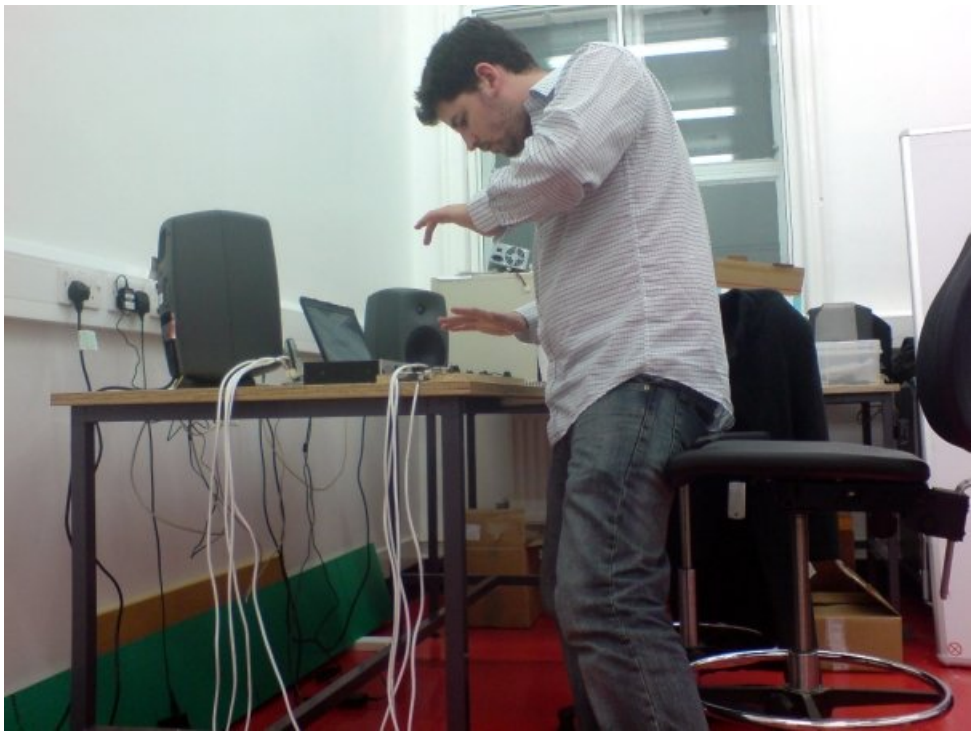

Figure 9.3: The composer working on *Cliqbuz* at Culture Lab

a triangle LFO on the amplitude of the second waveform, and the right-hand fingertips control the LFO on the amplitude of the first waveform; the further away from the sensors the quicker the LFOs. Instead of controlling the global amplitude of either waveforms, *Cliqbuz* uses the wrist controls to vary the width or envelope of the LFOs, making it possible to remove the effect of the LFOs completely by overlapping the attack and decay of the triangle LFO. For concert performance in a multi-channel environment a foot pedal was added to allow the performer to spin the stereo output of the instrument around the performance space.

The appeal of this instrument to an audience is in the real-time performance aspect, which is why no score is written for performance. Instead the piece should react to the context and environment. Firstly, audiences are not always accustomed to experiencing real-time electroacoustics when so much electronic music is based on prerecorded sound material. Secondly, the performer bypasses the problems associated with acousmatic music as the distinction between acoustic and electroacoustic is blurred by the visual correlation or legibility of gesture and consequence. Conversely, the instrument itself is often not visible during performance or at least not obvious to the audience, instead only the performers gestures are visible. This means that the audience enjoys both the sonic results of the performance and the clear movements or gestures that produce the music, which is amplified by the fact that the instrument is a non-contact instrument, leaving only the movements of the hands clear to the audience. In the words of one of the people who attended a performance of *Cliqbuz*, 'That instrument is a little bit like magic'. The video that accompanies this commentary shows the composer staring at the computer screen during the performances, this was due to technical issues and in a concert environment none of the technology that is used is visible to the audience.

Figure 9.4: The on-screen display of the *Cliqbuz* synthesis engine

Chapter 10

Circle Theory

> There is geometry in the humming of the strings, there is music in the spacing of the spheres.
>
> Pythagoras of Samos

Circle Theory is more a musical toy than a piece of music. It is designed so that it can continuously create music with or without human interaction. The basis for the *Circle Theory* was taken from a concept demonstration by Bill Orcutt, the creator of the graphical programming environment, Lily[1]. Lily itself is based on the design of Max/MSP and PureData but exists as a plugin for the popular web browser Firefox[2]. Orcutt made a modification of a circle packing algorithm, originally from a blog called Crickets Chirping[3]. In Orcutt's example he sent the data for the location and size of each circle via OSC to ChucK, the audio processing language developed at Princeton University, where the size of a circle controlled the amplitude of a sine wave and the proximity to the centre controlled the pitch of that sine wave, mapped upon a C major scale. Each of the circles is movable using mouse control and the algorithm is such that in most cases moving one circle causes the rest to move in response.

To make *Circle Theory* a stand-alone piece of music much was needed to be changed from this

[1] http://blog.lilyapp.org/

[2] http://www.mozilla.com/firefox/

[3] http://www.cricketschirping.com/weblog/2007/06/18/processing-sketch-circle-packing/

basic concept. Firstly, there is no need for the circles to be mapped onto a tonal scale so the code was adapted to produce continuous sine wave pitches instead of discrete, tonal step, changes. Secondly, if the circles constantly varied in sizes according to a random walk then the all the circles are going to be affected by this constant movement which makes it possible to leave the circles moving on their own without human interference. Lastly, the position of the circle controls the panning of its sine wave. The original design was to place the sine across a two dimensional field in 4 channels, however most performances required simply a stereo feed for practical reasons.The JavaScript code for creating the circles, packing them and sending the OSC signal can be seen in listing 10.1 and the ChucK code for receiving the OSC signal and creating the sine waves can be seen in listing 10.2.

The piece is performed in a dark room with an LCD projector projecting the image on the computer screen above or behind the performance space. The piece can then be left to run its own course or be controlled by a performer according to choice. The piece is ended by pressing the Enter key, whereupon all the circles slowly start to decrease in size until they are too small to make any sound.

Listing 10.1: JavaScript code for creating SVG circles and send the OSC signal out via Lily

```
var circles = null;
var iterationCounter = 0;
var dragCircle = null;
var numCircles = 10;
var offset =  0.95;
var toggle = 0;
var ampradius = 0;

function Circle(_x,_y,_radius,_id) {
        // Draws a circle at (_x,_y) with radius _radius and an id number.
        this.id = _id||0;
        this.x = _x||0;
        this.y = _y||0;
        this.radius = _radius||0;
        this.myColor = makeColor(64,64,64,64);
        this.draw=function() {
                fill(map(int(this.distanceToCenter()),1,275,255,50),
                    map(int(this.distanceToCenter()),1,275,255,50),150,1);
                stroke(64,64,64,64);
                strokeWeight(2);
                ellipse(int(this.x), int(this.y),
                        int(this.radius*2), int(this.radius*2));
```

```
                var distMod = int(map(int(this.distanceToCenter()),
                        0,200,0,42));
                if (this.radius > 3) {
                        sendMess(int(this.id)+1+" "+int(map(
                                this.distanceToCenter(),0,350,35,1700))+
                                " "+map(int(this.radius),7,(20*numCircles),
                                0,0.5)+" "+float(map(float(this.x),
                                0, 600,  2, 2)));
                }
                else {
                        sendMess(int(this.id)+1+" 0 0 0");
                }
        }
        this.contains=function(_x,_y) {
                var dx = this.x  _x;
                var dy = this.y  _y;
                return sqrt(dx*dx + dy*dy) <= this.radius;
        }
        this.distanceToCenter=function() {
                var dx = this.x  width/2;
                var dy = this.y  height/2;
                return (sqrt(dx*dx + dy*dy));
        }
        this.intersects=function(c) {
                var dx = c.x  this.x;
                var dy = c.y  this.y;
                var d = sqrt(dx*dx + dy*dy);
                return d < this.radius || d < c.radius;
        }
}

function setup() {
        // Initial parameters for the display.
        size( 600, 600 );
        smooth();
        fill( 0 );
        frameRate( 20 );
        // Create numCircle number of circles.
        circles = createRandomCircles(numCircles);
        background(0);
}

function draw() {
        background( 0 );
        for (var i=0; i<circles.size(); i++) {
          getCircle(i).draw();
        }
        for (var i=1; i<numCircles; i++) {
          iterateLayout(i);
        }
}

function comp(p1, p2) {
        var a = p1;
        var b = p2;
```

```
        if (a.distanceToCenter() < b.distanceToCenter())
                return 1;
        else if (a.distanceToCenter() > b.distanceToCenter())
                return  1;
        else
                return 0;
}

function Vector3D(_x,_y,_z) {
        this.x = _x||0;
        this.y = _y||0;
        this.z = _z||0;
    this.mult=function(f)
    {
        this.x *= f;
        this.y *= f;
        this.z *= f;
    }
    this.normalize=function()
    {
        var f   = Math.sqrt(this.x * this.x + this.y *
                this.y + this.z * this.z);
        this.x = this.x / f;
        this.y = this.y / f;
        this.z = this.z / f;
    }
}

function iterateLayout(iterationCounter) {
        // Control the interaction of the circles frame by frame.
        var circs = circles.toArray();
        circs.sort(comp);
        var ci = null;
        var cj = null;
        var v = new Vector3D();
        for (var i=0; i<circs.length; i++) {
                ci = circs[i];
                for (var j=i+1; j<circs.length; j++) {
                        if (i != j) {
                                // Change the size of the circles slightly.
                                ci.radius = ci.radius + (abs(random(2))
                                        +offset)* 0.075;
                                if (ci.radius > 75 && toggle == 0) {
                                        offset =  1;
                                        toggle = 1;
                                }
                                cj = circs[j];
                                var dx = cj.x  ci.x;
                                var dy = cj.y  ci.y;
                                var r = ci.radius + cj.radius;
                                var d = (dx*dx) + (dy*dy);
                                if (d < (r * r)  0.01 ) {
                                        v.x = dx;
                                        v.y = dy;
                                        v.normalize();
```

```
                                        v.mult((r sqrt(d))*0.5);
                                        if (cj != dragCircle) {
                                          cj.x += v.x;
                                          cj.y += v.y;
                                        }
                                        if (ci != dragCircle) {
                                          ci.x  = v.x;
                                          ci.y  = v.y;
                                        }
                                }
                        }
                }
        }
        var damping = 0.1/float((iterationCounter));
        for (var i=0; i<circs.length; i++) {
                var c = circs[i];
                if (c != dragCircle) {
                        v.x = c.x width/2;
                        v.y = c.y height/2;
                        v.mult(damping);
                        c.x  = v.x;
                        c.y  = v.y;
                }
        }
}

function createRandomCircles(n) {
        var circlesList = new ArrayList();
        while (n   > 0) {
                var c = new Circle(random(width), random(height),
                        (random(n)*1.25)+random(0), n);
                c.myColor = makeColor(0,.5);
                circlesList.add(c);
        }
        return circlesList;
}

function getCircle(i) {
  return circles.get(i);
}

function keyPressed(e)
{
    // If the spacebar is pressed make new random circles.
    var intKey = (window.Event) ? e.which : e.keyCode;
    if (intKey == 32) { //spacebar
        offset =  0.95;
        toggle = 0;
        circles = createRandomCircles(numCircles);
    }
    // If the return key is pressed make the circles get smaller.
    if (intKey == 13){ //return
        offset =  1.05;
    }
}
```

```
188
189 function mousePressed() {
190     // Allow mouse interaction with the circles.
191     dragCircle = null;
192     for(var i=0; i<circles.size(); i++) {
193         var c = getCircle(i);
194         if (c.contains(mouseX, mouseY)) {
195             dragCircle = c;
196         }
197     }
198 }
199
200 function mouseDragged() {
201   // Make the circle move when the mouse drags it.
202   if (dragCircle != null) {
203     dragCircle.x = mouseX;
204     dragCircle.y = mouseY;
205   }
206 }
207
208 function mouseReleased() {
209   dragCircle = null;
210 }
```

Listing 10.2: ChucK code for receiving the OSC signal and processing the sine waves

```
1  // Make 10 sine waves with reverb and panning.
2  SinOsc s1 => JCRev r1 => Pan2 p1 => dac;
3  SinOsc s2 => JCRev r2 => Pan2 p2 => dac;
4  SinOsc s3 => JCRev r3 => Pan2 p3 => dac;
5  SinOsc s4 => JCRev r4 => Pan2 p4 => dac;
6  SinOsc s5 => JCRev r5 => Pan2 p5 => dac;
7  SinOsc s6 => JCRev r6 => Pan2 p6 => dac;
8  SinOsc s7 => JCRev r7 => Pan2 p7 => dac;
9  SinOsc s8 => JCRev r8 => Pan2 p8 => dac;
10 SinOsc s9 => JCRev r9 => Pan2 p9 => dac;
11 SinOsc s10 => JCRev r10 => Pan2 p10 => dac;
12
13 .5 => s1.gain;
14 .2 => r1.mix;
15 .5 => s2.gain;
16 .2 => r2.mix;
17 .5 => s3.gain;
18 .2 => r3.mix;
19 .5 => s4.gain;
20 .2 => r4.mix;
21 .5 => s5.gain;
22 .2 => r5.mix;
23 .5 => s5.gain;
24 .2 => r6.mix;
25 .5 => s5.gain;
26 .2 => r7.mix;
27 .5 => s5.gain;
28 .2 => r8.mix;
```

```
.5 => s5.gain;
.2 => r9.mix;
.5 => s5.gain;
.2 => r10.mix;

// Receive OSC information to control the sine waves.
OscRecv recv;
8989 => recv.port;
recv.listen();
recv.event( "/circle/notes, i i f f" ) @=> OscEvent @ oe;

while( true )
{
    // Apply the OSC data to the sine waves constantly.
    oe => now;
    while( oe.nextMsg() )
    {
        int i;
        int j;
        float f;
        float pa;
        oe.getInt() => i;
        if (i == 1){
                oe.getInt() => j => s1.freq;
                oe.getFloat() => f => s1.gain;
                oe.getFloat() => pa =>p1.pan;
        }
        if (i == 2){
                oe.getInt() => j => s2.freq;
                oe.getFloat() => f => s2.gain;
                oe.getFloat() => pa =>p2.pan;
        }
        if (i == 3){
                oe.getInt() => j => s3.freq;
                oe.getFloat() => f => s3.gain;
                oe.getFloat() => pa =>p3.pan;
        }
        if (i == 4){
                oe.getInt() => j => s4.freq;
                oe.getFloat() => f => s4.gain;
                oe.getFloat() => pa =>p4.pan;
        }
        if (i == 5){
                oe.getInt() => j => s5.freq;
                oe.getFloat() => f => s5.gain;
                oe.getFloat() => pa =>p5.pan;
        }
        if (i == 6){
                oe.getInt() => j => s6.freq;
                oe.getFloat() => f => s6.gain;
                oe.getFloat() => pa =>p6.pan;
        }
        if (i == 7){
                oe.getInt() => j => s7.freq;
                oe.getFloat() => f => s7.gain;
```

```
84                  oe.getFloat() => pa =>p7.pan;
85          }
86          if (i == 8){
87                  oe.getInt() => j => s8.freq;
88                  oe.getFloat() => f => s8.gain;
89                  oe.getFloat() => pa =>p8.pan;
90          }
91          if (i == 9){
92                  oe.getInt() => j => s9.freq;
93                  oe.getFloat() => f => s9.gain;
94                  oe.getFloat() => pa =>p9.pan;
95          }
96          if (i == 10){
97                  oe.getInt() => j => s10.freq;
98                  oe.getFloat() => f => s10.gain;
99                  oe.getFloat() => pa =>p10.pan;
100         }
101     <<<"got (via OSC):", i, j, f, pa >>>;
102     }
103 }
```

Chapter 11

Conclusions

> I certainly had no feeling for harmony, and Schoenberg thought that that would make it impossible for me to write music. He said, 'You'll come to a wall you won't be able to get through.' So I said, 'I'll beat my head against that wall.'
>
> John Cage

As was stated in the introduction, the interests in this portfolio are the interaction between human and machine. Further to this is an interest in the interplay between order and chaos. In two cases the mouse and keyboard approach to electroacoustics have been abandoned firstly in favour of a custom built interface using Z-interface and Ethersense and secondly by using a control device that was designed for use with the Nintendo Wii. The synthesis engines for these two instruments were built specifically for use with their associated interfaces. The third instrument, *Circle Theory*, sticks to the standard computer controls but uses them in a way that allows for greater manipulability than normally possible, by moving one thing in this instrument you effect the totality of its constituent parts. All three instruments approach the gestural aspects of control in unconventional ways and should therefore leave no-one more or less capable of using them for performance. Each instrument requires a unique method of performance practice that can be learnt by almost anyone, regardless of musical training, due to the fact that preconceptions of how instruments should work were not taken

into consideration in their creation. The possible exception to this is *Cliqbuz*, on the grounds that its inspiration was grounded in the Theremin, but that pioneering instrument itself was innovative in its control mechanisms. *Rezoplucker* requires the performer to control it with flicks of the wrists and a few buttons, but due to the fact that these gestures are the end in themselves and do not make contact with an external device the movements can be more fluid than with conventional instruments.

The physical performance restrictions of these instruments are such that the performer is guided through the possibilities and not left with much room to stray from the intended sound-world. *Rezoplucker* uses the Wii controller to limit gestural movement, not physically, but by limiting the acceleration that can be detected, and the performer themselves limits the tonal environment that they can perform within before the piece is performed. In terms of physical restrictions, human hands are more comfortable making certain movements than others and this will further shape the musical output of the piece. For example, rotating the wrist of your hand whilst making gestural flicks is easier in some combinations than others and this can vary from person to person. *Cliqbuz* will only detect the hands at certain distances and even then it is not possible to make certain shapes with your hands in order to perform certain musical consequences. On top of this the synthesis engine crosses the controls for the right and left hands so that they rely on each other to perform any music at all. *Circle Theory* doesn't place any physical restriction on the performance that isn't already inherent in the control of most computers due to its reliance on the mouse and, to a lesser degree, keyboard. The instrument does, however, follow a strict set of circle packing rules that force the musical output to conform to its rules and structure. For example, small circles will almost always be forced into the centre whilst larger ones sit at the edges. The sonic result of this could be described as a form of subharmonic synthesis where the lower frequencies are quieter than the higher ones.

In this portfolio we have seen these three instruments as self-contained computer based music devices but there is no reason why the ideas and principles of these pieces could not be transformed

in hardware or software to be adaptable to other media or environments. The principles of this type of composition could be adapted and extended easily to incorporate non-linear mathematical functions in the form fractals, chaotic algorithms and neural networks. Currently the instruments are capable of being used in a solo or ensemble environment but the nature of the physical instrument could also be taken out of the context of the concert hall and into sound installations or wearable sound design equipment. The fundamentals of the instrument's design could be seen as a template on which to carry the work into new media such as hand-held devices like the iPhone and PDAs or alternatively bespoke equipment.

The real-time performance and audience interaction aspects of this music are also convenient ways of avoiding the problems of acousmatic listening in electroacoustic music, a problem I face with the other part of the portfolio, that of creating electroacoustic composition in a non-realtime and non-improvised environment. The challenges in sustaining interest of electroacoustic music over a large scale work where there is no performer to allow legibility of sonic results are difficult to overcome especially with abstract sounds that cannot be traced to any real world events or actions. There is no once solution to this problem as this is a fundamental issue in the art of composition especially in a time when the most listened to music is either short popular music or secondary in function such as soundtracks to films. It is my hope that the music in this portfolio finds its solution by presenting itself less as traditional music and more as ambient soundscapes. By soundscapes I do not mean real-world sounds presented as an immersive audio environment as found in *The World Soundscape Project*, but as a synthetic analogue to this. For example, *Heart of Light* presents believable industrial sounds that are all completely synthesised, *Light of Heart* focuses mostly on physical modelling to create a sound-world that is based in reality. The other pieces, such as *Pavanne* and *Christ ist Erstanden*, explore and expand upon the vocal sounds to make a total wash of sound. *Spanish Ladies* and *Down Among the Dead Men* use principals of electroacoustic synthesis in unusual ways, *Spanish Ladies* subverts the simplest ideas of granulation in its attempt to layer repetitive rhythmic ambience at the forefront of the piece whilst *Down Among the Dead Men* is an exploration

of complex synthesis with hidden vocals. Each one of these pieces explores the boundaries between order and chaos; between beauty and noise. It is my opinion that this boundary is where one can find the cutting edge of contemporary electroacoustic composition.

The running theme in this portfolio is the exploration of innovative techniques to create music made from sounds that have a richness of sonic textures and that show musical intention. The two contrasting methods of creation are unified in most cases by the fact that the statically composed pieces are made from similar systems, but are solidified at the point of composition instead of at the point of performance.

Bibliography

- Amato, Ivan. 'Muscle Melodies and Brain Refrains'. *Science News* 135, no. 13 (1989): 202–203.
- Bongers, Bert. 'An Interview with Sensorband'. *Computer Music Journal* 22, no. 1 (1998): 13–24.
- Bullen, Frank. *Songs of Sea Labour*. London: Orpheus, 1914.
- Chappell, William. Popular Music of the Olden Time a Collection of Ancient Songs, Ballads and Dance Tunes Illustrative of the National Music of England Part Two'. London: Cramer, Seals & Chappell, 1859
- Di Scipio, Agostino. 'Iterated Nonlinear Functions as a Sound-Generating Engine'. *Leonardo* 34, no. 3 (2001): 249–254.
- Jeffrey, Peter. 'Music Manuscripts on Microfilm in the Hill Monastic Manuscript Library at St. John's Abbey and University'. *Notes* 35, no. 1 (1978): 7–30.
- Knapp, Benjamin & Lusted, Hugh. 'A Bioelectric Controller for Computer Music Applications'. *Computer Music Journal* 14, no. 1 (1990): 42-47.
- Manning, Peter. *Electronic and Computer Music*. New York: Oxford University Press, 2004.
- Riedel, Johannes. *Leise settings of the Renaissance and Reformation era*. Madison: A-R Editions, 1980.

- Saunders, William. 'Sailors Songs & Songs of the Sea'. *The Musical Quarterly* 14, no. 3 (1928): 339–357.
- Schweitzer, Albert. 'The Origin of the Texts of the Chorals'. *Bach Chorales Website* (1908). See the website: http://www.bach-cantatas.com/Articles/CT-Schweitzer.htm
- Taylor, Arthur. *Notes and Tones: Musician-to-Musician Interviews*. New York: DaCapo, 1993.
- Varèse, Edgard. 'The Liberation of Sound'. *Perspectives of New Music* 5, no. 1 (1966): 11–19.