Playing with Technology - an approach to composition

Iain McDonald

Submitted in partial fulfilment of the requirements
of the degree of Masters in Music (*Music Technology*)

26/10/07

Faculty of Arts
University of Glasgow

Abstract

Approaches to the use of technology in music composition can be organised into two categories characteristic of the concepts of *bricoleur* and *engineer* as found in Claude Levi-Strauss' 'The Savage Mind'. This paper begins by arguing that the *engineer* approach is of more importance to the future of technology in music composition. It then documents the creation of five pieces, each intimately entwined with the use of technology and utilizing an *engineer* approach, outlining their associated intentions, technical implementations and concerns. Finally the ambiguous nature of the role of composer in these pieces is outlined and discussed.

Note

More details about the pieces discussed within this thesis are included on the accompanying disc in the form of a site capable of running on most platforms. This site contains video, audio, images and software documenting various aspects of the creation and final state of these pieces. Either Firefox or Safari applications are recommended for browsing the site.

Contents

Chapter 1

Introduction

Over the last 50 years the deployment of computers and other electronic devices has become increasingly prevalent in both music composition and production. Software typesetting environments such as *finale* have replaced the physical act of scoring music, allowing the composer access to tools which not only increase efficiency but add convenient editing functions; Digital Audio Workstations (DAW) have made editing and seamless re-sequencing of recorded audio available to a large cross-section of the population; and a range of other devices and software packages, too long to list, have afforded similar conveniences concerning musical creation.

Although these tools offer the aforementioned advantages within the creation of music they do so with respect to traditional western compositional paradigms[5], essentially increasing efficiency within this practice[3]. However, in doing this they obscure other advantages of using technology within composition, namely the ability to consider a more generalized approach to composition [22] [14].

One way to avoid this is to take an approach to the use of technology in composition wherein the composer is not only responsible for the creation of music but also, to some extent, the creation of the systems and tools used to realise this music. This type of approach can be seen in works such as Trevor Wishart's *Vox-5* and Jean-Claude Risset's *Duet for One Pianist* among others. In taking this approach the composer can develop a much more intimate relationship with the technology and therefore take advantage of its ability to aid the exploration of previously inaccessible compositional areas[27].

This report concerns five pieces, each intimately entwined with the use of technology, which were created over the last year by taking this kind of approach. Primarily the importance of this approach is outlined in chapter two followed by a review of the intention, implementation and concerns regarding each of the pieces in chapter three. Finally the main issue raised with regards to the creation of these pieces is discussed in chapter four.

Chapter 2

Background

Approaches to the use of technology in music composition can be organised into two broad categories characteristic of the concepts of *bricoleur* and *engineer* as found in Claude Levi-Strauss' 'The Savage Mind'[13]. These concepts describe two modes of human creation based on differences in attitude towards materials, tools and processes. The term *bricoleur* is used to refer to one who uses whatever materials and tools are at hand to accomplish projects. In contrast the *engineer's* projects are not limited by a finite collection of cultural artifacts. Instead the *engineer* is able to construct new materials and tools to realise his specific projects.

When considering these modes within the context of technology use in composition the *bricoleur* can be thought of as an avid user of tools for the creation of audio (i.e. software packages and black boxes that are immediately functional) to assist in the accomplishment of his goal. The *engineer*, on the other hand, prefers to use tools for the creation of tools (i.e. programming languages) to achieve his goal. It is important to state at this point that Levi-Strauss went to great pains to emphasize that, while these represent radically different forms of thinking, there is no superiority of one over the other[13]. However, we will now go on to discuss why the approach of the *engineer* is far more important to the future of technology within music than that of the *bricoleur*.

The problem with the *bricoleur* approach is not specifically the mode of thinking, but rather that the tools 'at hand' for use are, in a majority of cases, recreations of existing tools. The discipline of 'User Centered Interface Design' states that in order to create a well-designed tool we must first fully understand the task it aims to accomplish [19]. As our exploration of the uses of technology in composition is subject to continuous fluctuation, due to advances in technology, this could explain why many compositional tools have continued to model themselves on our traditional composition paradigm[5]. However, this could equally be attributed to the commercial nature of much software and their drive to accomplish "wide sales within a small marketplace"[18]. Irrespective of the cause, adopting this approach simply serves to produce tools that increase efficiency with regards to this type of practice[3] rather than facilitating the exploration of the new compositional landscape technology affords us.

Conversely the *engineer* approach allows us this freedom of exploration. Instead of relying on the existence of tools which harbour their own intentions, specific tools can be constructed by the composer with specific tasks in mind. The tools used by the *bricoleur* make use of 'interfaces' which are, by definition, an attempt to bridge the gulf between system and user therefore freeing the user from thinking in terms of the system [19]. The *engineer* approach allows for a much more intimate relationship wherein the user can come to understand the system as it is. This is important as the main advantage of working with technology is that it can open up our approach to composition by forcing us to consider a more generalised thought process[14].

In approaching technology in this way the composer is no longer tied to traditional compositional rules. Instead, he is free to create specific models for composition that can, theoretically, be based on anything from which data can be extracted. In this way movements on a skateboard[1], bridge vibrations[2] or the movement of cushions can all be considered as candidates for the basis of a compositional model. This approach also presents the means to explore ways in which these models can be interacted and intervened with.

The following section provides explanations and implementations concerning five pieces which were composed with an *engineer* approach to the use of technology. It was felt that it was not enough to simply supply the finished 'piece' as one would with notated music, rather, the whole model created requires explanation. As each piece essentially constitutes a different model of composition, relying heavily on the technology used, the implementation section was made as detailed as possible.

[1]Simon Morris' *Musique Concrete*
[2]Bill Fontana's *Harmonic Bridge*

Chapter 3

Pieces and Implementation

3.1 Programming/Scripting Languages

In the creation of these pieces the following were used:

- **Pure Data**
  Pure Data (PD) is a real-time graphical programming environment (similar to MAX/MSP) for audio, video and graphical processing video originally developed by Miller Puckette and others at Institut de Recherche et Coordination Acoustique/Musique (IRCAM)[1]. PD has since become open source and now has a large developer base working on new extensions to the program.

- **Objective-C**
  Objective-C is an object-oriented programming language used primarily in Mac OS X and GNUstep. Within this research it was used inside Cocoa[2]: an object-oriented application environment designed specifically for developing Mac OS X-only native applications.

- **AppleScript**
  AppleScript[3] is a scripting language devised by Apple Inc. that allows users to create automated workflows that perform repetitive multi-step tasks on their behalf.

[1]http://www.ircam.fr/ [accessed: 16th Oct. 2007]
[2]http://developer.apple.com/cocoa/ [accessed: 16th Oct. 2007]
[3]http://www.apple.com/applescript/resources/ [accessed: 16th Oct. 2007]

3.2 Live Break Core

3.2.1 Intention

This system was designed to allow a playful approach to the creation of 'breakcore'[4] within a live context. Due to the nature of this music most attempts at live performance result in playing preconceived tracks back to back whilst applying effects in real-time. While this works it tends to produce a stagnant environment within which both the artist and audience feel somewhat uneasy. This is usually attributed to the limited level of instant control the artist has over events on both a micro and macro scale.

This system is partially based on my approach to the composition of this type of music and endeavors to provide the user with control at the expense of effects[5]. Instead of focusing on ways in which the user could further 'spice up' preconceived tracks it gives them an environment in which music of this nature can be constructed 'on-the-fly' as it was thought that this would provide a more stimulating experience for both parties involved. It is important to note that it is not designed as a studio tool and that its development is iterative and in some ways still ongoing.

3.2.2 Overview

The system provides four tracks, each capable of holding and manipulating a single audio file which can be loaded into the track at any time. Once loaded, the audio file is indexed; a process which creates a list of rhythmically meaningful points within the audio file using analysis algorithms. The audio file can then be played as whole or reconstructed in a different order. Several playback attributes can be manipulated in realtime as well as buffer and time-stretching facilities. Furthermore, three standard effects are provided to allow simple shaping of the audio streams.

3.2.3 Technical Implementation

The PD framework consists of four main elements: Loading; Playback; Effects; Control. These will be outlined in the following sections.

[4]A genre of music typified by it's use of high BPM's, dense percussive rhythms and samples from both popular music and 'classic' jungle/rave/drum and bass.

[5]Similar systems have been built by other artists within this field such as 'Tim Exile' and 'Hardoff'.

Loading

The loading element of the framework allows the user to choose an audio file, either Audio Interchange File Format (aiff) or Waveform Audio Format (wav), from the hard-disk which is then stored in an array. It was found that when loading audio files, changes to certain parameters were required before playback would operate in the manner desired. As this was a time consuming process an algorithm was designed to set these parameters automatically using a numerical code (e.g. 2-1-4-8-1) allowing the user to focus fully on more important tasks. This code prefaces the file names of all audio files intended for use within the framework however, non-coded files can still be loaded without the use of this feature.

Once the audio file is loaded into an array it is indexed; a procedure that creates a list of rhythmically important index points within the audio file. There are two types of indexing offered, each tailored to a specific type of audio file. The first type of indexing caters for audio files which consist of a list of sounds separated by silence. This approach indexes each sound by monitoring the amplitude of the audio file as it is played. If the amplitude remains at 0 for $>= 50$ milliseconds the algorithm becomes primed and as soon as the amplitude becomes $! = 0$ the timed occurrence of this event is recorded as a new index point within the index list.

The second type of indexing caters for audio files which consist of continuous audio and essentially combines two different indexing approaches within the same algorithm. The first divides the length of the audio file by the number of beats it represents. This gives the length of one beat which is then used in an algorithm ,shown in Figure 3.1, to create the list of index points.

```
indexPoint = 0
for(i = 0; i<=numberOfBeatsInBar; i++)
{
indexArray[i] = indexPoint;
indexPoint = indexPoint + x;
}
```

Figure 3.1: Index point algorithm.

The second approach analyses the sample using the PD ‘bonk~’ object. This object looks for rapid changes in the spectral envelope as these are much more reliable indicators of percussive attacks than changes in the overall power[20]. Low and high threshold parameters allow the user to specify the sensitivity of the object and therefore what constitutes a ‘substantial attack’. The timed occurrence of each attack is recorded as a new index point within the index list.

Playback

There are two options concerning the playback of samples. Samples can be:

- played back based on trigger information received from a user controlled control surface
- played back following a user defined rhythm and accompanying algorithm

The first of these options uses trigger information received from a MIDI-controller (in this case the Korg padKontrol[6]). The user can choose whether or not trigger information is quantized and gated as well as which index points each pad will trigger (currently each pad can trigger two different samples based on the velocity at which it is struck). The second option requires a more in depth explanation.

Each track has it's own 2 by 52 grid, as seen in Figure 3.2, which can be used to

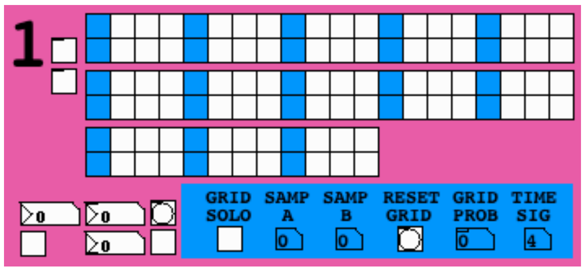

Figure 3.2: The grid for programming basic rhythms.

program basic rhythms. The lines in the grid can be assigned different index points using controls situated underneath the grid[7]. The columns represent a metered timeline which is stepped through at a speed designated by tempo, time-signature and track resolution. Figure 3.3 shows a standard four beat pattern entered into the grid where the first line is set to trigger a bass drum and the second line, a snare drum.

To complement the programmed pattern an algorithm was designed that, on finding no event programmed (i.e no cross on either line), would choose an unused

[6]http://www.korg.com/gear/info.asp?a_prod_no=KPC1&category_id=8 [accessed: 16th Oct. 2007]

[7]A possible configuration would be to assign an index point indicating a bass drum to one line, and an index point indicating a snare drum to the other.

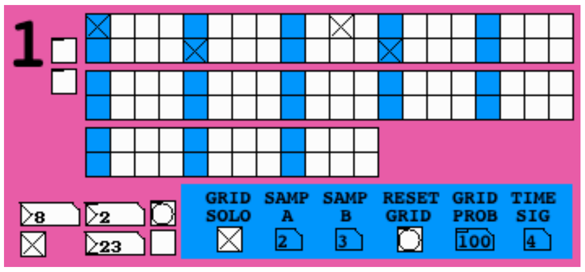

Figure 3.3: A simple four beat pattern represented within the grid.

index point and play it. Conversely, if both boxes are ticked the algorithm will play nothing.

It was found that as the pattern repeated it could become musically stale. In an attempt to combat this a user-controlled probability function was added allowing the user to specify how well the computer follows the defined pattern i.e. the percentage probability that it will play what is written. Finally a hi-hat feature allows a single index point from the audio file to be chosen and played based on probability values. There is a separate probability value for four consecutive beats which are looped through at the given tempo and time-resolution.

The combination of these elements provides a quick and easy way to produce patterns that have a strong rhythmic structure but also exhibit subtle changes that keep them interesting. In addition trigger information from the padKontrol can be used to either specify immediate location within the track's timeline and, if struck with sufficient force, cause an accent or change the playback pitch.

The timing of playback is ultimately managed by a sole metronome which feeds individual metronomes located within each track. This allows a global tempo to be specified, providing continuity between tracks, as well as individual resolutions for each track's timeline. This is particularly important given the temporal limits of the timelines.

Effects

In addition to playback parameters, such as 'pitch' and 'length', buffer and time-stretch effects are also supplied. When triggered a tempo-related slice of the audio stream is captured and loops until the user chooses to return to normal playback. The size of the audio captured (buffer) is chosen by the user, however, once captured the size of the buffer and speed at which the buffer is played can be varied as well as reversed. The time-stretch function is a simple algorithm that plays a specifically sized sample of the audio file whilst incrementing its starting position. The user can alter both the size of the loop and the value at which the start point increments. Alongside these, delay, filter and distortion effects were provided for rudimentary shaping of the audio.

Control

The control aspect of the framework was particularly important to allow effective and efficient manipulation of important parameters. Currently three different sets of controls are used to allow easy access to all parameters. Keys on the laptop keyboard are used to trigger events which have an on/off state in a 'push-to-make' [8] manner. In addition an array of 16 knobs on an Evolution X-Session MIDI-controller are used to control parameters which have a numerical state. Each line of the controller (8 knobs) and line of keys on the laptop keyboard can be assigned to any grouping of tracks. Alongside these control mechanisms a padKontrol MIDI-controller was used to allow the triggering of user definable index points of the loaded samples as well as control of timeline position and playback pitch.

3.2.4 Concerns

Within a system such as this both control and ease-of-use are important factors for consideration. The user must be able to express themselves sufficiently through the system however, in granting a high level of control the system should not become impossible to use. This is especially pertinent within the field of live electronics as the standard live environment presents several distractions and less room for error than the studio environment. Although the system, in its current state, offers a level of control which is deemed to be adequate the ease-of-use of the system is still felt to be lacking. One way in which this could be remedied is through the improvement of the interaction model used by the system with regards to either the control mapping strategies; devices used to enable interaction or both.

Mapping strategies refers to the manner in which physical interaction is mapped to system parameters - a process which can draw on three basic types; *one-to-one*, *one-to-many* (divergent) and *many-to-one* (convergent)[23]. In addition these basic strategies can be combined to create a complex mapping strategy termed *many-to-many*[8]. Currently all parameters accessible to the user are mapped to physical gestures using a manner which draws on both *one-to-one* and, at times, *one-to-many* mapping strategies. These types of strategy are generally used as they are simple to apply to a system[10]. However, through more prominent use of *many-*

[8]The key must be held down to be on

to-many strategies one can create an interaction system which allows the same, if not a higher, level of expression and engagement whilst lowering the cognitive load for the user[7] [16] [9] (an example of the implementation of this type of mapping can seen within the FMOL[10], T-Stick[15] and Sound Sculpting[16] systems).

Another method through which the operation of the system could be eased lies in the selection of different interfaces. At the moment the system utilises several interfaces[9] which each promote their own method of interaction; array of pads[10]; array of knobs[11]; keyboard as trigger array; keyboard as numerical interface; track-pad; and screen for visual feedback. Whilst the interaction methods of each of these interfaces suit the parameters they are mapped to, the overall effect can at times be overwhelming. This could be solved through the use of a single interface which could either be used in a multi-functional manner or would lend itself to a *many-to-many* mapping strategy. Ideally this interface would also supply visual feedback thus allowing the user to concentrate on a single interface for the duration of the performance. This could also further enforce the theatrics of live performance within the system by clearly illustrating that the performer is “necessary and is in fact controlling the music”[4]. Interfaces which would be suited to this type of application include the lemur[12], monome[13], and the more recently developed tenori-on[14].

There are also technical concerns which arose from the design and creation of this system, most important of which was the limitations of PD’s base architecture concerning threaded computing. The use of threads within a program allows several tasks to be carried out in a pseudo-simultaneous fashion. This is achieved through switching between threads (each undertaking a different task) several times a second. Without this type of support the program can only carry out one task at a time which must be seen to its end before another can begin. Even though PD now makes use of threads many of the objects written for the program have not been updated to take advantage of this resource.

The common ‘soundfiler~’ object used for loading audio files into an array within PD is one such object which does not take advantage of the multithreading abilities of recent PD releases. Unfortunately this means that when loading an audio-file using this object all other processing, including audio playback, must stop until loading has been completed. Thus every-time one wishes to load an audio-file a

[9]See 3.2.3 ‘control’

[10]Korg padKontrol

[11]Evolution X-session Controller

[12]http://www.jazzmutant.com/lemur_overview.php [accessed: 16th Oct. 2007]

[13]http://monome.org/ [accessed: 16th Oct. 2007]

[14]http://www.tenori-on.co.uk/ [accessed: 16th Oct. 2007]

silence within playback must be endured. This type of behaivour is unacceptable within a live context.

In an attempt to combat this problem an experimental object named ‘sndfiler~’[15], written by Tim Blechmann and Georg Holzmann, was compiled. This object provides the same functions as the previously discussed ‘soundfiler~’ object with the additional ability to make use of the threading functionality now embedded within PD. Whilst this allows the user to load audio-files without experiencing drop-outs in other processing, most importantly audio playback, the experimental nature of both the object and the use of threads within PD means that the overall system can become unstable at points. While this is not an optimum situation it is preferable to the previously mentioned problems regarding the use of the soundfiler~ object.

[15]http://grh.mur.at/software/sndfiler.html [accessed: 16th Oct. 2007]

3.3 Podcasts

3.3.1 Terminology

This piece makes use of Podcasts as a constantly replenishing supply of source material. A podcast is a feed published by an author in RSS 2.0 format, although ATOM 1.0 can also be used, which contains a list of enclosures that provide access to items or episodes of that podcast alongside other complementary data. Podcast consumers use a type of software known as an aggregator, such as iTunes, to subscribe and manage their feeds. Once a podcast is subscribed to the aggregator regularly checks the feed for new content which, if found, is then downloaded automatically.

3.3.2 Intention

In many ways, podcasting can be thought of as a new type of broadcasting, however, there are key differences between it and radio. Although both podcasting and radio deliver content to the listener, radio is limited in that the listener must be in both the right time and space if they wish to listen to specific content. Due to both the medium in which podcast episodes are presented (audio file) and delivered (through the internet) the listener is no longer limited by these scheduling and location factors.

In addition, the relative ease with which one can produce a podcast[16] means that the divide between those who listen and those who create found within radio is almost completely eradicated. In this way podcasting can be seen as realisation of radio experimenter Tetsuo Kogawa's suggestion that there should be "the same number of transmitters as receivers"[11]. This not only leads to a medium which encompasses a rich tapestry of thought but also sonic material.

This piece uses the spatial and temporal freedom present within the podcast medium to create an installation which juxtaposes podcast episodes and their related qualities.

There are of course, similarities present between this piece and John Cage's *Imaginary Landscapes No.4* in the use of broadcast technology. However, while Cage used the indeterminate nature of broadcast radio as the sound source he presented his piece within the confines of a fixed score, thus giving it a rigid structure[6]. In

[16]If one has access to the tools required to listen to podcast content they have the tools to create podcast content

contrast this piece is essentially an interactive installation[17] wherein the structure is fluid; created by those creating podcast material.

3.3.3 Overview

Podcast episodes are used as both source and control to create a theoretically infinite audio stream. Four podcast episodes chosen from a list of available episodes are loaded, played and analysed within PD. Attacks found within each episode are used as triggers for both selection of source audio from the episodes and control regarding playback options. The framework for the piece uses both iTunes and PD. Communication between the two applications was achieved using text files and a recursive AppleScript.

3.3.4 Technical Implementation

The framework consists of three main elements: Podcast subscription; Application Communication and Podcast Maintenance; PD Framework. These will be outlined in the following sections.

Podcast Subscription

iTunes was used as an aggregator to facilitate subscription to podcasts and, providing the installation is connected to the internet, subsequent automatic downloading of episodes as they become available.

Application Communication and Podcast Maintenance

Communication between applications and the maintenance of podcasts are taken care of by a recursive AppleScript and two text files. One text file contains the filenames of all podcast episodes that the PD framework can potentially access. The other contains the filenames of all podcasts that have already been accessed by the PD framework. Each time the AppleScript runs (every 15 minutes) it completes the following tasks in order;

1. **Delete used Podcasts**
   matches filenames found in the second text file with podcast episode filenames on the hard disk. This prevents the hard disk from becoming full.

[17] Albeit one in which interactivity is the same as the sound source and is thus subject to the same temporal and spatial flexibility present within the medium of podcasting

2. **Prepare new podcast episodes**
   deletes metadata, such as artwork, from episodes that have been downloaded using iTunes. The object responsible for reading mp3's within PD had trouble reading mp3's containing metadata.

3. **Document available podcasts**
   Writes a new text file containing all the names of podcasts available for use in the PD patch.

PD Framework

The desired output of the installation was a theoretically infinite audio stream that represented the podcast episodes playing at any given time. The best way to achieve this was to use the podcast episodes as both audio and control source. To these ends an algorithm was created, using the 'bonk~' object in PD, which would analyze the audio stream in real-time looking for 'substantial attacks'[18] (here referred to as rhythmic events). Any rhythmic event found by the analysis algorithm triggers the recording of the audio stream into a buffer for playback. Their occurrence in one podcast also trigger the playback of another channel's buffer. Therefore one audio stream's rhythm is superimposed on material sourced from another audio stream. Furthermore, rhythmic events cause the channel to take a snapshot of the current audio amplitude.

Certain aspects of playback are affected by the synchronicity and order of rhythmic events across audio streams. Here synchronicity refers to the possibility that separate audio streams may have rhythmic events that occur within a set time of each other (true synchronicity was deemed too improbable to be of use). Order refers to the sequence in which the audio streams involved in a synchronous event created rhythmic events e.g. all four audio streams had a rhythmic event within a set time in the order 2, 1, 4, 3.

This is measured at two levels, the first of which analyses the occurrence of rhythmic events across two audio streams in a specific order i.e audio stream one has a rhythmic event followed by a rhythmic event in audio stream four. The trigger information received from this analysis is used to control individual channel parameters such as overall speed of buffer playback; playback vibrato and the gradual slowing of buffer playback. The channel affected is the channel whose audio stream has its rhythmic event first; in the aforementioned example channel one would be affected.

[18]See 3.2.3 'loading'

The second level analyses synchronicity across all four audio streams using a similar algorithm. If ‘synchronicity’ occurs across all audio streams the algorithm outputs the numbers of the audio streams that were first and second. These numbers correlate to a specific way in which overall playback is affected; the first decides what aspect of playback will be affected and the second decides in what way.

Many of the parameters controlled by the ‘synchronicity’ of rhythmic events also make use of the snapshot data mentioned earlier. In these cases snapshot data either provides a value or informs a choice regarding a playback parameter.

3.3.5 Concerns

The main technical issue relates to the amount of source material required and the possible limitations of the medium through which it is acquired. Due to the concurrent playback of four podcast episodes within the system each minute created is done so from four minutes of source material. Thus the system requires approximately ninety-six hours of source material to run for twenty-four hours.

If the system was set-up with the intention of running for an extended period of time it would need to download the aforementioned ninety-six hours of source material each day, roughly equating to $5,317.2$MB[19]. Therefore, to run successfully the system relies not only on access to an internet connection capable of transferring data at approximately 492 kilobits-per-second(kb/s)[20] but also that a significant amount of podcast content is uploaded per day. These issues are raised here as they represent the most vulnerable part of the system, however, it is felt that, given a suitable internet connection, the system should run without fault.

[19] This is an approximation based on ninety-six hours of audio encoded using ‘MPEG-1 Audio layer 3’ (MP3) compression at 128kbps

[20] Average broadband operates at 512 kilobits-per-second

3.4 Data

3.4.1 Intention

Thanks to the ever decreasing cost of computers much music is now created with the use of commercial music software that essentially transforms the computer into a multitrack recorder, sequencer or synthesiser. However, many using this software are doing so through illegal means, specifically through the use of ‘cracks’ available on the internet. Essentially these cracks subvert the built in security of commercial music software thus allowing access to it without buying it.

This piece explores examples of cracked commercial music software, made to aid synthesis, sequencing and recording of sound, in their most elemental form: data. At this level the software becomes amorphous due to the commonality of the language; a string of 1’s and 0’s is open to interpretation through the lens of any file type. It is this amorphous property of digital data that is utilised within this piece to transform music software from tool for the creation and organisation of sound to sound itself.

3.4.2 Overview

A technique referred to as ’data bending’ was used to create the source material. Data bending involves reading one file type as another e.g. a Rich-Text Format (.rtf) file as a Joint Photographic Experts Group (.jpeg) compressed image file. In the case of these pieces non-audio files were sourced from illegally downloaded audio software that had had its security features breached. These non-audio files were subsequently read as audio files and then sequenced using a combination of Audacity, an open source audio sequencing program, and PD.

3.4.3 Technical Implementation

Although much of the work for this piece was done manually the limitations of Audacity meant that algorithms were designed within PD to allow easier sequencing of audio events as well as generation of content. For the most part these algorithms utilised a combination of pattern sequencing and probabilities.

3.5 Cushions

3.5.1 Intention

An advantage of computer use within audio lies in the ability to create sound without the constraints inherent in working within the physical realm[22]. With its use we can build sound-models, e.g. physical models for sound synthesis, based on any object; even ones that are, in reality, impossible due to their technological or physical make-up[28]. In addition, unlike physical sound generators such as musical instruments, once the model is specified we still retain the freedom to change the character of its behaviour.

However, while this increases our sonic palette exponentially the lack of physical presence within these sound-models presents a separate problem. In creating a sound-model within a virtual space we are cutting ourselves off from experiencing a physical relationship with the object and therefore limiting our interaction with the model. While this can be resolved through the use of interfaces the differences between sound-model and interface can give rise to physical-to-virtual mappings that are alien to the user.

This piece aims to create a simple 'sounding-object' which exists within both the physical and virtual realm simultaneously. This will be achieved through the use of a base model as the foundation for both the physical model and sound-model, representing the physical and virtual realms respectively. It is hoped that through this process the user will identify one object that produces sound when acted upon rather than an interface which relays data to a sound-model. On completion these 'sounding-objects' will become the focus of an installation.

3.5.2 Overview

Accelerometers connected to a bluetooth chip were positioned inside three equilateral triangle pyramid shaped cushions. Movements or forces exerted on the cushions were measured by the accelerometers and the resulting data was passed using bluetooth technology, via an application, to a PD patch. The PD patch's analysis of this data was then used to manipulate a physical-model based on the cushion and control other parameters of the sound-model which in turn formed the basis for real-time audio synthesis.

3.5.3 Technical Implementation

The framework consists of four main elements; Model; Data Acquisition; Data Analysis; Audio Synthesis; which will be outlined in the following sections.

Model

The base model, which both the physical and sound model use as a foundation, was an equilateral triangle pyramid as seen in Figure 3.4. This was chosen as it was

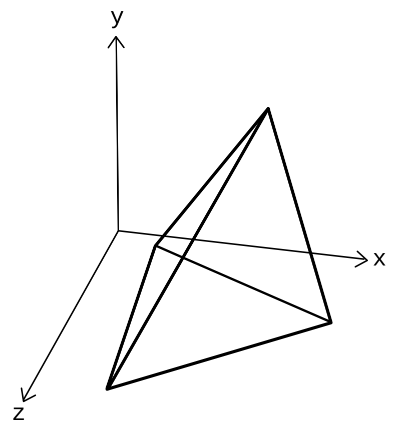

Figure 3.4: Base model used as foundation for both physical and sound models.

deemed a simple, aesthetically pleasing shape which would, due to its symmetrical nature, defy consistent orientation. This was important as orientation was seen as a potential state which could be monitored and used to indicate specific changes within the sound-model.

While the chosen model designated the overall shape of the physical model the means and material used to construct it were still in question. As the final aim was to produce‘sonic-objects’ which people would feel comfortable interacting with the physical model was realised as a cushion. It was felt that this design would not only encourage interaction but would allow users to be as physical with the object as they wished without having to worry about potential damage to the surroundings or the electronics inside the cushion.

Information regarding the implementation of the sound-model can be found in the ‘Audio Synthesis’ section.

Data Aquisition

When initially looking into accelerometers and bluetooth chips the costs were deemed to be too high considering the budget available. However, this changed with the release of the Nintendo Wii games console which used controllers (Wiimotes) that allowed the user’s motion to be tracked and interpreted by the console. This was achieved through the utilisation of three accelerometers, allowing measurement over three axis' as shown in Figure 3.5, and an IR sensor which all relayed data to the console via a bluetooth chip. However, consumer electronics meant these controllers were available at a much lower cost.

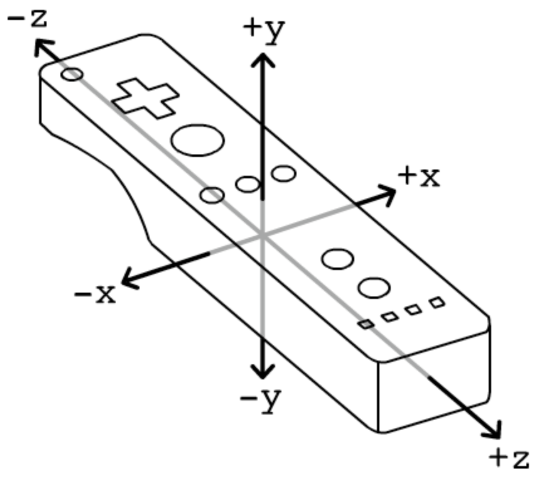

Figure 3.5: Three axis' measured by accelerometers in Wiimote.

Whilst the Wiimote embodied the technology required at a much lower price it also created a new problem. The controller was never intended for use with a computer, only a Wii games console, and as such used a proprietary handshake to pair with the console. This meant that whilst bluetooth enabled computers could see the controller they could not create a stable connection allowing them to receive data from it.

Fortunately there was much impetus within the development community to access the function of the controller with regards to use with a computer and subsequently development packages and drivers were released by individuals providing the basic framework to achieve this.

One such framework available for Mac, WiiRemote.framework[21] released by Hiroaki, was used in the creation of a small application named 'WiiToTCP' which allowed the computer to create connections to three separate controllers, receive

[21]http://sourceforge.net/projects/darwiin-remote/ [accessed: 16th Oct. 2007]

data from them and then send this data via Transmission Control Protocol (TCP)[22] to ports[23] which the PD patch was listening on.

Data Analysis

Before setting up algorithms to analyze the data from the controllers it was first necessary to fully investigate how they reacted to movements. This was achieved through analysing a graph read-out of the data received from the controller as it was moved[24]. This process revealed several things.

- If left stationary the data from the controller would still fluctuate.
- The controllers measure gravity which is affected by the accelerometers angle to earth.
- Whilst stationary these measurements are approximately 100 to 150 when varying the controller between two extremes on one axis
- Two controllers give slightly different readings when placed in the same position - they are not calibrated.

These issues were taken into account when designing analysis algorithms to recognize specific cushion movement.

In order to allow controllers to be interchanged a function was designed to allow for their calibration. The controller is left stationary in six different positions before it is inserted into the cushion. Once the reading settles in each position the relevant button on screen is clicked. This process creates three different calibrated versions of the data; the original; one that varies from 0 to 50; and one that is centered on 0. These are then used for further analysis.

There are three different algorithms that analyze calibrated accelerometer data. Each one looking for a data trend that would represent a specific type of physical exertion being directed upon the cushion and therefore the controller inside it. The types of physical exertion the algorithms are designed to recognize are:

- **hit** - a sharp impact is applied to the cushion
- **shake** - the cushion is shaken vigorously

[22]A network protocol which allows the creation of connections over which streams of data can be sent.

[23]End-points for sending and receiving data.

[24]Courtesy of Hiroaki's 'DarwiinRemote' application

- **bow** - smooth oscillation between two points

The first algorithm determines whether the cushion has been hit. The total sum of accelerometer readings is taken as the input. Each reading is then compared with the fifth last reading. If there is a difference of > 20 between them the cushion has been hit. This algorithm works on the basis that hitting the cushion will produce a sharp increase or decrease in accelerometer readings.

The second algorithm determines whether the cushion is being shaken. The total sum of all accelerometer readings is also used as its input. This data is used to calibrate a center reading x. If the reading then alternates between $> x + 2$ and $< x - 2$ five times per second the cushion is being shaken. The algorithm works on the basis that shaking the cushion will result in a continuous acceleration then deceleration of the overall accelerometer readings.

The third algorithm determines whether the cushion is being bowed. The algorithm is similar to the shake algorithm in that it takes the total sum of all accelerometer readings as it's input, uses this to calibrate a center reading x. If the reading then alternates between $> x + 4$ and $< x - 4$ the cushion is being bowed. However, this algorithm also measures the intensity of the bowing which is done by averaging the acceleration over 2.5 oscillations. This is based on the theory that a more intense bowing technique will produce a higher average acceleration.

Accelerometer readings are also used to vary parameters such as link properties within the physical model, pitch and Fast Fourier Transform (FFT) manipulation of the synthesized audio. While these are not directly linked to specific physical exertions they allow an extra level of control to be derived from the cushions movements thus creating a more complex sound object.

Audio Synthesis

The heart of the synthesis engine is a 3D physical model based on the equilateral triangle pyramid model[25]. This physical model was made using a Mass-Spring-Damper (MSD) system which allows masses to be placed in different loci within a virtual space and then connected using links. Once a force is imposed on a mass within the model it transfers through links to adjacent masses. Link properties such as rigidity and damping mean that the force will decrease as it passes from mass to mass. The model is anchored to fixed masses to ensure it will not float off once a force is imposed. If the algorithms discussed in the previous section show the cushion is either being hit, shaken or bowed these characteristics are artificially

[25]See 3.5.3 ‘Model’

simulated on the physical model. However, the physical model in itself does not create the audio, it simply imitates a real-world physical reaction to forces. The audio synthesis is achieved through a process called ‘scanned synthesis’.

‘Scanned synthesis’ involves the constant monitoring of the masses within the physical model[24]. Each time the model is updated the new X, Y and Z position, with regards to their original position, of every mass in the model is scanned and written to an array in a specific order. Essentially this process creates three dynamic wavetables, one for each axis, that represent the physical models state and can be read at an audio rate. The speed at which the wavetables are read through determines the pitch of the audio output.

3.5.4 Concerns

Although the final iteration of this system is deemed to be a success in relation to the original intentions it is within its creation that the limitations of both time and technology were most visible. Indeed there were several points where the creation of the system was forced to diverge from the chosen path due to either the complex computing knowledge required or technological limitations; the most notable of these concerned the synthesis and interactivity elements of the system.

Physical modelling synthesis was originally chosen as the basis for the systems synthesis engine as it offered a means by which a pseudo-natural audio output could be created that would be both representative of the physical shape of the real-world objects(cushions) and the forces applied to them. However, upon creation of a physical model within PD it became apparent that this would be impossible.

The ‘msd3D’ object used for the creation of physical models requires that it be ‘driven’ by a separate metronome object; each time the metronome ‘clicks’ the position of masses within the physical model are updated. Therefore, to create a high-quality audio signal the physical model must be driven at an audio rate i.e. the model must be updated 44,100 times per second. After several failed attempts and subsequent follow-up research it became apparent that PD was simply incapable of generating metronome ‘clicks’ at the speed required. Scanned synthesis was chosen as it was the closest achievable alternative that allowed realisation of the original aim, even if this realisation was only partial.

Further problems were met when investigating how to implement interactivity with the system through the analysis of accelerometer data. As mentioned earlier[26] *one-to-one* mapping of physical interface data to system parameters is simple to implement, however, it would not aid in the aim to create an expressive environment[9].

[26]See section 3.2.4

Therefore it was decided that a *many-to-many* mapping strategy would be adopted and in particular a gestural recognition element would be implemented within the system.

Much of the literature on the subject suggests that the most effective means for gesture recognition are either Hidden Markov Models[21] [12] [26] or Neural Nets[17] [1] [25]. Despite this neither model could be implemented because of the knowledge required; this area and its associated models represent a specialist area the scope of which was beyond this project. While this did not prevent the inclusion of gestural recognition within the system it did limit the spectrum of possible gestures to those simple enough that algorithms could be successfully designed to recognise them.

While these setbacks were not un-passable blockades making the piece impossible the process of negotiating around them did shape it somewhat. Indeed the differences and compromise between artistic vision and technological or practical feasibility were most recognisable within the creation of this piece.

3.6 Fruit

3.6.1 Intention

Still life images are recognisable as part of the heritage of western art. However, the iconic portrayal of bowls of fruit forever sealed in a single state has arguably passed over into the public consciousness. This piece seeks to release the image and place it back within its temporal context. In doing this the fruit is finally allowed to follow its natural course and decay. It is this process of decay which is highlighted by the addition of electronics and the subsequent sonification of what is essentially the fruits state of being.

3.6.2 Overview

Pieces of fruit are set up on a table in an arrangement indicative of 'still life' artworks. These pieces of fruit are then connected in series using electrodes which initiate a chemical reaction generating a voltage. Different groupings of fruit are then connected to circuits which produce a square-wave audio output indicative of the received voltage. As the fruit decays the voltage output by them decreases and the frequencies generated by the circuits change. Differences in fruit groupings will produce different rates of change within the voltage and therefore the frequency output by the connected circuit.

3.6.3 Technical Implementation

The framework consists of two main elements; Fruit Battery; Circuit; which will be outlined in the following sections.

Fruit Battery

The 'fruit battery' experiment is commonly used in many lower education establishments to display how a piece of fruit can create a voltage output. The experiment involves inserting a copper and zinc electrode into a piece of fruit. The reaction between the electrodes and the fruit creates a voltage. The voltage produced is somewhat reliant on the type of fruit used- a lemon can on average produce approximately 2.5v to 3v. To produce a higher voltage pieces of fruit can be connected in series in the same way normal batteries would be.

Circuit

The circuit itself is a common circuit based around a single 7555 timer chip[27] that allows the user to specify a frequency of operation. This can be done either by effecting the contol voltage with any type of variable resistor or supplying the circuit with a secondary voltage which can be compared to the control voltage. In this form it acts like a crude Voltage Controlled Oscillator (VCO) with square wave output.

[27]A low-power design of the standard 555 timer chip capable of implementing a variety of timer and multivibrator applications.

Chapter 4

Conclusion

Although I feel the pieces outlined in the previous chapter can be seen as somewhat successful in satisfying their initial aims, the process of their creation has raised one issue in particular which I would like to now discuss. This issue pertains to how I define myself in relation to the creation of these works and raises the question: am I truly the *composer* of these works?

Partial motivation for this question resides in the differences between the final product in these pieces and that traditionally expected within composition. Common composition practice strives towards an end in which a musical blueprint (score) is created or a piece is presented in recorded format, a factor which can be attributed to an obsession with the 'fixity of the work'[5]. With regards to the pieces mentioned in the previous chapter only one fits this description (data).

The final product of the remaining four differs in that they are dynamic systems that produce an audio output based on real-time interpretation of audio, physical events or the state of fruit. Whilst each system produces a similar audio output each time it is used the probability that these audio outputs will do anything more than exhibit a resemblance is small. This is because within these systems the 'music' is abstract, existing only in thought until it is realised through the use of the system.

Originally I assumed that because I had designed and created the systems I was, by default, the composer of the audio they output. However, on reflection this seems to be an over simplification of the relationships involved. While it is true that I supplied the rules of the system regarding the definitions of causes and their subsequent effects, during run-time control is delegated to either audio input, fruit or user; I can no longer influence the system's output.

The question of who is the composer then becomes far more complicated. Can I take the credit for the audio output as its composer even though I have delegated direct control over its creation? One could argue that the rules I have embodied in these systems are akin to those inherent in musical instruments. When playing a tuned guitar the action of plucking the ‘A string’ whilst fretting the 3rd fret of that string will produce the note ‘C’. This is absolute; the outcome of this action will always result in the same basic note[1]. Therefore a musical instrument can be thought of as a system embodying a specific rule set which governs how it reacts to events imposed upon it.

In this manner the systems I created would appear to be more consistent with the above definition of a musical instrument. Given this similarity the composer, or rather improviser due to its spontaneous nature, of any audio output would be the person or thing imposing the events on the system and not the creator of the system; no one would argue that Bartolomeo Cristifori di Francesco is the composer of all music played upon the piano simply because he invented it. Therefore although I can claim credit for the creation of the system I cannot assume credit as the composer of its audio output unless I am imposing events on the system myself. In this light I am an engineer of tools which aid composition, not a composer.

While this argument seems logically sound there are further factors that differentiate the systems in question from musical instruments. Due to the symbiotic relationship between musical instruments and compositional practice the design and creation of these instruments has had to adhere somewhat to the demands of compositional process[2], a point which their use of the 12-tone equal temperament pitch model attests to. Thus these instruments can be viewed as being built upon or in conjunction with a compositional model that exists separated from the instrument.

In contrast each of these systems is not solely based on this or any other previously defined compositional rules. In fact their design and creation also involved the creation and utilisation of specific compositional models entwined with both the technology and ideas central to that system. Whilst the compositional model embodied in most classical musical instruments allows for a particularly wide range of possibilities these compositional models offer a far narrower spectrum. This limited range of possibilities restricts any composition using the system to a relatively small niche.

It could be argued that by limiting the area of composition to such a small number of possibilities I am at least partially responsible for the audio output. Indeed we have established that as I do not have direct control over the events imposed on the system I cannot claim complete credit for the composition. However, in limit-

[1]Although there will be some difference present in the nuance of transients within the attack of the sound

ing compositional opportunities available, to whoever or whatever is imposing the events, to the point where each separate audio output of the system is distinguishable as being from that system I feel that I am more than simply an engineer of tools.

This question is further convoluted by the realisation that, in the process of creating these systems, my focus and passion was the technology rather than the music. Instead of beginning with a musical idea or aim which would then be accomplished through the use of technology I began with the technology. The technology would suggest certain possibilities that I would then mould into what I felt was an interesting aural output. The music then was somewhat a by-product of playing with this technology. This would indeed suggest that I am an engineer.

Putting technology first in the creation process could be attributed to an error on my behalf. However, I would argue that this problem is somewhat inherent in my research topic choice. In asserting that I would create pieces which were both reliant and entwined with technology I essentially shackled myself to its use. Subsequently any ideas for pieces were limited in that they had to contain technology. It is important to point out that this limitation was narrowed further as any technology used had to be of a type which was accessible, fitted within the budget and that I could develop an understanding of within a year[2]. As technology was the one absolute within the creation process it is not surprising that it became the primary focus.

However, one could argue that the process of putting technology first is comparable to writing a composition for a pre-specified group of instruments. Within composition of this nature the composer is limited to a group of instruments which suggest certain possibilities[2] in the same manner the choice of technology did regarding my process of creation.

This leaves me in a state of flux, lost between two absolutes. It would seem that what I am doing is situated somewhere between the discipline of a composer and that of an engineer. The feeling of uncertainty concerning my role within the creation of these artifacts is unnerving.

[2]See sections 3.2.4 and 3.5.4

Bibliography

[1] K. Boehm, W. Broll, and M. Sokolewicz. Dynamic gesture recognition using neural networks; a fundament for advanced interaction construction. *SPIE, Conference Electronic Imaging Science and Technology*, 98, 1994.

[2] W. Branchi. The State of Anxiety. *Computer Music Journal*, 7(1):8–10, 1983.

[3] C. Brown, J. Bischoff, and T. Perkis. Bringing Digital Music to Life. *Computer Music Journal*, 20(2):28–32, 1996.

[4] J. Chadabe. The limitations of mapping as a structural descriptive in electronic instruments. *Proceedings of the 2002 Conference on New interfaces for Musical Expression*, pages 1–5, 2002.

[5] S. Emmerson. Crossing cultural boundaries through technology? In S. Emmerson, editor, *Music, Electronic Media and Culture*, pages 115–37. Ashgate, 2000.

[6] C. Hamm. Privileging the Moment: Cage, Jung, Synchronicity, Postmodernism. *The Journal of Musicology*, 15(2):278–289, 1997.

[7] A. Hunt, M. Wanderley, and R. Kirk. Towards a model for instrumental mapping in expert musical interaction. *Proc. of the 2000 International Computer Music Conference. San Francisco, CA: International Computer Music Association*, pages 209–211, 2000.

[8] A. Hunt and M.M. Wanderley. Mapping performer parameters to synthesis engines. *Organised Sound*, 7(02):97–108, 2003.

[9] A. Hunt, M.M. Wanderley, and M. Paradis. The Importance of Parameter Mapping in Electronic Instrument Design. *Journal of New Music Research*, 32(4):429–440, 2003.

[10] S. Jordà. FMOL: Toward User-Friendly, Sophisticated New Musical Instruments. *Computer Music Journal*, 26(3):23–39, 2002.

[11] T. Kogawa. Toward Polymorphous Radio. In D. Augaitis, D. Lander, and W.P. Gallery, editors, *Radio Rethink: Art, Sound and Transmission*, pages 287–99. Walter Phillips Gallery, 1994.

[12] H.K. Lee and J.H. Kim. An HMM-based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):961–973, 1999.

[13] C. Levi-Strauss. *The Savage Mind*. University of Chicago Press, 1966.

[14] G. Loy. The Composer Seduced into Programming. *Perspectives of New Music*, 19(1/2):184–198, 1980.

[15] J. Malloch and M.M. Wanderly. The T-Stick: From Musical Interface to Musical Instrument. *Proceedings of the 2007 Conference on New Interfaces for Musical Expression*, pages 66–69, 2007.

[16] A. Mulder, S. Fels, and K. Mase. Empty-handed Gesture Analysis in Max/FTS. *Kansei, The Technology of Emotion. Proceedings of the AIMI International Workshop*, pages 3–4, 1997.

[17] K. Murakami and H. Taguchi. Gesture recognition using recurrent neural networks. *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, pages 237–242, 1991.

[18] G.L. Nelson. Who Can Be a Composer: New Paradigms for Teaching Creative Process in Music. *Proceedings of the Fifth International Technological Directions in Music Learning Conference*, 1998.

[19] D.A. Norman. *The psychology of everyday things*. Basic Books New York, 1988.

[20] M. Puckette, T. Apel, and D. Zicarelli. Real-time audio analysis tools for Pd and MSP. *Proceedings of the International Computer Music Conference*, pages 109–112, 1998.

[21] G. Rigoll, A. Kosmala, and S. Eickeler. High Performance Real-Time Gesture Recognition Using Hidden Markov Models. *Gesture and Sign Language in Human-Computer Interaction*, pages 69–80, 1997.

[22] C. Roads. *The Computer Music Tutorial*. Mit Press, 1996.

[23] J.B. Rovan, M.M. Wanderley, S. Dubnov, and P. Depalle. Instrumental Gestural Mapping Strategies as Expressivity Determinants in Computer Music Performance. *KANSEI-The Technology of Emotion*, 1997.

[24] B. Verplank, M. Mathews, and R. Shaw. Scanned synthesis. *The Journal of the Acoustical Society of America*, 109:2400, 2001.

[25] J. Weissmann and R. Salomon. Gesture recognition for virtual reality applications using datagloves and neural networks. *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, 3, 1999.

[26] AD Wilson and AF Bobick. Parametric hidden Markov models for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(9):884–900, 1999.

[27] T. Wishart. The Composition of "Vox-5". *Computer Music Journal*, 12(4):21–27, 1988.

[28] T. Wishart. *On Sonic Art*. Harwood Academic Publishers, 1996.