Software Influenced Composition

Investigating the influence of software on compositional methods in electroacoustic music

James Robert Surgenor

A thesis submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

The University of Sheffield
Faculty of Arts and Humanities
Department of Music

Oct 2019

Abstract

This research presents an autoethnographic investigation, through composition, performance and software development, into how composition is influenced by the use of software, utilising acousmatic composition as a case study.

Through the composition and performance of music, in addition to the development of software, it presents a framework for investigating acousmatic composition and its relationships to both the use and development of software in creative applications. Using this framework, it proposes a number of influences on compositional practice, and suggests that in the research of composition, software functions as part of both composition and performance rather than being an encapsulation of the tasks themselves.

It is accompanied by a number of compositions, sound examples and computer programs that both informed the investigations themselves, and explain their results.

It suggests a number of avenues for future research in both composition and performance, with relation to the utilisation and development of software, taking these results into account.

Dedication

For my grandparents.

Maureen Boyle & James A. Boyle
Ivy Surgenor & Robert O. Surgenor

Acknowledgements

This research would not have been possible without a great many people. I would like to thank the following:

My parents, Michèle and Peter. I cannot begin to express my gratitude for their constant love and support—this work would not have been possible without such a supportive team behind me. My brother Phillip and sister-in-law Allie, for supporting me over the years. My supervisor, Professor Adrian Moore for his time, patience, unwavering support and constant guidance—I cannot thank him enough for the support and opportunities he has given me. Dr. Adam Stanović, for his advice, support and friendship that have been a constant source of guidance throughout my time in Sheffield. Dr. Dave Moore, for constantly pushing me to believe in myself; his patience, technical support and friendship have been a source of great strength. Elsa Marshall, for providing an amount of tea, science-fiction and companionship that made the final year possible. My colleagues at the USSS for lending their time and ears: Alejandro Albornoz, Ian Baxter, Chris Bevan and Dimitrios Savva. Dr. Stephen Pearse for his friendship, guidance, encouragement and technical knowledge. My colleagues at the Department of Music for the community of support we worked hard to create: Ioanna Filippidi, Helen Gubbins, Tim Knowles, Shen Li, Yiyun Liu, Michael Walsh, Fengyi Zhang. Abbie Hill, for sharing a great deal of time and stories. Katharine Stone, for reguarly checking I was ok. Katie Williamson and Matthew Malone for their friendship and support. The amazing support staff in the Department of Music who have provided an incredible amount of help throughout my studies: Victoria Berry, Alley Bridge-York, Claire Cooper, Matt Jones, Rachel Plumb, and Nick Walsh. To everyone who has been a part of this adventure—I thank you.

Contents

CHAPTER 1

Introduction

This research has been focused on understanding the relationship between electroacoustic composers and the various software that they choose to utilise to create their music—more specifically, identifying the influence(s) that the software has on the compositional methods that develop. In order to do this, we must first outline such a compositional method, from which we can identify software's area of effect. Having done this, we can then begin to question what these effects might be.

Electroacoustic is primarily used as an encompassing term to describe a variety of musical activities involving electro-mechanical (re)production of sound: music requiring some form of electronic mediation—often that of reproduction over loudspeakers. As such, this term is of little use to us in our investigation, as there are potentially an incalculable number of approaches to the use of technological mediation. As we are trying to identify a compositional method, from the outset we must settle on the form of composition we are discussing.

Acousmatic composition has been chosen as a case study. In terms of composition, acousmatic music is primarily concerned with the exploration of musical properties in recorded and/or generated sounds, exploring musical relationships between them and their variations, created through manipulations with available technologies. The choice of acousmatic music allows us to focus on our question, without the need to consider additional complications such as the addition of musical instruments in live-electronics pieces, or the addition of visuals in the mixed-media realms.

Acousmatic music can be seen as music designed to be approached with, or presented so as to induce,

an acousmatic listening stance. This approach emphasises the importance of considering sound itself a musical entity, regardless of what created it[1]. The advent of technology led to the ability to record and reproduce sound, freeing composers to treat any sonic entity at their disposal as musical material, not just the sounds of musical instruments. As a result, there are no visual stimuli during performances, thus allowing listeners to focus on the sound itself. Schaeffer investigated the use of the available technologies not just to reproduce sound, but to manipulate it and create new musical materials. This practice of transforming sound and exploring relationships between them continues today, although in a different form. Schaeffer was experimenting in the late 1940s and 1950s, but just around the corner the digital revolution lay in wait. The technologies at our disposal today in acousmatic music are primarily software-based and are therefore the focus of this research.

For much of the last 40 years it has been acknowledged that the technologies we use have an impact on the ways that we work. Computers and the software running on them are often viewed as tools. As a result this impact is seen as an accepted risk or limitation. Despite the warnings that have quietly rumbled through the last four decades, little work has been done to understand what these influences are, and what role they play in composition. This research has sought to address this by conducting an autoethnographic study into acousmatic composition. It establishes a compositional framework within which these questions can be asked, and identifies the roles and influences of software in this context. Music and software have been written throughout as a primary data source.

1.1 Research Question

The main question of this research is whether software has an impact on compositional methods, what the impact might be, and how software might be designed in the future to accommodate these findings.

- How does software operate within the composition of acousmatic music?
- How does software influence the composition of acousmatic music?
- Can these influences suggest alternative designs for software?

[1] Acousmatic is a term derived from the teaching of Pythagoras, who taught from behind a screen or curtain. This enabled the students (Akousmatikoi) to focus on what was being said, rather than being distracted by the source of information. Schaeffer applied this term to musique concrète, where we aim to focus on the sound rather than its perceived source.

1.2 Methodology

The choice of method for this research is a result of attempting to address concerns from previous investigations. Perhaps the most obvious are interviews and/or observations. Interviews, although often rich in information surrounding compositional approaches, have tended to result in responses that converge on spontaneity and serendipity when dealing with compositional methods (Clowes, 2000; Eaglestone et al., 2007; Tracy, 2002; Upton, 2004). Observations, including speak-aloud techniques, have their usual concerns relating to the possible impact on the behaviour of those being observed (Eaglestone et al., 2007; Upton, 2004). As a result, this research has been carried out as an autoethnographic study.

Autoethnography is an under-utilised technique in this field of study (Magnusson, 2011). It is a process of positioning the researcher themselves (auto-) in the research, evaluated within a social and cultural context (-ethno-), through written documentation (-graphy). As such, this thesis tells the story of my lived experiences as an acousmatic composer, and serves as both data and analysis. As a method, it can be used to challenge long-held beliefs within fields. I must therefore begin by identifying myself within the wider context of what I'm researching. I am from Belfast, Northern Ireland, UK. I am an electroacoustic composer. I can write music in several electroacoustic domains, but I primarily write acousmatic music, and can therefore be considered, perhaps more accurately, an acousmatic composer. Before beginning this research, I studied Music Technology and Sonic Arts at The Queen's University of Belfast (2010–13), and Sonic Arts at The University of Sheffield (2013–14). I am therefore very much within the UK acousmatic tradition, whatever that might mean, and so many of my experiences are steeped in the works of Manuella Blackburn, Jonty Harrison, Adrian Moore, Denis Smalley, and Trevor Wishart.

Autoethnography has also been chosen to remove concerns about the different approaches of individual composers. It allows for variations *between* composers to be controlled. There are variations in practice that make them not entirely comparable (Adkins, 2014). There are variations in practice between pieces, even for a single composer, but such is the nature of creating a new work. Composition and the use of software is therefore best explored through autoethnography.

In terms of a more specific approach to this research, given that I am both a composer and a programmer, a cyclic flow of composition – reflection – programming was undertaken. Reflection mostly took the form of post-activity documentation: notes, an extended composition notebook, and personal reflection. These notes and notebooks are not included as part of the thesis, as their contents form the thesis itself. The programming was then informed by the reflections, and the cycle began again.

The research takes the form of a number of investigations into the various aspects that arose during this process. I began by identifying what I do when I compose (see Chapter 1), by composing a piece (*Twist and Turn*). From there, I gained insight into how software operates within my composition, and designed software to investigate those aspects (see Chapter 2: *linkEngine*, *le2*, *SOClib*, *lwae*, *chronon*).

1.3 Overview

The thesis looks at the two activities in acousmatic music: composition and performance. Chapter 2 introduces my compositional framework, as well as the piece *Twist and Turn*. Chapter 3 presents the role of software and its influences, including the software duality, as well as introducing the *SOClib* and the piece *Under the Spreading Chestnut Tree*. Chapter 4 identifies where software influences current performance practice, as well as exploring the links between composition and performance suggested in Chapter 2, investigating the influences of software on performance. It presents the software *SuperDiffuse*, and the piece *Under the Fractured Chestnut Tree*.

CHAPTER 2

Composition

2.1 Overview

The first stage in understanding the impact of software on compositional methods is to have an understanding of compositional method. This chapter presents the background for and the results of the initial investigation of this research: the composition of an acousmatic work, *Twist and Turn*, and the development of the *Fluid Framework* for investigating composition.

2.2 Existing work

As noted, we must first outline what composition might be. Importantly, we are not trying to define composition—this is often where the concerns surrounding individual differences arise—but to form an appreciation that allows us to question it. Adrian McKerracher states that 'Any effort to clarify the meaning of creativity, although productive, risks limiting this important concept to a singular definition at the exclusion of other valuable interpretations' (McKerracher, 2016). McKerracher presents a survey of metaphors commonly used to describe creativity, dividing them into the following broader categories: madness, possession, evolution, incubation, illumination, divergence, algorithm, investment, place, breaking a boundary, organism, and democratic attunement. If we first begin with the assertion that composition is a deliberate act, then it cannot rely on notions of mysticism. This is not to say that there are no elements of the unexpected, but rather that everything is *expectable*—the results of certain actions

may not have been predicted, but they must always be *predictable*—whether the composer is themselves able to predict the outcome is irrelevant (Boulez, 1986, p. 5)[1]. Applying this assertion to McKerracher's metaphors, we can disregard those of madness and possession.[2] The remaining metaphors occur frequently in other literature surrounding composition, and as such these references will provide examples of their meaning. As such, it is useful to keep these metaphors in mind throughout these discussions.

Horacio Vaggione challenges the idea that composition can be broken down into a set of processes or an algorithm (Vaggione, 2001). He suggests that it is something other than the sum of its activities, and whilst it may make use of plans and algorithms and scientific ideas, we must be careful not to reduce composition to one of these elements. This does not mean that we cannot gain an understanding of composition, but again reinforces that we should not aim to define it.

We have seen that composition/creativity ought not be defined, but can be appreciated; that it does not rely on mystery, but can involve the unexpected; and that it contains elements of chance and algorithm, but cannot be reduced to them. Taking this into account, we can contextualise existing research.

2.2.1 Rules and Restrictions

Composition involves making decisions. We are interested in how these decisions are made and what influences them. Margaret Boden proposes that creativity occurs when existing ideas are combined in new ways that form new ideas (Boden, 1995). Boden highlights that when presented with absolute freedom, our creative faculties cease to operate, proposing these rules and restrictions can answer the question of beginnings. Boden appears to suggest that these rules come from the creator[3] themselves, and their *conceptual spaces*. Boden's conceptual space is described as the mental realm within which the creator's mind is operating. It is perhaps easier to view the conceptual space as the framework of rules and restrictions that is constructed, within which the creator can be creative. It is the questioning within these spaces, and expansion of their boundaries where creativity occurs. This idea is additionally expressed by Jonathan Harvey, re-contextualising thoughts by Stravinsky, '[software aids] compositional freedom because they 'narrow down' the field of possibilities which then expand to reveal new worlds.' (Harvey, 1986, p. 190).

Our question remains where these rules and restrictions come from and how they develop. Monty Adkins's concept of *nodalism* offers some assistance here (Adkins, 2014). Adkins suggests that these

[1] For discussion of myths and mystery, see Gruenwals (1974), McClary (1989), Boden (2004), and Burnard (2012)

[2] Indeed, McKerracher notes that these metaphors are from historic philosophies, included to represent previous thoughts on the topic.

[3] I have used the term creator here as Boden is not specifically talking about composition, but rather creativity as a whole.

rules and restrictions can be thought of as memes: cultural ideas and practices thought of as behaving like genes with regards to evolution (mutation, etc), coming from Dawkins (1976). Nodalism posits that these memes are nodes within a neural network, combining the ideas of Boden and Dawkins. Nodes of thought and practice can mutate and evolve: either by the connections with other nodes (no matter how disparate), or by exposure to other nodes from other neural networks (the sharing of ideas and experiences with other people). The key factors of nodalism are that these networks are *decentralised*, and *distributed*. Nodes do not require a discernible starting point, and they can exist in isolated or localised forms without relying on others. This can account for the likes of individual differences and geographic variations in practice. If a node from another network is added, a copy is assimilated, rather than impacting the other network/node—although, any mutations of the node may indeed be incorporated back into the original. This is very similar to open source software, where each developer has their own copy of the program's code, and any changes are sent back to the maintainer of the project who might decide to incorporate those changes. These nodes are the result of everything someone has encountered, and new forms of practice or moments of specific creativity can be seen as the combination of disparate nodes in an individual's network. Nodalism can be viewed as the mechanism by which Boden's conceptual spaces expand.

2.2.2 MOSART Project

As part of the MOSART (Music Orchestration Systems in Algorithmic Research Technology) Project, there were a number of investigations into electroacoustic composition and the software in use. The methods used included interviews (Upton, 2004), surveys (Clowes, 2000), and observation (Nuhn et al., 2002; Tracy, 2002). Whilst each of these produced interesting and valuable results, it is important to be critical of the issues with the methods used. Interviews with composers may tend towards mystery and extremely personal aspects of their activities, as one might expect. There are also concerns around experimenter bias: most questions were asked from a software engineering perspective, as the ultimate aim was to inform the development of future software. There are also concerns over the design of such surveys, as well as the means through which they are disseminated and collected[4]. It is also a concern that Observation techniques may cause participants to behave differently.

Upton's work, 'An investigation into the compositional processes of Electro-acoustic composers', interviewed composers to identify their compositional processes. The term 'compositional processes' is understandably problematic given that it implies a reductionist stance. Regardless of this, however, Upton identified that composers work differently, with 'some valuing the scientific aspects of sound as primary

[4]In this case, the surveys were only sent to an online mailing-list

working elements' and 'some valuing the more functional properties of sound'. Upton is referring to composers who focus on the potential musical function of a sound within a phrase or piece, rather than for example, the investigation of particular synthesis techniques.

Nuhn's work, 'A Qualitative Analysis of Composers at Work', used observation, speak-aloud protocols, and a reflective interview. While the experiment seemed not to work as planned, Nuhn identifies some interesting aspects of composition. It concluded that more research needed to be done in order to understand the sensitising parameters required to conduct further research. It was clear, however, that composers used both convergent and divergent thought processes[5] in order to compose, and that time spent away from composing is vital. These exemplify McKerracher's metaphors of *divergence*, as well as *incubation* and *illumination*.

An interesting discovery as part of the MOSART project is the inclusion of the idea of memory as an important factor in creativity. This concurs with Adkins's nodalism, where disparate nodes of the network make connections to stimulate new ideas.

At this point, we are only concerning ourselves with composition. As such, the other findings from the MOSART project will be discussed later in Chapter 3. I am consciously separating the MOSART Project's findings on software here in order to avoid making assumptions about the role played by software.

2.2.3 Plans and Situated Actions

While researching human-computer interaction (HCI), Lucy Suchman identified the concepts of *plans* and *situated actions*:

> …this study adopts a view of plans just as formulations of antecedent conditions and consequences of action, which account for action in a plausible way …plans are necessarily vague, insofar as they are designed to accomodate the unforseeable contingencies of actual situations of action. Reconstructed retrospectively, plans systematically ignore the necessary ad hocness of situated action in favour of an account of the action as in accord with the plan …we need to look at how it is that actors use the circumstances that a particular occasion provides—including, but crucially not reducible to, formulations like plans—to provide for their action's developing purpose and intelligibility (Suchman, 1985)

Suchman describes situated actions as 'ad hoc responses to the actions of others and to the contingencies of particular situations', demonstrating this with the navigational methods of the Trukese who, '[begin] with an objective rather than a plan [and set] off toward the objective and [respond] to conditions as they arise in an ad hoc fashion …his effort is directed to doing whatever is necessary to reach the objective'

[5] Nuhn actually uses the phrase 'voyage of discovery'

(Suchman, 1985). Suchman goes on to note that this navigational style, in direct contrast with the plans of European navigators, uses the situation currently being dealt with and the information arising from it to progress, allowing the navigator to be able to 'point to his objective at any moment, but he cannot describe his course'.

As we have seen, composition is often viewed as a plan: a logical set of steps that produce the desired output. Upon building models to replicate it, something is always missing. These issues were exemplified by the work of Nick Collins, who used Music Information Retrieval (MIR) to build a set of tropes (essentially building Markov chains) from the canon of electroacoustic music, that he used to emulate the creative process of a composer (Collins, 2012). The identification of musical features through machine listening is interesting, and useful when used as part of an analytical framework, but as evidenced by Collins's Autocousmatic project the use of this to form models of compositional processes is problematic. The issue here is that composition is not the result of a plan, but rather the result of a series of situated actions that may or may not produce plans.

These concepts of plans and situated actions potentially account for the unforeseen. Situated actions tend not to be accounted for in the existing ideas of composition we have seen—they are the missing part of existing theories of which they themselves speak. So it is clear that we must not aim for a model of composition, but our understanding of what the act of composition is must be informed by situated action. This is not to say, however, that nothing can be learned from attempting to model composition.

2.2.4 Existing Models

While it is important to be wary of definitions, if we are to suspend our criticism of modelling creativity briefly, it is useful to consider what we can learn from those that have attempted it.

Richard Polfreman undertook a task analysis to develop a model of composition for use in the development of a piece of software. Task analysis is often used in the research of usability. It aims to identify what steps are required, often through observation, for a certain task to be completed. To do this, one must understand what the task is in its entirety and what completion means for the task. A summary of Polfreman's model is shown in Figure 2.1. Although Polfreman's research goes much further than the General Task Model shown here, from this overview we can clearly see two strongly linked operational paradigms: a knowledge-based research and planning paradigm, and an activity-based production paradigm. It is worth highlighting some aspects of this model, as they are present in what we have already seen, as well as other work yet to be discussed. In the knowledge/research paradigm, we see the

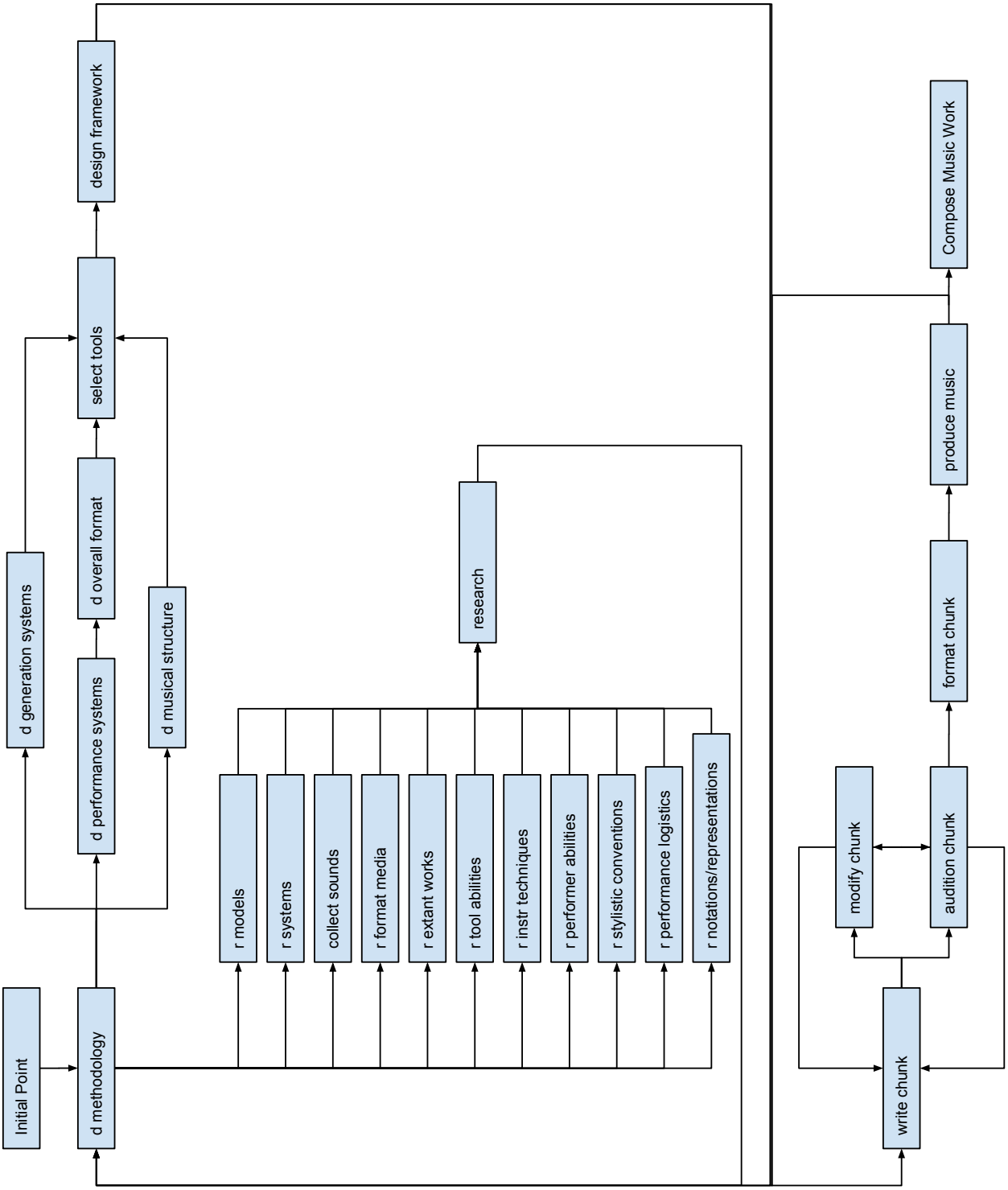


Figure 2.1: Polfreman's General Task Model (GTM) of Top Level Goals. Adapted from Polfreman (1999)

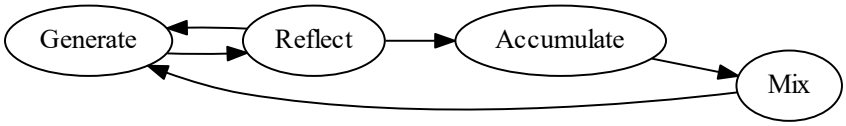


Figure 2.2: Moore's *work-reflect process* (A. Moore, 2016, p. 161)

design of *generation systems* and musical structures and selection of tools (see discussion of Wishart and Moore) and the design of a framework (see Boden and nodalism). There are also the incorporations of *stylistic conventions*, *extant works* (see Boden and nodalism) etc.

In Figure 2.1, we see certain elements labelled with 'd', for design, and 'r' for research. These denote the category of the goal within the model. The nodes within the diagram are top-level goals and as such are not aimed at identifying intricate details of the task but rather serving as descriptions of larger goals. Overall, Polreman's model contains three goals. Firstly, the design of a framework that 'covers both music-related constraints (e.g. instrumentation, musical structure) and practical ones (e.g. tools to be used in the composition process, the final format of the work)'. Secondly, the research goal, involving aspects such as the investigation of extant works, collection of sounds and stylistic conventions. Finally, Polfreman identifies the 'produce music' goal, shown at the bottom of the diagram. This incorporates activities that go towards the 'creation of the final deliverable product'. Importantly, Polfreman states that 'in general terms, the goals of *design framework* and *research* are carried out, at least partially, before that of *produce music* which leads to the completion of the piece'. Finally, it is important to note that Polfreman incorporates the idea of *partial completion* for these goals, which can lead to partial completion of other goals. Additionally, the composer can jump between these goals at any point as work on the piece progresses. It is worth keeping in mind the aspects of this model as we discuss other approaches (Polfreman, 1999).

Adrian Moore identified a process for composition based around his own practice, shown in Figure 2.2. Here, Moore suggests that the Generation of materials may start with the recording, or synthesis, of sound; most likely the starting point of a piece. Then, the Reflections on the generated sound drive new generation(s) of material which begin to Accumulate. The process by which these new generations are developed, i.e. what transformations are utilised, is based on the Moore's Theory of Opposites: '…we choose transformations based upon a reaction against a sound's most potent descriptor.' (A. Moore, 2016, p. 93). The Mix stage is where a composer begins to combine these materials to form composites, which are then treated as materials and the process repeats. These then grow from individual sound

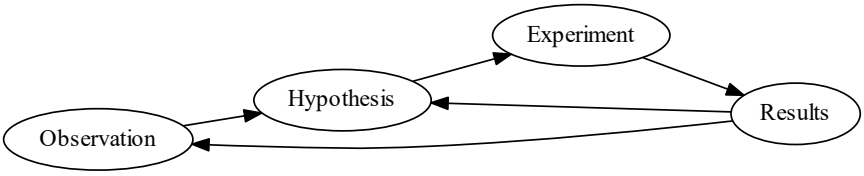


Figure 2.3: Scientific Method

objects, into phrases and sections, and eventually an entire piece. It is interesting to compare the process to that of the scientific method (Figure 2.3): an observation is made about a sound (reflection), an experiment is devised and implemented (generation, mixing), with the results stimulating further investigation (accumulation, reflection). The links between science and art have been discussed many times[6] but it is interesting to see such a direct parallel coming from composition research. It is also important to point out the links between Moore's process and Polfreman's model. If we remove the knowledge-based paradigm from Polfreman's model, we see a variation on the work-reflect process. Neither one is wrong, but the difference highlights their contrasting approaches to the question. Polfreman's model can be viewed as grounded in Suchman's idea of *plans*, whereas Moore's process utilises *situated actions*—suggesting that a piece is derived from the development of materials; coming from the sounds themselves (A. Moore, 2016, p. 7). It is important to consider that in Moore's process, although knowledge is not specifically mentioned, it is by necessity embedded in each of the stages. I would additionally propose that each stage has its own knowledge with which it is informed and expanded. This is in contrast to Polfreman's model which suggests that knowledge is a separate activity.

Trevor Wishart has written on three major topics concerning electroacoustic music composition: aesthetics (Wishart, 1996), sound manipulation techniques (Wishart, 1994), and structuring of materials (Wishart, 2012). Wishart (1994) is concerned with identifying what can happen to certain sounds when put through certain manipulations, creating an understanding of sound, process and output. This is certainly a large part of composition, and indeed Moore agrees that most time is spent in this generation/reflection stage. Wishart states that each book covers a specific topic, so *Audible Design* does not cover why specific processes might be chosen. He notes that this is covered in *On Sonic Art*, but as this text focuses primarily on aesthetics and what musical aspects of sound are at the disposal of a composer, the *why* here is somewhat different. As such, the rationale behind choosing certain procedures is left largely to the reader. Wishart's third and most recent book, *Sound Composition*, discusses the structuring of

[6] See Auger (1981), Jaumotte (1972), Lionnais et al. (1969), Loeb (1971), Monro (1997), Preusser (1973)

sound materials once they have been created. In it, Wishart documents how he composed a great number of his most influential pieces. From it, we learn that Wishart's primary compositional interest is sound transformation. As such, the book details how he decided to transform from 'sound A' to 'sound B'. This could be seen as proposing that one knows where one wishes to end up, with it being a matter of getting there, however this is not the case—this text focuses on what happens *after* manipulations are performed and materials are generated.

2.3 Identifying Composition

In order to identify my own compositional practice(s) I first wrote a piece. It is however interesting to note that I did not start out *writing a piece*, but rather with some related activities. When I arrived in Sheffield to undertake my research, I decided to take it as an opportunity to work with some new sounds. I had been composing with the same materials for a number of years and this was as good a time as any to work with new sounds—a clean slate for the start of a new adventure. At this point, I was not creating a new piece, but gathering sounds that might eventually be used in a future piece.

At the time, I had been watching videos on how foley was created for films and television and was intrigued by what appeared to be the frequent use of fruit and vegetables as sources. Foley artists perform with these physical objects to (re)create or replace sounds for visual media. Thinking in terms of acousmatic music, the links are strong and inspiring. The sounds created during foley are not only removed from their sources, as with acousmatic music, but new sources are inferred from their application to visual media. This can be seen as one evolutionary step away from acousmatic music. As such, I decided to investigate what would happen if some of the practices from foley were used in acousmatic music, where new sources are inferred from listening rather than visual media. I organised a recording session with a fellow PhD student, Chris Bevan, to capture some sounds with fruit and vegetables. As is common in Sheffield, composers will often hold long recording sessions together. This may not be common elsewhere, but the main reason is most likely to do with practicalities: studios and microphones take time to set up, so working together makes everything easier. The understanding with these joint recording sessions is that each composer will bring the objects that they wish to record, and the other will act as a recording engineer. Additionally, composers will tend to share the recordings with one another afterwards, so everyone gains from the collaboration. All of this was true for this session. Of the objects I brought, the most important turned out to be oranges, celery and plastic straws. We recorded using two microphone arrays: a spaced pair (cardioid) and a mid-side pair (omni mid). I chose this as I was not

ultimately sure what would work best and did not want to spend a great deal of valuable recording time figuring that out. The clue here is the use of mid-side, as it allows us to alter the stereo-image after the recording is made; although it is not without its side effects. My concern was with the sounds I wished to capture, not the technology used to capture it. We recorded about an hour's worth of audio, although the session lasted several; most time was spent setting up microphones, or discovering what manipulations of the objects resulted in interesting sounds.

Following the recording session, I returned to the studio a day or so later to edit out the in-between-takes. I was left with ultimately very little material and was quite demoralised at this discovery. A lot of the sounds that I thought had been really interesting at the time of recording seemed now to be useless, or worse, to have some technical issue in the recording rendering them useless. As such, I did not take these sounds any further and began working on other aspects of my research, mostly reading and software development.

A month or so later, I was working on learning a particular aspect of the SuperCollider programming language: the pattern system (PS). It provides a relatively simple way to describe complex patterns of behaviours that can be applied to sound generation. The PS is primarily an event scheduling system and anything that SuperCollider can do can be an event. The patterns can be changed at any time and the PS will automatically cross-fade between the two, avoiding any glitches. Quite often it is used in live-coding performances and due to its nature is primarily used to create rhythmic, beat-based material. I was interested in discovering a way to utilise the pattern system to generate something other than regular, rhythmic material. It turned out to be possible with a variety of techniques, from randomising the duration between each event, to having each event manage its own duration; thereby only using the PS to trigger the start of an event, allowing events to overlap. To test this, I was triggering a simple playback Synth[7]. Each triggering of a Synth was an event, and each Synth could playback a soundfile, or section thereof, at a particular speed. The soundfiles I used were from the aforementioned recording session, but it is important to note that I was at this stage only testing what would happen as a result of the PS being used in this way. I grouped the soundfiles according to source, and began experimenting with the patterns. If the soundfile remained the same for each event, as one might expect, it produced a form of granular synthesis. However, I now had full playback controls for each grain, including which soundfile it used. I could therefore have large grains, exceeding 1 or 2 seconds, and create very dense textures of sound. I sat in the chair and listened to the various results I was producing. It was only when I realised I had

[7] A Synth in SuperCollider is a collection of unit generators (following the work of Max Matthews). Put simply, it is a way of describing how sound is synthesised/processed, and should not be confused with a synthesiser (although they can be written to perform the same function). Other languages may use the term *instrument* but this is naturally problematic.

been sitting for 20 minutes listening to this generation algorithm that I realised that there was something interesting. If I could sit and listen to this process for such a long time, then there must be something to work with. It was at this point I decided to write a piece.

I began by simply recording long outputs from the process, letting it run as before but this time capturing the output. This made the sounds concrete—these recordings would not change, whereas the output of the PS was constantly changing. I would simply walk away from the computer, returning sometimes 20 minutes later. Once I had these for each group of sounds, I did the same for combinations of them. I now had clouds of sound for each group and each combination of groups. I noticed that certain combinations of groups, when given certain playback rates, were generating what sounded like rain and so purposefully set about to record a cloud of sounds with this texture.

It occurred to me that these textures were on the whole quite noisy, given their sources. I added a bandpass filter to the playback Synth. Now each grain had its own controllable filter. I re-ran the patterns, incorporating this new filter control. As I was actively generating materials, I was constantly recording the output to capture what was going on. I began with a wide bandwidth, letting most sound through, and after several minutes of one width, I modified the pattern to be slightly more narrow. I continued this process, narrowing the bandwidth until almost pure sine waves were left. The snaps of the celery were producing incredibly interesting results at this point, exciting the filter into action and leaving a pure tone. It should be noted that by this point I was creating multiple grains simultaneously, each with a harmonically related filter frequency—thereby creating chords. These chords changed randomly according to a set of possibilities I had predetermined. The chords were chosen so as not to fully resolve onto each other, or at least as best I could, in order to avoid a tonal centre. Once I was left with the sinusoidal chords being articulated by the source recordings, I began to open the filters again and return to the noisier material. It took a long time to figure out the chords, and the bandwidths to move to, so it must be made clear that there were several recordings of me trying to achieve this flow from noise-pitch-noise.

This ABA structure is almost exactly how the piece ended up. I opened my DAW[8], and began to assemble the various parts I had created; layering the clouds of sound together. The piece ended up primarily as a collage of these elements, structured around the noise-pitch-noise element. If one was to look at the timeline of the project, *Twist and Turn* is quite simple. No more than 10 tracks.[9] I was initially

[8]Digital Audio Workstation—ProTools at this time.

[9]During my PhD, I transitioned to Cubase as my DAW, and no longer have a ProTools license to open the Twist and Turn session. This somewhat highlights the issues with a subscription-based licensing model for software—I still have the session, but I cannot open it.

worried that the piece was too simple, but whenever I listened to it there was something in its simplicity that was exciting. It did what it needed to do and nothing more. Interesting sound materials, shifting into interesting harmonies and back into a combination of the two.

While structuring the piece, and indeed generating the materials themselves, it must be noted how much listening and reflecting was a part of the process. Not just on a sound object level, but on a macro, structural level. I would stop, take a moment, and listen to the whole piece (in whatever state it was). This would allow me to situate the phrase or section I was working on within the context of the piece as a listening experience. Whilst listening, I would take notes of times and what I noted about the piece. These could be as vague as "something needs to change/happen here", or as detailed as "filter material X at time Y to make room for material Z". A key feature in this activity was the temporary nature of these notes. They were not taken electronically but rather pen and paper, not in a notebook but on a notepad or refill pad; often simply scraps of paper nearby, whatever was available. The role these notes played in the act of composition was not as a permanent record, but as suggestions for what might be done. On reflection, it is clear that I was not even following my own advice. Often, I would contradict my notes, or simply ignore them. The reasons for this included anything from changing my mind, to solving the issue another way, or simply getting used to what I had previously considered needed changing. There were also several occasions where I did not execute what I had noted because on reflection, it was *good enough*. This mostly occurred closer to the deadline for finishing a piece when it became important to prioritise certain aspects of the composition over others in order to finish on time. If there was time at the end, I could go back and fix whatever was still problematic.

It is also important to note why I finished *Twist and Turn*. It took about three months to write, the longest I had ever spent on a piece. Importantly, I was not solely composing. Most of this time was not spent working on the piece. I was stuck on the structuring of the elements for a long time, so postponed working on it; only as I had the luxury of time at this stage, given the lack of a fixed deadline. However, Sound Junction (SJ) was approaching. SJ is Sheffield's electroacoustic concert series, a weekend of music from the USSS[10], and invited guests. I had already not performed at the first of these since starting my PhD and I was determined to have the piece finished in time for a world première at SJ, so I did. It is important to note, during this final stint of composition, I was asking other composers to listen to the work-in-progress. Alejandro Albornoz was also working on a piece at the same time, in a different studio, so he kindly listened to an early version of the piece and provided valuable feedback.

Twist and Turn was premièred at SJ, 15th May 2016. Due to its tonal nature, it was a very different

[10]University of Sheffield Sound Studios

kind of piece than what I had previously produced. Composers familiar with my previous work noted this change in style and commented that it was a good direction, whatever that might mean. I recall my colleague Dimitrios Savva telling me what he would have done differently, which I found very interesting; it must be noted that Dimitrios and I often had these kinds of discussions, so this was not unusual, although it may appear so given the context. Ultimately, as the composer with a finished piece, advice such as this initially seems bizarre, but will always be taken away and reflected upon. After the concert, several attendees commented on the piece, mostly positively with some focusing on aspects of the composition and others on specific aspects of the performance. I am a keen performer of electroacoustic music, with many years of experience in sound diffusion. A passing thought I have often had is whether my skills as a performer are able to 'cover up some crimes' in the composition itself. It is fair to say in my case that this is not true, as I am constantly thinking about performance while writing my music. If some apparently troublesome aspect present in another listening environment (i.e. the studio) seems resolved in performance, it is precisely because that is the way it was composed—one tends to write for the expected listening environment, as Moore writes, 'Although the composer is in control, unless you want to play your music only to yourself, you should consider your audience. It is important to listen to your music as others might hear it otherwise you tend towards self-affirmation …' (A. Moore, 2016, p. 4). Moore is talking about compositional aspects in this quote, but given that your work will most likely only be heard by an audience in a performance we must always be considering what our performance will be. In the case of acousmatic music, the established performance is that of multi-channel sound diffusion.

Shortly before the première, there was call for submissions to the Music Nova competition. I remember feeling conflicted about submitting a piece that had yet to be publicly performed to a competition, but I decided to enter regardless. At this stage in my career as a composer, the aim of entering the competition was not for the prize itself, but rather for establishing myself as a composer. In all honesty, I had forgotten that I had entered when an email arrived informing me that *Twist and Turn* had been awarded first prize[11]. As a result, *Twist and Turn* now has a history of national and international performances, which is also useful to reflect upon.

2.3.1 Twist and Turn: Reflection

The composition of *Twist and Turn* has highlighted several aspects of my practice, namely the use of algorithms, layering of textural materials and considerations of performance practice. I am using Emmerson's

[11] Musica Nova, 2016. Category A: Autonomous Electroacoustic Music

Language Grid (Emmerson, 1986) [12], as it affords a greater reflective analysis of my work.

When looking at *Twist and Turn*, the syntax is dominantly abstracted; it focuses on aural relationships such as density of textures, pitch and gestural shapes—the materials are chosen and positioned based on aural relationships. As mentioned in 2.3 certain elements are algorithmically determined but their position and relation to other materials was decided through abstracted means—in practice, moving sounds in relation to each other with respect to time, using a DAW to both adjust and assess what I felt to be correct. In terms of musical discourse, the piece is a combination of aural and mimetic, with neither being particularly dominant; the pitched sections are roughly equal, in length and prominence, to the more textural, non-pitched sections.

The introduction to *Twist and Turn* presents the majority of the materials used throughout the piece. As more of the piece was written this section was revised to provide additional signposting for what was to come. It begins with a fluctuating drone, struggling to stabilise itself. On the third attempt it explodes into the expansive granular textures that form the core of the piece; patterns of three are used throughout the piece to both create and subvert anticipation and expectation. In order to avoid becoming predictable themselves, the sets of three repeating gestures that are used to create anticipation take two main forms, based on the same underlying low frequency material: an attack gesture, and a rise-fall drone gesture. The first example can be seen in Figure 2.4 at markers 1–3, as well as 5–7. The same signifying sounds are used in isolation at markers 10 and 11. Throughout the piece, these sounds are used to facilitate transitions rather than serving to mark them out. Markers 5 and 12 provide examples of the drone gesture, with Markers 1–4, 6, 7, 10 and 11 exemplifying the more abrupt attack gestures. In combination with the textures, I created rhythmical accelerations and decelerations using short grain sizes and dynamic grain rates (the number of grains per second). These are entirely derived from the textural materials, providing a sense of sonic coherency within the piece. The section then tries to establish itself three times, lasting slightly longer each time, as well as introducing new materials, before being swept away by an attack gesture with Section A slowly forming as a result (see Figure 2.4, markers 1–4). These textures then ebb and flow in accordance with their underlying algorithm, attempting to become more pitched; these transitions to pitch being made interactively in real-time, as described previously, rather

[12] Emmerson presents two continua onto which we can position a piece: syntax and discourse. The syntax of a piece exists between the extremes of abstract and abstracted, and the discourse lies between aural and mimetic. Abstract syntax describes where the relationships between sounds and the structuring of materials is primarily chosen through some systematic process rather than purely how the sounds relate to one another sonically. Abstracted syntax is therefore the opposite, where sounds are chosen for their aural relationships to one another and the structure is determined by similar means.

In terms of musical discourse, aural discourse describes the structuring of materials in ways that ascribe to the traditional concepts such as pitch, harmony or rhythm. In contrast, therefore, mimetic discourse is that which utilises the listener's ability to associate sounds with their own real-world experience of sound - imitations of real-world sounds, or even 'aspects of human culture'.

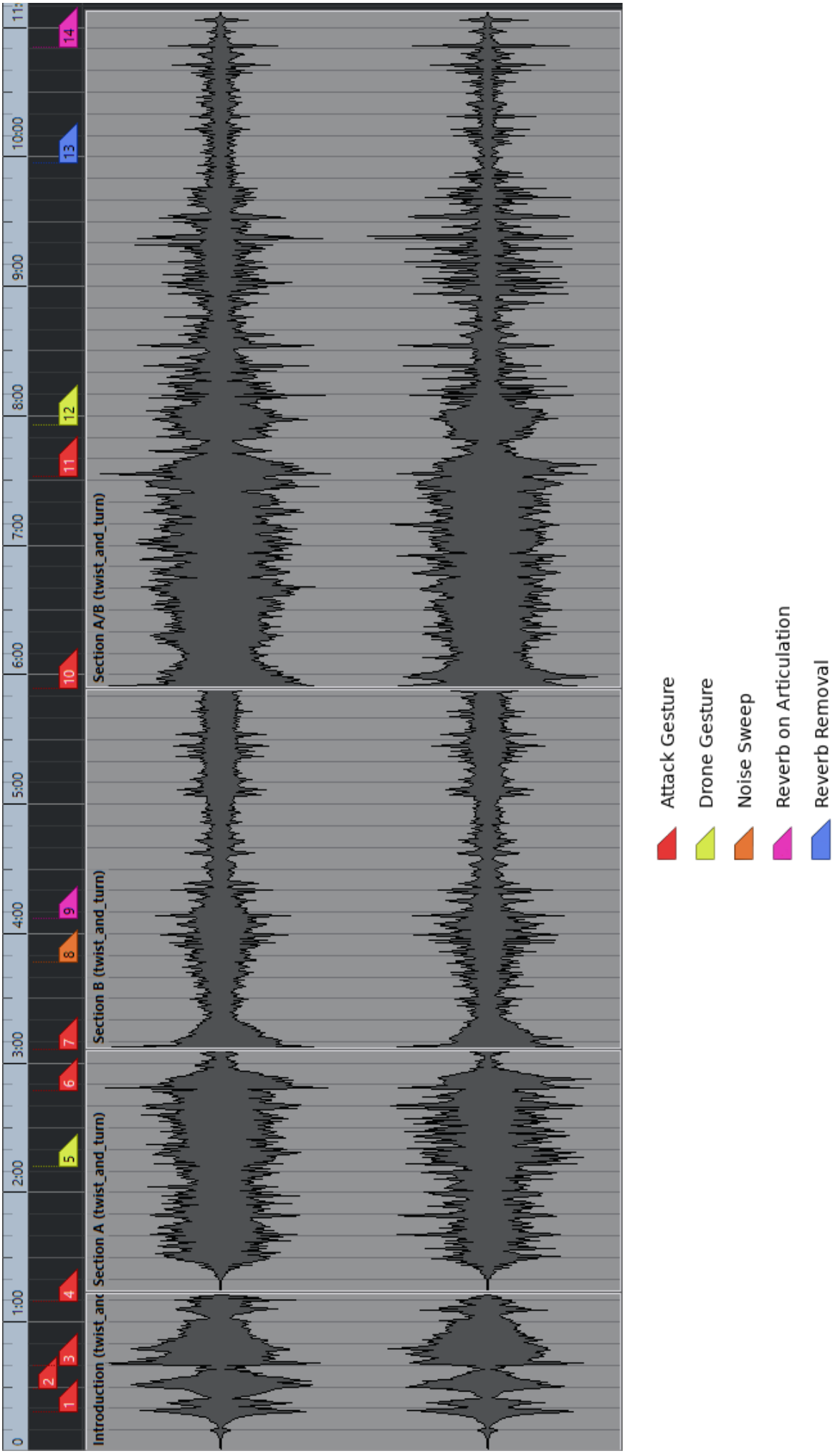


Figure 2.4: Twist and Turn: Waveform

than algorithmically. The reason the transitions to pitch are successful is that the the filters that create the pitch are always present in materials that transition into pitch—there is no cross-fade into pitch, the bandwidths of the filters are just slowly tightened until pitch emerges naturally. As a result, the movement of the filters is hard to detect because they have always been there—this provides a particularly unique sense of transition that the piece relies upon.

Section A begins with an isolated granulation of one source set (in this case, plastic straws). Other source sets emerge, in particular that of the oranges, and although is it not necessarily immediately obvious, the accelerating iterative materials heard here are, in fact, the algorithmic orange materials being put through additional granulation, as with similar materials in the introduction. The straw material is swept away by a drone, revealing the orange materials. An attack gesture was added to coincide with the pitches becoming more dominant, introducing Section B. It is not important that the listener identifies the sources here, as the piece is exploring developing textures through granulation, specific sources are only mentioned here as a way to highlight their repeated use in different contexts, which additionally functions as the preparation of those materials for later use.

Section B primarily utilises pitched materials, but again contains iterative materials to provide contrast. Some layers are also sent to a reverb for additional contrast. I added a filter swept noise signal whose timing was purposefully aligned with an articulation in the underlying material, serving to highlight this moment. Section B then gestates for a time, with the rapid activity of the materials providing performance opportunities in diffusion; for example, dynamic spatial movement. An attack gesture eventually signals and transitions into the final section of the piece.

Section A/B begins with the more aggressive materials, utilised in the introduction. As with the previous section, this establishes itself for enough time to allow for some more dramatic performance practices, with the material itself suggesting dynamic spatial movement. Another attack gesture is used to sweep away the majority of the textures, transitioning into a more calm, contemplative refrain of the textural materials from earlier before becoming more pitched. From marker 12, most materials are sent to the reverb and back again, with no reverb by marker 13. Reverberation is used twice more, once to highlight a particular articulation and again on the final articulation to conclude the work. This transition to reverb and back again facilitates a homecoming of sorts, signalling the ending of the piece, but it was primarily to allow the piece to be diffused more widely in performance and then returning for a close, more intimate finale. As shown, reverberation in this piece functions not only as a suggestion of distance but also as a naturally occurring colouration device.

Variation is a key feature of the piece. The constant fluctuation of granular textural material seem-

ingly becomes uniform and serves to be predictable, forming a background rather than foreground. The variations, or perhaps more accurately ornaments, mostly took the form of the additional iterative materials that accelerate or decelerate, often deliberately placed to highlight certain changes in underlying material. My approach here was to use these additional materials to articulate the underlying material, not distracting from it but rather to provide an additional layer to accompany the ongoing material—providing a shifting contrast between foreground and background, renewing the textural materials as foreground. In addition, as the piece progresses I added randomised vibrato (by varying the centre frequency of the filters) to create interesting variations in the later pitched materials.

In conclusion, this piece was largely successful and well received. The most important aspects have been highlighted here and have demonstrated both the use and influence of software in my composition.

2.4 The Fluid Framework

Documenting and reflecting on all of these experiences has led me to the following understandings of my own composition. It is important to start this section by highlighting that what follows is not a prescription for composition, but a reflective description. The aim here is the understanding of my composition focusing on the influence of software in order to begin questioning its impact. The framework that follows is somewhat compromised by the linear nature of text, as each aspect in some way informs the others. I have termed these *elements* of my composition.

2.4.1 Overview

Firstly, I am a composer—a composer of acousmatic music. I identify myself as such, and in so doing I align my activities within an established framework. As with nodalism, this framework is formed from what I have encountered; in this case, the UK acousmatic tradition. Furthermore, as a composer, I compose pieces of music. Although these pieces have starting points which may vary from piece to piece, what is more fundamental is that I decide to start them, something only I as the composer can do. Our question then becomes how these decisions become manifest. What are the factors that coalesce to require and drive a piece to be begun? In the case of *Twist and Turn*, I knew that I had to write a piece for my PhD, so I recorded some sounds. However, I started composing when I became interested in the sounds I was working with. In addition, I had to write a piece for an upcoming event, so it is clear that there are multiple forces at work. Upon deciding that I am writing a piece, it is clear that I am setting up a

framework of rules and restrictions within which to be creative (*cFramework*[13]): choosing to work with a particular set of sounds, a particular process or algorithm, etc. But is it also evident that there are rules and restrictions I am not actively choosing, that are imposed upon me from some of my other choices; as mentioned, even identifying myself as an acousmatic composer has this effect. I then proceed to operate within this *cFramework*, following Moore's process. Some elements of composition contribute more to the *cFramework* than they are influenced by it (such as *Reasons*), but each element contributes to and is impacted by the *cFramework* for a given piece. A diagram summarising the elemental interactions is provided at the end of the chapter, in Figure 2.5.

2.4.2 Reasons

The first element of my compositional work is reasons. Reasons are the general *why*. Why I compose, why I compose electroacoustic music, why I compose acousmatic music. What is important is that we recognise that they exist, and that it is where I identify with a particular practice or set of practices. In terms of the *cFramework*, *reasons* can be seen as being responsible for my aesthetic decision making. They naturally apply to all of my compositions, and I would propose that they are likely only to be affected by exposure to other philosophies or activities, as outlined in nodalism. It should be evident that these reasons persist within the other elements. I am never too far away from why I am doing something, whether I realise it or not. I can always explain what I'm doing and why it's necessary. They are the reasons behind starting *any* piece.

2.4.3 Motivations

The next element to consider is motivations. In contrast to motivation in psychological terms, which can often include the aspect of *why*, I more specifically consider these the driving force(s) behind starting a *specific* piece. As such, they are unique to each piece. The motivations for a piece can be influenced by the motivations of a previous piece, or any of the other elements. If the motivations change during a composition, it is most likely that a new, or different, piece is being written. By reflecting on my practice, I have identified the following motivations: personal, commission, competition, and exhibition. With the exception of personal, the motivations will all contribute aspects of aesthetics, duration, and format (number of channels etc.) to the *cFramework*, however they are not actively chosen but rather passively applied.

[13] *cFramework* is the framework of rules and restrictions that are constructed—hereafter, this term will be used to distinguish between the Fluid Framework

Personal motivations are those of an *ars gratia artis* nature: as a composer, I must write a piece. This also includes the kind of spontaneous composition that arises from flashes of inspiration. As mentioned, *Twist and Turn* utilises this motivation. Personal motivations are unique in that the idea of completion is not necessarily embedded. This gives them the ability to change. As noted with *Twist and Turn*, the motivation for starting was personal, but its completion was driven by another (exhibition). As one might expect, this motivation potentially imposes the fewest rules and restrictions; although the *cFramework* from Reasons will apply here, and will be further refined by the Intentions element.

Commission motivation is where one is asked to write a piece. This will most likely be for a specific event, and so a number of compositional aspects will be affected. The event might have a particular function, so aesthetic decisions may be impacted. In addition, things like the duration and format (number of channels etc.), may be dictated by the commission. It may be possible to view the personal motivation as a 'self-commission', but a true commission will provide a certain *cFramework*, whereas the personal motivation relies on the Intentions element to provide the *cFramework*.

As noted with Sound Junction, there are certain regular events that one might compose for. These constitute the *exhibition* motivation. They might include conferences, gallery exhibitions, installations, and regular concert series. The distinction here is that we are not necessarily being *asked* to compose a piece for it. The expectation of the event is to play an existing piece, but there are certainly occasions when I have used such events as motivation to write a new piece, specifically writing a piece for the event. As with a Commission, *cFramework* aspects include aesthetics, duration, and format.

Although *Twist and Turn* was submitted to a *competition* after completion, there have been occasions in the past where I have contemplated writing a piece specifically for a competition. Having spoken to other composers, several have admitted to doing so. Some competitions even require new compositions, acting as both commission and competition. Again, *cFramework* impact is that of aesthetics, duration, and format.

2.4.4 Intentions

When I use the term intentions (and later, receptions), it is not necessarily as deemed by Rob Weale as part of 'The Intention/Reception Project' (Weale, 2005). Weale was referring to the ideas that composers were trying to convey to listeners, how effective they were at doing so, and how such communications could be improved. Here, I am simply referring to compositional intentions—what I as a composer want to do for a given piece. Intentions for me are the conscious, actively chosen rules and restrictions that are specific

to that piece. It may be the desire to work with a particular sound, or set of sounds, a particular process or technique, or even a particular structure. It is additionally the element where aspects of the *cFramework* can be rejected, with the exception of absolute limitations; for example, the maximum duration for a competition submission cannot be altered, but aesthetic restrictions like the use of pitch/harmony/rhythm can be challenged. Intentions may very well be the start of a piece, but in this sense they will mostly lead to motivations—"I want to write a piece exploring this sound" may be the intention that leads to the personal motivation of, "I am writing a piece exploring this sound". Intentions are influenced by the Reasons and Motivations, but where Intentions lead to a change in Motivations, there is the creation of a different piece. Intentions are also influenced by the Receptions element, but only insofar as it applies to a future composition, as the current piece must be finished for Receptions to occur.

2.4.5 Activities

The other elements contribute to and develop the *cFramework* for a given piece, but there comes a point where one acts. Activities are these compositional actions. Once I have decided to work with a particular sound in a particular way, I have to actually do that work. Activities primarily utilise the *cFramework*, but the outcomes from it can result in changes to Motivations or Intentions. For example, processing a sound in a particular way may result in a new intention being developed, shifting the focus of the piece. However, where the intentions for a piece shift so much that these new developments become the focus, it is entirely possible that the motivations for the piece change, thereby beginning a new piece.

Activities can be thought of as consisting of two components: approaches and methods. These cannot be separated in practice, but it can be useful to describe them separately. The term approaches here refers to the theoretical aspects of action. The choice of a particular process with a particular sound, or the choice of which sounds to combine to form a composite. Methods are the doing of the action; for example, having decided to apply a filter to a sound, one must then *apply* a filter to the sound. This requires knowledge of what is available, both in terms of what processes might produce interesting results, and what software actually implements those processes. I view approaches as "what I bring to the table", and methods as "what I do while I'm there". Together, these form the Activities element, which follows (at least for me) Moore's *work-reflect process*; as evidenced by *Twist and Turn*.

Recording or synthesis of materials is part of the Generate stage, and so fits here. *Twist and Turn* began with a recording session, but without motivation or intention to write a piece. It may be argued that *Twist and Turn* began here, but the recordings I made were not for a specific piece. I was, however, inspired by

some sounds I was working with, which then resulted in a personal motivation and the development of some intentions for a specific piece. It is therefore possible to start a piece with Activities. It may even be the case that all pieces arising from personal motivation start here. As we have seen, Activities can influence the Intentions for the piece, or a future piece; are clearly influenced by the Intentions for the piece; and influence the motivations of a future piece. It is hopefully self evident that this is the element where technology, including but not limited to software, is utilised. With this, we see the position of software and its area of influence. We can therefore turn our attention to what these influences are.

2.4.6 Receptions

As shown, once a piece has been completed it must be performed. Performance is an embedded need in electroacoustic music, as without it the piece remains an intangible stream of binary digits. With *Twist and Turn*, and indeed all of my pieces, this need manifests itself in the composition. However, the receptions element encompasses what happens after completion: both performance, and the results thereof. It is here that my role as composer (for this piece, and this piece alone) halts, and I switch to being performer. This composer/performer duality is somewhat unique to electroacoustic music. Whilst I would argue that only future pieces are hereby affected, it may be the case that once a piece is performed a composer chooses to revise it. However, I have never done so. My philosophy has been, and most likely will remain, that a piece is a timestamp of the position in my compositional development; so once a piece is performed, the only lessons to take are for future works. As I perform this music, I learn how sound moves in space, which may inform certain structural or other compositional elements in a later piece. I also receive feedback from specialists and non-specialists that may inform future decisions: the next motivation, set of intentions, particular approaches or methods, etc. Receptions essentially constitute the performance, and Reflection (as *work-reflect*) for the finished piece, which goes on to inform the next piece. Receptions therefore play a vital role in composition, even though the role of composer is not directly a part.

2.5 Conclusion

Composition cannot be defined, but it can be appreciated. It can be understood without resorting to mysticism. Through the composition of *Twist and Turn*, the Fluid Framework was identified, which does not seek to prescribe composition but to provide a detailed enough description to conduct further

Element	Components	*cFramework*
Reasons	—	Aesthetics
Motivations	Personal, Commission, Competition, Exhibition	Aesthetics, Duration, Format
Intentions	—	As from Reasons and Motivations, addition or subtraction of rules specific to piece
Activities	Approaches, Methods	*indirect via influence of Intentions and Motivations*
Receptions	Performance, Response (specialist, non-specialist)	*future work*

Table 2.1: Fluid Framework: Summary of Elements

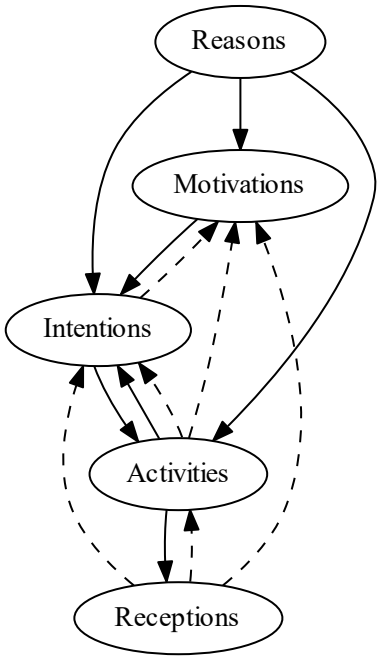


Figure 2.5: Fluid Framework: Elemental Influences

(dotted lines indicate influence on a future work)

investigation. My composition comprises five elements, each containing their own knowledge-sets, and relying upon and informing/interacting with the others.

The use of software has been identified as a key factor in most aspects of composition and is situated within the Activities element. The Activities element consists of Approaches and Methods, and software plays a part in each of these components. Knowledge of what software is available, how it works, what it might do to sound, will inform the Approach to working with material. In the case of *Twist and Turn*, I knew what SuperCollider's PS was capable of, what filters would do to the sound, etc. Following on from this, we see that Methods are not just informed by software but are in fact dictated by the software: how a piece of software works *is* how the composer will work. It cannot be anything other than this. The Methods are facilitated by the software. Moore's text is particularly interesting in that it discusses the use and rationale behind the software, whilst providing the software itself. Wishart's text also provides guidance on the use of software, referencing the Composer's Desktop Project (CDP), but as mentioned, it appears to focus on the creative output of the software. The CDP can perform the manipulations Wishart discusses, but because other software could do the same, the specifics of using the CDP itself are not covered. It is interesting to note that Wishart outlines the rationale behind the processing itself, so one could use the text to develop software based around these concepts.

CHAPTER 3

Software in Composition

In the previous chapter the use of software was located within the context of composition. However, its role remains unknown. As we will see in this chapter, the studies that examine the role of software do so from a perspective of identifying the task of composition itself, with the goal of creating software better suited to the task. In order to do this, however, they must identify what *completion* means for the task. Software is therefore viewed as the sequence of actions needed to compose. We can relate this approach to Suchman's concept of *plans* as opposed to *situated action* (see 2.2.3). Existing work posits that composition is a rule-governed activity that can be systematically encoded in software.

In contrast to existing work, this research has identified software as playing a much more integral role, not just replicating human activity, but actually facilitating it. The composition of acousmatic music, as we understand it today, would not be possible without the use of software. Rather than posing a recursive ontological dependency, this simply shows that the software is not separate from composition, but rather a part of it—software serves to enable the act of exploration required by the activity itself, as in situated action.

In viewing software as the embodiment of the process of composition how the software functions as part of that task is not considered. This chapter will seek to address this by taking forward the idea that software plays a role in composition, discussing existing work in the field and presenting a number of investigations into software in my compositional work. It begins with investigations into the two software paradigms of real-time (RT) and non-real-time (NRT), assessing them as part of composition. The design

of the software created will be discussed, as it is important to be able to identify any limitations in the assessment of the conclusions drawn from its use. The compositions created alongside the software will also be discussed in order to contextualise both the software and the conclusions.

3.1 Existing work

3.1.1 MOSART Project

Returning once more to the work of the MOSART project, they confirmed that composers used a variety of software. Their initial assumption was that this proved problematic, as switching software disrupted the work being done. However, it became clear that because composers use a combination of divergent and convergent thought processes, this switching ultimately proved beneficial as it encouraged divergent thought (Nuhn et al., 2002; Upton, 2004). As a result, they concluded that the creation of a single software environment for composition should not be the goal of future work.

Tracy began by reviewing the work of Nuhn et al. As mentioned previously, this study attempted to utilise naturalistic observation. Unfortunately, the environment in the study was not natural, with the composer not using their own computer and usual software. In addition, the composer knew they were being observed, which calls into question whether their behaviour was natural at all. Taking this into account, Tracy's conclusions are certainly marred by these concerns. However, Tracy concluded, in relation to software, that (Tracy, 2002):

- Improvising with the software is part of the compositional process
- Software limitations can be motivating as well as frustrating
- Knowledge of a program's capabilities may be at the root of some frustrations
- The[re] may be an unwillingness to use manuals amongst composers

3.1.2 The importance of memory

The MOSART project also identified the importance of memory in composition. This was on a rather large scale, reducing to the idea that a composer must have a working recollection of the sounds at their disposal. Eaglestone's idea was to incorporate a searchable database to reduce the requirement of memory and possibly assist, or pre-empt, composers in their choices (Eaglestone et al., 2007).

However, significantly earlier, when discussing the design of interfaces for computer music, Otto Laske identified several important roles of memory in composition. Laske proposed a memory model used in composition, suggesting six key factors that interact with *productions* (to be understood as a heuristic), that combine to form a *production system*. Events occurring at various levels of time (audio, conscious, interpretive), are perceived at corresponding cognitive levels (sonic, syntactic, semantic). The ways in which a composer's time levels and cognitive levels interact with short-term and long-term memory dictate the kind of interactions that one can have (O. E. Laske, 1978).

3.1.3 Attitudes to software

Separating the attitudes towards software and technology in general is rather difficult, as they tend to be discussed interchangeably. The following presents a summary of the attitudes that can be found in existing literature.

In a letter to the *Computer Music Journal*, Otto Laske suggested three different attitudes towards the use of computers in art: result-oriented, tool-oriented, and process-oriented (O. Laske, 1984). He classed these as analytical, pragmatic and cognitive respectively. Rather than explain these as Laske does, it is perhaps more interesting to consider the somewhat similar work of Nuhn et al. The conclusion drawn with regards to attitudes here is that composers fall into one of three groups:

> …composers for whom creation of computer related tools (i.e. software, hardware interfaces) is a natural accompaniment to composition and is inseparable from the process of composing. …[those who are] more concerned with engaging with the sounds themselves, i.e. composing with the tools available to them, but seem to feel the need to deal with aspects of tool creation, i.e. computer programming etc. …[those who] are similar to the second [group], with regards to their natural preoccupation with sounds rather than with the tools [, however] …they do not worry about the technical sophistication …but will rather try to push the tools they know to new boundaries to create original sound events. (Nuhn et al., 2002)

From this we can draw an interesting parallel between these attitudes and the standard computer users groups: Super-User, Power-User, User (respectively). There are those who write the software, those who know enough to modify the programs to suit their needs, and those who use whatever is available to them.

It is also important to appreciate how these attitudes have shifted over time. Laurie Spiegel proposed a series of assumptions that, although purely speculative, ring somewhat true. In summary, Spiegel posits that as the influence of commercialism grew in computer music, attitudes to software moved away from the composer being both the "tool maker and tool user", to the "tool user and the tool builder [being]

different people who never interact", amongst a variety of other important aspects (Spiegel, 1996). This may in part explain the different attitudes seen by Nuhn et al. (2002).

3.1.4 The role of software

As with creativity, it is useful to consider metaphors used in the description of software to form an understanding rather than a definition. It is, however, once again difficult to separate the computer from the software, particularly when referring to early writing on the subject given the landscape in which software was written. Regardless, what follows is a summary of the metaphors relating to software in composition.

Initially, studies focused on using computers to control external hardware, like a synthesiser (Bayer, 1977; Lawson & Mathews, 1977). However, as computers got faster, we began to see more varied descriptions emerge: the computer being a musical instrument (Bedworth, 1998; Bertelsen et al., 2009; Charrieras & Mouillot, 2016; Mathews, 1963); a tool (Bedworth, 1998; Keane, 1986, p. 116; McNabb, 1986, p. 148; Truax, 1986, p. 160); a calculator, type-writer, or word-processor (Pope, 1992); a creative partner or assistant (Bedworth, 1998; McAlpine et al., 1999; Miranda, 2001). These metaphors can help us identify how computers are being used. By far the most common is that of *tool*, or *instrument*.

Where *instrument* is used, we tend to see a very strong association with traditional musical instruments and an attempt to emulate them in either control or timbre. In contrast, the use of *tool* varies, often having two different uses. Firstly, where software has no particular role and is merely a tool to accomplish a task. This is also usually where other metaphors of calculator or type-writer are seen. Secondly, where software can be used, but where it is not fixed; where computers are viewed as a general purpose problem solver that can be made to work with a variety of problems—whether the software *currently* does something, is not a limiting factor, if one can modify it.

These differences might be seen as the attribution of agency to the computer, but it is perhaps more appropriate to apply the concepts of technological determinism and technological constructivism (Magnusson, 2019, p. xi). Here, Magnusson uses *determinism* to describe the idea that software entirely dictates composition, with *constructivism* relating to the creation of software being dictated by composition, providing a warning that we should not rely too heavily on either one. If one's attitude to software is that of *tool*, then it is fair to attribute this to constructivism. Alternatively, if one views the role of software more highly, it is likely to be rooted more firmly in determinism.

3.1.5 The influence of software

It is somewhat of a truism that software has an influence on composition. This has been written about many times: Belet et al. (1992), O. Laske (1984), Pope (1992), Pope et al. (1995), Truax (1986). However, the research conducted so far has failed to ask and/or answer the question of software itself. We have been warned that software has an influence, but as yet this influence has been overlooked. Of those that have questioned the influence, the prevailing attitude has been that of *tool*, and so the most common approach has been task-based. As a result, the studies have considered software a single entity aimed at replicating a human activity. However, when we consider software in the context of composition, I would argue that it is not simply a tool encoding some aspect(s) of compositional activity. Instead, if we ask the question, "how does the software operate?", we quickly discover that this has not been an aspect that has been previously considered. The questions have so far been "what software do composers use?", and "how might we make improvements to the software currently in use?". For example, Eaglestone et al. (2007) did not question the software paradigm as a factor in the use of software.

We have seen the perceived role of software, and a number of attitudes towards it. The factor we are most concerned with is that of its influence. The attitudes and roles we have seen suggest a certain perceived importance of any influence there might be. If one has no say in the software, its influence cannot be negated, and so any influence it may have can be safely disregarded as a part-of-the-task. If one can mediate the influence, it can be managed. Yet, if one can create one's own software, the influence can be controlled, within limits of the realm—becoming somewhat of a compositional parameter.

3.2 Background

At this point, it is important to consider briefly my own attitudes to software and technology in general; identifying into which of the groups I fit. This is included in the hope that the context it provides will alleviate concerns of bias in the discussions that follow. Rather than simply state which group I believe I fit into, I decided that it was better to consider how I could test this.

In order to classify attitudes towards software, building on those we have seen, I would like to propose the following statements as their appropriate litmus tests:

- "If the software existed, I would use it. If it did not do what I needed, I would use an alternative that provided the functionality I require."

- "If the software existed, I would use it. If it did not do what I needed, I would try to modify it to

suit my specific needs. If this was not possible, I would try to find an alternative."

- "If the software existed, I would use it. If it did not do what I needed, I would try to modify it . If it did not exist, I would make it."

Taking into account technological constructivism and determinism, these can be seen as increasingly deterministic. However, if we combine these with the metaphors we have seen, a different perspective emerges. I personally align with the final statement. It is not that I wish to reinvent the wheel, but in cases where the technology doesn't exist, I will try to create it. In terms of metaphors, however, I do not particularly agree with any of them. As such, considering my view as deterministic based on the statement alone is misleading. For example, the relationship one has with a tool is not that which I have with computers, and I am reluctant to use the metaphor of assistant as it ascribes too much agency to the computer. I would propose an additional metaphor, that of *substrate*. One might consider the plastic of magnetic tape, acting as a substrate for recordings, but I am instead thinking of agar jelly. Agar jelly is used in the field of biology to grow bacterial cultures, providing all of the required nutrients. Like software, it must be made, and then used under certain conditions to produce intended results. However, it is important to note that the substrate does not act alone. It requires intervention by us, and in turn our interventions change the substrate itself. The addition of penicillin, for example, is something that we might apply to agar, but the substrate itself reacts to this process with the combination of the two modifying the result of the experiment. The metaphor of substrate conveys a push-pull relationship, of one influencing the other to cultivate something new—in this case, music.

3.3 Defining RT and NRT

The way that we define real-time and non-real-time becomes of primary importance in our investigation. The difference is not about speed, as one might expect. RT software is not necessarily faster than NRT; in fact, quite the opposite. NRT software was initially much slower than RT, but as computers gained speed there was a point at which it could run much faster than RT. To generate a sound of 1 minute would take 1 minute in RT, whereas it can now take a few seconds in NRT; although previously it would have taken much longer than in NRT with slower machines. When we talk about RT, we must be careful to specify that which we are discussing. Following the *Computer Music Journal* through its history, we see a fascination with real-time, but with a definition that changes over time. In the beginning, excluding Bayer (1977) and Lawson and Mathews (1977) which refer to controlling a synthesiser, we see a definition of

RT as it pertains to the generation of sound. However, as time continues and computers improve, as generation becomes a standard feature of personal computers with the likes of the PDP-11 (Truax, 1977), the focus shifts rather unmarkedly to referring to the ability to control parameters of the generation; to being about real-time control of software. As a result, from this point RT will be used to describe in a general sense, but where required, *RT-G* and *RT-C* will refer to RT-generation and RT-control respectively.

In order to determine whether or not a piece of software is RT or NRT, it became necessary as part of this research to develop a simple test that could determine the paradigm. As noted, the main difference is the interaction with the software and when it occurs in the process. With RT software, one runs the program and can modify the result as it is produced. With NRT software, everything is configured in advance, and we later observe the result. As such, the test for being RT and NRT relies on the notion of what is set up in advance and where observation occurs in the process. This is the basis for the *domino test*. As with a set of dominoes, one might carefully set up the Rube Goldberg-esque domino show, and watch as everything falls into place. Changing the course of the toppling dominoes once set in motion is impossible and we must rebuild the entire display. Any software matching this description is inherently NRT.

In contrast, if we were able to change the course one might be able to consider the software to be RT. There is one exception to this: in cases where changing the course causes an audible glitch, the software is NRT. The following examples are provided to give clarification to this caveat:

- CDP. Processes and parameter values are chosen in advance and passed to the relevant executable. CDP is therefore inherently NRT.

- Csound. Instruments and scores are designed in advance and passed to the csound executable. Csound is therefore inherently NRT. While it does provide RT-G, in my experience RT-C has been unreliable—as such I would be reluctant to consider Csound as RT.[1]

- SuperCollider. Synths are designed in advance. Once a Synth is requested, parameters can be changed in RT, and the Synth itself can be replaced at any time without glitching the audio. SuperCollider is inherently RT.

- A digital audio workstation (DAW) (e.g. Cubase). Processing is chosen in advance, parameters

[1]It is hard to follow how Csound operates. The code is quite old and hard to understand, and some may even be legacy code that the program no longer uses. The RT-G appears to be using a blocking call to the audio API, but there are references to threads. It is difficult to see how these threads are being used. The RT-C appears to be handled through pipes with a FIFO (first in, first out) mechanism, but it is not clear how the changes are communicated—if the FIFO read is called in the audio thread, this can block the audio and cause a glitch.

can be changed in RT, but one cannot add a process (i.e. plugin) without glitching the audio stream. DAWs are inherently NRT.

- PureData. Patches are designed in advance. Patches can be changed in RT without glitching the audio stream. PureData is inherently RT.

Using the terms RT and NRT themselves, it is not necessarily clear in which paradigm the software works. NRT, while it was initially slower than RT, can now run significantly faster in some contexts than RT. In others, for instance with intense processes such as complex FFT (fast Fourier transform), NRT can take longer than RT. The same can be said, however, for RT processes—in intense processes it can also be too slow to generate audio. RT has time bounded constraints. If the audio is not calculated in time, the output will glitch. Having said this, if the RT audio process writes to disk correctly, by buffering on a separate thread, these glitches can be limited to the RT-G and not the recorded output (i.e. the audio) of the process. By this point, however, any benefits of RT are lost as the feedback loop is disrupted by the glitching audio.

In some ways, it is possible to view RT software as no different to NRT software—with RT being NRT that is guaranteed to run within real-time-safe boundaries. The buffer size configured in programs sets the amount of frames that the software has in which to calculate in RT. Increasing the size of the buffer to give the computer more time before running out may help in some circumstances. However, it will also increase the amount of delay between a parameter being changed and its result made audible. This in turn results in the software being perceived as not entirely RT.

These definitions of RT and NRT only take into consideration the computational requirements, and cannot account for the way in which we interact with these processes. Therefore, it is important to understand how the control of software in these two paradigms functions in relation to composition. This understanding is the focus of the following investigations.

We must first consider what it means to be RT. RT-G alone is not enough, as the usage of the software is that of NRT. RT-C is then a required addition, but what form this control takes is also important. If the software allows something to be done in RT, it should not glitch the audio in either the RT output or the soundfile output. If adding a plugin to a DAW causes a glitch, then that aspect is not RT. Taking into account the context in which the software is used is important here. A DAW is not used in a RT critical context, so a glitch when adding a plugin is not a requirement, as the primary context is NRT use—positioning a sound or control and then listening to the result. Therefore, even though it provides RT-C and RT-G, its usage suggests NRT, in a similar way to Csound. As the software is being used, if

it allows the user to do something that glitches the RT-G, then it cannot be considered a RT program. If however, the program provided RT-C and RT-G, but for instance did not allow the user to do something during the process of RT-G that caused a glitch, it would be classed as RT. The key here is the RT-G, the audio stream, and whether the software allows us to do something that isn't RT-safe.

In summary, NRT software offers neither RT-G nor RT-C. RT software allows RT-G and RT-C, but can only be considered RT if it performs within RT-safe limitations. As presented so far, it seems that there is some aspect of working with RT and NRT systems which we have yet to explain. Later in this research, an addition to the classification of paradigms was discovered that removes the need for the caveats: see discussion of Data.

3.4 Developing software and composition

During this research, several pieces of software were created alongside compositions in order to evaluate their impacts. It is important to outline the goals of the software and the investigations. Firstly, software is created from composition. As Wishart outlines, software designed from a technical standpoint will produce technology, 'an arbitrary, number-crunching program will produce an arbitrary result' (Wishart, 1994, p. 5). Building on this, we might consider that software designed from a musical standpoint could produce sounds that would be of greater musical interest to the composer. Wishart's writing in particular goes to great lengths to explain the sonic reasoning behind the transformations that the software might be capable of rather than focusing on the implementation. Secondly, I am not aiming to create the best software, but to create software that may both ask and answer questions about composition. The initial question is one of requirement, but not from a HCI standpoint. Rather, we are concerned with identifying what it is about NRT and RT software that we need to question further. To answer this, I will follow the line of questioning from Chapter 2—how does the software interact with Approach and Method. In so doing, we approach the question of impact not from a standpoint of how well the software embodies the composition process, but from a position that considers software as a part of the composition.

If we imagine for a moment two pieces of software: one operating in RT, and the other in NRT, where the underlying sound processing algorithm is the same, and the parameters are set to the same values, what then could we say about their outputs? They would be equal, down to the very sample. This being the case, what role then does the operational paradigm play in the use of software, if the difference is not in their sonic output?

In developing software for composition, I have taken an *agile* approach. With an agile approach,

software is developed in a modular fashion through periods of rapid iteration. The modularity we speak of can be determined by what are known as *user stories*. A user story tends to be based on statements similar to the following: “As a user, I have to be able to…”, and “As a user, I would like to be able to…”. Taking these stories, I can then implement them so that everything that is required is possible. Desired features may or may not be implemented, depending on workload and applicability. Importantly, user stories are not used to build a set of requirements, but a set of features; each story can be responded to individually. This is in contrast to *waterfall* approaches, where a list of requirements is obtained, and the software is written. If a requirement is not accounted for the entire project may need to be revisited.

The methodology used in the following investigations is outlined below. Taking the *Fluid Framework* as a starting point:

- Develop software in order to question its use in composition
- Use software to compose short phrases
- Document thoughts on Approach and Method
- Identify potential influences of software

This was conducted with both RT and NRT software. The software used was written in C++, allowing the control of experimental variables outside of the operational paradigms—providing the guarantee of the same processing whether in RT or NRT.

Much of the existing work looking into software has not taken into account the operational paradigm of the software. Instead, choosing to focus on the user interface, whether it is text-based or graphical. While this information is valuable, it cannot be used to answer the question of how the software interacts with composition, because it focuses on how the user works with the software to complete the task, rather than how the software influences the user’s actions. Throughout the following investigations of both NRT and RT, software will be considered as part of the process, and not an encoding of the process itself.

3.5 Investigating NRT

3.5.1 linkEngine

The first experiment was the development of *linkEngine*: a NRT audio processing engine. Following on from the Fluid Framework’s concept of Activities, the question became how NRT software impacts the

ideas of Approach and Method. To evaluate Approach, I must outline my thoughts about working with sound in relation to the software available. The evaluation of Method requires the use of the software and critical reflection on the interaction with the software.

As a NRT engine, it offers no controls once it has been asked to run. Everything must be specified in advance. The design of the *linkEngine* came directly from the Fluid Framework: take a sound, decide what to do with it, and do it. The user specifies a chain of processes on an input file to produce an output file, specifying each link in the chain. As the goal of *linkEngine* was to question the use of NRT software, it was not really necessary to have a large selection of processes, but it contained enough to investigate with: gain, delay, high-pass filter, low-pass filter, channel split, and normalisation. Any number of processes could be specified, and there could be multiple chains within one request. These chains were described in XML, and these XML files were passed to the program via the command line to be executed. A minimal example is shown in Listing 1.

```
<LINK>
  <CHAIN inFile="in.wav" outFile="out.wav">
    <PROCESS name="Gain">
      <PARAM name="gain">0.5</PARAM>
    </PROCESS>
  </CHAIN>
</LINK>
```

Listing 1: Simple *linkEngine* XML example

linkEngine could transform one sound into another, but it could not really produce two outputs from a single input, nor could it take two inputs to produce an output. However, as noted by A. Moore et al. (2013), this is not necessarily required: one-to-many processing in this context can be broken down into separate one-to-one processes.

The core of *linkEngine* is the `Engine`. This is responsible for reading the XML descriptions and building the processing chains. The main function of the program passes the XML file to the `Engine` and requests that it runs. Upon running, the `Engine` will initialise the next chain, then process it, before moving on to the next one.

A chain in *linkEngine* processes a single audio file. It contains a single audio bus on which all the audio is written and read. During the initialisation stage three things happen. First, the input file for the chain is opened. From here, the audio bus is initialised with the same number of channels as the input file. Then the processes are initialised, having their parameters reset, allowing the length of the

processing tail (the maximum number of additional frames to be added by any one process, like a delay) to be calculated. When the chain is asked to process, it will create the output file and run through each process in the chain, in the order specified in the XML.

The processing is handled by `Effects`. These are what I have been referring to as a *process*. An `Effect` is a simple processing class that requires it to accept: an input bus, an output bus, a number of channels to process, and a number of frames to process. The function signature is as below:

```
1 virtual void process(float** in, float** out, const int numChannels,
  ↪  const int numFrames);
```

A feature added much later in the development of *linkEngine* was the error reporting. It was important for me to have human-readable error messages that explain where the program has gone wrong. If the XML file does not exist, it will report exactly that; if a breakpoint is missing a time or a value, it will report which parameter within which process is missing a value, etc. This made tracking down mistakes in the specification much easier.

The overall design of *linkEngine* was to provide a processing engine that had a way of interacting with processes. The engine acts as a host for these processes and guarantees that they all behave in a standardised way. It would be possible to implement any processing as a *linkEngine* `Effect`. Even complex pre-process analysis is possible, because each process is given the details of the file that its audio is coming from. The `Normalise` process is a good example of this, but the initialisation could really contain anything; waveset creation or FFT analysis to name a few. Therefore, *linkEngine* provides a framework in which NRT processing can be written using a simple API[2]. Adding a new process is a matter of deriving from `Effect` and implementing the constructor, `process` and `reinit` methods, as shown in Listing 2. These processes are not dynamically loadable (i.e. plugins), as there are limits to the number of dynamic libraries a single executable can link against, but also because greater integration into the engine is made possible. As the software is open source, it is easy to add or modify processes—and even the engine itself—as may be required.

3.5.2 NRT: On approach

As a result of developing the software myself, I was conscious that the knowledge of its limitations was changing my approach to the software, even without necessarily using it. I knew that it could not

[2] Application Programming Interface

```
#include "effect.h"

class MyProcess : public Effect
{
  MyProcess(const EffectData& data)
    : Effect(data)
  {
    // name, minVal, maxVal, initVal
    addParam("myParam", 0, 1, 0);

    // do we add extra frames?
    addsTail(false);
  }

  void reinit()
  {
    // initialisation code
  }

  void process(float** in, float** out, const int numChannels, const
   ↪  int numFrames)
  {
    // DSP code
  }
};
```

Listing 2: *linkEngine* API example

process things independently, or at least some complex combination of `split` and multiple chains would ensue. Consequently, I ended up utilising more simple sets of processing, even though greater complexity was possible. As shown in Listing 3, the `split` process could be placed anywhere in the chain to produce independent output files per channel, and from there another chain could be used to process those files further. The result of using a direct, in-place processing system was that although it was a literal interpretation of the composition process, it failed to encapsulate the complexities of the kinds of processing utilised—it was impossible to take the output of a specific *process* and have both it and another chain operate on it. Although this might have been made possible with an additional `render` process, one aspect of the *linkEngine* that was promising was that because each chain knows about its own input and output, it would be possible with little effort to have the program scale to high-performance computing environments and process all chains in parallel for large-scale performance improvements. This would require the chains to be completely independent of each other, so even the existing `split` operation would cause problems with this as it creates a dependency between the chains.

So far we have seen how *linkEngine* influenced my Approach to even thinking about what might

be possible. Knowledge of processes, although limited by design, was definitely a factor in producing even short phrases with which to test. However, when it came to *using* the software, the NRT aspect played its part. Not only did I have to know what processes were available to begin with, I also had to have an understanding of what both that process and its parameters would do to a sound. The most common word that recurred throughout my notes was, perhaps as one might expect, *experimentation*. The software produced *nothing* without everything being accounted for. As such, a great deal of time was spent planning and writing the chain specifications. This could quite often be frustrating, as without *sound* being produced, it often felt as though no progress was being made on the composition itself.

```
<LINK>
  <CHAIN inFile="in.wav" outFile="out.wav">
    <PROCESS name="Delay">
      <PARAM type="static" name="delayTime">22050</PARAM>
      <PARAM type="static" name="feedback">0.9</PARAM>
    </PROCESS>

    <PROCESS name="Gain">
      <PARAM type="breakpoint" name="gain">
        <BREAKPOINT time="0" value="1" />
        <BREAKPOINT time="10" value="0.5" />
      </PARAM>
    </PROCESS>

    <PROCESS name="Split" />
  </CHAIN>

  <CHAIN inFile="out-0.wav" outFile="out0-processed.wav">
      <PROCESS name="Delay">
          <PARAM name="delayTime">512</PARAM>
          <PARAM name="feedback">0</PARAM>
      </PROCESS>
  </CHAIN>
</LINK>
```

Listing 3: Complex *linkEngine* XML example

3.5.3 NRT: On method

If we consider what is produced from this process, there is a complete set of instructions that can be used to reproduce the output; provided the input and the chain description are retained. However, it is also important to consider the characteristics of the output files. Whilst the duration of the output can be increased, with something like the `delay` process, it is also important to realise that this is a fixed

time—a known duration that can be, and in fact is, calculated based on the input parameters. So, whilst processes in NRT can alter the duration of the input file, it can by its nature only create output of fixed duration. At which point, the output is heard, evaluated, and the next iteration of the NRT processing begins. As a result, the utilisation of NRT appears to lend itself to iterative developments of material that by its nature produces closely related, but clearly distinguishable sounds. There is a clear delineation for each sound, and if all intermediary sounds are retained, there exists an uninterrupted *evolution* of material from one sound to another.

We have seen that NRT produces finite, concrete sound objects that have an evolutionary relationship to one another. Although we have seen method already in terms of how I work with the software itself, it is also important to now consider the resultant method of working with those sounds as materials themselves, once they have been made. It is perhaps important to note these two forms of method: how to use the software, and what is done with its output. The audio files produced from NRT can be thought of as discrete sound objects. They are somewhat indivisible in that to change its properties, I must either modify the NRT processing itself, producing an entirely new sound object with different properties, or use it as the starting point for further development. Either of these result in the creation of a new sound object which, by virtue of the process, does not modify the original sound file itself. By taking this into account, I am therefore either required to combine them into larger, more complex objects, or use them to generate new material. Additionally, in cases where the processing did not prove useful, it is important to consider that this too informs the next process.

3.5.4 NRT: Conclusion

It is now possible to identify, with respect to NRT software, how my compositional approach and method are thereby affected. By using *linkEngine* to create short phrases, it became clear that two aspects proved greatly important.

Firstly, the assumed knowledge of the software and its capabilities and idiosyncrasies. This impacts not only how I think about the sounds, but also how I consider working with them. If a process is too complicated to achieve, I am less likely to do it, potentially at the expense of musical exploration. A greater degree of planning is required in NRT as a result. The delay between deciding what to do and hearing the results is greatly increased by the amount of time spent in the specification of parameters. I have deemed this an experimental approach: design an experiment, conduct it, analyse the results. This constitutes an effect on *approach*.

Secondly, the use of the software results in a collection of discrete, finite sound objects. Whether these sounds are used or not within a phrase or piece, something is learned from these intermediary sounds. When continuing to develop these materials, I either add them together to form more complex sounds, or choose some aspect on which to focus for the next round of processing. Taking any two sounds, we have a some tangible path of sound objects in between: a clear evolution of materials. This can prove very useful in acousmatic composition, as I am able to work with a collection of sounds that are related to one another; I can worry about how to combine them into phrases later, knowing that there are stages in between on which I can rely. This presents an effect in relation to *method*.

In conclusion, I have observed that the use of NRT enforces an iterative, experimental approach, which in turn results in an additive working method.

3.6 Investigating RT

Following on from the NRT investigation, I explored RT in the same way. I took the *linkEngine* and added RT capabilities. As with *linkEngine*, there were several iterations of program to incorporate ideas as they were encountered.

3.6.1 le2

le2 was the successor to *linkEngine*, as the name implies. As mentioned, it added a RT audio engine. It was also the first of the audio engines from this research to utilise a node-based graph system. The audio engine would wait for a graph—in this case, akin to a chain as before—to process, and then process it in RT. In a later re-write of *le2*, hereafter *le3*, the parameters were modified to allow for RT-C, accepting input from MIDI (OSC was later planned), although no way was added to specify these programmatically. RT-C was tested using the later version, but source for this version has been lost. For these reasons, *le2* is not distributed with this thesis, as it does not fully represent the software used in this investigation. However, the observations as outlined in this section were later additionally confirmed by the use of both *SOClib* and *lwae* (see No-Time), and so its discussion remains important.

With the addition of the node system, processes were now able to exchange audio. In order to accomplish this, the idea of having a single audio bus was no longer feasible. It became necessary for each node in the graph to have its own buffer on which to operate; therefore allowing any node to operate on any other node's audio without the need for an intermediary output file. To accomplish this, it was first necessary for the nodes to know about one another's input and output types. For example, a process that

required a mono input could not connect to a node that produced a multi-channel output—nodes have to be compatible. This was implemented using a bit-flag, as demonstrated in Listing 4.

```
// Based on audiographnode.h
// known integers representing individual states

// 0001
const int mono = 1;
// 0010
const int stereo = 1 << 1;
// 0100
const int multi = 1 << 2;
// 0111
const int any = mono | stereo | multi;

// using the bitwise AND, we can check if certain
// bits are set

int input_required = stereo;
int output_given = mono

int compatible = input_required & output_given;

// if both bits are 1 then 1, else 0
// 0010
// 0001
// ||||
// vvvv
// 0000

if(compatible != 0) {
    // compatible - can connect
}
else {
    // incompatible - cannot connect
}
```

Listing 4: Bit-flag example

As the goal was to make as few changes as possible to *linkEngine*, it used the same XML specification to describe processing. The assignment of RT-C was carried out by whatever graphical user interface (GUI) was implemented, as it would have direct access to the nodes, their parameters, and `Parameter::setMIDIConfig`.

It operated in much the same way as *linkEngine*, reading and building processing chains, processing them in blocks, and outputting the result. The only real difference was the RT-G and RT-C. *le2* in its initial state provided RT-G without RT-C. The difference with the addition of RT-G was that I no longer

had to wait for the process to finish before opening another program to listen back to what had been created. We can therefore conclude that the addition of RT-G on its own really only acts as a time-saver. In later versions that allowed RT-C through GUI my observations were quite different.

3.6.2 RT: On approach

In knowing that I could hear the sound immediately, without necessarily considering the settings of all of the parameters, I noticed a change in my approach to working with the software. I would not hesitate to select a process without knowing the effect it would have on a given sound. In direct contrast to NRT software, it appears that RT has less of an assumed knowledge-base. Coupled with the feedback of parameters on the audio, I was quick to begin to associate controls with their audible change. This, therefore, presents itself as an interactive discovery process between input and process, through parameters. RT-C offers the way in which these correspondences are learned. The terms *play* and *improvisation* were most often found in my personal notes. These are representative of my approach, rather than method, as it reflects how I was thinking about working with the software: knowing it was improvisatory, I would perhaps consider using combinations of a sound or process with which I was not familiar. It is interesting to relate this to the findings of Tracy (2002), who concluded that improvisation was an important part of the use of software. Improvisation is not a term I have associated with NRT, so it is fair to conclude that the software used in Tracy's investigation was actually RT, although this was not accounted for given that operational paradigms weren't considered.

3.6.3 RT: On method

The influence that RT software had on my method was noticeably different than that of NRT. The improvisatory approach to interaction led to a somewhat improvisatory method. In *le2* and its variants, the output lasted as long as the input. This is not particularly common in RT programs, but was still enough to identify, albeit to a lesser extent, a resultant working method with the output of the RT process. With the utilisation of RT, parameters are under the sole control of the user, as and when the control is desired. As a result, it is possible to make audible mistakes: moving a parameter too quickly or not smoothly enough, or not moving a parameter at exactly the desired moment, to provide a few examples. This is the only point at which RT processing can be audibly distinguishable from NRT. In the case of mouse control for single parameters, I found that the inaccuracies in the control method would quite often result in audible differences that could only be attributed to RT-C.

le2 and *le3* produced files of the length of the soundfile (or slightly longer as necessitated by the processing tail). *le3* is unusual in this regard, compared to other programs that offer RT-C. Quite often, a RT program will loop the input. This puts *le3* in a unique position, whereby objects have a more finite state than they otherwise would. In programs that produced indefinite output durations, I found myself rehearsing parameter changes, resetting controls to known-good states before proceeding to re-enact what I had discovered to be interesting, and these would all be captured within a single soundfile. In the case of *le3*, each attempt at parameter control was captured in a separate soundfile. The important point to consider here is that from one iteration of the process to the next, parameters began from where they were at the end of the previous iteration. In the case of continuous output, if a resetting was desired, it was captured in the soundfile itself.

In terms of the soundfiles generated from RT, they tend to contain an amount of audio not necessarily intended to be captured. As a result, the output of RT software tends to need editing to extract what I am calling the *golden moments*—where all the parameters aligned and something of interest occurred. It is perhaps more obvious with RT software utilising continuous output, but even with programs like *le3* using finite output, there are many soundfiles that contain little usable material.

We can then determine RT software to utilise a subtractive method. The sounds that are created are likely reduced to golden moments. Although, it is important to note that once these moments are extracted, there can be little material in between them. Between one moment and the next, the parameters may have moved so far apart, with the transition being technologically marred, that there is little relating them. This is in direct contrast to NRT, where every iteration is conceivably a usable sound object with a clear lineage; enabling what Wishart refers to as transformation throughout his texts.

3.7 The Software Duality

Whether by choice or not, previous investigations into software we have seen have ignored the software itself, focusing on the composer and encoding their practice within the software. We must only ask the question “How does the software work?” to expose this concern. This is not to say that nothing can be learned about composition from software, but rather providing another approach to the investigations. It is again important to note that this research is not concerned with identifying or creating the best software, or even an all-encompassing program, but rather using software in general to answer questions about composition. As seen with Clowes (2000), if we concern ourselves with specific pieces of software our findings are most likely limited to the lifespan of that program. We must therefore be careful to deal

in more general terms, whilst additionally avoiding gross generalisations. So far, investigations have taken the point-of-view of the User. This ultimately leads to software being regarded simply as a tool: buttons are pressed and a result comes out. If, however, we were able to take the stance of a Power-User or Super-User we realise that software itself has a way of working, into which our own activities while using it must fit—which we might be able to modify.

Software is written, primarily, based on the way we want to interact with it; the difficult part being the design of such interactions. If one were to survey the approaches to the design of audio software, one would quickly see the two operational paradigms emerge: RT and NRT. This in itself is not a new concept, but has been overlooked by existing research. Without taking into account how the software operates, any conclusions are surely limited. These paradigms directly correlate to the development of the computer.

Audio is stored as discrete values of amplitude at specific times, typically ± 1. Initially, computers simply were not fast enough to generate the numbers as they were needed to (re)create an analogue audio signal. These numbers, the audio samples, were stored on tape that could then be played back. As such, the generation and playback of the samples were by necessity separate activities. Such an approach is NRT. The desire was then, as one might expect, to be able to generate the samples as quickly as they were required, calculating at least 44100 samples per second. This may sound simple, but if these numbers are not simply being read back (from a storage medium), they must be computed in full. To illustrate this, take a sine-wave, the basic building block of audio:

$$n = \sin\left(\frac{2\pi f n}{N-1}\right)$$

Here, n is the given sample, f is the frequency, and N is the number of samples (in this case, the sample rate). Using the simplest of optimisations, if n is the only variable, the value of $\frac{2\pi f}{N-1}$ can be stored in advance, allowing the instructions to be reduced to a multiplication and a sine of the result. Even though sin is likely a single instruction, unless it uses a lookup table it might take multiple CPU cycles to calculate the result.

Computers were not fast enough to perform the calculation of the next sample(s) in addition to the outputting them to a digital-to-analogue converter. As such, they wrote their output to a tape (Mathews, 1963) and played it back later. This feedback system is noted by Joel Chadabe (Chadabe, 1977).

In NRT software, such specific concerns over the number of CPU cycles a process will take are not really necessary; the program will just take slightly longer to run and the delay between requesting and

auditioning the sound is protracted. However, when it comes to RT, these concerns cannot be ignored. If the RT process has not completed within the necessary time limit, the audio will glitch, disrupting the immediate feedback loop that RT software creates.

RT software has long been seen as a target. This is discussed in the work of Wishart (Wishart, 1994, pp. 7–8). Although, importantly, Wishart is also cautious about RT at the expense of NRT. He points out the benefits of both, and the results of this research agree with that assessment. This research relates strongly to the ideas of Wishart with respect to explicit and intuitive knowledge (Wishart, 1994, pp. 5–7), providing corroborating evidence to support them, and identifying potential causes and additional impacts not previously discussed.

We have seen that the utilisation of NRT software produces data for every stage: input, process, parameter changes, and output. Taking into account the output data, we can see that these can be considered discrete, concrete sound objects. As a result, an additive method is adopted. In contrast, when using RT software, the data for parameter changes are not necessarily stored or recallable. In addition, the RT output data can either contain errors inherent to the use of control input devices or, in the case of software that can create continuous output, multiple sound objects that may or may not be desired later on. The importance of the output data here is that it will often result in the need to edit or remove content: resulting in a subtractive method.

This presents software paradigms as diametrically opposed: polar opposites in both approach and method. As we have seen from other research, this does not necessarily present a concern: alternating between different software was highlighted as a benefit (Eaglestone et al., 2007), and the operational paradigm could be considered to have the same effect. It could be argued from the evidence provided in this chapter that RT encourages divergent thinking, whereas NRT favours convergent thinking.

We are not concerned with determining the best paradigm to use, but rather understanding how each paradigm impacts composition. For example, generating a soundfile of an hour's duration with slowly changing parameters would be terribly difficult and time consuming in RT, whereas it may be considerably easier and faster to do so in NRT. However, NRT does not necessarily provide an immediate response to parameter changes, so perhaps requires a greater degree of advance understanding. Neither one is better than the other, but their operational differences are important to take into account in any investigation of composition. It would appear that software dictates my method and informs my approach. As has been discussed before, utilising different software is more of a positive than a negative, but our question becomes one of necessity. Is it possible for software to present itself in such a way that it does not require the user to consider the operational paradigm?

3.8 Data

When it comes to software and composition, it is important to consider what is being produced from each paradigm. This has been mentioned repeatedly in previous sections. In NRT software, all the information that is needed for the execution of the synthesis is given in advance: the control, or 'performance' parameters are specified. From these, it produces the new generated sound. In the case of acousmatic music, there will also be the use of an input soundfile. If we were to run the process again, it would take the input file and apply the control parameters again, producing the same output file.

In the case of RT software, there is a marked difference. The program is started. What can be loaded at this point depends on the program, but typically it starts in an initial blank state. The performance parameters are not specified in advance. The input file is not specified in advance, but rather dynamically loaded into memory after the program starts. In terms of what is generated, only the output file is created. If one were to run the process again, the parameters would be where they last were, and it would be impossible to recreate exactly the previous interactions—RT is essentially stateless and where controls are changed interactively it cannot in practice be recreated later on.

Wishart mentions the creation of data. The use of the CDP is particularly evident in the following:

> As a record of this process (as well as a safeguard against losing my hard-won final product) I would keep copious notes and copies of many of the intermediate sounds, to make reconstruction (for variation) of the final sound possible. Today, I store only the source and a brief so-called "batch file" …[that can] create the goal sound. (Wishart, 1994, p. 4)

Wishart goes on to suggest the creation of software that, '…also recorded my hand movements [that would]…store these intuitively chosen values and these could be reapplied systematically in new situations, or analysed to determine their mathematical properties so I could consciously generalise my intuitive knowledge to different situations' (Wishart, 1994, p. 9).

The conclusion we can draw from these discussions is one of reproducibility. NRT is inherently reproducible. RT is more likely than not, irreproducible. How this difference then manifests itself in compositional practice is therefore one of the key questions in this research.

RT–NRT modification

It might be possible in some cases, depending on the software, to have RT behave like NRT. In the case of software in dataflow languages like Pd, it is possible to modify the program to use predefined parameter changes (most often by specifying lines). In something like SuperCollider, it is possible to do something similar, using envelopes and/or sequencing through a Routine.

Paradigm	Input Data	Parameter Data	Output Data	Data loss
Real-time	Partial	No*	Yes	Yes
Non-real-time	Yes	Yes	Yes	No

Table 3.1: Software data by paradigm

*see RT–NRT modification

NRT–RT modification

It is impossible for a user to run NRT software as RT. Not only does RT have specific constraints, but it must also connect to audio hardware to function correctly, requiring significantly more code. However, an example of the NRT–RT modification can be seen in Csound. It is possible to run Csound in RT, as the code connecting to audio hardware is already in place, but the way it handles RT-C is not particularly graceful and often leads to glitches.

Conclusion

In accounting for the data for the operational paradigms, we can identify the differences in method. In accounting for the interactions, we can identify the differences in approach. These relationships are key to understanding the influence of software at a more fundamental level. It is only by taking into account the data from software that we can understand the difference between RT and NRT. In general, RT software is not reproducible, in contrast to NRT. The difference is not just about speed, nor is it entirely about interaction. The final aspect to consider in determining the operational paradigms are the data produced along the way. For a program to be NRT, it must be supplied with information and produce the same output each time. A RT program must provide RT-G and RT-C, but does not necessarily provide enough data to allow the output to be recreated. The differences in data between the paradigms are summarised in Table 3.1. As a composer, I have come to understand these implicitly, but it is important that these be actively considered in research into composition, as well as the design of software.

3.9 Investigating No-Time

As a result of the NRT and RT investigations, I wondered if it was possible to create software that operated outside of the two paradigms. We have previously mentioned software that can switch, or be manipulated into working in either way. For example, Csound can run in RT, but control becomes very difficult and in many cases is unstable. This may be a result of the CsoundQt program's own instability, but regardless

of this the RT aspect is very much an afterthought. We are required to learn an entirely new language (FLTK), or attach long-winded commands to our scores, making them unreadable as their original states would be. SuperCollider can be run in NRT mode, but we must codify our activities into a rather awkward score syntax that is almost too complicated to be useful[3]. I would like to borrow a term from the study of English literature in order to demonstrate my concerns here. The work of Joanna Gavins on Text World Theory poses two main concepts: world-repair and world-replacement. To summarise all too briefly, world-repair is the idea that at some point in the narrative of a story, information is revealed that causes the reader to re-evaluate certain aspects of the world within which the story is set. World-replacement is when this information so drastically reshapes the reader's view on the world that they must entirely reconstruct the world they created in their minds, based on this new information. For a moment, let us apply the same concept to interaction with software. We develop ways of working with software, be it NRT or RT. This would be the world that we construct. Thinking of how both Csound and SuperCollider switch paradigms, it may be understood as a form of world-replacement: we must entirely reconstruct our approach and interactions based on this context switch. As these examples show, it is often a difficult change to make. The question then becomes whether or not it is possible to design software that does not force this re-evaluation.

This, then, is where the *No-Time Ideal* emerges. Here, I define No-Time (NT) as software designed to operate in either paradigm, RT or NRT, without forcing a contextual shift. If the paradigm is not something I am actively considering as a composer it would be ideal if it was not a factor in the software itself. The investigations that follow present a first attempt at discovering and outlining the requirements of NT, as well as potential approaches to software implementation.

3.9.1 SOClib

SOClib, the Sound Object Composition Library, was a first attempt at No-Time software. It is a Quark for SuperCollider that provides an object-oriented approach to creating and processing sound objects. The design was to allow interaction at a sound object level, with parameters and their values being abstracted behind the programming interface. Once values have been set, the object can simply be told to *play*, recalling all of its processing and parameter settings. The creation of sound objects is simply:

```
1 ~mySoundObject = SOC_SoundObject("path/to/soundfile");
```

[3]There are additions to the language that can make this process easier, but we are discussing the native NRT/RT behaviour of the software.

From there, it was a matter of adding processes to the sound object. Most processes do not generate their own audio, so the first process to add is usually soundfile playback:

```
1 ~mySoundObject.add(SOC_Player());
```

The `SOC_Player` gives access to simple playback parameters, such as speed and amplitude. The most important thing about *SOClib* is that it employs the idea of No-Time. If you want to use it in RT, you can. If you want to use it in NRT, you can. To use it in RT, you can ask for the GUI window:

```
1 ~mySoundObject.gui;
```

At which point, a graphical interface will appear, representing the sound object and its processing. It contains a waveform view of the entire soundfile it relates to, but a selection can be made to reduce the SoundObject to a specific section of a given soundfile. This makes it possible to have several SoundObjects referring to different points within the same soundfile. As a result, it is not entirely necessary to extract clips from the sounds in advance of using *SOClib*, although it would still be a useful stage, as the selection is not necessarily saved.

For use in NRT, the inbuilt scheduling of SuperCollider can be used. I primarily opt for `Routine`s given my experience with them. An example of a `Routine` as used with *SOClib* is given in Listing 3.1.

The development of *SOClib* was driven by the piece alongside which it was written. It was designed to be completely modular, providing a framework within which to quickly add new processes as they were needed. As a result, it is not particularly well written, but as evidenced by the music created with it, it is very much a useful collection of processes. Its design is influenced by that of *le2/3*, consisting of controllable parameters within a process, the difference here being that instead of an audio graph *SOClib* uses a `SOC_SoundObject`. It is perhaps of greater use to explain *SOClib* in situ, and interesting to note that its development exemplifies that of *situated action*.

3.9.2 Under the Spreading Chestnut Tree

As mentioned, *SOClib* was created alongside a piece, which would eventually be titled *Under the Spreading Chestnut Tree* (UTSCT). I had just read Orwell's *Nineteen Eighty-Four*. It had a profound effect on the way I viewed the world around me, and came at a time when his words rang perhaps truer than ever. I felt compelled to write a piece. Not with the goal of retelling the story, which would ultimately be futile,

```
(
  var soundObject, gainProcess, routine;

  soundObject = SOC_SoundObject("/path/to/soundfile.wav");

  gainProcess = SOC_Gain();

  soundObject.add(gainProcess);

  routine = Routine({
    var gainParam;

    // get the gain parameter of SOC_Gain
    gainParam = gainProcess.getParam("gain");

    // set the value
    gainParam.value = 1;

    // wait 1 second
    1.wait;

    // change the value
    gainParam.value = 0.5;
  });

  routine.play;
)
```

Listing 3.1: Simple *SOClib* Routine example

but rather a sonic environment inspired by its setting. UTSCT, therefore, fits firmly within the *Personal* motivation.

It was around October 2017 when I finished the book and decided to write the piece. Luckily, as a result of the time of year, by the time I got to making source recordings in early November, Sheffield was experiencing the remnants of a large storm. At the time, I lived in an area that contained a small wild area with trees. As I was walking home I was taken by the sound of the storm through the trees and borrowed portable recording equipment the next day, hoping that the weather would return. Luckily it did, and I braved the weather that evening in the hopes of reducing the surrounding traffic. I recorded roughly 30 minutes of a soundwalk through the trees and surrounding area as the weather kept up. Rather fortunately, it was not raining.

After returning to my apartment, I was keen to listen back to the sounds, so set about offloading them from the recording equipment. The cheap hi-fi speakers attached to my computer at the time revealed some promising materials, but on listening back in the studio the following day, it was clear that what had

been recorded was not what I, as a listener in the environment, had experienced. I was primarily left with fluctuating noise that barely portrayed what I had heard. I'm sure that the experience of the environment whilst making the recording was only really something I could connect with upon subsequent listening, having been the one who was actually there; the connection being made by my own memory. As this piece was aimed at presenting my experience of the book, this would be something I'd have to address in the composition of the piece.

There were four main elements of the book that I wished to encapsulate in the piece: the environment of the world, the Department of Truth, Room 101, and the ending. As the goal was not to retell the story, adherence to the overall theme for the piece was not limiting in any real sense, mostly providing a frame within which to work. Having these four elements, it seemed quite obvious to me that the piece ought to be in sonata form—not for any classical reason, but rather a way of exploring two smaller ideas (environment and DoT) that merge into a more complex environment (Room 101) to produce a larger piece.

Once in the studio, having listened back and made a series of clips from the source recordings, I thought about what I might do with the sounds. I set about trying to create the environment section, which would be the first subject of the sonata form. The source sounds did not have the sense of space and motion that I had experienced, and so I set about trying to artificially create this sense of dynamism. I wanted to retain some naturalism in the sound, so it was important that any processing did not introduce too many digital artefacts.

First subject: The environment

I decided that granular synthesis would allow me to control the parameters I required and having large, overlapping grains would retain enough of the original sound to remain convincing. As such, I added the `SOC_Granular` process to *SOClib*. This posed a particular problem with the library, given that *SOClib* relies on the sound finishing at some point; granular synthesis in this case being a continuous generation process. However, the need to continue composing superseded the need to find a solution to that particular issue and so it remains unresolved. I proceeded to create an `SOC_SoundObject` with the opening soundwalk material, adding the new `SOC_Granular` process. Being unsure about the resultant sound, I instinctively opted for RT operation. After modifying some parameters, it was clear that the process was appropriate, however, the aspect of spatial movement was still rather constrained. At this point, I added the `SOC_Balance` and `SOC_Pan` processes to *SOClib*. These allowed me to pan the sound between two speakers with a random element. Given the dramatic nature of the desired

sound, the randomisation was created with stepped randomness, making the sound jump from point to point, as opposed to smoothly transitioning. As the output from the granular process was stereo, I used `SOC_Balance` to control the panning. Once I had set the amount of randomisation to an appropriate value, I began working in RT to see what controls would produce interesting effects. As the pan controls had been set, I was utilising the controls of the granular process. In order not to lose anything while changing parameters in RT, it was important to record the audio output of the system. As a result, I ended up with long recordings of granular material panning randomly. Through the process, I learned how the sound was reacting to the parameter changes, and discovered some interesting parameter movements. Unfortunately these changes were too quick and frequent to be sensibly encoded using the `Routine` scheduling, but had there been a way to easily do so, I would have. This is not a limitation of the NT concept, but rather the *SOClib*'s control implementation; had *SOClib* used a more dynamic control system utilising control buses it would have been much easier to codify the changes in parameters I was performing in RT. Regardless, I ended up with some very dynamic environmental material, and taking this back into the DAW session I was able to combine it with the original source recordings to create the opening of the piece, starting with the source and gradually introducing the granular material created an environment that represented what I had experienced, but also adding a sense of the unreal that helped to create a sense of unease.

Second subject: The Department of Truth

The second subject, based around the Department of Truth, explores the idea of information being created, edited and discarded. This section relies almost entirely on source-bonded material, utilising largely raw recordings. Throughout this section, silence is used as a compositional element to emphasise the void left when something is removed. The section used little processing, rather using sequencing in the DAW and simple delay-based and filtering processes with DAW plugins.

The use of the DAW for this processing was one of structural necessity. The filter sweeps, for example, needed to happen at a more macro level and needed to be adjustable at a structural level for the piece, rather than at an individual sound object level. The timing for the parameter changes were not in relation to the sound object itself but rather the other sound objects that comprised the phrase or section, the reason being that if those sound objects moved in relation to each other, it may be necessary to adjust the durations of certain parameters to account for these changes so having them encoded in the object would require the generation of a new sound, which may prove very difficult, particularly with respect to RT. These should be thought of as trans-object changes, related to the objects around it within a phrase.

As the DAW is responsible for sequencing sounds in time, so too is it responsible for these structural level, trans-object, parameter changes that are not necessarily part of the sound object themselves, but rather part of how they relate to other objects in the phrase. Since the composition of UTSCT, this idea has been incorporated into *SOClib* with the addition of composite sound objects: a sound object made up from other sound objects, with a time offset in between. If this was used to create an entire phrase, and the parameter system updated to be better scriptable in the SuperCollider language, the need for the DAW-based processing might somewhat diminish.

Development: Room 101

Starting at around 04:11, the piece slowly intensifies, culminating in the sonic representation of Room 101. It was during this section that the NT system came more greatly into play. This section makes heavy use of the `SOC_Playback` process, with combinations of fixed, scheduled and RT playback-rate changes. There are also a great number of RT granular explorations. Importantly, whether RT or NRT, I would begin with RT and move towards either NRT, or if the movement was too active, have a specific trajectory in mind to perform during the RT recording. Previous sections of the piece were exported as their own sound files, and processed using the granular processing, including similar parameter changes to the initial material. One thing I noted during the process was, again, that had I been able to script these dynamic changes to precise values, it would have been a preferred way of working.

The culmination in Room 101 is a 10kHz sine wave punctuated by a rather large, compressed attack gesture. Both of these were entirely synthesised using SuperCollider specifically for the section. The duration of this particular section was timed to coincide with my own limit for tolerance of the sine wave at a given level. The processed environmental recordings return here to give relief, and the section fades away to the original source recordings that begin the closing section, this transition forming the recapitulation of the sonata form.

Coda

A coda was added to UTSCT. In part due to a frustration with the ending of the book, this section served thematically to present the idea of hope. The material here is entirely synthesised using the Karplus-Strong algorithm (Karlplus & Strong, 1983), controlled by random algorithm, all within SuperCollider. The tonalities and timbre are suggested throughout the piece, being the source of many of the pitched elements.

The ending has proved somewhat divisive, with several listeners questioning its necessity. It is interesting to note that this is really the only compositional decision directly related to the theme of the piece. I believe that a piece should stand on its own without the use of programme notes, as anything to be communicated should be done in the music itself, if that is even the goal of the piece. For me, the ending is required, but it raises the question of whether or not there should be alternative versions of this piece, both with and without the coda.

Initial Conclusion

In cases of NT, when I am unsure of how a sound will behave, there is a clear trend towards first utilising the RT aspects. Alternatively, where there is a greater certainty of output, I have a tendency to begin with NRT. As the idea of what processing a sound requires becomes more concrete, my desire to utilise NRT processing becomes greater. In the case of *SOClib*, due to the limitations of the control system, the creation of complex parameter changes became very difficult. As a result of writing UTSCT, it became clear that the idea of a composite sound object was needed, as well as the addition of breakpoint controls that allowed NRT control to be specified more usefully. Although the breakpoint system is yet to be written, the `SOC_CompositeSoundObject` was later added, and it allowed a single sound object to be made up of several others with individual time offsets. In theory, an entire piece could be made this way. However, the likes of `SOC_Granular` that have no fixed duration proved problematic and so this has yet to be finalised. Since work on *SOClib* gave way to other investigations, it has become evident that a `SOC_Envelope` process that can terminate objects gracefully would likely solve this.

To conclude this initial investigation, it has become clear that there is a tendency to work from RT to NRT, or use NRT throughout, provided that NRT provides a way to adequately specify dynamic changes in parameter values. In the absence of adequate parameter control, RT will be utilised and an approach of good-enough will prevail. Additionally, whilst not full world-replacement, *SOClib* provides an example of world-repair—switching from the graphical RT-C to the text-based scheduling for NRT was a change, but because the use of `Routine`s are natural SuperCollider, it did not present as great a challenge. It may be the case that if everything was graphical, or indeed text-based, this may remove even the world-repair aspect of switching operational paradigms.

3.9.3 Under the Spreading Chestnut Tree: Reflection

As mentioned, the motivation for this piece was personal. The main intention was to create a piece inspired by Orwell's Nineteen Eighty-Four but intentions also included the use of field recordings and the use of sonata form as an overall structuring device. In terms of more broad activities, my approach was primarily to improvise with the field recordings and discover interesting relationships to develop further. The approach resulted in the generation of materials, through the methods (discussed below) that were suggested by the materials themselves. This led to more approaches for working with these new materials and the piece developed from there. Among my methods were the use of *SOClib* to granulate and randomly pan materials, and the positioning of materials in time with the DAW to create phrases from those materials. Additionally, the DAW provided reverb and filter-sweeps, much in the same way as the previous piece.

In terms of influences from the previous composition, I did not want to use the same structure, or to use pitch in exactly the same way. I did, however, use similar attack and drone gestures to facilitate transitions and prepare sounds for later use—primarily the coda—as well as granulation of the previous sections to generate an additional section (constituting approach, using the method of *SOClib* granulation and panning in RT), and the use of filter-sweeps between materials. In addition, I was keen to have this piece be very performative and as with *Twist and Turn* it contains several sections that may appear slightly long in a stereo listening environment, the goal being that the use of diffusion would be the focus of the work for that time; for example, the development and recapitulation sections, where their dynamic materials are composed to suggest large spatial movements that can be explored in performance. Another key influence from the previous piece was ensuring that each section was as long as it needed to be. In previous pieces I have moved on too quickly to another section or introduced new material—this is mostly driven by a sense that something always needs to be happening, but I learned a lot about allowing materials to gestate from composing, in particular, *shift*[4] and *Twist and Turn*; although it is still very difficult to overcome this need for constant activity.

Positioning the piece on Emmerson's language grid, I consider UTSCT to have a combination of abstract and abstracted syntax, with primarily mimetic discourse. Although the overall structure is imposed by sonata form the sections were not rigidly constrained by it—the phrases within the sections are still very much governed by aural relationships between materials and not that of conventional pitch or rhythm, with the exception of the coda.

[4] *shift* is a composition from my MA portfolio, composed in 2014

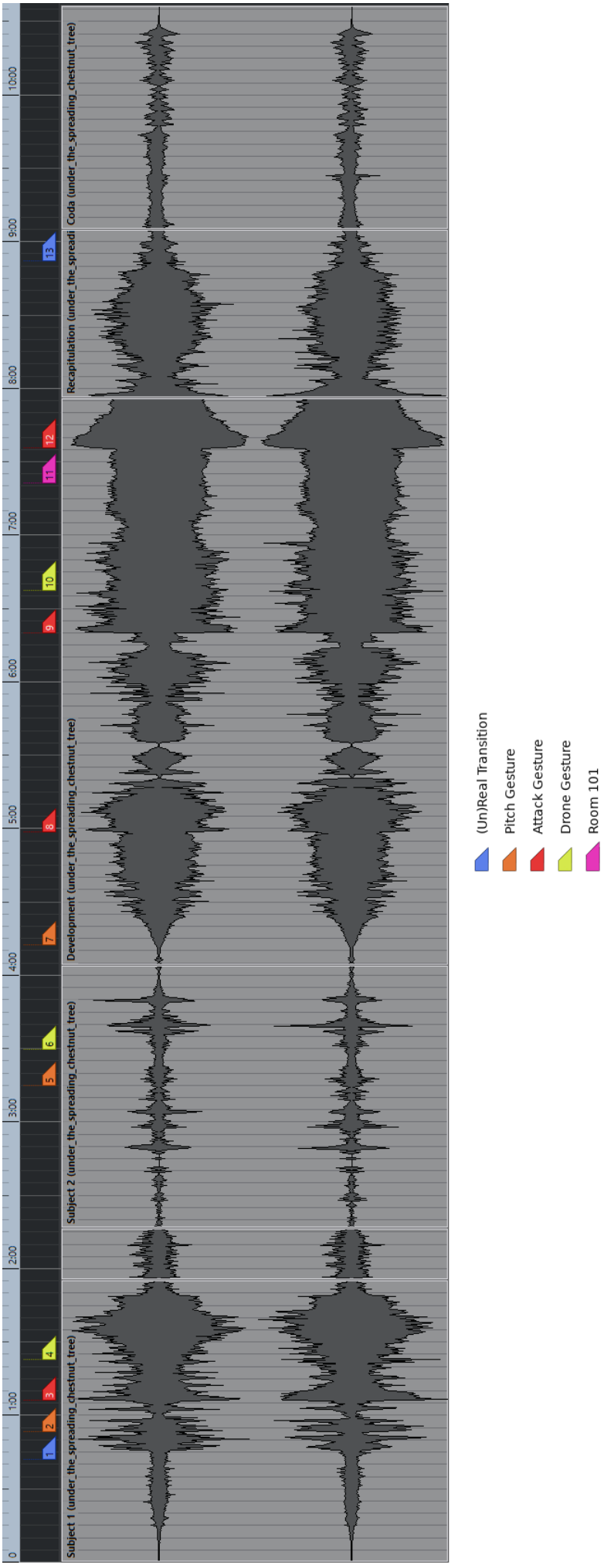


Figure 3.1: UTSCT - Waveform

This first subject of the sonata form serves to present the real-world and somewhat harsh environment—this being the intention for the subject, realised by the use of field recordings. The reality of this environment is then challenged by the introduction of the granular materials, and this real/unreal transition is mirrored at the end of the piece—see markers 1 and 13 of Figure 3.1 respectively. As with the previous piece, I used patterns of three to set up and subvert expectations; an example of this can be heard in the first subject which swells three times, each time introducing new materials.

The bridge between the two subjects serves, rather literally, to move the piece from outside to inside, using the sounds of footsteps and doors to facilitate the transition. I used this to my advantage during performance, mimicking the transition in physical space by reducing the spatialisation down to a pair of distant loudspeakers—in the case of Sheffield this happened to be on the stage of the auditorium and resulted in the effect of a performer physically walking across the stage. It proved very effective, with several listeners commenting afterwards and provides an example of using the specific traits of a given venue to highlight certain compositional aspects during performance, and as I knew the piece would be performed in Sheffield I was conscious of this during composition.

The second subject represents the day-to-day job of Winston Smith at the Ministry of Truth—the sound of written words being crossed out, with the silence that follows; the transition to the modern equivalent, using computers as the method of deletion; the constant, mindless, unquestioning repetition of these tasks being highlighted with the use of delays and echoes–using of technology for its own effect, rather than for any particular musical/functional purpose—the effect of the technology being the purpose, mirroring the intention to replicate repetition. Whilst this section contains a number of references to the intention for the piece, this only represents an influence on my approach—the exact method(s) of processing, positioning and phrasing were still devised through listening and reacting, primarily through the use of RT aspects of the *SOClib* and NRT aspects of the DAW.

The development section explores the first and second subjects, notably using tape-based manipulations to create vibrato and pitch variations. I used contrary motion in pitch between several sources to create tension and build momentum in the section. This is swept away by the first of three attack gestures at marker 8, revealing pitched materials (derived from reversed excerpts of the theme from the coda, heard between 05:17-05:24). A deceleration gesture leads into a slow accumulation of granular materials, providing a build into an attack gesture at marker 9 of Figure 3.1 which ushers in more dynamic materials. This then culminates in Room 101, starting at marker 11. The third attack gesture is used at the introduction of the 10kHz sine wave, seen at marker 12.

The pitched material used in the coda is prepared throughout the piece, although in the form of drones

and short excerpts, as seen at markers 2, 5, 7 and 9 of Figure 3.1. As mentioned, the coda was created as a reference to hope and I often regarded it, at least in my own mind, as Julia's theme. The use of a randomised algorithm serves to represent the idea of beauty from chaos, or perhaps more appropriately a sense of hope from disorder/despair. It isn't important that a listener identifies any of the representations in the piece, as mentioned it should be able to stand on its own.

UTSCT is the first piece I have written with such a strong sense of symbolism, even if I was conscious to keep it constrained and rather abstract. As such, being in new territory, I am not sure how successful certain aspects were. I am convinced that the coda was required by the piece, but as others have commented it could perhaps have been more successfully prepared. On reflection, I am conscious that certain aspects were chosen for extra-musical reasons and it may be the case that these are the least successful moments. However, I remain pleased with the resulting piece and its performance.

3.9.4 Something from Nothing

Something from Nothing (SFN) was another piece created using *SOClib*. It focused on utilising the processes that did not have a fixed duration: `SOC_Granular` and `SOC_Stack`[5] in an attempt to confirm previous findings with RT software.

SFN was primarily created for the Sound Junction concert series. This would be my last opportunity to perform at Sheffield before the end of my studies and as such I required a new piece to perform. SFN therefore exemplifies the *Exhibition* motivation. At the time, I was conscious of my long-forgotten instrument, a clarinet, that was sitting in the corner of my flat. I had always loved the way that it gave me the ability to create sound from silence, through breath to pitch. As with UTSCT, this theme served as a structural element within which to work.

Work on the piece began with the recording of clarinet material. In a moment of uncharacteristic certainty, I knew how I wished to set up the microphones. I used two microphones, traditionally positioned to record clarinet: at the bell and the upper joint. However, in the studio I treated these as a stereo pair, panning hard-left and hard-right. This was done to make use of the stereo image as I played, without the need for physical movements that would create extra unwanted noise. I recorded a variety of breath sounds, and as I moved around the instrument, various panning was encoded in the source recording. This created some rather dynamic material to work with, not only having timbral variation but with the addition of spatial motion. I recorded categories of sound, corresponding to the three sections: breath, breath-to-pitch, pitch. Naturally, silence would be handled by the lack of sound itself. Having made

[5]The Stack process is based on CDP's stack.

recordings, as has evidently become custom, I proceeded to extract the material that I thought would be useful—as before, mostly removing what had been marred by technical errors.

At this stage, it is important to highlight the DAW I was using: Cubase 10. This particular version contains a host of new features that make the composition of acousmatic music much easier than other DAWs. For example, the opening material was not only created using the new Sampler Track[6], but utilised Cubase's Restrospective MIDI Recording[7]. Additionally, while writing various phrases and sections of the piece, I heavily utilised the new improvements to Direct Offline Processing (DOP) as introduced in Cubase 10. This was particularly important, given that DOP provided semi-concrete sound objects—the data from all processing were stored alongside the input files. With regards to the investigation, this addresses the challenge from *SOClib* and relative trans-object changes (parameters that might require adjustment if objects move within relation to each other), as even the destructive processing is non-destructive; at least until "make offline processing permanent" is activated.

There were, of course, some processes that were not possible in Cubase. For example, granular processing is not available, so it was all achieved with *SOClib*. The final materials were also created using SuperCollider—with the chordal material, similar to *Twist and Turn*, being created by filtering the original material with band-pass filters to form chords, articulated by the underlying material, using a randomised chord generation algorithm.

The granular processing created soundfiles of long duration, each containing a number of sound objects that may or may not be related aurally. Additionally, most of them contained materials that would not be used in the piece. This confirms that software operating in RT, as is essentially the case here with *SOClib*, utilises improvisation with subtraction. Interestingly, as with *Twist and Turn*, I struggled with the ending of the piece. I exported the entire piece as it was at that point and used it as the input to the granular processing. The resultant material was used almost verbatim as the ending of the piece. As the position in the input soundfile was manipulated in RT, different sections of the piece came to the fore, and sections began to combine in interesting variations.

3.9.5 Something From Nothing: Reflection

Representing an exhibition motivation, several intentions for the piece were guided by knowledge of the concert format and venue. As a result, the duration of the piece is around 10 minutes and contains dynamic gestural materials particularly suited to diffusion. It is perhaps important to note that the du-

[6]The Sampler Track, since Cubase 9.0, allows the creation of a single-sample sampler instrument.

[7]Retrospective MIDI Recording allows the user to recall the most recently played MIDI, even if recording was not active. The earliest references I could find to this feature are from Cubase 4.

rations for each of the pieces composed as part of this research are similar in duration. This is likely a subconscious acknowledgement of the requirements for performances—writing a piece of long duration severely limits performance opportunities as concerts tend to include a number of shorter pieces, commonly 10–15 minutes, to allow a greater number of composers to perform.

The main intention for the piece was the use of clarinet materials, exploring noise to pitch. The previous pieces used object-based, mimetic sounds and field recordings, so the choice of instrumental sources was driven by that influence as an alternative. As a result of using a pitched instrumental source, I decided to approach the use of pitch in a similar way to that of *Twist and Turn*—the harmonic material of SFN being derived from filtered noise material, taken from the instrument, in the same way.

The overall structure of SFN was again derived from the intention of the piece—exploring the pitch-noise continuum. As a result, the piece naturally became ternary form, similar to *Twist and Turn*. Positioning it on Emmerson's language grid, I consider it to have abstracted syntax and mimetic discourse: the materials were chosen, developed and structured based on their aural relationships; and they are mostly source-bonded, with sections relying on harmonic relationships.

It begins with a set of three phrases, getting progressively longer and more developed until the third which expands into the rest of the section: see Figure 3.2, markers 1–3. Similar to *Twist and Turn* SFN uses iterative granular materials to provide ornamentation and variation. Instead of a violent attack gesture, however, SFN makes use of a more subtle stacking gesture (both ascending and descending) throughout, serving as a recurring motif with differences in stack offsets providing sources of variation: see markers 4, 5, 8–12, 14, 17 and 19 for the most notable examples.

The second section begins with a transition from noise to pitch—a raw source recording of a performance, at markers 6 and 7. It perhaps wasn't the best instrumental performance but it was the only one that didn't have technical issues; the earlier recordings had better performances but the preamps weren't calibrated correctly and the later performances were marred by my own fatigue—as such the choice was quite limited. This is punctuated by descending gestures, which introduces the underlying drone material. As these gestures lose energy they are replaced by breath and key-noise gestures, and these are then slowly filtered to produce pitches from the breath materials—in the same way as *Twist and Turn* filters are always present in this material but they slowly become more focused to reveal their centre frequencies, producing pitches and harmonies, see markers 13 and 14 for filtered breath materials. This then gestates for a time. As with the previous pieces this was to allow material to develop and facilitate performance opportunities—in this case, rather than having dynamic materials as the focus, SFN uses the separation between the gentle drone materials and the articulations to allow for spatial exploration—

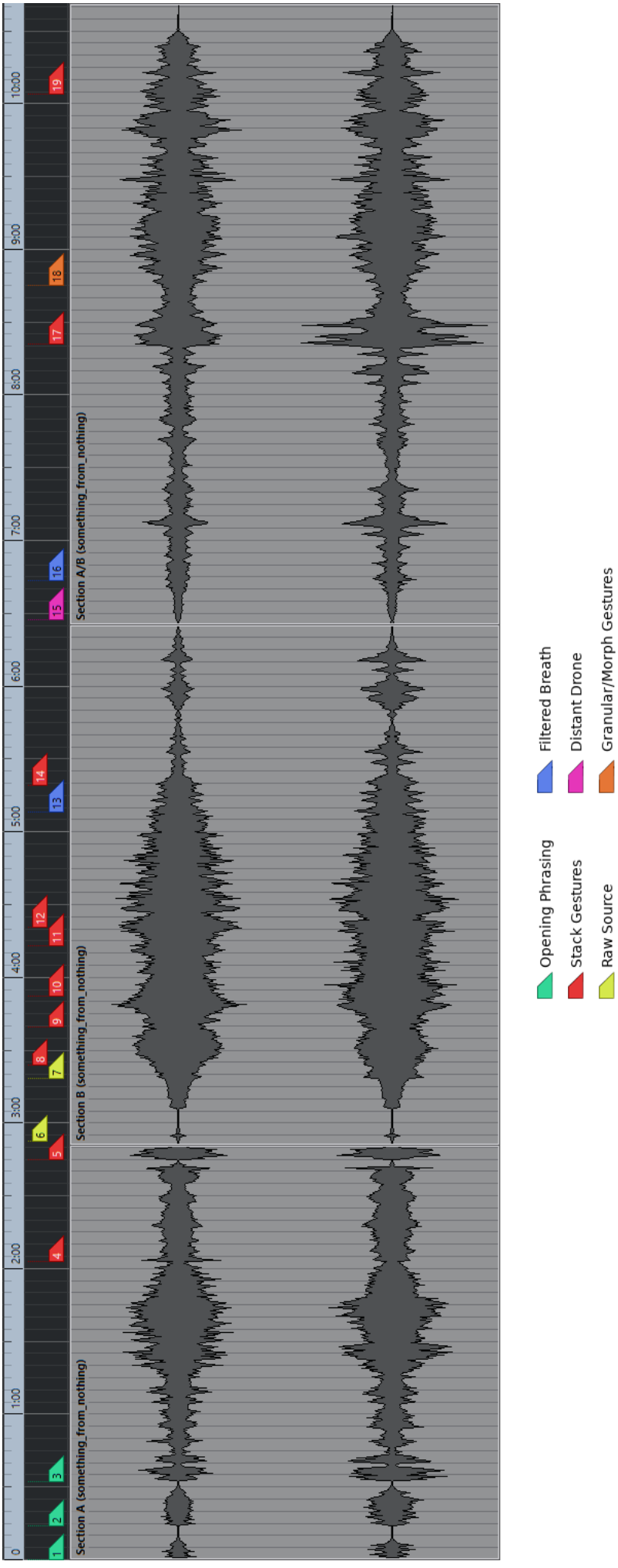


Figure 3.2: SFN: Waveform

the more pitched materials can be positioned around the audience to create an envelopment, with gentle movements between speakers (reflective of the ebb and flow of the materials) being highlighted by the ornamentation and other materials.

The second section then fades out, followed by a distant drone (applying additional reverb) referencing the opening section (see marker 15). The silence between the fade out and the drone entering is deliberately timed to allow a movement in performance: sending the sound to more distant speakers, mimicking the distance in the material. Again, the drone crossfades into the final section in such a way that allows the piece to be effectively moved back into a closer speaker position in performance. The final section begins with a return of the opening materials from the first section quickly followed by the filtered breath material from the second section. The filters tighten more than before leaving almost pure sine waves articulated by the underlying breath material. A descending stack gesture then ushers in a granulation of the previous sections at marker 18, similar to both *Twist and Turn* and UTSCT. By exporting and using the other two sections, scrubbing backwards and forwards in time with large grain durations, it becomes possible to combine elements from all the previous sections into an interesting blurred collage that ebbs and flows between snippets of known material in new ways. These dynamic morphs allow for dramatic spatialisation in performance.

The pacing of this piece is quite different to the others in this research. It is much slower and fairly consistent, ebbing and flowing in line with the materials. I was conscious, as with previous pieces, to allow the materials to gestate and given their sources the materials have plenty of variation over time. This is particularly useful in performance given that there are moments of subtle movement but also moments of significantly greater activity which can be accentuated by spatialisation.

SFN is a tribute to my past and present, and somewhat representative of my hope for the future. While composing I was conscious that it would be the last piece of this research and that it was quite possibly the final piece I would perform in Sheffield. I have not previously had much emotional investment in the materials I use in my music, and I have no doubt that this changed the way I approached working on the piece. In fact, Adam Stanović commented after the first performance that an emotional attachment to the materials was audible, in a way that hasn't been manifest in my other works. I feel that this piece is successful in much the same way as *Twist and Turn*: there is something in its simplicity that is both comforting and interesting. I am pleased with this piece and as with the others I will continue to learn from it and its performances.

3.9.6 lwae

lwae (lightweight audio engine) is a C++ audio library aimed at creating NT systems. It combines the *linkEngine* breakpoint system with the *le2* RT engine, adding a switchable value for each parameter. In *lwae*, everything is calculated at audio rate[8], however, to avoid zipper-noise all RT parameter changes are linearly interpolated over 2ms.

The core of the library is the `AudioGraphNode` class. As before, these nodes can be connected together in any combination, provided the nodes can be connected. One of the key differences is the inclusion of a switching mechanism for all parameters—allowing them to hold both a RT and NRT value, and dynamically switch between the two. To switch, a call to the `Parameter::setState` method is made, passing an argument of either `State::DYNAMIC` or `State::AUTOMATED`. The dynamic state corresponds to RT control, and the automated state corresponds to NRT breakpoint control.

The parameters themselves provide a `value` method, which will return the value of the currently active state. This is implemented rather simply by the use of a pointer. Changing the state alters what the pointer refers to, and a pointer swap is a guaranteed atomic operation so should not result in any data tearing. Each parameter is ticked per sample, which is used to advance both the interpolation of the RT value and the position within the breakpoint.

Nodes can be connected together as before, but to ensure that they are processed in the correct order, rather than relying on manual ordering, it was necessary to re-write the audio graph. The `AudioGraph` class in *lwae* is a container for `AudioGraphNode`s, as with *le2/3*. In contrast to *le2/3* nodes are automatically sorted as their connections are made to ensure dependencies are accounted for. The topological sorting algorithm used here is an implementation of what is known as *Kahn's algorithm for topological sorting*, which can sort directed acyclic graphs, such as those in audio processing. This was important, as it allowed the creation of graphs in an ad-hoc fashion, which is most likely the way a GUI application would be used. As nodes are added to the graph, the ownership of the node is transferred. If the graph is asked to connect nodes it does not own, it will take ownership. This means that the use of the `AudioGraph` will ensure not only the correct node order, but also that all allocated `AudioGraphNode`s are deleted alongside the graph. There may be an occasion when a node is added to the graph that doesn't have any connections, an orphaned node. In order to reduce CPU load, particularly in RT, these nodes shouldn't be processed. Due to the topological sorting, the `AudioGraph` can identify such nodes, and remove them from the active processing graph. The idea of an active graph for processing, in addition to the list

[8] It must be noted that at higher sample rates, this may prove problematic.

of all nodes, is a key difference from other software in this research, that was made possible by the use of the topological sorting.

In order to facilitate as much flexibility as possible, it was necessary to redesign how audio files were handled by the system. In *lwae*, disk input and output are themselves implemented as `AudioGraphNode`s. This means that they can be placed at any point in the graph, and even allow graphs to input and output multiple files, potentially with completely independent processing. In order for this to be possible, it was necessary to make the file accessing RT-safe. As such, both `DiskIn` and `DiskOut` utilise separate threads that read/write buffered audio to disk, preventing the audio thread from blocking and glitching the RT-G.

Initially, *lwae* used the XML specification of the *linkEngine* variants. However, as will be discussed in the next section it became necessary to utilise a simpler system of communication that allowed graphs to be created from graphical programs.

3.9.7 Chronon

The chronon was a theoretical particle of time, proposed as a way of understanding some of Einstein's theories. Later, other theories removed the need for the idea of time being a particle, but it was based on the idea of wave-particle duality. In much the same way, the software presented here, *Chronon*, provides an insight into the software duality of NRT and RT.

It was developed alongside, and utilises, *lwae*. It provides a graphical front-end for *lwae*, allowing the creation of audio processing graphs. Crucially, learning from *SOClib*, it does require an operational domain shift on the part of the user. Additionally, it allows the user to select which of the parameters from whichever of the processes in the graph they wish to work with. Upon running, the user can listen to the product as it is generated, but they are also free to take control of a parameter in RT and have *that* become the output. The RT interaction will then be recorded and pulled back into the system to be used as if it had been the NRT input from the beginning.

Until this point, *lwae* had been using the XML specification of the previous engines. This proved useful, but ultimately the use of XML was an unnecessary overhead. As such, *AGScript* was created. *AGScript* (AudioGraph Script) is a lightweight command-based scripting language used to create audio graphs for *lwae*. It uses a simple set of commands that can easily be remembered once the concept of graphs is understood. It consists of 6 commands, each taking a number of parameters. The full specification is provided in Listing 5.

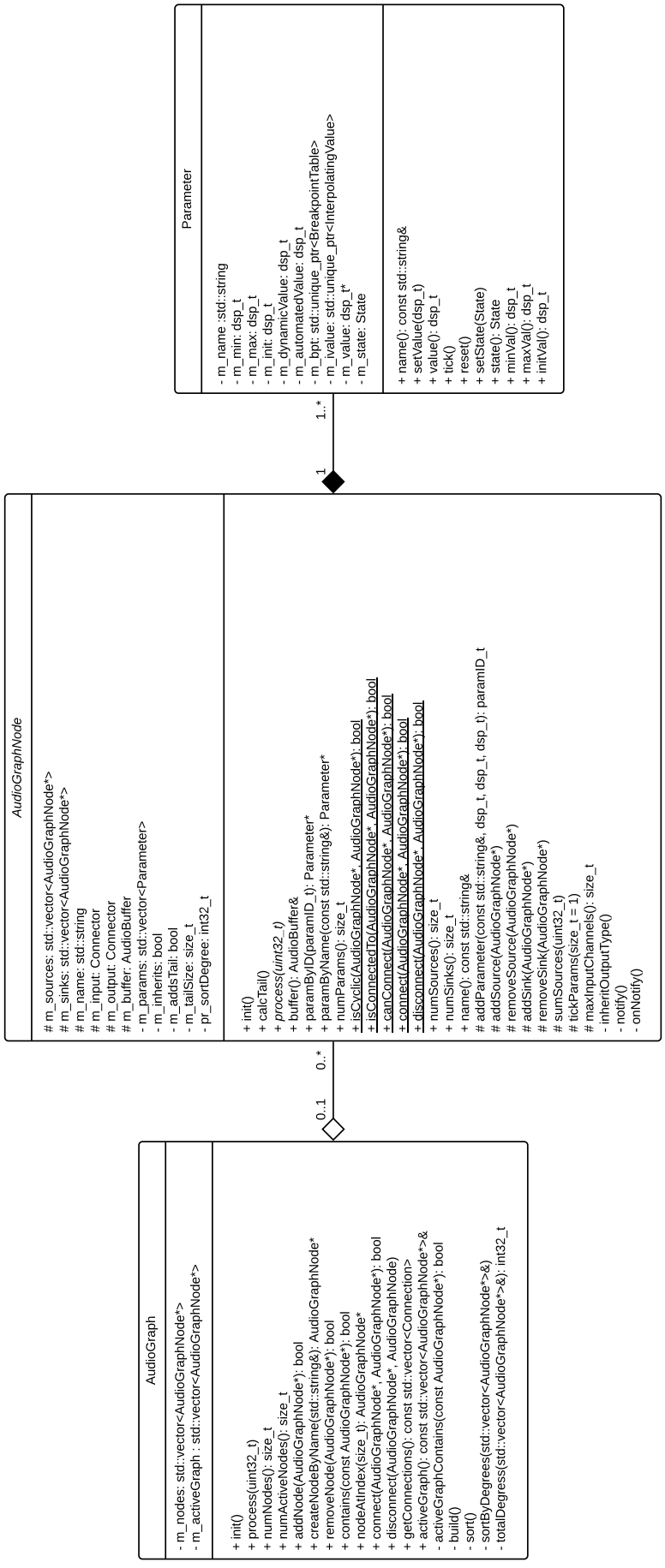


Figure 3.3: AudioGraph and AudioGraphNode UML diagram

```
AddNode <id> <nodeName> <xPos> <yPos>
ConnectNode <srcID> <sinkID>
SetNodeProperty <id> <propName> <propValue>
AddBreakpoint <id> <paramName> <time> <value>
SetParamMode <id> <paramName> <mode>
SetParamValue <id> <paramName> <paramValue>
```

Listing 5: AGScript specification

AGScript can describe any audio graph that *lwae* can create and run in a NRT context. This is why it does not have commands like RemoveNode, for example. *AGScript* was developed as part of *Chronon* to make save files easier to create/read. The *AGScript* parser written as part of Chronon connects directly to the respective *lwae* functionality. *AGScript* represents the NRT aspect of lwae. *Chronon* itself provides the RT functionality—connecting directly into *lwae*'s RT aspects, while creating a *AGScript* record of its activity at the same time. A simple example of *AGScript* can be seen in Listing 6.

```
AddNode 0 DiskIn 0 0
AddNode 1 Gain 0 100
AddNode 2 DiskOut 0 200
AddNode 3 RtOut 200 200

ConnectNode 0 1
ConnectNode 1 2
ConnectNode 2 3

SetParamMode 1 gain dynamic
SetParamValue 1 gain 0.5
```

Listing 6: Simple AGScript Example

Chronon provides a graphical representation of the graph and allows it to be edited. Importantly, it cannot be modified during RT-G. It became clear from my earlier investigations that being able to modify the graph in RT was not necessary, as I would decide the *processing* in advance where possible, leaving the exact parameter values to RT-C if allowed. The parameters that are controlled in RT are chosen in advance of any interactions. In contrast to *SOClib*, however, these are not presented on a process-by-process basis. Instead, the user can choose the parameters from whichever process in the graph to control in RT—in effect, designing a control interface. These are currently displayed as simple sliders, but the software could by modified to display any graphical widget. Those that are not assigned as RT-C can still be set to specific values, either static or automated with breakpoints, in advance of generation. Having decided the process, and the parameters to control, the user can then generate audio. It is possible to

assign all the parameters from all the processes to RT-C.

Chronon features a *render* function, that runs a loaded *AGScript* file entirely in NRT to produce an output. However, it additionally features a *play* function. Here, the program will operate as RT-G only, utilising the breakpoints as previously requested. If and when the user modifies a control in RT, *Chronon* will signal the parameter to switch to dynamic mode, offering RT-C. At this point, the value of the slider is passed to the parameter directly. However, before the value change is requested, it obtains a time-stamp from *Chronon* (with a clock that restarts each time the program is asked to create audio). This time-stamp, along with the value and parameter, is stored in a queue as a record of the event. This is used to fill in a breakpoint table, that is crucially not part of the audio engine. This secondary, hidden, breakpoint table is used to generate NRT save file, once the processing stops. *Chronon* is responsible for creating these *AGScript* files, whereas *lwae* is only responsible for handling RT events (responding to parameter changes, processing audio, writing soundfiles). This separation of duties aims to prevent RT glitches. The *AGScript* file created post-processing is then loaded into the program for further work.

As a result, *Chronon* handles the *AGScript* and user interaction, whereas the audio engine (*lwae*) is responsible for the audio. I have tried as best as possible to follow model-view-controller paradigms where possible. If the source files, in addition to the *AGScript* file, are retained, it would be possible to utilise these to recreate the output soundfiles.

3.9.8 Limitations and future work

The requirements for a NT program are that it provides NRT and RT functionality, in such a way that it does not constitute a context shift resulting in world-replacement. It should produce data in line with NRT, but provide RT-G and RT-C.

Chronon and *lwae* come close in this regard. A longer study of the use of a NT program such as this would be necessary to understand more of its limitations, but it is a promising start. I have found that abstracting the RT/NRT paradigms behind the NT facade has led me to focus less on how the software forces me to work, and more on how I wish to work with it. It is important to note though that there are many aspects that could be improved. Having said this, *Chronon* serves as a proof-of-concept—an initial implementation. It does not provide all the functionality a composer might desire, but it was not created for that purpose.

Future work would include the addition of undo to *Chronon*, and more processes for *lwae*, to name a few. Additionally, it would be interesting to investigate the addition of a sequencing engine to facilitate

the creation of more complex sound objects, similar to composites in *SOClib*. With regards to NT, this research has outlined the requirements for such a system, and demonstrated its possibility.

Perhaps the most exciting avenue for the *NT ideal* is the further study of composition. By capturing all data, NT systems could be used in studies to analyse not only what composers used in their work, but might also provide insights into where composers chose not to investigate further— particularly if a sequencing engine was incorporated.

3.10 Conclusion

Software has historically been viewed as a tool for composition. However, on more investigation, we have discovered this not to be the case. The misconception has been perpetuated by the use of software in instrumental composition, where it serves as a mere number-crunching device for generating combinations of notes and rhythms. The impact of software has largely been ignored, either by sheer dismissal or with research based on an assumption of prescriptive and formulaic approaches to composition, as with HCI research.

Software traditionally ascribes to either RT or NRT operational paradigms. The definition of RT shifted, seemingly largely unnoticed, to focus on interaction rather than generation. This research has proposed the Domino Test as a way of distinguishing which paradigm a particular piece of software best fits. Upon discovering which paradigm, a series of investigations were devised in order to discover the impact. RT and NRT software do not affect the sonic entity produced. This realisation is either unsurprising or latent, but leads to more important questions: if it does not affect sound, then what does it affect? The difference between them is one of interaction, certainly, but how does that manifest itself in composition? Perhaps most importantly of all, does this difference matter?

This chapter has proposed that we interact with RT software in an improvisational way, and as a result this forces composers to adopt a subtractive method. In contrast, NRT software imposes a more contemplative, iterative approach, that produces smaller more finite sound objects, enforcing an additive method.

When I, as a composer, approach a musical problem, I am almost immediately confronted with software. In the event that it is a musical problem I have faced before, even though I cannot predict the actual sonic output, the paradigm of the software becomes less important; the software that helped solve it can be used to produce the anticipated output. Alternatively, if I'm not sure, I am often more inclined to investigate with RT software initially. My uncertainty manifests itself as an improvisatory need, and

the instant sensory feedback loop provided by RT software helps to quickly discover ways that both do and do not work in the particular situation. Rather than a journey with no destination, it is perhaps better to think of any destination being acceptable, navigating in response to the situation at hand: situated action. However, as these RT experiments continue, I begin to discover the combinations of parameters that produce the most interesting output. At this point, I will switch to a more NRT approach, and in the case of RT software will attempt a RT-NRT modification if possible.

The concept of narrowing down possibilities through RT experimentation, resulting in NRT, or NRT-like approaches has not been discussed in other work. More often than not, RT is held as the gold-standard; however, this is most likely due to the promise of instant gratification. We have seen that RT shifted meaning, focusing on interaction rather than generation, but also that RT interaction is by definition somewhat mythical. A RT computer is an impossibility—we strive instead for a fast-enough approach, where the computer can respond within our reaction times and appear to be RT. When we communicate with a computer in RT, we are really saying, “I wish that *had* happened”. The computer is always catching up. When we look at the development of RT audio software, we see the ring-buffer. In the context of RT control, this is a queue of commands saying “the user wanted this to happen”. In more musical terminology, a ring-buffer is the core of a delay-line, so we see that RT is merely a delayed response to user input, with the hope of being fast enough that we do not notice. CPUs can do branch prediction, which tries to guess the branch of a conditional expression, but that cannot predict user input (and even if it could, it could not guess what the input would be).

As demonstrated, software not only informs my approach, but dictates my method. We have seen that provided the underlying audio processing algorithm is the same, RT and NRT software, when given the same parameter values, will produce identical audio output. As a result, it is clear that the impact of software lies not in the audio, but elsewhere: how I work with it, and by extension how I work with and think about my sounds. The operational paradigms of RT and NRT result in polar opposite compositional activities: improvisation with subtraction and iteration with addition. From existing work, we know that this is not necessarily a negative aspect, as divergent thought processes can be vital for creativity, but it is not clear whether this is something I must consider, even passively.

NT has been presented as a potential ideal when it comes to the design of software, inspired by the notion that the operational paradigm is not something actively considered by the composer. To be considered NT, software must provide RT-C and RT-G, as well as NRT processing. It must provide the necessary data to recreate any sound it has generated. Parameters should be switchable, during RT-G, between RT-C or NRT-style breakpoint control. In addition, the act of switching should not constitute a

world-repair context shift and should ideally also avoid world-replacement. Whether NT is operating as RT or NRT, the user should not be actively considering which paradigm is in use at any given moment. The idea is to provide paradigm shifts that do not represent a challenge to the user: the reproducibility of NRT combined with the interactivity of RT.

CHAPTER 4

Software in Performance

In Chapter 2, it was identified that performance had a role in composition. This chapter will briefly outline how acousmatic music is performed, highlighting the software and technology involved. It presents the idea that performance is an embedded need in acousmatic composition, and investigates the use of performance systems in the composition of multi-channel acousmatic music. It presents the software and techniques used in the creation of a multi-channel work based on these ideas.

4.1 Performing acousmatic music

The performance of acousmatic music is known as sound diffusion. The fixed-media is performed over multiple loudspeakers distributed within a performance space. The performer, most often the composer themselves, will use a system designed to allow the sound to be manipulated throughout the space in real-time during the performance.

In 1999, Jonty Harrison outlined the BEAST style of diffusion (Harrison, 1999). He suggested that sound diffusion can serve as a way to present a piece written for a certain number of channels in a larger venue than that in which it was written. The system proposed by Harrison utilises additional loudspeakers, at least eight, positioned around the space in an extension of stereo. The goal, as noted by Harrison, is not necessarily to move sound dramatically around a space, but rather to provide a changeable focal point of sound to ensure that each listener can experience the sound in the space equally well. The spatial information written by the composer, for example in stereo, may not necessarily translate well into a

given venue. Diffusion offers this required control and redistribution within the venue. It would appear that Harrison's view of the performer in this context is a form of live-mastering engineer. The loudspeaker configurations as proposed here have had a lasting effect on the performance of electroacoustic music in the UK, with most concert loudspeaker arrays seemingly being arranged in-line with Harrison's recommendations[1].

4.2 Hardware-based and software-based

From what can be seen of diffusion systems, at least in the UK, there appear to be two approaches: focusing on either hardware or software. What I am considering a hardware system is one based around a traditional mixing desk: either analogue or digital. These are usually fixed installations, and appear to be designed in conjunction with the venue itself; as seen with the likes of the Sonic Laboratory at the Sonic Arts Research Centre (SARC) in Belfast, based around the Studer Vista 5 console (Queen's University of Belfast, 2019).

Whilst these systems have the benefit of often being custom designed for the venue, they pose a few challenges. Firstly, that of cost. Taking SARC as an example, prices for the console itself can start from around £100,000[2]. Although it would be possible to reduce the cost of a hardware system, the second difficulty with this approach is that of permanence inside a venue. For mobile rigs, or even those with no fixed venue, like those at Birmingham or Sheffield, it becomes difficult to design the system, let alone transport it. Additionally, hardware systems tend to be difficult or costly to expand later on as they are specified for the installation. Having said this, an expansion might also be made difficult if the space is fully equipped in the first instance and as a result has no additional space to expand.

Alternatively, it is possible to make use of a software-based system. In contrast to hardware-based systems, the software approach can be independent of specific hardware. It requires: a computer running specialised software, a soundcard with the required number of outputs, and some sort of hardware control surface with MIDI or OSC. In terms of cost, compared to hardware-based systems it is significantly cheaper. Additionally, it most often would not require the replacement of the entire system in order to expand it later on—by replacing the soundcard, or adding more DACs depending on the situation. The software that runs on it can also easily be modified or updated—providing its source is available. It is most likely for these reasons that many institutions have opted for software-based systems. Software-

[1] For more discussion of diffusion, see Austin (2000, 2001), MacDonald (1995)
[2] Rough estimate provided by HHB Communications Ltd.

based diffusion offers the most flexibility at the least cost, making it an ideal choice. As mentioned, there are several systems already in existence. It is interesting here to note that BEAST transitioned to a software-based system (Wilson & Harrison, 2010). Of those that utilise software, each institution in the UK appears to have its own in-house system: University of Birmingham has BEAST (Wilson & Harrison, 2010), University of Manchester has MANTIS, University of Sheffield had M2. It remains to be seen whether this is related to how applicable the system is, the openness and availability of the software, or simply Not-Invented-Here Syndrome (Antons & Piller, 2015).

Although there are systems available, they are either closed-source (M2), platform-specific (BEAST, M2), have limited channel counts, or are overly complicated and difficult to learn. If we were to design a new system, it should be: agnostic to channel counts, both input and output; open-source, so that one might learn from and adapt the system to our needs; and most importantly cross-platform, so that it is not hardware-specific. These, alongside the expectations mentioned previously, were the main influencing factors in the development of *SuperDiffuse*.

4.3 SuperDiffuse

SuperDiffuse is an n-channel diffusion system for the performance of acousmatic music. It is written in SuperCollider and distributed as a Quark. It was designed with the goal of running the Sound Junction concert series at The University of Sheffield, with large-scale public concerts twice per year. Its predecessor at TUoS was M2, written by Dave Moore, to which much is owed. As it was designed to run concerts, a concert was the starting point of the design. Each concert is made up of pieces, so there must be a way of adding, removing and swapping pieces. Each piece might be made up of any number of channels of audio, and each piece might need its own, per-channel, proportional distribution between the loudspeakers. Each loudspeaker must be able to be controlled from a physical input device, most likely a fader over MIDI or OSC. In cases where there are not enough physical controls to control each loudspeaker individually, it must also be possible to control multiple loudspeakers from a single control fader.

If the goal is to run a series of concerts, then speed and simplicity are key. This is exemplified most when the venues are not permanent. At Sheffield, the 32-channel speaker array is only set up in the venue for the duration of the festival (2-3 days). Therefore, time is a vital factor. Performers may only have 10 minutes to rehearse their piece, which is often the duration of the piece itself. The system must not be an obstruction here. The more features a system has, the more complicated its set up becomes. Once the

speakers are connected, it must begin to produce sound. Each year, the speakers may be connected to different outputs on the system, and speakers may be added, removed, or swapped; so it is not possible to have the same profile for each concert. Each feature of the system must incur a minimal lead-time. If pieces can be added to a concert in advance, and the output layout is also known, then the first two steps of the process can be removed, further reducing lead-time. From starting SuperCollider, the creation of a 32-channel concert (adding a piece, configuring routing, and MIDI configuration) can take as little as 2 minutes[3].

SuperDiffuse uses a model-view-controller (MVC) framework to communicate between the data layer (the Concert itself), and the GUI that appears. By design, it is entirely possible to create a concert without the GUI and control everything from SuperCollider's language.

The version of *SuperDiffuse* submitted as part of this thesis is fully documented in the SuperCollider Help System. The following sections present a more technical outline of the system.

4.3.1 System Design

The overall design of *SuperDiffuse* is quite simple. There are two audio buses: input and output. There is a single control bus, where the external controller values are written. There are a number of groups to ensure that processing happens in the correct order. Input Group, Input FX Group, Patcher Group, Output FX Group, Output Group.

A playback synth, streaming audio from a soundfile on disk, writes to the input bus. An output synth reads from the output bus and multiplies it by the control bus values (representing the amplitude values), writing to the soundcard. There is no direct connection between input and output buses. The only way to connect is by creating a series of `SD_Patcher` synths. These are mono synths that can read a single channel from the input bus and write it to a single channel on the output bus. As a result, there are no limits to the routing capabilities—any input can be routed to any output in any combination. When a matrix point is added, a patcher is created between the corresponding input and output channels of the respective buses. Each patcher synth has its own amplitude, which is taken from the value in the matrix, acting as a DCA—this is what facilitates proportional routing.

The use of a routing matrix was inspired by M2[4] (D. Moore, 2004). Alternative panning/routing systems, like VBAP (Pulkki, 1997) or ambisonics (Austin, 2001), require that the system knows the position of the speakers in the space. Given that *SuperDiffuse* was designed to be independent of specific

[3] It should be admitted that my familiarity with the system is a factor here, but it is possible to turn up at a venue and create the concert in a short space of time—this has been the case for several smaller concerts in Sheffield.

[4] *SuperDiffuse* does not use the same control system as M2, however.

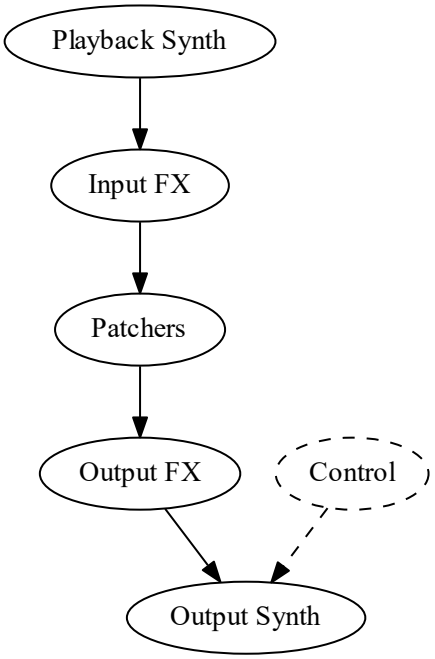


Figure 4.1: *SuperDiffuse* signal path

hardware, such a requirement was unsuitable. The proportional routing can still be used to create phantom images, but it is up to the user to specify the amplitudes directly.

Before the patcher synths read from the input bus, the `Input FX` are applied. This is a group set aside to perform in-place manipulations on the input bus. Synths added to this group should read from the input bus and `ReplaceOut` (replacing the audio on the bus). The group ensures that this happens *before* the routing. Once the patcher has written its audio to the output bus, the next stage is the Output FX group. Similar to the Input FX, this allows post-routing processing to take place. Following this, the output synth applies the amplitudes as defined by the control system and the audio is sent to the loudspeakers. The signal path is outlined in Figure 4.1.

The control system, whilst not being as sophisticated as M2, is still well suited to diffusion. There are master faders[5] that control the real level of a single output channel, but *SuperDiffuse* uses an abstraction layer to offer configuration possibilities. The graphical sliders presented in the GUI are actually control faders[6]. These communicate with the master faders using the Observer Pattern (Gamma et al., 1995, pp. 293–304). A master fader takes its value from a control fader—it *subscribes* to the value of the control fader. This way, a control fader can have any number of master faders attached to it; thus allowing for a single control fader to set the values for multiple master faders. Any time the control fader changes its

[5] `SD_MasterFader`
[6] `SD_ControlFader`

value, it notifies all of the master faders that are attached to it and each master fader responds by setting the real value of the output level to which it corresponds.

4.3.2 Future improvements

SuperDiffuse has, at the time of writing, reached version 1.4.0. It has been used in venues across Sheffield to facilitate in excess of 25 public concerts, ranging from 8 to 32 channels. It is, however, neither perfect nor finished. Firstly, speed is a great concern. While most pieces load within a reasonable time, the speed at which the waveforms are drawn becomes troubling in larger pieces—in duration, or channel-count, or worse, both. This has the potential to stall a concert. So far, given that the concerts at Sheffield contain pieces from a variety of composers, the applause of the audience between pieces is enough to cover the transition. However, in feature-concerts, with a single special guest composer, there have been several instances where I have been asked if it is possible to have pieces run into each other. This is firstly not possible given the piece selection mechanism, but it is also made potentially worse by having a large enough pause between loading pieces. In order to address this, it would be necessary to write a custom SoundFileView that could store waveform information to disk, thus avoiding the cost of calculating it each time the piece loads.

In terms of technical concerns, there is currently no way to have pieces of mixed sample rate within a single concert. This would involve restarting the server at the new rate, which would in turn require the entire concert to be restored in order to recreate the synths, groups, etc. Unfortunately there is no clear way to eloquently achieve this within SuperCollider itself. Therefore, I would need to implement a custom audio engine to allow this behaviour. As a result, work on such a standalone version, without the need for SuperCollider, is under way.

The most requested feature has so far been the addition of live input. This was not originally added as it was not required for the events at Sheffield, but it would allow a broader range of performances, not just fixed-media music. There are several solutions, but the best is as yet undecided. A future version of the software will allow for this, and will probably necessitate the implementation of Mooney's *coherent audio source sets* (Mooney, 2005; Mooney & Moore, 2008). This would allow routing based on combinations of different sources (live input or soundfile playback), to be sent through the routing matrix. An implementation is outlined in Figure 4.2. The control could be simplified by having, as with M2, the control embedded within the routing matrix, with faders controlling the matrix itself, rather than the input and/or output of the matrix.

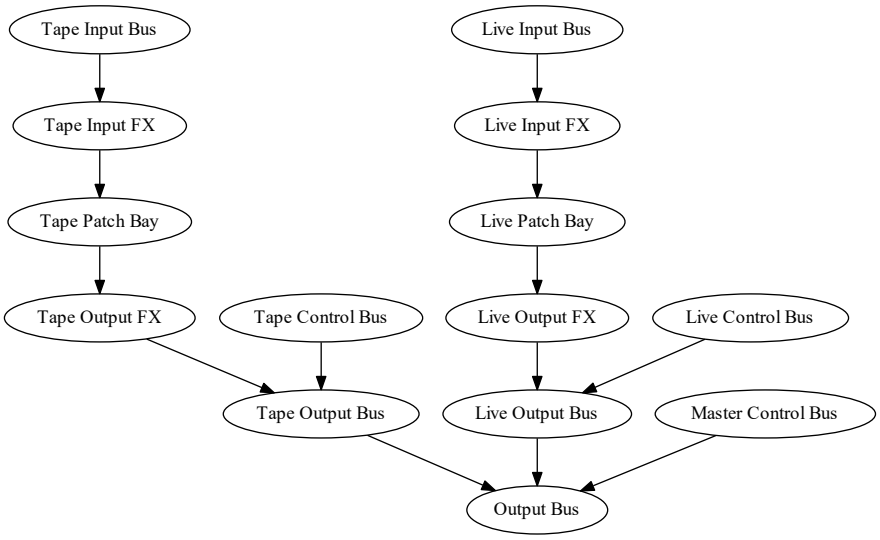


Figure 4.2: Future *SuperDiffuse* signal path

It would also be advantageous, as has been suggested by others, to have VCA-like faders. In this case they would really be DCA (digitally controlled amplifier), but the principle remains the same: a fader that controls others relative to their own position, similar to a group fader. These were initially not implemented because of the complexity of configuring them, adding additional start-up time to the overall system. However, there are ways to implement this that could minimise the impact, but significant development time would be needed to ensure it functioned correctly.

There are also additional effect synths that it would be useful to have. For example, being able to delay specific speakers (as an output effect), would be useful for compensating for distant loudspeakers. Additionally, the likes of phase decorrelation would prove useful to avoid cancellation within certain spaces (Kendall & Cabrera, 2011).

In order for *SuperDiffuse* to be able to suit all electroacoustic concert types, it would be necessary to have video playback to facilitate mixed-media works. In almost all cases, it would be possible to utilise the live input system to accommodate this, having a separate system feeding both the video and audio, going to the projector and *SuperDiffuse* respectively. However, in cases of fixed, mixed-media (video and tape, for example), it would be useful to have a video playback engine within the diffusion system. This would allow the audio to be diffused while outputting the video, without the need for external synchronisation (pressing play on two devices). In its current state, SuperCollider does not support video playback, and so it would not be possible to implement. However, I plan to investigate the incorporation of video playback into the standalone version.

In conclusion, *SuperDiffuse* has been largely successful as a performance system. It has been demon-

strated in a large number of public concerts, and has received a great deal of feedback and interest from many of those that have used it. Much of the feedback has in fact already been incorporated, but there is still much that can be done. As more features are added, and to make it more distributable, it seems that the standalone version is the correct direction.

The SuperCollider version did provide the inspiration for the work to follow, however. *SuperDiffuse* was written to facilitate performances. We found that composers had more time to rehearse and reflect on the spatial realisation of their works. As I observed this, I had the opportunity to reflect on the future development of the software. I realised that because it was written in SuperCollider, it was already capable of capturing an audio recording of the performance without the need to modify the code. This was initially exciting, as the piece could be performed in the same space, over the same system, after the initial performance. I was not considering this a replacement of performance practice, but was interested in the idea of preserving performances that could be later analysed—something that may in the future result in historically informed performance practices for diffusion. However, I also realised that the system could be taken into the studio and used to create multi-channel soundfiles. In a studio context the creation of multi-channel audio recordings, a by-product of using SuperCollider, could provide another approach to creating multi-channel source materials for use in multi-channel compositions.

4.4 Multi-channel composition

A. Moore (2016, pp. 188–201) echoes Harrison's thoughts on performance, but goes on to expand on some practicalities when it comes to multi-channel compositions, incorporating both Smalley's idea of *canopies* (Smalley, 1997) and the ideas of *fractured* composition (A. Moore, 2008). Here, Moore suggests that splitting sounds into separate stems for combinations of live triggered sounds as well as fixed-reproduction and diffusion would allow more complex and potentially rewarding compositions. This idea appears to have been explored by a few composers, as noted by Wilson and Harrison (2010).

Peters et al. (2011) conducted a large survey of the technologies used in the composition of spatial music. Their findings are too many to document here, but it is worth drawing particular attention to a few of the most relevant here. Spatialisation tends to be primarily used for concert contexts, with composers aiming to "enhance the listening experience", use spatial aspects "as a paradigm for artistic expression", or "organi[s]e and structure sounds". Crucially, they discovered that "more than 50 percent of the composers 'never' use, or do not have access to, seven- or eight-loudspeaker configurations.". Of the venues for performance, "acoustical conditions", "technical limitations of the venue", "time constraints", and

"non-ideal loudspeaker and audience location" were reported as the main challenges faced in venues. It is perhaps for these reasons that some of the more complex, and perhaps more accurate, spatialisation techniques are simply not used.

Wishart (1996, pp. 191–235) documents a variety of potential spatial motions that one might aim to utilise. He also discusses the aesthetics of spatial motion and how they might serve musical purposes. Many of the movements proposed by Wishart are based on the use of geometric patterns within the listening space (or frame). It has been shown that a lot of material can be observed to not willingly translate to arbitrary motions in a studio, so musical consideration is of key importance. The reasons for this may have something to do with the speaker arrangements as proposed by Harrison, although it appears likely to be more greatly rooted in psychoacoustic phenomena (some of which are related to the nature of loudspeakers in a space), as noted by Kendall and Cabrera (2011).

Multi-channel composition is something I have never investigated, for two main reasons. Firstly, as has been pointed out by others, increasing the channel count also changes the nature of the performance. I have always greatly enjoyed performing stereo pieces in concert, and the benefits of specific positioning of individual materials never outweighed the aspects of diffusion that I liked most. Secondly, it is unnecessarily complicated. The point at which I might apply spatialisation would be while working in a DAW. The DAW has different requirements when it comes to multi-channel audio—often referred to as surround sound, after the cinematic heritage of the supported layouts. Unless I am writing in a standardised speaker layout[7] (and, crucially, BEAST-style eight is not a standardised layout), it is very difficult. The reason for utilising a DAW here has been, as with other materials, I thought of spatial changes as trans-object which may need to be modified later if their relationships change with respect to time.

4.5 Performance in composition

For much of the last 60 years, composition and performance have been seen as separate activities. However, in acousmatic music, we see a trend where the composer is also the performer. To answer the question of why this is the case, we must look again at what this composition is in its physical, or rather digital, form. We do not wish to discuss the concept of *work* here, but rather the medium through which performance is enabled—a sound-file on a computer hard-drive, a CD in a player. The information contained therein is digital—binary digits. Whether CD or digital file, what is stored is the stream of binary

[7] Cubase 10 Pro supports: LRCS, 5.0, 5.1, LRC, LRS, LRC+LFE, LRS+LFE, Quadro, LRCS+LFE, Quadro+LFE, 6.0 Cine, 6.0 Music

digits that represent the discretised fluctuations in amplitude, that when converted to a continuous voltage represents the movement of a speaker cone, that when detected by the human hearing system may or may not be perceived as sound. What we realise is that until this music is performed, it does not exist in any tangible form. Unlike notated music that is stored in human-readable form, fixed-media music can only be read by machine, whereas instrumental music might be able to be read from a score, producing a rendition in one's mind. It is only through the mediation of technology that acousmatic works become perceivable as music. We can therefore conclude that this music has an embedded need to be performed, in a different sense than instrumental music, lest it remain this intangible stream of binary digits. This embedded need can be seen as why the composer is most often the performer. The performance can be viewed as the final stage of composition. This is not to say that a performer of acousmatic music cannot exist, but rather explains why composition and performance are so strongly linked in acousmatic music. Given the embedded need for performance, we can conclude that performance must play a large part in the composition itself. Indeed, we have already seen that notions of performance influence certain compositional decisions. This notion builds upon that which we have seen before, and I am proposing that composition and performance are, at least in this context, part of the same process. Our question becomes how then might it influence decisions that are not wholly based on the practicalities of a given venue—each venue will require different performance techniques, but we are instead interested in decisions that one might make irrespective of the venue, for example, some venues may have loudspeakers at height but the choice to use them will be specific to that venue during a performance, rather than something to necessarily consider whilst composing.

In their article, Wilson and Harrison (2010) are primarily concerned with presenting the idea of n-channel pieces, as opposed to fixed channel-counts. In doing this, we see the diffusion system become more of a live performance system that can trigger and position sounds dynamically during performance. This is an exciting idea, and one I hope to explore, but for now I am focusing on composition for a known number of speakers.

Knowing how a sound ought to be positioned amongst loudspeakers is only part of the solution. We must realise it by the utilisation of software, placing the sound in the desired position. As we have seen with other software, the mouse is the primary input device. It represents a single, point-source on a two-dimensional plane. A majority of software uses this point-source as the positioning of a sound source within a top-down view of a speaker array. As such, they require the layout of the speakers to be known, and as we have seen this cannot always be the case. Additionally, an issue arises when movements between arbitrary sets of loudspeakers are desired, as a point-source cannot represent this.

I am considering these behaviours whilst composing in the first instance, not just during performance. These thoughts are echoed by Smalley (Austin, 2000), among others (MacDonald, 1995). As two mutually inclusive activities, it is therefore important to consider the role that performance plays in composition.

Considering space as a compositional element that is explored through performance, it is also important to consider multi-channel composition. We have already looked at software in composition, but we must also ask what role performance software plays in composition. Having looked at the design of SuperDiffuse, we can use it to investigate.

As a composer, I have learned how sound behaves while moving through space through my experiences as a performer. When composing for two channels, a standard pan control makes absolute sense: left and right. However, when composing in multiple channels this point-based system has always eluded me, mostly because the commonly available point-based panning does not accommodate the BEAST-style main-eight configuration. Using such a system, I am able to position a point-source within the speaker-field, and move it as such. This might be suitable for certain kinds of materials, but how might I implement more complex movement that is not based on point-sources? It is incredibly difficult to position a sound at two points and move them independently; for example, in diffusion it is possible to position the sound on a pair of distant loudspeakers while also moving around a set of more nearfield loudspeakers which, instead of functioning as panning, serve as two independently localisable scenes. It is even more difficult to have the same sound moving between specific sets of speakers that are not related in terms of a standard speaker configuration.

4.6 Under the Fractured Chestnut Tree

SuperDiffuse was created at the same time as UTSCT. As a result, multi-channel composition became an interest. I realised that if there was a way to utilise the diffusion system in the studio, then it should therefore be possible to encode spatialisation at the point of composition. Answering this question became the focus of this research—could a performance system could be utilised as a spatial composition system? This investigation differs from the work of Wilson and Harrison (2010) in that the stems would be used in the composition and the piece would remain fixed.

It was at this point that the composition of UTSCT began. As a result, from the beginning, I planned for this piece to be multi-channel. However, I felt that it was important to first confirm my suspicions about how the sounds would behave in space—discovering if my studio-based intuitions were reasonable.

Therefore, I proceeded to write the piece in stereo, as I normally would, with the intention of using the première of the piece to investigate how the sounds behaved in a concert setting. I planned to re-write the piece after this, creating the multi-channel version, incorporating what I had learned from the performance. The following represents my thoughts and experience of composing and performing in this way.

During the composition stage, several moments clearly stood out as spatialisation opportunities. Firstly, for the opening I wanted the environment to feel like it was being presented rather than a fully enveloping sound. The soundworld was to be observed as a foreign environment, as if projected on a screen, to avoid listeners considering themselves a part of the world itself. Therefore, I knew that I wanted to start the piece on a pair of loudspeakers. It is probably most common for this sort of material to be presented over a great number of channels to create such an environment in the space, but this was not an environment I wanted to place the audience *in*, but rather observe. As the granular environmental sounds enter, I knew that I wanted to use the spatialisation as a special effect to enhance the aspect of the untoward, having the sound swirl throughout the space. Secondly, the Department of Truth. This section is a small room, containing footsteps moving from outside to inside. I knew that I wanted to use the space to highlight this aspect, so I wanted to try to move from a larger group of speakers to a single pair. It was also clear that Room 101 needed to be loud enough to have the desired effect, so that, when it is swept away by the environmental material, it serves as respite.

The performance of the piece involved the use of *SuperDiffuse*, over the Sound Junction 32-channel rig, set up in Firth Hall. The first stage for me in performing any piece is to listen to each pair of loudspeakers within the space individually. I took a moment to do this, listening for spectral variations and sound localisation, for a number of sections. This was during the rehearsal slot allotted to me for the performance later that day. Having gained an understanding of the loudspeaker sets, I began to have full run-throughs of the piece, experimenting with what I consider to be scenes—sets of loudspeakers that suit certain sections of the piece. It is unclear whether this corresponds directly to the concept of *coherent loudspeaker sets* (CLS), as proposed by Mooney and Moore, which appears to have been discussing the configuration of the diffusion system itself rather than the act of diffusion (Mooney & Moore, 2008). However, the term seems an accurate description of my practice here. In this discussion, I am using the term CLS to describe a specific subset of loudspeakers within a larger array.

After the performance, I returned to the studio. Armed with the knowledge of the materials' behaviours in a larger concert space, which mostly corresponded to my studio-based intuitions, I began to

deconstruct UTSCT. Studio 6 of the *Soundhouse* features an 8.x configuration[8]. I started by recording a full stereo diffusion of the piece using that configuration, almost exactly as I had described earlier in this discussion. This was then used as the basis for the more detailed work to follow. This soundfile is provided in the accompanying sound examples to this thesis. Listening back to this, I knew that I wanted to make certain materials behave in more specific manners. I proceeded to create a set of stems for the piece, simply exporting the tracks from the Cubase session. This later proved problematic when it came to working with specific materials, as I often reuse the same tracks for different sound objects in an effort to reduce the session track-count, so figuring out which material was in which stem required some effort.

The design of *SuperDiffuse* provided a separation between the system itself and the GUI that users interact with. The `SuperDiffuse_Concert` is an object within the SuperCollider environment, and it provides direct access to most of its underlying features from within the SuperCollider language. The idea was to encode specific spatial information into the individual stereo/mono stems to produce multi-channel versions. Each stem was individually loaded as a piece into the system and its multi-channel result was recorded. The stems did not need to be the same length, but they did all have to start from the beginning of the piece. This was to ensure that upon reconstruction, the stems aligned and the piece itself remained unaltered by the process, other than the addition of spatialisation. For this to be possible, it was necessary to write an extension for *SuperDiffuse* that made the creation of diffused stems easier. It essentially provided a record button that would setup a multi-channel audio file, then trigger the playback of the given piece (i.e. stem) within *SuperDiffuse*. Provided each recording began at the beginning of the piece, the diffused stems would line up again.

The reconstruction of the piece was rather simple. I took the diffused stems, along with the stereo/mono stems I did not choose to spatialise, and added them together again to produce the final multi-channel piece. I refer to UTFCT as a spatial reconstruction, rather than a multi-channel work, as it is impossible to say that I would have made the same compositional choices had I been working in multi-channel from the beginning. What follows is a summary of the techniques I used during this process, and some recommendations for further investigation.

4.7 Techniques for studio-based diffusion

Of the writings I have seen, none appear to deal with the practicalities of diffusion, choosing instead to focus on aesthetics. Over the years, I have developed a set of techniques that I tend to resort to, which

[8] It also has an additional front-centre speaker to facilitate 5.x

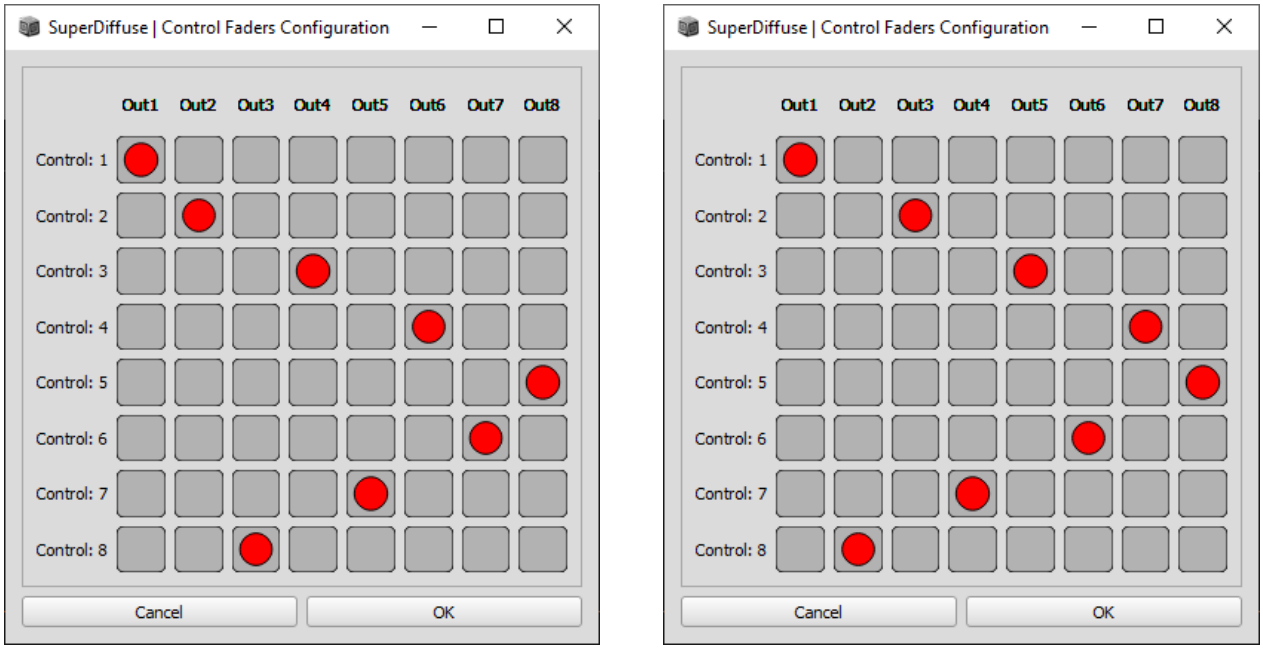
may help explain my choices. Some of what follows is passed from composer to composer anecdotally, but little appears documented.

Firstly, always bring something up before bringing something down. This goes for moving between loudspeakers. In my experience, sounds only appear to transition between speakers once the initial CLS is reduced. In addition to affecting the movement, this technique also guarantees sound is always produced; if attempting both at once, it is entirely possible to misjudge the crossover between the CLSs and have the level audibly drop. The worst case of this misjudgement is accidentally having all faders down, resulting in there being no audio at all.

When controlling larger numbers of speakers, in my experience, a natural instinct when an enveloping sound is desired would be to use many loudspeakers. However, in reality, given the phase alignments, this only serves to increase the overall volume, not the sense of immersion. If the system could decorrelate the signals going to each speaker, this might help to compensate, as described by Kendall and Cabrera (2011). Additionally, lower volumes allow the space to be articulated with the sound, rather than having the sound imposed on the space. All too often I have heard material played, simply too loudly. With many materials, this can cause distortion in the ear itself: most likely distortion product otoacoustic emissions (DPOE). The act of playing at a reduced volume can in most cases remedy this, although these concerns are mostly raised by performance rather than composition.

From my own experience, when composing in stereo, I must consider left and right. Translating this to performance, I have to consider this encoded left and right within the front-back and up-down fields of performance. Having performed in various venues over a number of years, I am always considering how certain materials will behave in a space. For example, consider a noise signal that pans from left to right over a given time. In performance, I can place that left-right anywhere front-back-up-down statically in the venue, and it is still a left-right movement. However, if I move the same sound from front to back, for example, this then becomes a diagonal movement across the space. The act of performance has entirely modified the spatial characteristics of the sound. Suppose then that I take a sound composed of sinusoidal material and do the same. Given the perception system of the human ear, the movement will not be so pronounced, if perceivable at all. If I then take a combination of the two sounds, if things align as such, it is entirely possible to split the material in the two channels across the space—having the sinusoidal material appear stationary, whilst the noise material moves diagonally across the space. Composition in stereo is not reduced to left and right. Through performance it becomes something altogether different.

Although not possible, in general, during performance, we can create motion between speakers in many of the patterns suggested by Wishart when using the diffusion system in a studio context. Many of



a) Clockwise b) Counter-clockwise

Figure 4.3: *SuperDiffuse* circular motion

these are made possible by having assignable faders, as with *SuperDiffuse*. Rather than having traditional mappings, we can use alternative fader assignments to create movement patterns, while maintaining traditional *wave* movement patterns with physical controls—having a learned physical motion map to different spatial motion. The configurations for circular motion (both clockwise and counter-clockwise) are shown in Figure 4.3. Other movements are possible through the proportional routing matrix, such as reducing a stereo set to a single speaker while moving through the space. In combination, a great variety of motion is possible. Diagonal movement, for example, can be achieved in a few ways. If a stereo signal pans from left to right, then the diffusion becomes a motion from front to back. Alternatively, in the case of a mono signal, it may only require routing the same sound to each loudspeaker, controlling one at a time. It would also be possible to route a stereo signal to CLSs, and have positional control on this level too. This particular approach would have the effect of rotating the listening frame, as outlined by Wishart.

So far, we have primarily considered stereo and mono materials as source for stem-based diffusion. Given that the diffusion system can handle any number of input channels, it would potentially be possible to create a single multi-channel file containing all the stems, using that to record spatialisation. This was briefly investigated, but disparity between the number of input channels and output channels made it too easy to accidentally remove entire stems if they were forgotten about. I found that it was much easier to work with smaller channel-count materials and build up the overall spatialisation from there—encoding spatialisation at a sound object level, first, rather than a structural one.

The creation of stems was somewhat arbitrary for UTFCT, being related to the tracks in the DAW. If spatial reconstruction were to be attempted again, it would perhaps be of greater benefit for an alternative creation process. Perhaps grouping materials by spectral characteristics would be best, as this is often how we might consider sounds when thinking of movement. It is important to note, however, that the creation of stems would only be necessary for spatial reconstruction. If the system was used throughout the composition itself, rather that afterwards, each sound object would itself be multi-channel from the beginning. At this point, a need for contextualised listening becomes apparent. One might not wish to hard-code spatial movement, as it might depend on its relation to other materials within the phrase or section. Therefore, a composition-focused diffusion system would be ideal. It should provide a way to audition one set of materials that are not affected by the control changes, whilst creating a spatialised recording of other, independent sounds.

Another feature, that was not utilised in the case of UTFCT, is the possibility of loudspeaker-specific processing. *SuperDiffuse* already provides this with its `FilterSets` for input and output, but as these are focused on a performance environment, they serve a functional purpose. The `SuperDiffuse_Concert` object in SuperCollider provides access to the underlying groups and buses. As such, it is possible to add any processing Synth to the Input and Output FX groups, on a channel-by-channel basis. In the studio, this could foreseeably be used creatively to have sounds be manipulated differently as they are moved throughout the space. For example, consider a per-loudspeaker filter, each with different settings—suddenly, a source material is given multi-channel spectral-split, dependent on its position in space. This is a rather simple example, but there are many more complicated avenues still left to explore.

4.8 Conclusion

The use of the performance system during composition has been shown to present an additional method for creating multi-channel source materials. Its main advantage is the utilisation of the knowledge-base, including that of physical controls, concerning the movement of sound in space that I have gained from performance experience. From the creation of UTFCT, it became clear that several features were missing from the software, and that a different approach had to be considered. The concept of spatially reconstructing a piece is one I find interesting, but I am more excited by the use of it to compose a piece from the start. The use of the diffusion system in the studio opens up a new world of possibilities to explore space-specific processing.

Taking into consideration the work of NT from the previous chapter, we can begin to see another future

development. If a diffusion system was written from a NT perspective, it could prove a considerable boon in studio-based composition of multi-channel music. Currently, *SuperDiffuse* only records audio, not control. This was initially by design, as automated performance is a divisive topic. However, in the context of composition, it becomes a requirement. The future standalone version of *SuperDiffuse* utilises *lwae* as its audio engine, and so it is already partly there. The difficulty is in deciding what the GUI becomes, as the needs of performance and composition are different in this respect. It would be possible to write the diffusion matrix as an `AudioGraphNode`, and have it reside within *Chronon*. The goal of my future work would be the creation of a multi-channel composition environment that utilises NT, based on the proportional routing and control system of *SuperDiffuse*.

CHAPTER 5

Conclusion and Future Work

This research has taken a different approach in trying to understand the influence of software in the composition of acousmatic music. Throughout, software and music have been created in a cyclic manner, with each informing the other, holistically viewing software as part of the process of composition and performance. It has presented the idea that software can make a composer work in diametrically opposed fashions, but that this may not present a concern in itself provided it is not ignored and attributed to some other factor.

Software, composition and performance represent very exciting areas of research. As mentioned, it is my hope to continue to develop the concept of No-Time, and incorporate the ideas of performance systems into composition.

As an autoethnographic study, the conclusions within this thesis are naturally limited to my own experiences. It is hoped, however, that most of the work presented here resonates with others and can inspire future work, and perhaps lead to more software informed by its use as part of the composition of acousmatic music.

The software included as part of this research is available under open-source licenses. The submission of this research does not constitute the completion of these programs, and they will continue to be developed and the focus of my future work.

Software has previously been considered the embodiment of compositional activities. This research has presented that by taking an approach that engages with composition, performance and computer

programming we can search for new knowledge and experience.

APPENDIX A

Accompanying Materials

A.1 Sound examples

The sound examples that accompany this thesis provide clarification of several of the key points made in the thesis. There is a folder of examples for each chapter, as labelled, made up of examples from the pieces submitted alongside the thesis. The NRT folder contains examples of the iterative, additive process. The RT folder contains an example of an entire RT output, and the moments that were selected for use in a piece. The Chapter 4 folder provides examples of the stem-based diffusion used in the creation of *Under the Fractured Chestnut Tree*, as well as some examples of specific spatial movements. The exact contents are outlined in the README file contained within the main Sound_Examples folder.

A.2 Software manuals

The user manuals for the software distributed with this thesis are contained with the software distributions themselves. In the case of *SuperDiffuse* its documentation resides within the SuperCollider help system, accessible from within SuperCollider.

Bibliography

Adkins, M. (2014). Nodalism and Creative Practice, In *Proceedings of the Conference of Computation, Communications, Aesthetics and X, 2014*. xCoAx.

Antons, D. & Piller, F. T. (2015). Opening the Block Box of "Not Invented Here": Attitudes, decision biases, and behavioural consequences. *Academy of Management Perspectives*, *29*(2), 193–217.

Auger, P. (1981). On Creativity and Discovery in the Fine Arts and in the Natural Sciences. *Leonardo*, *14*(2), 144–145.

Austin, L. (2000). Sound Diffusion in Composition and Performance: An Interview with Denis Smalley. *Computer Music Journal*, *24*(2), 10–21.

Austin, L. (2001). Sound Diffusion in Composition and Performance Practice II: An Interview with Ambrose Field. *Computer Music Journal*, *25*(4), 21–30.

Bayer, D. L. (1977). Real-Time Software for a Digital Music Synthesizer. *Computer Music Journal*, *1*(4), 22–23.

Bedworth, J. (1998). Computer music and human expression. *Digital Creativity*, *9*(2), 115–120.

Belet, B., Jaffe, D., Brümmer, L., Maverick, V. & Spiceley, A. (1992). The Composer and the Computer. *Computer Music Journal*, *16*(2), 7–10.

Bertelsen, O., Breinbjerg, M. & Pold, S. (2009). Emerging Materiality: Reflections on Creative Use of Software in Electronic Music Composition. *Leonardo*, *42*(3), 197–202.

Boden, M. (1995). Creativity and Unpredictability. *SEHR: Constructions of the Mind*, *4*(2).

Boden, M. (2004). The Creative Mind: Myths and Mechanisms.

Boulez, P. (1986). Technology and the Composer. In S. Emmerson (Ed.), *The Language of Electroacoustic Music* (pp. 5–16). The Macmillan Press Ltd.

Burnard, P. (2012). Musical Creativities in Practice.

Chadabe, J. (1977). Some Reflections on the Nature of the Landscape within Which Computer Music Systems are Designed. *Computer Music Journal*, *1*(3), 5–11.

Charrieras, D. & Mouillot, F. (2016). Getting Out of the Black Box: analogising the use of computers in electronic music and sound art. *Organised Sound*, *20*(2), 191–199.

Clowes, M. (2000). *An investigation of compositional practices in the field of electro-acoustic music, with an evaluation of the main software environments currently in use* (Master's thesis). University of Sheffield.

Collins, N. (2012). Automatic Composition of Electroacoustic Art Music Utilizing Machine Listening. *Computer Music Journal*, *36*(3), 8–23.

Dawkins, R. (1976). *The Selfish Gene*. Oxford University Press.

Eaglestone, B., Ford, N., Brown, G. & Moore, A. (2007). Information systems and creativity: an empirical study. *Journal of Documentation*, *63*(4), 443–464.

Emmerson, S. (1986). The Relation of Language to Materials. In S. Emmerson (Ed.), *The Language of Electroacoustic Music* (pp. 17–40). The Macmillan Press Ltd.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Sotware*. Addison-Wesley.

Gruenwals, I. (1974). Mysticism, Science and Art. *Leonardo*, *7*(2), 123–130.

Harrison, J. (1999). Sound, space, sculpture: some thoughts on the 'what', 'how', and 'why' of sound diffusion. *Organised Sound*, *3*(2), 117–127.

Harvey, J. (1986). The Mirror of Ambiguity. In S. Emmerson (Ed.), *The Language of Electroacoustic Music* (pp. 175–190). The Macmillan Press Ltd.

Jaumotte, A. (1972). Art, Science and Technology. *Leonardo*, *5*(2), 165–168.

Karlplus, K. & Strong, A. (1983). Digital Syntheis of Plucked-String and Drum Timbres. *Computer Music Journal*, *7*(2), 43–55.

Keane, D. (1986). At the Threshold of an Aesthetic. In S. Emmerson (Ed.), *The Language of Electroacoustic Music* (pp. 97–118). The Macmillan Press Ltd.

Kendall, G. S. & Cabrera, A. (2011). Why Things Do not Work: What you need to know about spatial audio, In *Proceedings of the International Computer Music Conference 2011*. International Computer Music Association.

Laske, O. (1984). Definitions of Computer Art. *Computer Music Journal*, *8*(4), 9–11.

Laske, O. E. (1978). Considering Human Memory in Designing User Interfaces for Computer Music. *Computer Music Journal*, *2*(4), 39–45.

Lawson, J. & Mathews, M. (1977). Computer Program to Control a Digital Real-Time Synthesizer. *Computer Music Journal*, *1*(4), 16–21.

Lionnais, F. L., Agoston, G. & Bentley-Koffler, P. (1969). Science is an Art. *Leonardo*, *2*(1), 73–78.

Loeb, A. L. (1971). Structure and Patterns in Science and Art. *Leonardo*, *4*(4), 339–346.

MacDonald, A. (1995). Performance Practice in the Presentation of Electroacoustic Music. *Computer Music Journal*, *19*(4), 88–92.

Magnusson, T. (2011). Confessions of a Live Coder, In *Proceedings of the International Computer Music Conference 2011*. International Computer Music Association.

Magnusson, T. (2019). *Sonic Writing: Technologies of material, Symbolic and Signal Inscriptions*. Bloomsbury Academic.

Mathews, M. (1963). The Digital Computer as a Musical Instrument. *Science*.

McAlpine, K., Miranda, E. & Hoggar, S. (1999). Making Music with Algorithms. *Computer Music Journal*, *23*(2), 19–30.

McClary, S. (1989). Terminal Prestige: The Case of Avant-Garde Music Composition. *Cultural Critique*, *Spring*(12), 57–81.

McKerracher, A. (2016). Understanding Creativity, One Metaphor at a Time. *Creativity Research Journal*, *28*(4), 417–425.

McNabb, M. (1986). Computer Music: Some Aesthetic Considerations. In S. Emmerson (Ed.), *The Language of Electroacoustic Music* (pp. 141–153). The Macmillan Press Ltd.

Miranda, E. R. (2001). *Composing Music with Computers*. Focal Press.

Monro, G. (1997). This Is Art, Not Science. *Leonardo Music Journal*, *7*, 77–80.

Mooney, J. (2005). *Sound Diffusion Systems for the Live Performance of Electroacoustic Music* (Doctoral dissertation). University of Sheffield. Department of Music.

Mooney, J. & Moore, D. (2008). A concept-based system for the live diffusion of sound via multiple loudspeakers, In *Proceedings of the Digital Music Research Network,* Unpublished. Digital Music Research Network Conference 2008.

Moore, A. (2008). Fracturing the acousmatic: merging improvisation with disassembled acousmatic music, In *Proceedings of the International Computer Music Conference 2008*. International Computer Music Association.

Moore, A. (2016). *Sonic Art: An Introduction to Electroacoustic Music Composition*. Routledge.

Moore, A., Moore, D., Pearse, S. & Stansbie, A. (2013). Tracking Production: Identifying compositional methods in electroacoustic music. *Journal of Music, Technology & Education*, *6*(3), 323–336.

Moore, D. (2004). *Real-time Sound Spatialization, Software Design and Implementation* (Doctoral dissertation). University of Sheffield. Department of Music.

Nuhn, R., Eaglestone, B., Ford, N., Moore, A. & Brown, G. (2002). A Qualitative Analysis of Composers at Work, In *Proceedings of the International Computer Music Conference, 2002*. International Computer Music Association.

Peters, N., Marentakis, G. & McAdams, S. (2011). Current Technologies and Compositional Practices for Spatialization: A Qualitative and Quantitative Analysis. *Computer Music Journal*, *35*(1), 10–27.

Polfreman, R. (1999). A task analysis of music composition and its application to the development of Modalyser. *Organised Sound*, *4*(1), 31–43.

Pope, S. T. (1992). Editor's Notes: The Composer and the Computer. *Computer Music Journal*, *16*(2), 4.

Pope, S. T., Rahn, J., Cerana, C., Katayose, H., Pecquet, F. & Karpen. (1995). Touched by Machine?: Composition and Performance in the Digital Age. *Computer Music Journal*, *19*(3), 13–17.

Preusser, R. (1973). Relating Art to Science and Technology: An Educational Experiment at the Massachusetts Institute of Technology (M.I.T.) *Leonardo*, *6*(3), 199–206.

Pulkki, V. (1997). Virtual Sound Source Positioning Using Vector Base Amplitude Panning. *Journal of the Audio Engineering Society*, *45*(6), 456–466.

Queen's University of Belfast. (2019). *Sonic laboratory specifications*. Retrieved September 17, 2019, from http://www.sarc.qub.ac.uk/home/sarc/facilities/soniclab/

Smalley, D. (1997). Spectromorphology: explaining sound-shapes. *Organised Sound*, *2*(2), 107–126.

Spiegel, L. (1996). That was Then: This is Now. *Computer Music Journal*, *20*(1), 42–45.

Suchman, L. (1985). *Plans and Situated Actions: The problem of human-machine communication* (Doctoral dissertation). Palo Alto Research Centre.

Tracy, H. (2002). *An investigation of composers' attitudes towards the user interfaces of electro-acoustic compositional software and how to support the compositional process in a software environment* (Master's thesis). University of Sheffield.

Truax, B. (1977). The POD System of Interactive Composition Programs. *Computer Music Journal*, *1*(3), 30–39.

Truax, B. (1986). Computer Music Language Design and the Composing Process. In S. Emmerson (Ed.), *The Language of Electroacoustic Music* (pp. 155–173). The Macmillan Press Ltd.

Upton, C. (2004). *An investigation into the compositional processes of Electro-acoustic composers* (Master's thesis). University of Sheffield.

Vaggione, H. (2001). Some ontological remarks about music composition processes. *Computer Music Journal*, *25*(1), 54–61.

Weale, R. (2005). *The Intention/Reception Project: Investigating the relationship between composer intention and listener response in electroacoustic compositions* (Doctoral dissertation). De Montfort University, Leicester.

Wilson, S. & Harrison, J. (2010). Rethinking the BEAST: Recent developments in multichannel composition at Birmingham ElectroAcoustic Sound Theatre. *Organised Sound*, *15*(3), 239–250.

Wishart, T. (1994). *Audible Design*. Orpheus the Pantomime Ltd.

Wishart, T. (1996). *On Sonic Art*. Harwood Academic Publishers.

Wishart, T. (2012). *Sound Composition*. Orpheus the Pantomime Ltd.