_derivations and the Performer-Developer:

Co-Evolving Digital Artefacts and Human-Machine Performance Practices

Benjamin Carey

University of Technology, Sydney

Submitted to the Faculty of Arts and Social Sciences in Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy

2016

Certificate of Original Authorship

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Student:

Production Note:
Signature removed prior to publication.

Date: 9th of October, 2015

Acknowledgements

I have been humbled by the support, encouragement and enthusiasm shown by many people throughout the life of this project. Thanks to my co-supervisors Dr. Andrew Johnston and Dr. Jon Drummond for their guidance and encouragement, and most importantly for getting me over the finish line. My thanks also go to Dr. Kirsty Beilharz for her patience and guidance during the formative stages of my candidature. Thank you, Kirsty, for your timely advice, long discussions and for your trust in my abilities.

Thank you to all of the musicians who I have worked with throughout the project. Special thanks to my dear friend Joshua Hyde, for your continued enthusiasm and advocacy of the *_derivations* project, and your tireless encouragement of everything I do. Thanks also to Alana Blackburn, the first performer to take a step into the unknown with this project. Thanks for your friendship, enthusiasm and beautiful playing. Thanks to musicians Zubin Kanga, Antoine Läng, Evan Dorrian, Zane Banks, and Alexander Berne, all of whom I have had the good fortune of hearing improvise with *_derivations*. To the many people who have downloaded and performed with the *_derivations* software, my sincere thanks. It's humbling to have received comments, questions and links to performances and recordings made with *_derivations* from far-flung places across the globe.

To my many friends and colleagues who have helped keep me sane throughout the duration of this degree, my warmest thanks. A special thanks to Aengus Martin, Oliver Bown and Roger Mills for the many great discussions, jam sessions, gigs and hack-a-thons had over the life of this project. Thanks also to friends Lachlan, Jemima, Tristan, Mark, Smiddy and Tobias for your support and encouragement over these past few years.

To my beautiful family who I love dearly, thank you. Thanks to Mum and Dad for all the love, support, advice and encouragement. To my twin brother Nick, thanks for always being there, for keeping me sane, and for letting me vent from time to time. To Libby, thank you for being such a great listener and a ball of good energy. Thanks to Jenny, Roger and Lesley for your love and support. And lastly, to Zoé, who has endured all the ups and downs of this rollercoaster ride with me, thank you for always believing in me. I couldn't have done this without you.

Contents

List of Figures

Abstract

This thesis concerns the development and use of interactive performance systems designed for improvised musical performance. Written from the perspective of a performer-developer, the research traces the development of personal approaches to designing for musical interactivity in human-machine performance, culminating in the development of the *_derivations* interactive performance system and related creative outcomes.

The contributions and outcomes of this research project are as follows:

- The development of novel computer music techniques for use in interactive musical performance;

- A novel self-reflective study of the development and use of interactive musical performance systems from the perspective of a performer-developer;

- Theoretical perspectives on the design and use of interactive musical performance systems.

In addition to the published thesis, this research has generated significant creative outcomes in the form of software, studio recordings, documentation of live performances, video documentation and a publicly available website dedicated to the *_derivations* system. These creative outcomes are also presented as significant contributions of this research.

The creative practice underpinning this research is presented as a narrative of development, tracing advancements in the author's practice towards the stabilisation of the *_derivations* system and its accompanying performance practice. Designed for use by instrumental improvisers, *_derivations* uses live-sampling and timbral matching techniques to generate autonomous responses to the live performance of an improvising musician, engaging the performer in a playful, improvised musical dialogue. This thesis outlines both formative programming experiments and stabilised software artefacts, tracing the

author's creative practice to reveal the iterative and cyclical patterns of development engaged in throughout this research.

Employing a practice-based research approach, this project uses the creative practices of software programming and interactive musical performance to surface issues, concerns and interests concerning human-machine performance practice. A self-reflective methodology is employed to engage with emergent research themes arising throughout the development of my creative artefacts. The thesis concludes with three extended reflections-*on*-action that interrogate theoretical concerns relevant to the interactive computer music community. The first of these reflections addresses the relationship between human and material agencies in the practice of the performer-developer, whilst the second reflection interrogates the concept of musical interpretation in the context of human-machine performance. The final reflection proposes *symbiosis* as a novel interactive metaphor in the development of interactive musical systems.

Chapter 1. Introduction

1.1 Introduction

This thesis concerns the development and use of interactive performance systems designed for improvised musical performance. The research traces the development of personal approaches to designing for musical interactivity in improvised human-machine performance, culminating in the *_derivations* interactive performance system and related creative outcomes. Employing a self-reflective, practice-based methodology, this research situates creative programming practice and interactive musical performance as sites for investigating issues, concerns and interests related to improvised human-machine performance practice. By tracing the development of my creative practice, the research has sought to understand the design and use of such systems from a practitioner's perspective. *Reflection-in-practice* and *bricolage programming* methods have enabled the advancement of idiosyncratic interactive software and musical performances, while *reflection-on-practice* is used to uncover and examine theoretical issues surrounding these interrelated creative practices that are relevant to the wider research community.

The contributions and outcomes of this research project are as follows:

- The development of novel computer music techniques for use in interactive musical performance;

- A novel self-reflective study of the development and use of interactive musical performance systems from the perspective of a performer-developer;

- Theoretical perspectives on the design and use of interactive musical performance systems.

In addition to the findings outlined in this document, this practice-based research project has generated significant creative outcomes in the form of software, studio recordings, documentation of live performances, video documentation and a publicly available website. Accordingly, the discussion and analysis contained within this thesis should be

understood with respect to these creative outcomes. These outcomes are detailed in Appendices A – E and have been provided in digital form as part of the submission materials alongside this thesis.

1.2 The *_derivations* software

The focus of the creative practice outlined in this thesis concerns the development of the *_derivations* interactive performance system, a software program designed for use by instrumental improvisers in human-machine performance. The software uses live-sampling and timbral matching techniques in order to generate autonomous responses to the live performance of an improvising musician, engaging the performer in a playful, improvised musical dialogue. The development of this software system extends my personal interest in investigating musical interactivity in human-machine performance practice. The software is therefore embedded with my musical inclinations as an improvising saxophonist and electro-acoustic musician. The result of an iterative and emergent process of creative programming, this software was made freely available online in June 2013. Since this time, the system has been downloaded and used by numerous musicians worldwide and has appeared on several musical releases, including two releases that form part of the submission materials for this thesis (see Appendix B).

1.3 Performer-developer context

> "In the future, there should be more individuals in computer music who possess both high-level musicianship and technological skills. […] Naturally, performers are trained to handle their instruments very intimately, making them seem like extensions of their own bodies; mastering the craft of performing and handling the instrument seems difficult to achieve otherwise. As real-time computer music systems are closely connected with performance, I think that trained performers will be at an advantage in this field."
>
> (Kimura 1996)

Owing to the maturity and ubiquity of computer music environments such as Max[1], SuperCollider[2], Pure Data[3], ChucK[4] and others, performers seeking to expand their

[1] https://cycling74.com/products/max/

[2] http://supercollider.github.io/

[3] https://Pure Data.info/

practice to encompass digital technologies have a wide array of sophisticated tools and techniques at their disposal. Research and creative works presented to international fora such as the International Computer Music Conference, the Sound and Music Computing Conference and the Conference on New Interfaces for Musical Expression display the breadth of research in the computer music community, and highlight the inherently interdisciplinary nature of the field. In this area, practitioner-researchers are common and are often multi-skilled in areas such as musical composition, software development, interaction design and musical performance.

As articulated by violinist and computer musician Mari Kimura, the unique skillset and perspective possessed by instrumental performers can be a great advantage in the field of computer music. Given the inherently technical nature of computer music research and practice, musical performers seeking to engage with this field often extend their skillset to include creative software programming. Many performers have taken to augmenting their musical practices with the aid of flexible computer music environments, exploring personal methods of musical interactivity in both composed and improvised contexts. The immediacy of computer music tools such as Max make the environment ideally suited for use by performers seeking to develop their own performance systems and tools. In addition, such real-time environments make the process of programming, testing and refining personal software tools a liberating and autonomous process for the musician.

Whilst Kimura's perspective hinges on the benefits skilled performers can bring to the computer music community, in this thesis I consider the combined identity of the *performer-developer* as an integral part of advancing understandings of the new practices that are evolving in this field. In this research the notion of the *performer-developer* is used to describe my dual role as an instrumental performer and developer of interactive musical software. The unique relationship that develops between a performer and their emerging software makes this creative practice truly interactive. In this context, developers are themselves the ideal 'end-users' of the software being designed. As a result, the development process is cast as one of navigation and exploration over one of planning, design and execution. During the development process the decisions made by

---

[4] http://chuck.cs.princeton.edu/

performer-developers are not only inherently situated within the practical domain of their instrumental performance practice, but also with the emergent and cumulative interactive experiences gained through developing personal, idiosyncratic methods for interactive performance.

1.4 Background and context for the research

In 2006 I began developing a personal approach to electro-acoustic performance, extending my practice as a classically trained saxophonist with interests in experimental music. As an experienced interpreter of contemporary music, I developed an interest in so-called 'mixed music' works for instrument and tape, as well as the use of live electronics in contemporary musical performance. During 2006/07 I began to immerse myself in Sydney's improvised music scene, and it was here that I became intrigued by the highly unique and personal methods of musicians integrating live electronics into their performances. The work of local performers Jim Denley, Robbie Avenaim, Gail Priest, Peter Blamey, Dale Gorfinkel and Kusum Normoyle, as well as visiting musicians such as John Butcher, Oren Ambarchi and Kim Myhr inspired me to begin developing my own performance practice in this area.

Beginning with concepts of live sampling, looping and processing, I made use of Ableton Live with attached MIDI controllers (pedals, fader boxes, etc.) to sample, manipulate and structure recordings of my saxophone performance in real-time. These experiments culminated in solo-improvised performances for saxophone and laptop computer in which I played both the role of an instrumentalist and laptop musician simultaneously. While heightening my interest in developing an electro-acoustic performance practice, these initial explorations highlighted some practical and aesthetic challenges of solo improvised performance with live electronics. In these early performances a major issue was one of multi-tasking in performance, as I tried to maintain control over both my acoustic and digital performance simultaneously. Often I would play the saxophone solely in order to sample my sound for later manipulation, necessitating an alternation between playing the saxophone and physically manipulating recorded samples by hand. Aesthetically, these physical limitations restricted the structural variety of these improvisations and, from a performance point of view I found that playing two roles at once limited my potential for spontaneous improvisation.

In parallel with these experiments I also explored the phenomenon of saxophone-controlled audio feedback, inspired by the work of John Butcher as heard on the releases *Invisible Ear* (Butcher 2003) and *Cavern with Nightlife* (Butcher & Nakamura 2004). Using the tenor saxophone as a resonant chamber, I explored controlling the pitch and intensity of an audio feedback loop through the careful positioning of the instrument in proximity to a microphone, as well as through experimentation with unconventional fingering patterns. By using my instrument in this way, I developed an interest in the interactive possibilities of using an acoustic instrument to control and shape electronic sounds. Although limited in its ability to generate complex sonic results, the technique was satisfying due to the natural physicality of using the saxophone as an interface, and the emergent and surprising interactions set up by the delicate nature of audio feedback.

Given the unpredictability of a feedback loop, absolute control over the resultant electronic sounds was not always possible. This aspect became a large part of the performative focus in my improvisations using this technique, enabling a unique exploratory approach to interacting with electronically generated sound. My focus was less on complete control over the resultant sounds, but in navigating and exploring the delicate and emergent relationships between the instrument, microphone, fingering patterns and the unpredictable effects of room acoustics. My experiments in this area made use of both live and pre-programmed musical structures. In the semi-improvised work *m18d06* (2006) I made use of pre-programmed, timeline-based software automation to alter the input level of the microphone, as well as to control live sampling and triggered processing of the saxophone signal.[5] In contrast to the live sampling based approach described above, this approach mitigated the need for juggling between instrumental and laptop performance. However, the flexibility gained in performance was tempered by a rigid, timeline-based approach to structuring electronic signal processing in performance.

In the years 2007-2009 I gained experience presenting scored works for instrument and electronics. These 'mixed' works varied in the complexity of their integration of the performer with computer generated sound, ranging from traditional 'instrument and tape' compositions to more sophisticated works involving score-following and a variety

[5] A performance of this work can be streamed at the following URL: https://soundcloud.com/emeidos/m18d06

of fixed and live processed musical materials. As I soon discovered, this area of performance also comes with great logistical and aesthetic challenges with respect to synchronisation, room acoustics and musical interactivity.[6] From the perspective of my burgeoning electro-acoustic performance practice, I was particularly attuned to specific issues encountered in interpreting such works. For me, the reliance on a technical assistant (usually the composer) to present works in part diminished the autonomy of the performer in presenting the work in concert. I also became interested in the fine balance that existed between performer control and influence over electronic processes, and I began to look further towards surpassing the triggering and synchronisation issues encountered in my experience with mixed music works.

After reflecting on these previous experiences, I identified a number of important considerations that I sought to address in my electro-acoustic performance practice:

- ensuring physical performative freedom for the improvising instrumentalist
- relinquishing control over aspects of the performance to software systems
- enabling explorative interaction with electronic sounds
- maintaining variety in the timbral qualities of electronic sounds

In the trajectory of my creative practice outlined in this thesis, these initial themes guided my approach towards developing methods for engaging with the computer in improvised, human-machine performances. Identified through self-reflection, these themes represent the core issues, concerns and interests guiding the initial stages of the creative practice pursued throughout this project. Given my experience with managing both instrumental and electro-acoustic performance simultaneously, I sought to explore creative methods that could ensure a degree of freedom for myself as an improviser working with live electronics. Although ill defined at this early stage, the freedom I sought in my own practice was influenced by a preference for methods that avoided supplementing instrumental performance with external controller devices. In order to do this, I envisaged the development of forms of 'shared control' (Chadabe 1984) over electronic systems by implementing autonomous, algorithmic processes to control

[6] The unique challenges faced by performers interpreting 'mixed' electro-acoustic works have been described in depth by professional violinist and composer Mari Kimura (1996, 2004)

various levels of synthesis and processing in live performance. Such algorithmic processes may be guided by live performance, but also have the ability to influence the musician's performance in unpredictable ways. Similar to my previous experience with temperamental and often unpredictable feedback systems, I sought to develop software systems that could engage a musician in a dialogue with electronically produced sounds and structures. Such a playful, interactive relationship with a software system would encourage improvisation and dialogue between both human and machine in performance. Finally, in my practice I sought to explore these concerns through developing novel approaches to computer synthesis and processing in real-time systems. In the development of algorithms that could contribute autonomously to an improvised performance, I envisaged imbuing my systems with rich and complex timbral identities, as well as the ability to coherently integrate with an acoustic instrumentalist in performance.

Seeking to expand upon my nascent electro-acoustic methods, I began programming my own interactive software to explore novel ways of engaging with the computer in performance. Through developing these personal methods, I began to expand my understandings of interactive performance. However, whilst I began programming with a series of defined aims and goals, I soon discovered that the process of creative programming was much more than the implementation of pre-planned creative ideas. The programming of interactive musical systems is an emergent and iterative process. It relies upon creative decision making in response to challenges and opportunities that arise during programming and testing of interactive software artefacts. In the research presented in this thesis, this creative space became the site for engaging in *self-reflective practice*, posing questions about interactive musical performance and human-machine improvisation. Throughout this process, the above discussed concerns and interests surfaced broader issues surrounding my developing creative practice. Addressing these issues through practice and reflection both fed back into the practice itself, and also aided in the development of deeper theoretical contributions to the field at large.

1.5 Self-reflective practice

Given the tightly bound relationship between development and performance in this creative context, research that seeks to shed light on interactive musical performance

benefits from the tacit knowledge only accessible from inside the practical domain. In these practices, novel computer music techniques, modes of interactivity and understandings of practice are the result of specific artistic projects emergent technical and aesthetic concerns. For the performer-developer, programming and performing with interactive musical systems is an idea generating process, and a space for interrogating assumptions about performance practice and software development. Throughout these interdependent creative practices a complex relationship develops between personal aesthetics and theoretical concepts, and their encoding and refinement through software development and musical performance. While creative practices may be seen as primarily concerned with the creation and dissemination of artistic works, it is equally true that the process of artefact creation may in itself be considered a unique site for engaging in research. Therefore, this creative practice is ideal for enabling depth of insight into theoretical understandings of practice in the field of interactive computer music.

My unique position as a performer-developer working in this space is harnessed in this research to explore emergent research themes that have arisen throughout the development of the *_derivations* interactive performance system. This project has sought to examine researchable problems encountered in the plane of practice, relating emergent understandings of the practical domain to theory relevant to the computer music community at large. Although computer music practice is often associated with highly technical, software-driven research, I argue that the practice of developing novel computer music systems is above all a creative endeavour. The unique space in which these systems are developed, tested, refined and distributed provides rich and multi-layered knowledge about the state of the art in such burgeoning artistic practices. By examining software design and development as a creative practice engaged in by the digital arts practitioner, the complexity and novelty of finalised artefacts may therefore be understood as the result of an ongoing process of entanglement between theoretical concepts, material agency and musical performance practice. Through consideration of the trajectory of a developmental project, this thesis uncovers the entangled nature of software development, musical performance and theoretical understandings in this space.

Although the development of the *_derivations* software began within the context of my creative practice, the software is also envisaged for use by other instrumental improvisers. This aspect of the *_derivations* system evolved over time and affected certain

decisions during the software development process. As the software was distributed for use by third parties, considerable thought was put into interface design and documentation (see Appendix E). Facilitating meaningful interactions between instrumental musicians and software in computer music performance has been one of the driving forces behind my focus upon clear graphical user interface design, and various forms of documentation and communication related to the software. However, evaluating the usability and generalisability of this software was not the aim of this research. Instead, it was the complex and iterative process of design, testing and refinement of my personal interactive methods that focused my attention as an artist-researcher developing interactive software. Through a considered process of action and reflection, the development and use of my software artefacts has been used to uncover and examine researchable problems relevant to the field at large.

1.6 Structure of the thesis

Chapter 2 surveys the field of interactive music, outlining and critiquing key techniques and theoretical approaches. This chapter begins by examining some of the prevailing models and metaphors of interactive musical system design, as well as some more recent critiques and expansions of the notion of interactivity in human-machine performance practice. The chapter continues with a survey of specific design strategies, drawing upon examples from notable musical systems. Input analysis and segmentation approaches are outlined, and various generative strategies are discussed. Drawing from a number of examples of musical systems, the concept of sonic and algorithmic *derivation* is outlined as it applies to generative and interactive strategies of systems in the field. The notion of musical autonomy is then examined in relation to the concept of *Live Algorithms*, and appraised in the context of interactivity and mutual influence between humans and computer music systems. Finally, the relationship between rehearsal and performance is discussed in the context of human-machine performance, a comparatively under-researched concept in the field. Here I propose further consideration of this area in the design of interactive musical systems.

Chapter 3 outlines and justifies the methodological approach taken in the research. The chapter begins by outlining the significance of practice-based methods in the creative arts, positioning this project within recent methodological approaches to

research in this area. Scrivener's distinction between *problem-solving* and *creative-production* research projects is outlined, and Schön's notion of *reflective practice* is discussed to justify the self-reflective methods employed in this research. Turning to Andrew Pickering's *mangle of practice*, I consider the entangled relationship between human and material agencies as a core methodological consideration in the context of digital arts practice. Pickering's mangle is dissected and re-cast as a useful model for research in the creative arts, while McLean and Wiggin's concept of *bricolage programming* is outlined to situate the entanglement between human and material agency in the development of interactive software. The chapter concludes with an outline of the various data collection methods used throughout this practice-based research project.

Chapters 4, 5 and 6 detail the chronological development of the creative practice engaged in throughout the research. Titled *Wayfinding: Parts 1, 2* and *3*, these chapters outline the twists and turns of my creative trajectory, detailing technical achievements of developed software whilst outlining emergent issues, concerns and interests uncovered through *reflection-on-practice*. In Chapter 4 I outline formative approaches to designing for musical interactivity in my practice. Beginning with input segmentation and MIDI representation, the chapter outlines subsequent efforts at developing event-based methods musical interactivity and generativity, most notably with respect to Markov-based generative techniques. Alongside examples of successful software, this chapter also details tangential and ultimately unsuccessful lines of enquiry that served to solidify the creative trajectory of my subsequent programming practice.

In Chapter 5 I detail the shift in my programming practice towards live sampling based approaches to musical generativity, and the development of significant analysis/re-synthesis modules intended for use in interactive performance. The bottom-up development approach detailed in this chapter illustrates the emergent nature of my *bricolage* approach, culminating in the development of integrated systems built from existing components.

Emerging from the experiments discussed in the previous chapter, in Chapter 6 I detail the iterative development of the *_derivations* system. Throughout this chapter, increasingly specific programming challenges are encountered and reflected upon, uncovering the particularly emergent process of *interactive stabilisation* inherent in the

development of performer-developer devised software artefacts. The innovation of the core *phrase matching* algorithm of this system is discussed, alongside subsequent advancements in autonomous generativity (the *self-referencing* algorithm) and database management (*session databases*). The chapter concludes with reflections upon the use of the *_derivations* software in performance, and the evolution of this artefact from a personal performance environment to freely available and open-source end-user software.

Chapter 7 presents three sustained reflections upon issues, concerns and interests that have emerged from the development and use of human-machine performance systems. These reflections are presented as theoretical and analytical findings of the reflection-*on*-action undertaken in this research. The chapter engages critically and analytically with themes emerging from the process of reflection-*in*-action present in the iterative cycles of development, testing and performance outlined in Chapters 4, 5 and 6. The first reflection considers the complex and entangled notions of material agency and authorship in performer-developer devised creative artefacts. The development and use of such systems is considered in relation to concepts derived from actor-network theory, examining the complex relationship that exists between design and use of these artefacts. The second reflection interrogates the notion of musical *interpretation* as it concerns the context of improvised human-machine performance. Reflecting upon various modes of musical interpretation in performance, this section positions improvised human-machine performance as a mediated practice connecting developer, performer and software in an entangled relationship. With reference to the *_derivations* interactive performance system, the final reflection presents *symbiotic* musical interaction as a useful metaphor in the field of improvised human-machine performance. Considering mutual dependence as a desirable trait in the development of interactive musical systems, *symbiosis* is outlined as a speculative metaphor for musical interactivity in such burgeoning practices.

Finally, Chapter 8 summarises the theoretical and creative contributions of the research with reference to the contributions outlined in this chapter. Performances, collaborations and musical releases are also outlined, and the dissemination and ongoing use of the *_derivations* software is discussed with reference to submission materials detailed in Appendices A - E. To conclude, ongoing and future creative and research work is presented.

Chapter 2. Literature Review - Interactive Music Systems

2.1 Introduction

The creative practice pursued throughout this research is situated in the field of interactive computer music. This diverse field of research and practice concerns the development and use of bespoke musical systems used in live musical performance, with specific emphasis on the responsiveness, interactivity and generativity of such systems. This research is specifically concerned with the development of novel approaches to integrating interactive computer music software with improvised instrumental performance. These systems often contribute to live performances through a combination of advanced machine listening algorithms, sophisticated generative grammars and real-time sound synthesis and processing methods. In the field of computer music, interactive and generative systems have been approached from a variety of artistic and technical perspectives, and have emanated from an array of professional practices. Composers of electronic music make use of interactive approaches in the context of scored musical works, researchers develop autonomous improvising agents from models of improvised performance, and performers expand their practices by designing their own algorithms for interactive performance. Due to the nature of such real-time technologies, the boundaries between composers, researchers and performers are by now firmly blurred.

As noted in Section 1.3, it is the position of the *performer-developer* that frames the creative practice of this research. For such practitioners, software development is an integral part of the creative process. Designing bespoke software for one's own musical needs is a liberating ability for the performer working with live electronics in performance, and many performer-developers have taken to developing software that enables an interactive dialogue to develop between themselves and their computer music systems. In the following chapter I survey and critique key theories, techniques and approaches to designing for musical interactivity discussed in the literature on interactive music, with an emphasis on improvised human-machine performance. Throughout this chapter I also seek to highlight the unique and innovative work of work of performer-developers in this space.

2.2 Definitions, models and metaphors

Practitioners and theorists working in the area of interactive music have sought to define interactivity in musical software systems in various ways. A pioneer of interactive composition and performance, Joel Chadabe has described interactive composition as a two-step process involving first the programming and performance an interactive system. For Chadabe, the performance process is in itself a form of composition, as the composer/performer directs what he terms an 'intelligent musical instrument' (Chadabe 1984). Such instruments respond to a performer in ways that are complex and unpredictable, but are also grounded in the musical gestures provided by the performer interacting with the system. In Chadabe's work the process of developing interactive musical systems is also linked to the development of novel musical interfaces, such as the proximity sensitive antennas used in his seminal work *Solo* (1978).

In describing the diversity of approaches available to the interactive composer, Chadabe has provided some useful metaphors:

> sailing a boat on a windy day and through stormy seas
> the net complexity or the conversational model
> a powerful gesture expander
>
> (Chadabe 2005)

Without referencing traditional compositional and performance practices, Chadabe's metaphors emphasise the contrasts inherent in the various approaches to interactive musical composition. In the first metaphor, the author defines an interactive context in which the unpredictable and uncontrollable contributions of a musical system engage the performing in 'weathering' its atmospheric conditions. The system is untameable, and the performer must work with its idiosyncrasies in order to chart a course through dangerous waters. In the second metaphor, Chadabe defines an interactive model in which a two-way conversation between participant and system defines the musical interaction. This metaphor recognises an important middle ground between the a system's responsiveness, and its ability to instigate musical trajectories that remain largely unpredictable to a human performer. In the final metaphor, the author describes a type of system that seeks to expand the performed gestures of the human performer. This model acknowledges a

mode of interactive performance in which unidirectional control over the system's response characterises the interactive relationship between human and machine.

Other definitions and models of musical interactivity with computers have relied more specifically upon metaphors derived from traditional instrumental musical practices. In *Composing Interactive Music*, Winkler characterises interactive music as "…a music composition or improvisation where software interprets a live performance to affect music generated or modified by computers." (Winkler 2001, p. 4) Rowe has published widely over the past two decades on interactive music systems, and has defined such systems as "…those whose behaviour changes in response to musical input. Such responsiveness allows these systems to participate in live performance of both notated and improvised music." (Rowe 1992, p. 1)

In characterising the main features of interactive music systems, both Rowe and Winkler focus their attention on the ability of a system to analyse and respond to 'musical input' received from a performer. Various classifications and models are proposed by both authors to describe the way in which such systems interact with the performer via the analysis of this input. Rowe characterises systems as either *score-driven* or *performance-driven*, displaying *transformative*, *generative* or *sequenced* response types, and the author draws a distinction between systems that exhibit behaviour that either extends the performer's instrumental gestures (the *instrument paradigm*) or acts more like a virtual player (the *player paradigm*). (Rowe 1992, pp. 6-7) Winkler's interactive models attempt to draw links between the types of interaction afforded by such systems and the levels of control and influence inherent in established musical contexts. The models described are of the classical orchestra, the string quartet, the jazz combo and the free improvisation ensemble (Winkler 2001, pp. 21-8).

Trombonist, composer and programmer George Lewis's *Voyager* program embodies the composer's conceptual ideas about improvisation and musical interactivity into a system which describes as a 'virtual improvising orchestra' (Lewis 2000, p. 33). According to Lewis, a performer cannot directly control *Voyager* but may influence its contribution to a performance through their performance dynamics. The software acts of its own accord by using analyses of the musician's improvised performance to affect a series of complex internal algorithms controlling some sixty-four "asynchronously

operating" music-generating players (Lewis 2000, p. 34). The composer describes the performative effect of this interactive environment thus:

> I conceive a performance of *Voyager* as multiple parallel streams of music generation, emanating from both the computers and the humans—a non-hierarchical, improvisational, subject-subject model of discourse, rather than a stimulus-response setup.
>
> (Lewis 2000, p. 34)

Regarding the internal dynamics of his generative approach, Lewis cites the Javanese gamelan ensemble as a useful model for non-hierarchical and independent musical organisation in a large group environment. To Lewis, the capacities of his virtual orchestra to both analyse and generate are what drive such complex forms of interactivity. The composer explains that in this collective context, the success of such large-scale musical interaction "can be seen to depend not only upon the performative skill of the players, but on their real-time analytic capabilities." (Lewis 2000)

Of course, early examples and classifications of interactive music systems must be evaluated in terms of their technological context. Interactive music systems of the 1980s and early 1990s relied almost exclusively on the MIDI standard for the analysis, processing and output of musical information. Drummond considers early constraints in a recent survey of the field, noting that the analysis and synthesis of sonic material in these systems was restricted due to the computing power and speed of this period, cost of hardware, and the inherent constraints of the MIDI standard (Drummond 2009). In considering Rowe and Winkler's definitions of interactive music, Drummond draws attention to the limitation of sensory input in both approaches to event-based musical parameters afforded by the MIDI standard (notes, dynamics, articulations, etc.). The author is critical of the limitation of these approaches to the input of event-based musical material, and in making the case for gestural interaction with synthesis parameters he notes that "The morphology of the sound in a MIDI system is largely fixed and so the musical constraints are inherited from instrumental music." (Drummond 2009, p. 126)

Paine is similarly critical of both Rowe and Winkler's definitions and models of musical interaction, arguing that models based on existing musical practice may be

insufficient to describe the potential of interactive music systems to enable new ways of making music:

> In a situation where the system is designed to accompany or improvise with a musician, the construction of the responses within an agreed musical aesthetic makes sense; however, this approach does nothing to further our exploration of the inherent qualities of an interactive music system, it simply squeezes interaction into a known template.
>
> (Paine 2002, p. 297).

For Paine, a richer and more flexible model for musical interactivity than those inherited from instrumental music practice is that of human conversation. For the purposes of his model of interaction, Paine defines a conversation as:

> unique and personal to those individuals
> unique to that moment of interaction, varying in accordance with the unfolding dialogue, but is;
> maintained within a common paradigm (both parties speak the same language, and address the same topic)
>
> (Paine 2002, p. 297)

Paine's model is therefore more concerned with the design of mutual influence within an interactive context, and the uniqueness of the relationship between the individual's interaction with the system and how this can be reflected in system design. Interestingly, the author argues that these conditions of interactivity are most often met in responsive sound installations or immersive environments in which the public interacts with the system (Paine 2002).

Also seeking classifications of musical interactivity that extend beyond existing musical metaphors, Bongers has provided an expanded classification structure that considers the 'who' in an interactive musical context – that is, exactly who are such systems designed to be interacting with. Bongers' categories are *performer with system, system with audience* and *system with performer with audience* (Bongers 2000). Although such classifications do not take into account the specific types of musical interaction achievable within each category, they prove useful in defining the overall context in which interaction with a performance system is envisaged to take place.

Eldridge is also critical of traditional metaphors in classifying the possibilities afforded by interactive music systems, focusing attention upon the levels and direction of control and influence within such systems. The author asserts that much of what is described in interactive music discourse as 'interactive' is essentially a one-way form of interaction, with little attention paid to the ability of software to autonomously influence a performer: "…in many instances of published research, there is little evidence of 'mutual influence': traffic down Winkler's street is essentially one-way." (Eldridge 2008, p. 34) Eldridge's creative work is concerned primarily with addressing this imbalance, incorporating generative algorithms such as neural oscillator networks and dynamical systems into her interactive systems (Eldridge 2005, 2008).

Also concerned with issues of mutual influence are Tim Blackwell and Michael Young, who have coined the term *Live Algorithm* in the context of improvised computer music performance. According to Young, a *Live Algorithm* is "the function of an ideal autonomous system able to engage in performance with abilities analogous, if not identical, to a human musician." (Young 2008, p. 337) Blackwell and Young differentiate between what they term 'strong' and 'weak' interactivity by the level of autonomy such a system displays in a performance context. A weakly interactive system responds directly to instrumental input, with surprising or unpredictable behaviour often organised by stochastically designed processes. In other words, although the system may surprise and provoke certain courses of action in their human interlocutors, these systems are not necessarily cognisant of the performative context of the interaction when developing novel musical material.

According to the authors, a strongly interactive system would by contrast not only *respond* to a given performance situation, it may also *instigate* musical trajectories and genuinely surprise a human partner in a musically intelligent fashion. Such ideals would therefore enhance the ability of the system to participate in a mutually influential exchange with a human performer (Blackwell & Young 2005). The authors suggest that the ideal of strong interactivity is found in the human practice of free improvisation, a practice that eschews top-down organisation, instead displaying emergent forms of musical structuring that are contingent on the 'comprehensible' contributions of the players involved in the performance dynamic.

Bown, Eldridge & McCormack have labeled traditional models and metaphors of musical interaction as being representative of an *acoustic paradigm* (Bown, Eldridge & McCormack 2009). The authors argue that although traditional distinctions between instrument, composition and performer are commonly used to describe aspects of contemporary computer music practice, those working in the field are continually redefining the relations between these in their creative practice. In the field of experimental and improvised computer music performance, "…software developers commonly play an active part in the development of the musical concepts and the production of the music itself, and artist-programmers are common." (Bown, Eldridge & McCormack 2009) In looking towards a *digital paradigm*, the authors' discussion also highlights the interactive role that a software artefact itself can play in the broader context of musical culture, drawing a distinction between an artefact's capacity to facilitate interactive exchanges through *performative agency* (in performance time) and *memetic agency* (out of performance time).

The various authors discussed above highlight perceived limitations of early approaches to defining and classifying aspects of interactive music systems, with many critiques focused upon the incompatibility of metaphors descended from instrumental musical practice to adequately explain the types of interaction afforded by the diversity of current approaches. These authors also expand the scope in which musical interactivity can be defined by highlighting the flexible and malleable nature of contemporary computer music practice, including the different interactive contexts possible within the field as well as the interactive role that software artefacts themselves can play in musical culture out of performance time.

However, as Bongers' broad categories highlight, it may prove useful to focus attention upon the interactive context intended for the specific music system in question when applying or rejecting classification methods in the field. In defining these contexts, Bongers' three classifications importantly delineate the roles of each of the possible actors present in any interactive musical experience; namely those of the *performer, system* and the *audience*. Whilst Paine's proposed conversational model of interaction is perhaps more flexible in encompassing a wide variety of interactive experiences, the implication that the conditions of this model are most often met within a public installation context appears limiting. Paine's model is concerned with the nature of mutual influence between

human and system, and the uniqueness of the interactive relationship between the individual and the system in any interactive context. Although this author shares Paine's criticism of traditional 'note-based' approaches, this does not necessarily preclude the *performer with system* paradigm as a relevant avenue for exploration in the field of interactive music. Indeed, in recent years there have been a variety of approaches that favour the analysis and use of timbral informational from the instrumental signal rather than discrete pitch events. These systems strive to move away from event-based approaches, however they remain inherently connected to an instrumental performance paradigm, as evident in the work Cuifo (2005) Hsu (2005, 2006, 2008; 2010), Young (2008, 2009; 2003), Bown (2011; 2006), Bown and Lexer (2006) and others.

2.3 Design strategies – key concepts and approaches

Given the context of this research project, issues related to human-computer interaction in improvised musical performance are of great relevance. To return to Bongers' classification structure, in this thesis we are therefore concerned with issues surrounding interactive systems working within a *performer with system* interactive context. As outlined in the following survey of recent interactive systems, the issue of interactive context (i.e. *who* is interacting with the system) is crucial to the design strategies of system designers, in addition to the musical and aesthetic context envisaged for the interaction itself.

2.3.1 *Relationship between input analysis and generation*

Machine listening in interactive musical systems – the ability of a system to 'make sense' of the audio stream presented to it during performance – are of upmost importance in the design of interactive musical systems. Appropriately, the choice of a particular analysis and/or segmentation method is often tightly bound to the musical and interactive context envisaged for the system being designed. As discussed previously, canonical musical systems relied heavily upon the MIDI representation of acoustic input signals. As many practitioners have noted, such a method has unsurprisingly resulted in approaches to system design that privilege traditional methods of musical generation aligned to this standard. In addition, symbolic musical representations provided by MIDI have the added problem of quantising or omitting certain musical features, particularly with relation to timbre. Although drawing criticism from those seeking to expand

definitions of musical interactivity, some recent approaches to the design of interactive systems have made inventive use of event-based parameters in their generative designs. The appropriateness of these methods is naturally dependent on the musical and interactive context in which such performance data is to be used.

In the *Omax* project, the authors present an interactive improvisation system which models musical improvisation through the analysis of event-based performance data from an instrumental signal (Assayag, Bloch & Chemillier 2006; Assayag et al. 2006). In this system, fundamental frequency analysis achieved via the *Yin* algorithm (De Cheveigné & Kawahara 2002) drives a pitch-to-MIDI conversion system that parses acoustic instrumental signals into a MIDI representation of the live performance of an instrumental improviser. This MIDI data is then further modelled using sophisticated machine-learning methods. A virtual improvisation kernel is informed by this data, developing improvisational responses (via either MIDI output, or more recently via analysed and time-stamped recordings of the performance) enabling instrumentalists to interact with a virtual 'clone' of himself or herself (Assayag, Bloch & Chemillier 2006). In this instance, event-based analysis of performance gesture is vital to the system's response, as the system aims to statistically model an improvisation upon the analysis received in real-time from the performer.

Collins' work is heavily concerned with event onset detection in the analysis of human performers, a machine listening approach that serves to detect musical onsets from a live audio stream. The author's *DrumTrack* software is designed to track both the tempo and phase of a human drummer from live audio alone via beat induction methods (Collins 2005). This approach informed the design of an algorithmic improvisation system based upon this input analysis method. As Collins has noted, aesthetic considerations pertaining to the algorithmic generation process were considered in the development of the beat induction algorithm itself, informing its design:

> Certain decisions taken in the programming of the beat induction algorithm betray compositional decisions, such as the 90-190 tempo range without mid biased tempo prior that supports drum and bass style 160bpm+ drumming. Assumptions of 4/4 eased the pattern matching task, and the handling characteristics at phase transitions were revised to fit feedback from the performer.
>
> (Collins 2005)

Similarly, Gifford and Brown's work makes use of stochastic onset detection (SOD) techniques to derive onsets from musical audio signals, and specifically non-pitched percussion instruments (Gifford & Brown 2008, 2009). The authors describe a system that detects musical patterns from a live musician by analysing salient musical features such as pulse, metre and downbeats, further using this accumulated information to drive generative algorithms that perform alongside the musical performer. As the two previous approaches demonstrate, the use of event-based methods for analysis, prediction and generation of musical materials is relevant to the context of percussive performance. These musical scenarios are dependent upon rhythmic and metric accuracy in the analysis process in order to drive their generative responses to live input.

Other event-based methods can be found in the work of Hsu and Ciufo, whose systems used silence thresholding methods to track the appearance of musical 'phrases' from a live input stream (Ciufo 2005; Hsu 2006). In such methods, the amplitude of the live input is used to determine when the musician is playing, and when the signal has fallen silent. In both cases, a simple time threshold is used to report once the signal has fallen silent for an amount of time specified by the silence threshold mechanism. The crossing of these time thresholds are then used to report the end of a 'phrase' from the input. Both authors use such high-level event-based methods in order to segment analyses of a live performer into larger chunks from which to derive statistics on the musician's current performance state. These statistics are then used to affect the generative processes of their algorithmic improvisational systems. High-level analyses such as phrase segmentation help the system designer compartmentalise an input stream into manageable chunks of analysis data. Whilst still reducing the musical input into a series of symbolic representations, both authors use such segmentation measures in order to manage a large array of streamed sound descriptors, rather than to objectively analyse the musical input for perceptual phrase boundaries.

In contrast to such event-based methods, in working with the Korean flute *daegeum* Dobrian sought a nuanced approach to the use of the instrumental signal in interactive performance (Dobrian 2004). A method the author dubbed 'stealing expressivity', Dobrian sought a natural form of expressivity in his computer-synthesised material through continuous pitch and amplitude tracking of the flute signal. According to the author, such an approach was necessitated by the difficulty in employing event-based

analysis methods to this particular instrument due to specific sonic characteristics such as its wide vibrato (Dobrian 2004). Taking continuous amplitude and pitch data from the audio input stream, Dobrian outlines techniques for capturing and making direct use of this idiosyncrasy of the daegeum's performance. Thus, the analysis method is informed by instrumental characteristics, which in turn affects the approach to musical generation by the computer music system.

2.3.2 *Timbral awareness*

Many recent interactive musical systems have moved beyond event-based methods in both analysis and generation. Often these systems are concerned with the role of musical timbre in driving their generative output. Due to the increasing availability of high quality real-time spectral analysis tools in computer music environments, the real-time task of analysing the timbral characteristics of live musical performance has become a relatively trivial task. Systems making use of such tools are unsurprisingly concerned with musical contexts in which timbre is considered an integral part of the musical idiom itself, such as in the context of free or non-idiomatic improvisational practices (Bailey 1993). In addition, the wide variety of methods for analysing musical timbre via sound descriptor analysis has ensured an equally wide variety of approaches to using such analysis tools in interactive musical systems.

Hsu's improvisation systems make use of sound descriptors analysed from a live improviser to track salient perceptual features of the improvising musician, as well as to directly affect the material of improvising software agents (Hsu 2005, 2006, 2008; Hsu 2010). In earlier systems, the author has used sound descriptors in order to automate and control a series of virtual improvising agents. These improvising agents make use of gestural curves analysed and stored throughout a performance to automate their various synthesis and processing parameters (Hsu 2006). In a more recent system the author describes his approach to using timbral analysis as inspired by the idiosyncratic timbral techniques of saxophonist John Butcher, a free improviser well-known for his innovative use of timbre (Hsu 2010). In addition to tracking loudness and tempo of the improviser, Hsu's system tracks auditory roughness, a sound descriptor pertaining to the interference between various partials present in a complex signal. This measure was chosen due its

correlation to the musical concepts of tension/release and consonance/dissonance (Hsu 2010).

Young and Lexer (2003) discuss the use of Fast Fourier Transform (FFT) analysis as a creative tool in freely improvised electroacoustic performance. In this work, the authors suggest that visualising the audio spectrum of a live performer can aid the computer musician in developing responses to the acoustic performance of the improviser. In addition, the authors propose a method for using FFT analysis for providing real-time controls for synthesis parameters. Such an 'audio as controller' approach is justified by the authors as an improvement upon pitch tracking and amplitude following methods that favour a reductive approach to musical material (Young & Lexer 2003).

Johnston's work makes use of sinusoidal decomposition in his approach to the design of audiovisual virtual instruments based on physical models (Johnston 2009; Johnston, Marks & Edmonds 2005). In this work, traditional pitch tracking is combined with sinusoidal decomposition techniques using the well-known *fiddle~* external for Pure Data and MSP (Puckette, Apel & Zicarelli 1998). The work maps both analysed pitches and the amplitudes of the first three partials of the performer's audio spectrum to control a series of visual spheres on screen. The amplitude of the various partials has a direct affect on the brightness and colour of these spheres. Although directly controlling individual spheres via audio analysis, Johnston's work engages the musician in a dialogue with his audiovisual instruments as the activation of these engages a mass-spring model controlling both audio and visual responses to the musician's actions.

Other interactive systems enable system awareness of the current and past timbral context within an improvisational performance to drive algorithmic responses to instrumental performance. Ciufo's system *Beginner's Mind* uses real-time descriptor analysis to dynamically build a measure of the 'perceptual identity' of segmented phrases analysed from the live input over the duration of a performance. Ciufo's approach uses silence-thresholding techniques to segment the analysis of a live audio stream into phrase boundaries. The author uses Jehan's *analyzer~* external for Max (Jehan & Schoner 2001) to track various elements of the live performance including brightness, noisiness, amplitude and pitch information, using this information to relate the current performance state of the human improviser with data analysed previously in the same

performance. This approach enables the system to consult a growing list of pre-analysed material siphoned from the performer during an improvised session (Ciufo 2005).

As the previous projects demonstrate, the choice of analysis and system response methods in any given interactive music system is dependent upon the musical and aesthetic context in which this system is operating. This context can include the specific sonic characteristics of the acoustic instrument being analysed and invariably includes the type of interaction desired between performer and system.

2.3.3 *Sonic and algorithmic derivation*

As discussed previously, an important aspect of early discourse on interactive music is the level and direction of control and influence present in an interactive system. Balancing levels of control and influence within an interactive music system becomes an important aspect of the design of such systems, and is once more dependent on the type of interaction desired in performance. As has been noted by Blackwell and Young, for interactive systems to be considered strongly interactive, they must also be able to influence music performed by an instrumentalist in a non-trivial way (Blackwell & Young 2005). Describing interaction as mutual influence, Pressing has defined an 'interactive instrument' as one "…that directly and variably influences the production of music by a performer." (Pressing 1990, p. 20) Bongers has referred to this process as feedback, both within the system itself and between the system and the performer (Bongers 2000).

Expanding on the concept of influence in such performance contexts, an interesting and unique feature of electronic music performance is the degree to which a system's sonic vocabulary and/or generative grammar is directly derived from the input from a live performer. Paine's observation that "a perceivable relationship between the gestural input and the system output, is a central issue in the design of interactive systems" (Paine 2002, p. 298) becomes relevant here when we consider the means by which such systems develop their sonic responses to live performance input. Many designers of interactive systems make use of the sonic input of instrumental performers as a basis for both the sonic vocabulary and generative grammars of their system designs. This is an important factor considered by many system designers in their work. Here I define such approaches

as employing a form of sonic or algorithmic *derivation*, a term referring to how a system's sonic or algorithmic responses are directly appropriated from its external input(s).

2.3.3.1 Sonic derivation

Sonic derivation is an approach common in much contemporary electronic music practice, filtering into a broad range of performance modalities. Broadly speaking, in such approaches the live audio of one or more performers is used as the primary source material for improvised, pre-programmed or deterministic processing, sampling and manipulation in performance. The practice of 'live-sampling' – the recording, overdubbing and manipulation of sonic material performed live – has a long history in live-electronic performance, and is the primary means by which sonic derivation has been explored in much contemporary computer music performance. Live looping is a simple yet effective example of sonic derivation, in which material performed by a live performer is recorded and used to perform additive musical structuring during performance. Live looping of instrumental performance is an approach that became popular in the mid-1970s with guitarist Robert Fripp's *Frippertronics* tape-loop system, developed in collaboration with Brian Eno (Fricke 1979). Originating in the studio compositions of composers such as Karlheinz Stockhausen, Koenig and others, the use of tape loops in composition and performance was explored in the 1960s by composers such as Pauline Oliveiros, Terry Riley and Steve Reich (Collins & Escrivan Rincón 2007). However, Fripp's live performance system became widely known through his solo performances and the album *No Pussyfooting* (1973), which he recorded in the studio of the pioneer of ambient music, Brian Eno.

The practice enables a performer to construct musical structures based upon recorded and 'looped' segments of their own live performance. As a structural approach, this form of sonic derivation relies upon connecting current musical ideas with those that have been performed previously. Such approaches are often presented in a 'one person band' context, a form of performance practice in which the process of musical structuring is presented as a key performative element. Crucially, the instrumental or vocal performer maintains direct control over the live structuring of their looped materials. The contemporary practice of live looping is extremely broad in scope, and due to the availability of dedicated looping pedals and software the practice is widespread in popular

musical performance[7]. Contemporary musicians such as Zoë Keating, Camille Dalmais and Imogen Heap make use of live looping technology to build traditional song structures from sampled and repeated materials.[8]

As a form of process music, live looping is a conceptually direct form of electro-acoustic performance. However, despite the musical ingenuity of those engaged in the practice, the approach does not lend itself well to more complex forms of musical structuring. Musical material recorded during a performance, although divorced from its original context, is often presented largely unchanged, as the structuring of the recorded materials forms the conceptual interest of the approach. In addition, the linear and additive nature of the musical structuring process ensures that more complex methods of musical generation remain largely out of reach of the practice. Despite its limitations, the conceptual directness of the live looping approach remains broadly appealing. Indeed, for computer musicians engaged in the development of automated, interactive and generative systems, the act of recording and re-organising material siphoned from a live performer continues to remain a core creative focus for a number of practitioners.

Although not exclusively, live sampling is often employed by performer-developers seeking to expand their live performance practice. For instrumentalists working with music technology, the ability to sample, loop and manipulate one's own sound in real-time is an enticing extension to traditional instrumental performance practice. Such approaches range from the straightforward live looping approaches discussed above, to more complex, algorithmically controlled sampling processes. This approach to performance practice is very widespread amongst a variety of musical genres. Given the diversity of this field, a survey of such a broad variety of artists is outside the scope of this paper, however a number of approaches to sonic derivation and interactive performance are noted here.[9]

---

[7] Websites such as the long-running 'Looper's Delight' have contributed to disseminating information about the history, techniques and contemporary craft of live looping (LaFosse 1996).

[8] Artist websites can be accessed at http://zoekeating.com/, http://www.camille-music.com and http://imogenheap.com/

[9] Other notable performer-developers working with live sampling in their practice include Richard Barrett Karlheinz Essl, Christian Fennesz, Robin Fox and Pamela Z.

Kaffe Matthews makes use of live sampling in the context of solo improvised performance, using the sampler as a means of augmenting and extending both performative action and resonant spaces. Matthews, originally a violinist, makes use of the software LiSa[10] (developed at STEIM) for live sampling-based solo performances. The artist develops textural compositions by sampling sounds from the performance space via strategically placed microphones, and transforming them in quadrophonic surround sound. In her performances, the artist is interested in the site-specific nature of sound, using the blank space of the sampler to capture and transform sounds from the venue into in situ compositions. The immediacy of sampling as a compositional tool is an important part of Matthews' work: "What immediately thrilled me was that the sampler allowed you to make music without having to labor over it for hours every day, which was what I'd been used to doing." (Huberman 2004) In addition, Matthews views the process of working with computer music tools as a 'collaboration' with the machine, as she states "The computer often had good ideas." (Huberman 2004)

In *Fond Punctions*, performer-developer Alice Eldridge presents a generative performance environment for cello and computer in which levels of control and influence between player and system components are clearly defined. In this system, two generative algorithms (a homeostatic network and a physics simulator) are used to create dynamic and evolving textures based upon live samples of the instrumentalist's playing. In this example, performer control over the system is limited to the recording of the samples themselves (triggered via foot switch), as two generative processes subsequently control a granular synthesis engine that processes these samples during performance (Eldridge 2005). In *Fond Punctions,* the influence of one algorithm upon another determines the sonic output of the computer music system in its entirety. There is no further control over this process by the performer.

Eldridge has noted that the underlying practical motivation behind the development of the system was the desire for a 'hands free' mode of live-electronic performance (Eldridge 2005). However, as all raw sonic material is directly appropriated from the performer's input, and the decision as to which musical material is to be recorded is afforded to the performer, the performer has direct control over the overall sonic texture

[10] http://steim.org/2012/01/lisa-x-v1-25/

through both their playing and carefully chosen samples to record. In addition, the emergent nature of the generative processes further influences the performance of the instrumentalist in their subsequent performance. Although the performer has no direct control over the algorithmic processes at work within the system (unlike in a live looping performance) a balance between control, sonic derivation and influence has been achieved in this interactive performance environment.

Performer-developer Rodrigo Costanzo is an improviser and computer musician working with live electronics in the context of free improvisational performance. A trained classical instrumentalist, Costanzo's recent software systems, The Party Van[11] and Cut Glove[12] are complex, yet intuitive live-sampling based software tools that allow the user to improvise with materials sampled in real-time from a live instrumentalist. Costanzo makes use of these systems in both solo and group improvisatory contexts, and his systems have been shared and documented freely online. In addition to real-time performer control using hardware controllers[13], Costanzo's systems also make use of input analysis and algorithmic control in order to drive various parameters in his systems. A balance between direct and algorithmic control over low-level variables in his systems enables a degree of abstraction for the user from the complex process of sampling and manipulation.

Returning to Ciufo's work discussed previously, the author describes a live-sampling method in which a recording module captures the entire audio stream of an improviser during performance. In conjunction with a variety of statistics collected from sound descriptors analysed from the input, his system is able to relate the current performance state of the musician to live-sampled material collected throughout an improvisation (Ciufo 2005). In Ciufo's work, statistics are captured on various time scales of the performance, enabling the system to make decisions upon which material to process based upon numerous measures of musical historicity. A less direct example of sonic derivation can also be found in Dobrian's approach to expressive computer synthesis described previously. Although the algorithmic processes at work in his systems are unclear, the performer's influence over the system's sonic outcome through directly

[11] http://www.rodrigoconstanzo.com/the-party-van/

[12] http://www.rodrigoconstanzo.com/2015/06/cut-glove/

[13] http://monome.org/

deriving control parameters from the live flute signal is clear in his design approach. Hsu's (2006) approach to deriving synthesis parameters from the live signal of an instrumental performance may also be seen as a form of indirect sonic derivation. The author's use of gestural curves taken from sound descriptor analyses are used to create synthetic gestures derived directly from the performer.

In Sebastian Lexer's *piano+* system, the author describes an interactive environment in which distributed layers of direct and indirect control are used to capture and process live sampled material from his acoustic performance (Lexer 2010). Working in the context of freely improvised performance, the performer-developer's system participates in freely improvised performance through a combination of algorithmic processes (neural networks, stochastic algorithms), audio and sensor analysis and hardware controls. The interaction between these various layers of control enable the performer to improvise with an emergent computational system of great complexity, with the acoustic origin of the piano forming the basis of the system's sonic and algorithmic processes.

Sonic derivation is also found in the previously-described *Omax* project; however, in contrast to the previous artists works, this project requires both a live instrumentalist and a computer music performer to pilot the interactive software. In this project, the process of sonic derivation is *continuous* as opposed to *momentary*, with the system continually sampling the musician's live performance into a large audio buffer in memory. In contrast to live looping and the momentary live sampling approaches exemplified by Eldridge's work, the musical structuring process in *Omax* is automated through the use of low-level analyses on a continually recorded audio stream, stitching together a musical 'clone' of the performer from time-stamped segments of the recorded performance. A human *Omax* 'player' operating the software is then tasked with navigating an algorithm built from these analyses, effectively giving high-level control over the musical patterning process to another musician. The *Omax* system is discussed in more depth in the following section.

2.3.3.2 Algorithmic derivation

In addition to the derivation of sonic materials from a live improviser, numerous approaches to the design of interactive systems also make use of analysed patterns from

a live input in order to structure the computer's generative musical output. Such approaches effectively model an improviser's performance in real-time, enabling these systems to derive their algorithmic and generative responses directly from detailed analyses of a live performer. , these methods build spaces of generative potential interactively from the performer, enabling efficient search algorithms to extrapolate from recognised patterns in the recorded data.

Such modelling processes are at work in Pachet's *Continuator* system, a real-time application of Markov-modelling intended to model the stylistic characteristics of jazz improvisations (Pachet 2002). Pachet introduces his system as a bridge between two seemingly incompatible domains of computer music practice, namely interactive musical performance and musical imitation. The author's system makes use of variable order Markov models (VMMs) in order to efficiently and convincingly model musical sequences derived from a live performer. Such models are then used interactively to 'extend' the musical performance of an improviser with real-time continuations of their melodic material.

In the *Omax* project discussed previously, the authors also engage in a form of modelling, with the system acting as a real-time, interactive instantiation of a pattern recognition process (Assayag, Bloch & Chemillier 2006). This derivation process relies upon the *factor oracle* algorithm, an algorithm that recognises patterns arising in a string of characters (Allauzen 1999). Pitch tracking of the instrumental signal is used to build the oracle during a real-time performance, enabling the human operator to navigate a constantly expanding model of the performer's history in the generation of one or more clones of the improviser. This process has been recently extended to encompass models of the spectral content of an improviser's sound (Bloch, Dubnov & Assayag 2008), in order to autonomously generate clones based upon the connections between timbral materials captured from the improviser. A recent version of the software *WoMax* also incorporates an interactive visualisation of the factor oracle algorithm for the computer operator to navigate in performance (Lévy 2013).

An approach that blends both sonic and algorithmic derivation is found in the *soundspotting* technique developed by Michael Casey (2009). Soundspotting is a technique for generating streams of audio data by using content-based music information retrieval

methods (Casey et al. 2008). In a soundspotting system, a target signal (often a live instrumental signal) is used to query a large database of pre-analysed audio segments. The system uses the target signal to find the closest matching segments from within its database, using these segments to create a concurrent musical stream of concatenated database elements. Extending previous non real-time approaches such as Zils and Pachet's *musaicing* (musical mosaicing) (Zils & Pachet 2001) and Casey's own previous non real-time soundspotting system (Casey 2009), the approach uses the target signal to create musical output through the use of Mel Frequency Cepstral Coefficient (MFCC) feature extraction on the live input.

In Casey's description of the process, the result of the real-time soundspotting creates a hybrid musical instrument, an approach that favours controllable and 'learnable' results by an instrumental performer (Casey 2009). This type of system may therefore be considered closely aligned to Rowe's *instrumental paradigm*. However, Casey has also discussed the possibility of the system being used in numerous interactive ways, including using the live target signal to query a growing corpus of its own history, creating what he describes as an 'associative memory canon':

> It is clear that method of feeding the target through a self-referencing memory process produces a deterministic output, and it is this process that generates the ensuing canon without further intervention from a composer. Careful composition or selection of target materials leads to the construction of a counterpoint that is relational at each time instant to the history of a performance.
>
> (Casey 2009, p. 426)

Casey's approach to real-time content-based music information retrieval is an interesting example of both indirect sonic and algorithmic derivation. Here the sound of the live performer is recreated in real-time by stitching together returned segments of a large corpus of pre-analysed materials. In addition, the method by which these materials are concatenated is determined entirely by comparisons made between the data analysed from the performer.

Another approach to musical modelling and algorithmic derivation is found in Martin's Agent Designer Toolkit (ADTK) (Martin 2014; Martin et al. 2012). The ADTK is a toolkit for designing the behaviour of musical agents and is implemented as a

Max4Live device in the Ableton Live software environment. The aim of the software is to enable the design of autonomous agents that automate elements of a musical performance in the Live environment, in collaboration with a human user. In Martin's work, human performances using Ableton Live are analysed using machine-learning algorithms in order to present the user with models of a musician's interaction with the software. This approach is achieved by performing variable order Markov modelling on the various parameters used by a performer in a Live set. In addition, association rule learning algorithms are used to search for patterns and dependencies amongst the various parameters modelled by the markov modelling process. Martin's approach is unique due to its focus upon user interaction in the algorithmic derivation process. The user is presented with a list of parameters that have been modelled during one or more training performances. This user interaction process is then further used to determine the various rules that can be applied to the musical agents designed to automate musical performances.

2.3.4 *Live algorithms and musical autonomy*

Several authors discussed previously have sought to define approaches to interactive musical improvisation with regard to musical autonomy, self-organisation and emergent processes. These authors are often concerned with the study and simulation of artistic creativity and are closely aligned with the cross-disciplinary field of computational creativity.[14] Central to Young and Blackwell's concept of the live algorithm is the employment of generative algorithms that in most cases require no human intervention or modelling in order to drive their internal processes. These algorithmic approaches are based upon algorithms such as neural networks, genetic algorithms, search and sort algorithms and swarm dynamics in order to achieve autonomy in the generation of musical materials in performance.

As discussed previously, the employment of live algorithms in musical performance are primarily concerned with enabling strong interactivity in an improvised musical context (Blackwell & Young 2005). A large part of what enables such a strong form of interactivity to occur is the software's ability to instigate musical trajectories in a plausible

[14] http://computationalcreativity.net/

yet unpredictable manner. Central to the concept of live algorithms is the high-level breakdown of system components into analysis, and synthesis elements, with a hidden patterning process between the two. This otherwise simple outline of an autonomous musical system rests upon the functional characteristics of the patterning mechanism itself, which has been characterised as an abstract 'behavioural' system (Bown 2011). Importantly, the interface between the patterning mechanism and the analysis and synthesis layers is not specified, meaning that the design of live algorithms rests upon finding suitable relationships between the inherent dynamics of autonomous systems to suit the musical context envisaged by the designer.

Young's work employs a feed-forward neural network as the patterning component of the live algorithms used in the improvisatory works *piano_prosthesis, au(or)a* and *cello_prosthetis* (Young 2008). The feed-forward network is trained from audio analysis fed to it from the performance of a live instrumentalist, the output of which is mapped to real-time synthesis and processing parameters. As the author explains, this type of algorithm is well suited for use in improvisatory contexts due to its capacity for generalisation and tolerance for unpredictable input (Young 2008). Feed-forward neural networks usually require a training phase before generating output, however in Young's implementation of the algorithm the training phase occurs during a musical performance. Given the ability of the neural network to learn and adapt to its audio environment through training, the implementation of such learning algorithms can also be considered a form of algorithmic derivation. This is due to the inherent means by which the organisation of the patterning mechanism is directly contingent upon the input fed to it throughout performance.

Neural networks are a popular choice of algorithm for digital arts practitioners. Other approaches making use of neural networks in interactive musical systems are Beliharz et al.'s *hyper-shakuhachi* project employing neural oscillator networks (Beilharz, Jakovich & Ferguson 2006), Bown and Lexer's use of continuous-time recurrent neural networks (CTRNNs) to control a spectral filter (Bown & Lexer 2006) and Bown's use of CTRNNs as a behavioural module to control a composed generative algorithms controlling sample playback and synthesis parameters (Bown 2011).

Eldridge's work discussed previously is also concerned with musical autonomy in the generative process. Although *Fond Punctions* is not concerned with audio analysis of a live performer, the use of a homeostatic network and physics simulation places her work within the realm of autonomous musical generation in interactive performance. Similarly, Blackwell is concerned with the autonomous interaction between components of a generative system as a core element of the design of interactive systems. Blackwell's work has focused upon the simulation of swarm dynamics as well as the biological process of *stigmergy* observable in natural systems (Blackwell & Young 2004). Such biological systems are used to control granular synthesis algorithms and other synthesis parameters in his systems.

In Bown's work, the development and use of live algorithms is intimately connected to issues of collectivity and sharing in creative programming practices, as well as the ability for modularity and extensibility in the design of musical software systems (Bown 2011; Bown, Eldridge & McCormack 2009). Discussing modular design approaches in the development of live algorithms, the author outlines the use of binary decision trees (DTs) as a useful approach to enabling autonomous musical interaction with a live performer (Bown 2011). The DT is a form of classification algorithm used to analyse and classify data in a similar way to neural networks. The DT algorithm combines a hidden, randomly determined internal state with sound descriptor analysis streamed from a live performer. The algorithm uses both its internal state and the incoming audio data to carve up a multi-dimensional space based upon a series of decisions made by the algorithm. The output of this dynamical process is then used to trigger generative audio events programmed and arranged by the user.

These events are conceived as occupying the compositional part of the system, and may be reconfigured on the fly easily in response to the output of the dynamical system. The approach taken by Bown in this instance demonstrates the possibilities inherent in using such modular design methods. Such a methodology eschews integrated approaches to connecting analysis and synthesis parameters in an interactive system. Instead, the designer is free to 'mix and match' various hand-coded generative modules to suit the dynamical patterning behaviour of the output of the DT in relation to its reaction to the

live performer. Hence there is a considered separation between analysis, patterning *(f)* and synthesis elements of the system.[15]

Despite the conceptual and practical benefits of autonomous algorithms, in the context of human-machine musical performance the notion of machine autonomy must always be balanced with other considerations inherent to collective musical performance, and especially improvisatory practices. According to Bown and Martin, a musical system may be deemed autonomous if its future state is more likely predicted by analysing its own past states, and not those of its environment (Bown & Martin 2012). However, according to the authors, to the extent that musical autonomy means the capacity for such systems to display self-determination, "the autonomy that we seek in autonomous music systems is not something that should be maximised to the point of freedom from influence." (Bown & Martin 2012) In Young's writings, musical autonomy is considered as part of a conceptual scheme that also includes adaptability and musical intimacy (Young 2009). To Young, intimacy in such performance contexts refers to the degree to which computers and humans might both adapt and learn from one another in performance (Young 2009). The musical ideals of adaptability and intimacy must therefore be balanced against the abilities of such systems to display characteristics of self-determination. Concerned with *stigmergic* behaviours, Young proposes that adaptability in human-computer improvisation is a process in which two entities acclimatise to their environments, without direct concern for responsiveness, causality or intentionality. The musical environment is shared, and humans and computers adapt to this through shared exchange with the environment, not each other directly.

Young suggests that this conception of human-machine improvisation 'avoids the pitfalls of anthropomorphism', as concerns for human-machine interactivity and influence are no longer the main concern in the design of live algorithms (Young 2009, p. 2). Concurrently, the author proposes the concept of intimacy in these scenarios as an important element in social interaction. With improvisation conceived of as a social process, intimacy is proposed as metaphor for human-computer improvisational

[15] I have had the pleasure of performing with Bown's system in performance on numerous occasions. An album titled *Ben + Zamyatin* was released in 2013 on the Not Applicable label documenting three tenor saxophone improvisations between myself and Bown's software: http://www.not-applicable.org/?page_id=2023

exchange. This concept includes the ability of social entities to engage in self-disclosure and validation through a partner's response (Young 2009, p. 4), acknowledging the need for mutual influence and learning to occur between the two entities.

2.3.5 *Rehearsal and performance practice*

One aspect of interactive computer music discussed little in the literature is the role of rehearsal in the context of interactive musical performance. Although the relationship between rehearsal and performance differs depending on the performance practice context in question, it may be surmised that the process of rehearsal is integral to all performing arts, and especially so in musical performance. In discussing his model of conversational interaction, Paine notes that a true interactive relationship between interactor and system can only exist if a system is capable of changing and evolving. According to Paine, systems of this type would reflect the 'cumulative experience of interrelationship' present in human interactive contexts (Paine 2002, p. 298). Dynamic change and evolution is inherent in a human conversational context, not only in through the interactions between people at one given time and place, but also over longer timescales. Such a process is also naturally present in the rehearsal process in musical performance.

In the work of Bastien and Hostager (1992), the authors discuss the cumulative building of interpersonally shared histories between musicians over consecutive rehearsal sessions. Investigating the context of collective jazz performance, the authors note that performance-time interactions between musicians depend upon 'suprapersonal' common histories of individual participants. These histories are based upon individual players' cumulative interactive experiences with other musicians over several rehearsals and performance. In the context of 'free' or 'non-idiomatic improvisation' musical improvisation (Bailey 1993), the question of rehearsal becomes problematic, as this form of musical performance emphasises emergent properties and spontaneous actions unique to interaction in performance, without reference to a pre-defined musical score. However, the collective experiences of the individuals involved in freely improvised music do form part of what is an important attribute of any creative rehearsal process, that of dynamic change and evolution of ideas and actions through continued practice in the given artistic context.

The rehearsal process has been demonstrated to form an integral part of the design process itself in the field of interactive music. Johnston, Marks & Edmonds describe the process of developing an interactive artwork as a type of collaborative 'sketching' between software designer and performer, and highlight the use of tools such as Pure Data and Max/MSP in facilitating this type of experimentation in the design process (Johnston, Marks & Edmonds 2005). Hsu and Sosnick have discussed the process of rehearsal as being an integral part of an evaluative framework for comparing two separate interactive music systems (Hsu & Sosnick 2009). The two improvising musicians chosen for the study are involved in two rehearsal sessions per system studied (one short and one long), with the intention of these rehearsals being to familiarise the improvising musicians with the functionality of two separate systems in order to gather feedback in the form of a questionnaire for their evaluation.

Regarding system modularity and flexibility, some systems have been shown to allow the user to choose from a space of possible control strategies that may affect the way the program interacts during a live performance. Rowe's *Cypher* programme enables the user to define the interactive mappings that determine the way in which the system responds to musical input, allowing for experimentation in the rehearsal process of an interactive work using this system (Rowe 1992). As a violinist-composer-programmer, Kimura has explored performance practice issues in the presentation of music for instrumentalist and electronics. Making specific mention of how sound and room acoustics can affect the performance of works in this medium, the author proposes flexible programming strategies to allow for adjustments in the rehearsal process, particularly with respect to dynamic curves (Kimura 2004). Bown's system described above also makes the rehearsal process an integral part of the design of his live algorithm. In Bown's system, a grid of cells is presented to the user to choose which particular generative events are available for triggering by the decision tree's output. The DT itself is also helpfully visualised in graphical form for the user to aid in this process. The configurations of this grid can be changed at will during a rehearsal or performance, and then saved as preset files for recall in future performances (Bown 2011).

From the examples above it is clear that the rehearsal process can form an integral part of both the initial design and further evaluative phases of interactive music system design. In the work of Johnston et al., rehearsal sessions with a professional

performer/composer were demonstrated to have advanced prototype versions of the software artefact through the ability to test and refine aspects of the system in real-time. Hsu et al.'s work uses the rehearsal process to collect experiential feedback from two professional improvisers in order to compare between two already operational interactive music systems. As exemplified by the work of Rowe, Kimura and Bown, it is also clear that the parameters of some interactive music systems, although designed within the particular aesthetics of the system designer, have been left open to change by a performer during the rehearsal process. This is especially useful in the case of performer-programmers, who are in a unique position to make meaningful changes to a system's behaviour in order to satisfy their desired mode of performance, and often leave open these possibilities in the design of their performance interfaces.

However, if interactivity is synonymous with what Pressing has described as 'mutual influence' (Pressing 1990), perhaps discussion of the role of rehearsal should not be limited to its ability to enable the creation, evaluation or controlled manipulation of interactive music systems out of *performance time*. To take the role of rehearsal further in the design of interactive music systems, it would be of additional interest that designers design to enable and reflect the 'cumulative experience of interrelationship' that real-world rehearsal scenarios present. This view is inclusive of the effect that numerous performance time interactions can have on future interactions between a performer and the interactive system, but also the effect that these cumulative interactions can have on the evolution of the system's responses over a longer period of time.

In discussing creative engagement in interactive art contexts, Edmonds et al. have identified three main categories in which engagement with interactive artworks can occur, namely: *attractors* (attributes of an artwork that attract an audience's attention), *sustainers* (attributes that keep an audience engaged past the initial encounter with the work) and *relaters* (attributes that help a *continuing relationship to grow* between audience and system) (Edmonds, Muller & Connell 2006). Although the authors have developed their ideas in the context of audience participation in interactive art, the final two of these concepts also prove relevant to the design of interactive systems designed for use by expert musical performers. Given that the process of rehearsal is an integral part of all performance practices, the final category identified by the authors, *relaters*, is especially relevant to consider in the design of interactive musical systems. The rehearsal process,

in the context of musical performance, is where artistic ideas form and evolve over time in the context of performance practice. In the context of musical system design, a consideration of rehearsal in the design process as complimentary to designing for performance recognises the uniqueness of the rehearsal context as a creative space in musical performance. This also has the potential to open new avenues for musical interactivity based upon the ability of such systems to actively engage instrumentalists over several rehearsal sessions.

2.4 Conclusion

In this chapter I have surveyed key theories, techniques and approaches to designing for musical interactivity as discussed in the literature. In surveying recent system designs, I have highlighted recent trends towards timbral analysis in interactive human-machine performance, and its specific connection to freely improvised musical performance. In addition, the concept *derivation* has been proposed to describe the way in which some musical systems derive both sonic and algorithmic materials and processes from a live performer. Finally, through a consideration of interaction as a 'cumulative experience of interrelationship', rehearsal has been proposed as a valuable area of concern in the development interactive musical systems.

Chapter 3. Methodology

This practice-based research project is concerned with the development and use of interactive performance systems in improvised musical performance. The aim of the research is to understand the emergent creative practice of interactive system design from a practitioner's perspective, and to open up the process of design, development and use of these systems using a self-reflective approach. Through considered reflection-*on*-action, this research has sought to uncover researchable problems in the area interactive musical software development and performance. The interdependent creative practices of interactive software development and human-machine performance have provoked detailed reflection upon issues, concerns and interests relevant to the field at large. The twists and turns of the creative process are articulated in the form of a *narrative of development,* explicating the development trajectory of a major creative software artefact, and highlighting salient research themes that are further addressed through critical analysis in Chapter 7 of this thesis.

The present chapter outlines the self-reflective methodological approach taken in the research, contextualising and justifying this approach with respect to the relevant theoretical perspectives that underpin the enquiry. By engaging with interdisciplinary perspectives on epistemology, practice-based research and reflective practice in the creative arts, this chapter argues for the novel methodological contributions of this research to the field of practice-based research in the creative arts.

3.1 Practice-based and creative-production research projects

As a practice-based research project, the principle means by which this research contributes to new knowledge is through both the process and outcomes of the interdependent creative practices of interactive musical software development and improvised human-machine musical performance. In the following section the research is situated in the context of both *practice-based* research and *creative-production* research projects in the creative arts. Schön's notion of the *reflective practitioner* is then examined as it concerns practice-based research in the creative arts.

3.1.1 *Practice-based research*

According to Candy (2006), a research project is to be considered practice-based if its contribution to new knowledge is demonstrated partly through practice and the outcomes of that practice. Practice-based research projects provide original contributions to new knowledge through both the presentation of artefacts/outcomes developed through practice, and substantial textual contextualisations of these outcomes in the form of doctoral theses or other published materials. Although the significance of the outcomes of the research project must be described in such written documents, "a full understanding can only be obtained with reference to the artefact." (Candy 2006)

Central to the notion of practice-based research is the role of the 'practitioner-researcher' in carrying out the research project. As defined by Robson, a practitioner-researcher is someone who works as a practitioner in a professional setting whilst simultaneously undertaking a 'systematic inquiry' that is of direct relevance to their work (Robson 2002). In art and design contexts, practitioner-researchers hail from a variety of professions including practitioners from visual arts, design, music, dance, creative writing and other backgrounds. Gray (1998) has examined the emergence of practice-led and practice-based research in art and design over the past two decades, and has helped to define the role of practitioner-researchers and the specific approaches to research that occur in art and design contexts. For Gray, the practitioner-researcher identifies "researchable problems raised in practice, and responds through practice," and often plays a multi-dimensional role of that of a creator of research materials (art/design works), observer of self and others and as collaborator in collaborative work contexts (Gray 1998).

Considering the above, it is important to note the reflexive relationship between the practitioner-researcher and the overall context of the specific research inquiry. Due to the central role played by the researcher's practice in the generation of new knowledge, practitioner-researchers are obliged to acknowledge their personal 'stake' in the research, and therefore acknowledge that knowledge in such contexts is subjective and a result of personal construction (Gray 1998). Although practice-based research may be situated within collaborative settings (and may therefore involve the analysis of data collected from third parties), practitioner-researchers must acknowledge their role as active and

influential members of the environment in which their research takes place. This requires an acknowledgement of the subjective nature of the research enquiry, the relationship between the researcher's goals and intentions and the developed research materials (artefacts/outcomes) as well as the relationship between the researcher and any third parties involved in the study.

In a practice-based research project the scope, methodological approach and specific methods of the research inquiry are determined by the close relationship between the practitioner-researcher and their specific domain of practice. In art and design contexts, Gray and Malins suggest that a central characteristic of an 'artistic' methodology is one that embraces a pluralistic approach to the choice of research methodologies and associated methods, and a reflexive approach to research inquiry as a practitioner-researcher. In addition, adaptations and hybrid methodological approaches are common, as practitioner-researchers search for research designs that reflect the unique nature of the specific research inquiry in question (Gray & Malins 2004).

Common to many practice-based projects however is an acknowledgement that practice itself remains the primary tool for knowledge generation, and as such it maintains a central place within the chosen methodology. Practitioner-researchers may use their practice to examine latent research themes, explore developing ideas about practice itself or undertake experiments related to a central topic of interest (Scrivener 2000). Such explorations are carried out in the plane of practice directed towards the generation of artefacts. Although artefacts developed through practice-based research may embody knowledge generated throughout the research process, they must be analysed and evaluated in light of the unique practical context in which they were developed. As cited in Candy (2006), the artist Ross Gibson has articulated the view that the textual component of a practice-based research project should not be focused upon *explaining* the artefact, but rather work towards illuminating and understanding the *process* that gave rise to it. For Gibson, the text is rather:

> …an explicit, word-specific representation of processes that occur during the iterative art-making routine, processes of gradual, cyclical speculation, realisation or revelation leading to momentary, contingent degrees of understanding.
>
> (Candy 2006, p. 9)

Practice-based research projects can therefore be understood as both process and action-oriented with respect to both the generation and application of new knowledge through practice. As a result, throughout the research process research questions and themes may only reveal themselves as a consequence of moves within practice, making the practical domain a space for both generating and responding to research questions.

3.1.2 *Creative-production vs. problem-solving research projects*

The role of practice as method in practice-based research highlights some fundamental issues of research method and the communication of knowledge within creative arts research. Stephen Scrivener has suggested that although practice-based research centres upon the creation of artefacts, there exist fundamental differences between those artefacts that are developed as a response to justified and well-defined research problems, and those projects focused upon creative production that use practice as a vehicle for exploring complex research themes (Scrivener 2000). For Scrivener, this distinction is essential to understanding what kind of knowledge claims can be made by practice-based research projects in art and design. He advocates that in such contexts, due to the focus upon process and the entanglement between artefact development and emergent research interests, self-reflexive practice provides the most suitable means by which researchers in this area might make their research contributions.

Discussing research methodology in doctoral projects in art and design, Scrivener has outlined the differences between traditionally understood *problem-solving* research projects and what he terms *creative-production* projects typical of practice-based research in creative practice contexts. According to Scrivener, artefacts developed in *problem-solving* research projects are presented as either novel artefacts posited to solve well-defined problems, or as improvements upon already existing artefacts (Scrivener 2000). By contrast, *creative-production* research projects are concerned with the generation of artefacts as a means to investigate, explore and define research problems as well as to solve them. Problems arise through the practice of artefact creation, and research themes are developed and explored through subsequent moves in practice. Therefore, the development of artefacts themselves remains the main research focus, and the explication of the process of design and development therefore forms an integral part of the project's contribution to new knowledge (Scrivener 2000).

Although 'technological' research projects are typically more likely to fall within Scrivener's definition of *problem-solving* projects, elsewhere Holmes has argued that the technologically focused creative practices of new media art and other creative software approaches should be understood with respect to Scrivener's notion of *creative-production* research (Holmes 2006). The author argues that much new media art research exists in the form of 'hybrid' projects; projects displaying aspects of both *problem-solving* (technological) and *creative-production* projects (Holmes 2006). As Scrivener's definition of the 'technological' remains somewhat ill-defined, here I argue that the practice of software programming in the creative arts can clearly be characterised as a *creative-production* project, insofar as the creative practice of software development is presented as a core method of the research methodology.

Drawing a distinction between *problem-solving* and *creative-production* approaches, Scrivener outlines some criteria that are generally considered when judging the contribution of artefacts developed as a part of a *problem-solving* research project:

- artefact is produced
- artefact is new or improved
- artefact is the solution to a known problem
- the problem is recognised as such by others
- artefact (solution) is useful
- knowledge reified in artefact can be described
- this knowledge is widely applicable and widely transferable
- knowledge reified in the artefact is more important than the artefact

(Scrivener 2000, p. 4)

The criteria above outline the transferability of the knowledge reified in the artefact, and the expectation that the artefact itself is 'useful'. The artefact that is developed is said to embody a certain 'know-how' of the problem domain, with the knowledge reified within intended to be generalisable and reusable (Scrivener 2000). The artefact itself must be new or an improved version of a previous artefact, and most importantly it must be the solution to a problem that can be recognised and understood by others. Central to this notion is the assumption that the researcher has defined and justified a research problem that the artefact is posited to solve. A central concern of *problem-solving* practice-based research projects therefore involves the careful framing and justification of this problem.

This is by no means a trivial part of the research process, as researchers spend a significant portion of the early part of the project finding and justifying the existence of the problem. Inevitably, this process involves a good deal of trial and error, false starts and wrong turns along the way towards the eventual problem as outlined in the written document. The researcher's initial understandings and expectations of the problem domain may change after encountering unexpected 'back-talk' of the situation during practice, leading them to re-evaluate the initial understandings that gave rise to the problem (Schön 1983).

However, as Scrivener explains, in *problem-solving* research projects this 'finding phase' is decidedly not explicated in the written documentation. The problem is usually presented as a *fait accompli*, justified through reasoned argument leaving the problem-setting process out of the spotlight. According to Scrivener, in such contexts:

> …the student is expected to justify the existence of the problem rather than to explain how it was found. Hence, the problem is usually presented as if it were the natural consequence of rational analysis of past work in the field.
>
> (Scrivener 2000, p. 8)

By contrast, in a *creative-production* research project the production process is often at the centre of the knowledge-making agenda. In these contexts, the artefact is often generated in order to ask questions about a particular topic of interest, to problematise aspects of the domain of practice or to clarify and refine blurry problems and that are in a continual state of definition through engagement in practice. The artefacts themselves might therefore be considered as the end result of a search for clarity and understanding through practice the complex process of practical engagement that gave rise to them. In addition, Scrivener suggests that the personal motivations and past experiences of practitioner-researchers are more likely to determine the trajectory and scope of a *creative-production* project than the ability of their outcomes to solve well-defined research problems. Artefacts and outcomes of such projects are also not motivated explicitly by the need to be the first of their kind, or to prove themselves as improvements upon already existing artefacts (Scrivener 2000).

In light of these observations, Scrivener maintains that in creative-production contexts the artefact itself cannot be correctly judged by the same criteria as those developed through problem-solving projects (Scrivener 2000). The author maintains that the process of artefact creation in creative-production must be foregrounded, and that the practitioner-researcher should articulate the multiple issues, concerns and interests that have given rise to the developed artefacts. Scrivener acknowledges that issues, concerns and interests in such projects are themselves emergent, and that the process of artefact creation (as articulated through practice) embodies the push and pull between problem-setting and problem-solving that is the hallmark of a process of reflective practice (Scrivener 2000). As such, there is a need for practitioner-researchers engaged in *creative-production* projects to clearly articulate and document the process of production, and the moments of reflection in and on action that occur throughout a *creative-production* project. Doing so enables the practitioner-researcher to articulate the emergence of "researchable problems raised in practice" (Gray 1998), and to explain and analyse through reflection-*on*-action how these problems were addressed in practice. In Scrivener's words:

> The relationship between issues, concerns and interests and outcomes in a creative-production project is one that changes throughout the entire process. Thus, unlike a problem-solving project, where we can largely ignore the actual problem setting and solution processes, I am of the view that description of the creative-production process should be the principle means by which students demonstrate that they are self-conscious, systematic and reflective creators.
>
> (Scrivener 2000, p. 9)

In order to account for the process-oriented approach typical of *creative-production* projects, Scrivener outlines alternative criteria with which to judge their contribution to new knowledge:

- artefacts are produced
- artefacts are original in a cultural context
- artefacts are a response to issues, concerns or interests
- artefacts manifest these issues, concerns and interests
- the issues, concerns and interests reflect cultural preoccupations
- artefacts contribute to human experience
- artefacts are more important than any knowledge embodied in them

(Scrivener 2000, p. 6)

Instead of insisting that artefacts created be solutions to well-defined problems, Scrivener places weight on an artefact's role as a *contribution to human experience*. Artefacts may make claims of originality, but rather than having to justify their novelty against criteria of utility or generalisability of knowledge they embody, they may be demonstrated as original within a clearly defined cultural context. Furthermore, the artefacts must be developed in response to issues, concerns and interests that reflect cultural preoccupations, and these issues, concerns and interests must be manifest in the artefact itself. Finally, in direct contrast to artefacts developed in problem-solving projects, Scrivener suggests that artefacts themselves remain more important than any knowledge embodied in them (Scrivener 2000).

3.1.3 *The reflective practitioner*

As Scrivener has outlined, where practice is used as a core method for knowledge-generation, self-reflection both *in* and *on* practice becomes an indispensible tool for both improving practice and communicating the unique insights gained from these reflections to wider audiences. What separates practitioners from those practitioners engaging in practice-based research (practitioner-researchers) is the rigour with which such self-reflection is undertaken. To the practitioner-researcher, self-reflection becomes a core research method that enables the reader to enter into the complex cycles of action present during practice that have culminated in the developed artefact. Practice-based researchers therefore make implicit or tacit knowledge communicable through self-reflexivity, and in doing so reveal insights into creative process and point towards larger research themes beyond the self.

Self-reflexivity is a central concern of Donald Schön's theory of *reflective practice* (Schön 1983). Schön's notion of the 'reflective practitioner' is an often-cited framework for understanding what is knowable by practitioners in various domains, and how this knowledge base is harnessed and enhanced through self-reflexivity inside and outside the plane of practice. Schön has defined two important and complimentary types of reflection that form part of reflective practice; that of reflecting-*in*-action, and that of reflecting-*on*-action (Schön 1983). The former, reflection-*in*-action, is characterised as the reflection of practitioners during the actual act of practice, describing the kind of on-the-spot decision making informed by both the situation at hand and the practitioner's body

of knowledge and experience in the practical domain. By contrast, reflection-*on*-action is characterised as being further removed in time from the act of practice itself. As a theoretical and analytical appraisal of the practical situation, reflection-*on*-action is considered a critical skill for professional development and practice-based research that is related to the act of review, evaluation and analysis (Gray & Malins 2004).

For Schön, reflection-*in*-action is the complex interactive cycle of action and reflection that occurs in everyday practice. Schön places importance on the practitioner's tacit professional know-how – their 'knowing-in-action' – and the interplay between problem *setting* and problem *solving* that occurs within plane of practice. Reflection-*in*-action occurs when a practitioner simultaneously makes moves (changes) in the domain of practice, assesses the implications of these moves on the current task, evaluates and appreciates new situations or problems that arise and takes further actions that either confirm or deny the validity of their past history of actions (Schön 1983, p. 94). This iterative process is fundamentally directed towards transforming a situation from its present state into something better, and forms an integral part of how professionals advance in their everyday practice. Understanding the outcomes of practice therefore requires an understanding of the on-the-spot decisions made by practitioners, and therefore the process of reflection-*in*-action. This process is however generally unspoken and may be unremarkable to the practitioner during the act of practice, as it consists of personal and context-bound responses to a problem domain. For practitioners, reflecting-*in*-action is at once an expression of *know-how* in the domain influenced by past experiences, and also a function of the specific circumstances of the particular creative task or situation. The practitioner is at once posing problems, testing their applicability through action, evaluating the implications of these actions and appreciating new understandings of the domain of practice in light of the changed context.

According to Schön, when a practitioner reflects-*in*-action they engage in a "reflective conversation with the situation" (Schön 1983, p. 76). Such conversations take place when practitioners make changes to the situation, evaluate the effects and implications of these actions provided by the situation's 'back-talk', and subsequently respond to the situation anew having appreciated the implications of their past actions upon future decisions. Responding to this 'back-talk' the practitioner "reflects-in-action on the construction of the problem, the strategies of action, or the model of the phenomena, which has been

implicit in his moves." (Schön 1983, p. 79) The moves made by practitioners in the act of practice are founded upon tacit knowledge; models and appreciations of a situation that have accumulated from past experiences in the practical domain. In addition, these inherently subjective models and appreciations are further expanded and re-evaluated in direct response to unexpected problems encountered in practice. Schön's 'reflective conversation' metaphor therefore acknowledges the importance of action to the knowledge-making agenda, and the entangled nature of action and reflection in the process of practice.

Schön's conception of professional 'know-how' exercised and expanded through reflection-*in*-action emphasises its essentially tacit nature; considering appropriate courses of action in the moment the practitioner makes decisions 'on-the-fly', without sustained critical insight. The practitioner relies upon both their personal experience and the specific problems posed by the present situation in order to determine the next course of action. Reflection-*on*-action, by contrast, is a type of retrospective appreciation, evaluation and understanding of decisions made through critical reflection on past practice. When a practitioner reflects-*on*-action, they do so by evaluating decisions made in practice and the implications of those decisions on the future shape of the creative task, and potentially on their methods of practice in general. Reflecting-*on*-action requires a thorough understanding of the reasons for making decisions during practice, and of the specific context in which such decisions were made. According to Scrivener (2000), reflection-*on*-action is driven by a desire to learn from experience, in contrast to reflection-*in*-action which is driven by a need to deal with unexpected and unintended consequences of action within a practical context. When a practitioner reflects-*on*-action they: "reflect on knowledge and ways of working automated over an extended period" (Scrivener 2000, p. 9). In other words, reflection-*on*-action requires the practitioner to reflect upon and understand the process of reflection-*in*-action itself.

Fundamental to the notion of reflective practice therefore is an acknowledgement that knowledge generation in practice-based research is entangled with action, and that professional knowledge emerges through engagement in and critical reflection on practice itself. The process of reflecting-*on*-action requires the practitioner-researcher to appreciate past practice as a cumulative history of on-the-spot decisions made in response to the specific circumstances of the creative task. Importantly, in reflective

practice the practitioner-researcher's role as both a problem-*setter* and problem-*solver* is acknowledged. Reflection-*in*-action is foregrounded as the means by which practitioners deal with uncertainty in the act of practice. Part of this process involves the practitioner reframing his or her understandings of a problem in response to the situation's back talk. Such reframings often also include redefining the problem itself.

For Schön, the reality of professional practice is at odds with traditional notions of technical rationality that contend that practice is a form of "instrumental problem solving made rigorous by the application of scientific theory and technique' (Schön 1983, p. 21). Through an analysis of various examples of professional practice Schön argues for an understanding of practice as responsive, interactive and in a constant state of flux. His analysis points towards a recognition that practitioners are actively making sense of their own actions in response to circumstances that cannot have been predicted outside of real-time practice. Although learned theory and technique inform a practitioner's models and appreciations of the practical domain, these models are subject to modification in the face of uncertainty, and it is through reflective practice that practitioners expand their understandings of the domain, set new problems and ultimately address and develop elements of professional theory and technique.

3.2 Reflective practice as research methodology

As outlined in the previous sections, practice-based research in the creative arts places a great deal of emphasis on the process of practice itself, and how it can access and reveal research themes and questions that could not have been easily defined or posed from the outset of a research project. From this perspective, Schön's conception of knowing-*in*-action is integral to a practitioner's ability to guide and shape a creative task, as well as their ability to identify and reflect upon relevant lines of enquiry through practice. Taking Scrivener's lead, I have articulated how reflective practice may be used as the principle means by which practitioners might analyse, synthesise and communicate the complexity of research themes emerging from practice in such research contexts.

For the artist-researcher, reflective practice is the means by which research themes can be made communicable to a wider community. Concerns, issues and interests identified through practice are engaged with by the practitioner-researcher, and explicated in a

written research document. In this thesis, self-reflective practice is used to highlight research themes that have surfaced through the development of a significant computational artefact and its use. Through the *narrative of development*, certain themes are identified and demonstrated as being acted upon in order to advance practice. However, the interdependent creative practices of software development and human-machine performance have also surfaced more fundamental themes surrounding the design and use of interactive musical systems. These themes are the emergent result of both direct engagement in the practical domain, but also through sustained and critical reflection upon further literature in the field. Chapter 7 is presented as the culmination of these sustained critical reflections.

This sustained form of reflective practice represents the reflection-*on*-action engaged in throughout this research. In the following sections of this chapter I contextualise the reflection-*in*-action engaged in throughout my creative practice in some depth. With reference to the work of Andrew Pickering, here I argue that technologically focused creative practice research should be understood as part of a temporally emergent negotiation between human and non-human agencies. Pickering's *mangle of practice* is introduced to highlight the unique position of technological creative practice in advancing new knowledge, and the situated nature of such knowledge generation. Although Pickering's ideas were developed in the context of empirical scientific practice, his concepts relate well to Schön's conception of practice as a 'reflective conversation with the situation'. It is the emphasis on the agency of the material in Pickering's work that is the most relevant to technologically focused creative work. Following discussion of Pickering, the chapter concludes with a consideration of McLean and Wiggins' notion of *bricolage programming* in the creative arts. As an embodied conception of artistic programming practice, *bricolage programming* highlights both the exploratory and emergent nature of artistic software programming.

3.3 Introducing the mangle of practice

With its roots in the Sociology of Scientific Knowledge (SSK), Andrew Pickering's *Mangle of Practice: Time, Agency and Science* (Pickering 1995) is a performative understanding of scientific practice that acknowledges the entangled nature of human and material agency in the real-time production of scientific knowledge. His work provides a

thorough understanding of how researchers engaged in 'goal-oriented practice' converse with their materials, arguing that scientific knowledge generated from empirical research should be understood as the result of the push and pull between the researcher, their machines, and the material world under investigation. The importance of this theoretical perspective is that it treats empirical scientific practice as temporally emergent and situated in its specific practical context. Pickering describes this view of scientific knowledge as a shift towards what he calls the *performative* idiom, where the temporality of practice in all its complexity is foregrounded. To Pickering, this perspective stands in contrast to the dominant view of scientific culture as *representational*, where science is cast as 'an activity that maps, mirrors, or corresponds to how the world really is.' (Pickering 1995, p. 5)

This shift to the performative places scientific outcomes, theoretical understandings and developed artefacts as direct consequences of the interplay between human and machine agencies in practice, a process that Pickering defines as the *mangle of practice*. What the mangle suggests is that through the entanglement between human and the material, new questions are continually posed as new understandings of the problem domain arise through practice, as surfaced through *material agency*. This view is founded upon Pickering's argument that the world "is not, in the first instance, [filled] with facts and observations, but with *agency*". To Pickering, understanding scientific discoveries entails an appreciation of the complex interactions between both human and non-human agencies, thereby situating knowledge production as an interactive process that happens in and through time. For Pickering, "the world is constantly *doing things*, things that bear upon us not as observation statements upon disembodied intellects but as forces upon material beings" (Pickering 1995, p. 6), and scientific practice must therefore be understood in relation to this field of influences. As a theoretical perspective, Pickering's *mangle of practice* therefore problematises the notion of objectivity and closure in empirical scientific practice, situating scientific results within a temporally emergent process of give and take between humans and the material world.

Appropriating Pickering's ideas from the field of Sociology of Scientific Knowledge (SSK) to creative practice, *the mangle of practice* suggests that artworks (as with scientific facts) represent the end-point of a search for knowledge that is characterised by a negotiation between human and material agency in the plane of practice. From this

perspective the most suitable course of action is to open up practice itself to explicitly reveal the cycles of resistance and accommodation from which arise artistic practices, theoretical concepts and artistic outcomes. From the perspective of self-reflective practice-based research, Pickering's ideas point directly towards methods that get inside this process and allow a faithful explication of the mangle as it happened. As Jefferies has suggested, self-reflective methodologies are well-poised to provide an insider perspective of the messy nature of creative practice, giving the reader insight into the processes of mangling that are underway in such contexts (Jefferies 2012).

In this research, Pickering's *mangle of practice* provides an epistemological framework informing my overall methodological approach to examining the creative practice of interactive systems development, and the related performance practices that have evolved from the use of my software artefacts. Pickering sees practices themselves as being subject to the mangling process, and it is through reflective practice that I elucidate this process in Chapters 4, 5 and 6 of this thesis. Pickering's mangle is used to justify a self-reflexive approach to analysing practice, acknowledging the present research trajectory as being tied to the emergent understandings of software artefacts that have developed over time. The research themes that have been investigated through practice should therefore be situated in their temporal context, understood as products of an emergent interplay between human and material agency in the processes of artefact creation, use and evaluation.

Much of the uniqueness of Pickering's 'mangle of practice' lies in its fundamental questioning of commonly held epistemological perspectives in the Sociology of Scientific Knowledge. His critique of SSK is a critique of the positions of both social constructivism and technological determinism, and in particular the inability of either of these perspectives to deal with the outcomes of research from the perspective of engagement in practice. To Pickering, their understandings of scientific knowledge are *atemporal* (Pickering 1995, p. 4). A social constructivist perspective, typical in the field of SSK, contextualises scientific discoveries with respect to relatively fixed social and cultural factors influencing scientific knowledge and practice from the outside. From such a perspective, technological innovation as exemplified by advances in the scientist's apparatuses might therefore be understood as causally linked to larger societal and cultural factors. With respect to technological determinism, Pickering notes that a

temporally emergent understanding of scientific knowledge is also at odds with perspectives that attest that "the social is continually refashioned around technological imperatives." (Pickering 1995, p. 169) Pickering suggests that such an antihumanist version of history glosses over the complexities of engagement in the plane of practice that shape both society and our technology.

To Pickering, neither social constructivism nor technological determinism acknowledge the entangled nature of the relationship between human and material agency, and therefore neither can be used to explain the significance and depth of the outcomes of scientific research. Viewing scientific practice as temporally emergent enables one to understand its outcomes in relation to both the cyclical patterns of problem setting and problem solving inherent in practice, as well as the active and influential role that *material agency* plays in this process. To this end, Pickering highlights the vital role played by the researcher's apparatuses (their *machines*), with their use and reconfiguration – their intersections with human agency – credited with forming a large part of the understandings that arise from scientific practice. As part of the mangling process, materials and machines, as well as interpretive and phenomenal accounts of the research domain are 'interactively stabilised' through practice, as the researcher finds unique *accommodations* to unforeseen *resistances* that arise unexpectedly in the plane of practice. This cyclical process puts both human and material agency in an interactive relationship, a process Pickering describes as a *dance of agencies* (Pickering 1995, p. 21).

3.3.1 *The mangle and reflective practice*

Although developed in the context of empirical scientific research, it is the importance that Pickering places upon process that suggests that his 'mangle of practice' has much to offer practitioner-researchers engaged in reflective practice in the creative arts. In seeking to understand scientific knowledge as entangled in temporally emergent practice, Pickering makes extensive use of the personal accounts of practitioners in his own analyses. Importantly, the author attests that such personal accounts might be seen as an integral part of the 'mangling process' engaged in by the practitioner, and might be viewed as "products of the dialectic of resistance and accommodation, at once retrospective glosses on emergent resistances and prospective elements of strategies of

accommodation." (Pickering 1995, p. 53) Due to their foregrounding of process, Pickering's acceptance and encouragement of the use of scientists' personal accounts might therefore be interpreted as an acknowledgement of the role of reflective practice in shaping scientific knowledge. Pickering's appraisal of such reflective accounts emphasises the emergent nature of goals, plans, and interpretive accounts and the important role that practitioner reflection plays in the process of knowledge generation.

Pickering's perspective on practice as both interactive and temporally emergent finds obvious common ground with both Schön and Scrivener's conceptions of reflective practice in both professional and creative practice contexts. Schön's notion of practice as a 'reflective conversation with the situation' and his emphasis on the practitioner responding to the situation's 'back talk' are analogous to Pickering's understanding of practice as a dance of human and material agencies. What Pickering's perspective shows is that machines and the material world profoundly affect subsequent decisions made by practitioners in the plane of practice. The way in which the practitioner responds to unexpected *resistances* encountered in practice might be said to take a familiar form, that of reflection-*in* and *on* action as the practitioner searches for appropriate *accommodations* to these hurdles in order to move the practice forward.

By seeking to fully understand the real-time context of empirical practice and its implications for scientific knowledge, Pickering advocates for a process-driven conception of knowledge production in which the messy nature of problem solving is examined as it occurs through time. Recalling Scrivener's analysis of research approaches detailed in Section 3.1.2, we might therefore understand Pickering as arguing for the communication of the various twists and turns that exist in the problem finding phase of a *problem-solving* research project. Although the context in which Pickering developed his ideas differs from the artefact-driven approaches of practice-based research, his ideas resonate with conceptions of interactivity and emergence experienced by those engaged in such research in the creative arts. By situating knowledge production in the plane of practice rather than as an atemporal result of practice, Pickering's ideas suggest that artefacts (outcomes) developed in practice-based research should therefore be understood only through a consideration of the specific and changing circumstances that gave rise to their production.

3.3.2 *Connecting the mangle to creative arts research*

Considering the original context from which Pickering developed his ideas, it is important to examine the potential of its application by analysing how researchers outside of the Sociology of Scientific Knowledge have appropriated the mangle to their needs. New media artist and researcher Ashley Holmes has discussed Pickering's mangle of practice with particular reference to reflective practice in new media research (Holmes 2006). Situating new media creative practice within a critique of approaches to practice-based research outlined by Scrivener (2000), Holmes points to Pickering's ideas as providing alternative epistemological and ontological grounds with which to understand technologically focused creative practice. Holmes' appropriations of Pickering's ideas to this context are focused upon the connections between Pickering's dance of agencies metaphor and the "rigour of procedural engagement with software tools and/or algorithms' that characterises much new media creative practice (Holmes 2006, p. 6).

Holmes describes such engagements with technology as forming *grammars of practice*; a series of routinised practices that help to shape the trajectory of a given technological project. It is suggested that such grammars of practice are observable in and central to many fields of endeavour, therefore justifying the link between these seemingly disparate areas of practice. For Holmes, the connection between scientific practice and creative practice is concerned primarily with the negotiations between both human and material agency that typify these domains of practice, hence the invocation of Pickering's 'dance of agency'. Furthermore, the author highlights the connection between Pickering's notion of the *tuning* of goals and intentions in scientific practice, and the iterative cycles of planning, implementation, appraisal and review that are characteristic of artefact creation in new media software development (Holmes 2006, p. 6).

Further justifying his appropriation, Holmes interrogates Pickering's concepts in more detail, turning his attention to Pickering's definition of material agency and its relationship to digital arts practice. Here Holmes extends Pickering's concept of material agency to encompass digital technology, situating this concept with respect to software development in new media arts practice. Citing Steigler, Holmes illustrates that both human and technological (software) agencies can be seen as temporally emergent and mutually influential through practice (Stiegler 1998). Considering Pickering's concept of

material agency in its original context however, Holmes' appropriation requires further clarification. In Pickering's work, *material agency* is referred to throughout the text both in relation to the *material world* (the object of empirical scientific research) and the world of *machines* developed to frame and capture this material agency (the scientist's apparatuses).

This is an important distinction to make in this context, as the separation between these two types of agencies is directly connected to the doing of empirical scientific practice. From Pickering's point of view, material agency is encountered in both the use of machines to *frame* material agency, and in the *capturing* of material agency itself as a part of this framing process (Pickering 1995, p. 83). The importance of this distinction goes to the heart of Pickering's conception of his 'dance of agency', and the associated *tuning* process Holmes' adapts to his purposes. According to Pickering, the dance of agencies is characterised by periods of interchange between activity and passivity on the part of the practitioner during their interaction with their machines (Pickering 1995, p. 21). The active role relates to the building of and modifications to a machine designed to capture material agency, whilst the passive role sees the practitioners observing the results of the machine's capture of material agency. Viewed from this perspective, Pickering's dialectic between activity and passivity necessitates a conception of machine agency that acts in the face of human passivity during processes of scientific observation.

A challenge for technologically focused creative-production research is in reconciling this conception of material agency with alternative understandings of machine agency in creative practice. Returning to reflective practice, Schön has noted that much artistic practice relies upon the artist entering into a 'reflexive dialogue with materials', describing the personal and interactive relationship between the practitioner and their subject matter. Considering design as a 'reflective conversation with the situation', Schön acknowledges the entangled nature of both human and material in creative practice:

'Typically, [the designer's] making process is complex. There are more variables—kinds of possible moves, norms, and interrelationships of these—than can be represented in a finite model. Because of this complexity, the designer's moves tend, happily or unhappily, to produce consequences other than those intended. When this happens, the designer may take account of the unintended change he has made in the situation by forming new appreciations and understandings and by making new moves. He shapes the situation, in accordance with his initial appreciation of it, the situation "talks back," and he responds to the situation's back-talk.'

(Schön 1983, p. 78)

To use Pickering's terminology, the designer is therefore engaged in a process of *interactive stabilisation* whereby the consequences of action (as exhibited by the situation's 'back-talk') shape his or her subsequent moves in practice. Each move is informed by the consequences of action, leading to a process in which concepts, goals, materials and artefacts stabilise over time. This 'dance' between the human and material plays out as a human search for accommodations to material resistances encountered in practice. Each search for accommodations by the designer has the potential to create unforeseen resistances, resulting in outcomes that can only be understood as an emergent result of this interactive relationship.

Another central component of Pickering's mangle is the dialectic encountered between *resistance* and *accommodation* in practice. For Pickering, the 'dance of agency' is characterised by the interplay between *resistances* encountered in material agency, and human *accommodations* to these resistances in practice. In the context of creative programming practices, Pickering's conceptions of activity and passivity may be expanded to include the material agency expressed by the output of a computer algorithm. The dialectic between resistance and accommodation lies at the core of the mangle of practice, articulating a process in which goals and plans, interpretive and phenomenal accounts are mangled through the practice towards cultural extension (Pickering 1995, p. 22).

According to Pickering, a resistance "denotes the failure to achieve an intended capture of agency in practice" (Pickering 1995, p. 22). Although one might conceive of all artists encountering some form of resistance throughout the creative process, Pickering's sense of the word is specifically connected to the *capturing* of material agency, a core concern of empirical scientific research. However, the notion of resistance lends itself to a conceptualisation of creative practice as interactive and temporally emergent. In the context of creative practice, a resistance may be understood not as a 'failure to capture material agency', but as a *disconnection* between what is initially strived for in practice and what is found or discovered. Accommodating these disconnections can then take a variety of forms, as indeed they do in Pickering's model. This may include the artist changing the conditions that gave rise to this disconnection (taking another course

of action), or by incorporating the new discovery into the conceptual space that defines the practical domain, re-orienting the direction of practice itself. This enlargement of the artist's conceptual space is analogous to Pickering's understanding conception the *mangling* of a scientist's *interpretive accounts* of a phenomena under study (Pickering 1995, p. 74), through engagement in practice.

3.4 Bricolage programming and reflective practice

The entanglement between human and material agencies, as articulated by Pickering, is of direct relevance the creative practice engaged in by artist-programmers. A large part of the creative process in the programming of new software artefacts for interactive music involves a heavy dose of trial and error. In addition, the software programmer often begins with a set of concepts and goals that provide the stimulus for the creation of a new software artefact, however it is in the direct relationship between the programmer and their code that the overall artistic direction of the project begins to take shape. McLean and Wiggins (2010) have paid specific attention to the process of software programming in the creative arts, developing a theoretical understanding of the interactive and iterative process of programming as an artistic practice and the vital role played by the machine in this process. The authors introduce the notion of the programmer as *bricoleur*, an approach to programming previously described by Turkle and Papert (1992). *Bricolage programming* is contrasted with 'planned' approaches to software engineering, in which unexpected results are viewed as 'mistakes' or 'missteps' that deviate undesireably from an initial design. According to Turkle and Papert, *bricoleurs* by contrast "have goals but set out to realize them in the spirit of a collaborative venture with the machine." (Turkle & Papert 1992, p. 136)

From the above foundation described by Turkle and Papert, McLean and Wiggins outline an embodied view of programming in artistic contexts, maintaining that the interactive relationship between a programmer and their code enables an emergent process of design to occur. Through direct feedback from the execution of code – an expression of machine/material agency – programmers make artistic decisions that cannot be anticipated outside of practice. These decisions are directly influenced by the unexpected feedback of the software they are developing. Such deepening connections between artist and code leave open a space in which trial and error and tangential

experimentation are guided by the moment-to-moment experience of writing, executing and testing a software artefact (McLean & Wiggins 2010). This form of mediated decision-making recalls Schön's concept of material 'back talk' in the practical domain, an integral part of a practitioner's reflection-*in*-action.

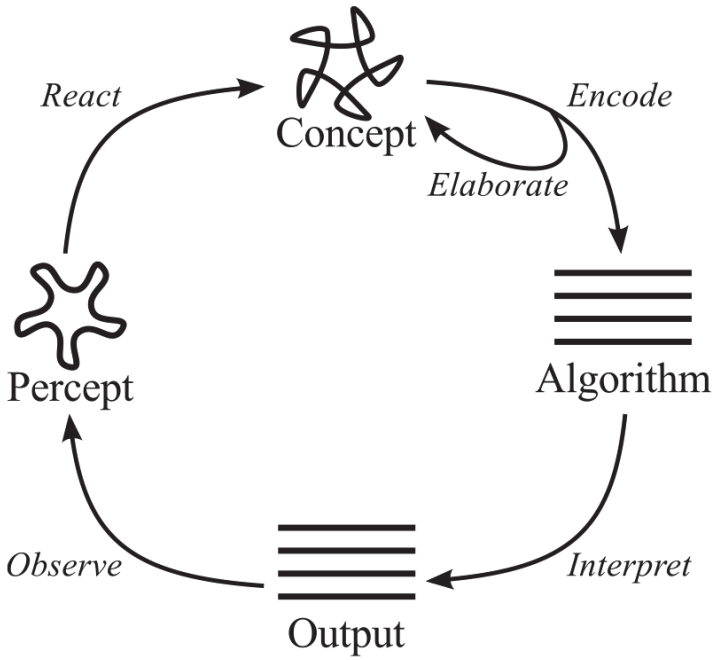


Figure 1: The Bricolage Programming cycle of action and reaction (McLean & Wiggins 2010, p. 2)

Bricolage programming is characterised as a cyclical process of action and reaction between the programmer and their code. In this cycle the programmer's initial concepts are encoded and elaborated into algorithms, whose output is then observed and appreciated by the programmer leading to potential revision and extension of their original concepts. In Figure 1 above, the elaboration of a programmer's ideas is represented as an inner loop between the programmer's concepts and the encoded algorithm. The outer loop represents the output of the algorithm followed by the programmer's perception of this output, and finally how this percept affects the programmer's initially defined conceptual space. The role of perception in the outer loop is crucial for the direction of the artistic project, with the program's output being evaluated by the programmer against the initial concept of what the program might have been. The feedback loop is characterised as being formative and evolutionary for the project, as the initial concept itself is then open to revaluation and change. The authors suggest that externalising the interpretation of the programmer's concept to a computer

program often results in a greater degree of surprise and unpredictability to the creative process. (McLean & Wiggins 2010)

With reference to Wiggins' 'Creative Systems Framework' (CSF) (Wiggins 2006), the authors define creativity as "a search in a space of concepts" (McLean & Wiggins 2010, p. 4) and introduce the notion of *transformational creativity* as it applies to artistic programming, a concept developed by Boden (2004). As the business of the artist-programmer is to develop software, the search the authors describe necessarily involves the encoding of concepts into algorithms to be interpreted by the computer. Interestingly however, the authors suggest that this creative practice has as much to do with the observation and perception of the computer's interpretations of the encoded algorithms as the actual encoding itself. As the authors note, in such contexts "observation may itself be a creative act." (McLean & Wiggins 2010) The boundaries of the programmer's initial search space (which define their concepts, intentions and artistic ideas) are continually being expanded and altered through the perception of the machine's interpretations of their encoded algorithms. This can be understood as a genuinely performative view of artistic computer programming, in which both the human and the machine play equally important roles in advancing the artistic work.

Figure 2 illustrates the *bricolage* programming cycle once more, further annotated with elements from Wiggins' Creative Systems Framework. The CSF incorporates three distinct elements that McLean and Wiggins have applied to their concept of *bricolage* programming, each helping to further refine this initial cycle. According to the CSF, *R* defines the search space of the programmer's concepts, *T* defines the traversal of this space whilst *E* defines the evaluation of concepts found in the space. With the CSF applied to *bricolage* programming, the authors regard the act of artistic programming as interactive, with the computer's output concretely influencing the programmer's concepts about their own work. Importantly, the programmer's conceptual space – *R* – is not fixed or closed but subject to expansion and change through interaction in the plane of practice. When viewed as a search space, the act of elaborating and encoding concepts into a working algorithm is considered a means by which the programmer *traverses* – *T* – the search space suggested by *R*. This traversal is characterised by the authors as a strategy by which the programmer employs "techniques and conventions employed to convert concepts into operational algorithms." (McLean & Wiggins 2010, p. 6)

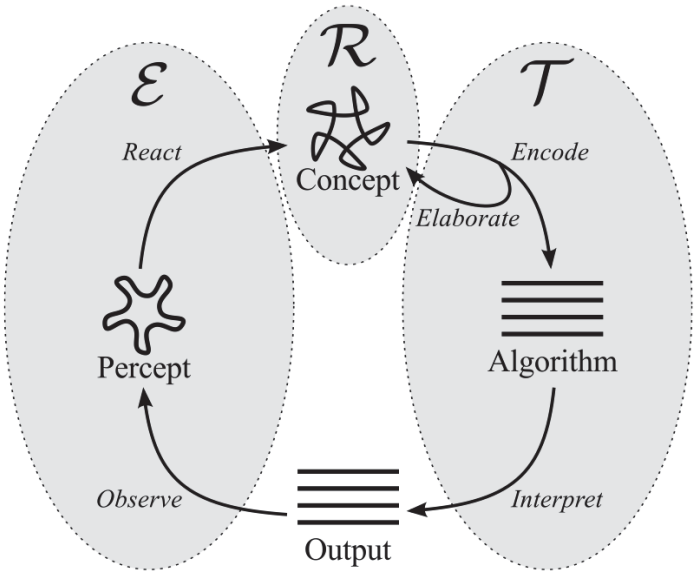


Figure 2: The Bricolage Programming cycle of action and reaction annotated with the components of the Creative Systems Framework (McLean & Wiggins 2010, p. 6)

3.5 Bricolage programming and the mangle of practice

McLean and Wiggins' concept is a useful description of the type of programming engaged in through this research project. Although each software artefact may be initiated by a series of specific goals, the *bricolage* programming cycle describes the emergent process of artefact development in contrast to top-down software engineering approaches to design. As a performer-developer, the software artefacts developed throughout this thesis were borne from exploratory processes that sought to investigate rather than to solve design problems. Chapters 4, 5 and 6 of this thesis articulate this process by detailing the development of my creative practice, highlighting the way in which technical outcomes engendered further areas of exploration that could not have been foreseen in advance. In addition, my unique position as a performer-developer ensured that physical, musical performance formed an integral part of the development of my software artefacts. When such algorithms are designed to interact with a human performer, the close relationship between performance and development not only aids in the interpretation stage of the *bricolage* model, but also continues to redefine the initial concept from which the algorithm is encoded. Musical performance is therefore

integrated directly into the feedback loop between myself as developer, and the output of these developing software artefacts.

3.6 Data Collection

As has been detailed in this chapter, self-reflective practice has been the primary methodological approach in this research. The development narrative pursued in Chapters 4, 5 and 6 is informed by the close consultation of various complimentary sources of data recorded between 2010 and 2014. These data sources therefore aided in the process of critical self-reflection. Three primary sources of data were collected throughout this project that give insight into the development process: research memos, computer code in the form of Max 'patches', and audio recordings. Throughout the three central 'wayfinding' chapters of this thesis, this research data was consulted extensively in order to articulate both the technical implementation of my developing software systems, and also to give valuable insights into the reasons behind certain creative decisions.

Whilst many of the Max patches are referred to directly within the text, the various research memos and audio recordings generated throughout the project are not explicitly referred to in this thesis. Instead, these valuable forms of qualitative data were continually consulted in order to trace the underlying technical and aesthetic trajectory of my creative practice. Below I outline the three sources of data in detail, and how they were used in this project.

3.6.1 *Research memos*

Throughout this research project I have undertaken a process of self-reflexivity through systematically recording research memos in a variety of forms. The memos exist as either handwritten entries written in A4 notebooks or as digital notes taken using the free cloud-based software program Evernote[16]. I have collated five A4 notebooks that contain memos recorded between January 2010 and May 2012, and a digital archive of Evernote data ranging from August 2010 to present. These research memos have served multiple purposes. They have been an indispensible part of the iterative development of

[16] https://evernote.com

my creative work, enabled prolonged reflection on my specific area of creative and performance practice, helped trace emerging themes related the field at large and provided a site for the planning of future creative moves. As the basis of my self-reflective practice, these memos are therefore the primary means by which I have engaged in reflection-*on*-action throughout the duration of my doctoral project.

Although there is often considerable overlap between various types of memos, these writings may be categorised into the following groups:

- Technical implementation plans
- Reflections on current development
- Programming problem-solving
- Reflections on performances
- Reflections on broad aesthetic goals
- Brainstorming next creative moves

Research memos were particularly useful in understanding the rationale behind the aesthetic decisions made throughout this research. By triangulating information from these written documents with the documented Max 'patches', a rich and accurate account of this complex creative trajectory spanning several years was created. In addition, the theoretical perspectives articulated in Chapter 7 of this thesis developed directly from research memos written throughout the development of my software artefacts. This illustrates that the process of reflection-*on*-action in this project was conducted at various stages throughout the research project.

3.6.2 *Max patches*

The software development undertaken throughout this project makes use of Cycling '74's *Max* software (formerly 'Max/MSP'), a 'visual programming language for media' (Cycling '74 2014). For the user of the Max environment, the act of programming requires the arrangement, structuring and manipulation of high-level graphical objects originally developed in the programming languages of C, Java or JavaScript. The source code that underlies a Max program written by the user lies beneath the surface of the documents created by the Max programmer. A finalised program in the Max

environment is called a Max 'patch'. As the act of developing a finalised program using the Max environment depends upon manipulating these higher-level objects, the role of the programmer-user is best understood with reference to the user-generated patches themselves, and not the underlying source code as represented in these lower-level languages. The Max patches files I have produced throughout this research contain all the information needed to interpret the creative intentions behind the functioning software artefacts. Therefore, for the purposes of this thesis these files may be conceptualised as the *source* of the resultant software artefacts.

Several hundred documents (patches) have been created using the Max environment as a part of this research project. Some of these patches are small programs that were developed to test specific programming ideas, whereas others are much larger and incorporate numerous other Max patches and other files as software dependencies. This working method encouraged code re-use and adaptation, as smaller programs were used as building blocks and eventually incorporated into larger working programs. It should be noted that this type of development differs from a hierarchical, project-centred approach to software development. Many of these files began their existence as rough 'sketches' that were later incorporated into working systems, and organising this large collection of files into a formalised code repository such as git[17] was not considered vital to my working methods in the Max environment.

Whilst some of the Max patches developed in this project included embedded comments on their functionality, many of these files did not. In order to make sense of the emergent development of my software systems, I have maintained an extended a textual description and preliminary analysis of these 'source' documents developed throughout the duration of this project. This document was developed retrospectively by surveying the archive of Max patches stored on the hard drive of my personal computer, describing their contents and functionality as well as grouping them with related files in order to account for the appearance of larger streams of creative enquiry. Given the emergent and naturally messy nature of my development process, this retrospective account of my work aided in making sense of the programming trajectory, and was triangulated with the

[17] https://git-scm.com/

'programming problem-solving', and 'technical implementation plan' research memos outlined above.

3.6.3 *Audio recordings*

Audio recordings made throughout the development of the *_derivations* software have provided an invaluable source of data to aid in reflection-*in*-action throughout the development process. Over time these recordings have also enabled critical reflection-*on*-action that has helped in shaping the theoretical understandings of human-machine performance practice discussed in this thesis. Throughout the development of the *_derivations* system several hundred audio recordings were made, some in order to document various stages of design process and others as direct outputs of my creative practice. In addition, throughout extended use with the *_derivations* software I have accumulated a large library of rehearsal databases or 'sessions' created with the software itself. Included in such databases are the reference audio recordings that document the solo contributions of instrumental interactions with the *_derivations* software.

The various audio recordings made throughout this project can be grouped into the following categories:

- Tests of individual system components
- Simulated system performances
- Studio human-machine performances
- Live human-machine performances
- Session database reference recordings

As mentioned above, the audio recordings created throughout the research are not referred to specifically throughout the thesis, nor have they been the objects of formal musical analysis. The majority of these are treated as creative artefacts of my developing interactive systems, and as such these were used as objects of critical reflection for both advancing practice and provoking broader theoretical considerations. In the case of session database recordings, given the machine listening and timbral matching approaches discussed in this thesis, these recordings were used extensively in advancing

the technical implementation of *_derivations phrase matching* algorithm (see Section 6.5 for more detail).

3.6.4 *Data excluded from the research*

Although developed in the context of my personal creative practice, it must also be noted that numerous third party musicians have also used the *_derivations* software in performance. Some of these musicians have worked directly with myself in close collaboration, whilst others have made use of the software independently in their own work. Through introducing *_derivations* to third parties I have had many fruitful conversations with musicians regarding interactive performance and human-machine improvisation. Whilst surveying the experiences of other musicians interacting with *_derivations* would have provided interesting qualitative data, in this research a decision has been made not to include information pertaining to these users' experiences with the software. As a performer-developer creating an idiosyncratic system such as *_derivations*, a self-reflective approach enabled me to focus upon the changing approach to development and use of this artefact from inside the *mangle of practice*. Such an approach provided a rich and unique perspective on the development of new technological artefacts and emergent artistic practices. Conducting user studies on expert musicians may have yielded interesting additional perspectives on the developing artefact and performance practice, however in the context of the current research it is my belief that such perspectives would have been outside the scope of this current research.

3.7 Conclusion

Through a cyclical process of action and evaluation, the creative practice discussed in this research relies upon the emergent and reciprocal relationship that has developed between both human and material agencies. The developing agency of my technical artefacts *mangled* my goals and interpretations of the domain of interactive, human-machine performance. The form of development followed throughout this research, as discussed in the following three chapters, has followed a *bricolage* approach to creative programming. By following such an open and exploratory form of software development, significant software artefacts were developed that embodied my developing understandings of interactive musical performance.

By engaging in self-reflective practice, both reflection-*in*-action and reflection-*on*-action have enabled sustained critical insight into issues, concerns and interests relevant to the field at large. Through reflecting-*on*-practice, the relationship between myself as *performer-developer* and my developing software artefacts is analysed and investigated. The next three chapters detail my creative practice as a *narrative of development*, describing and reflecting upon the software development process engaged in throughout this research. The first of the three reflections in Chapter 7 is dedicated to investigating this complex *dance of agencies* in relation to relevant theoretical literature. Here Pickering's conception of the *mangle of practice* gave rise to a sustained analysis of the practical domain of interactive system development itself.

Chapter 4. Wayfinding Part 1: Formative Software

4.1 Introduction

In the following three chapters I describe and reflect upon the trajectory of the creative practice I have undertaken throughout this research. As outlined in the previous chapter, a concern for an understanding of interactive system design as a creative practice, as well as making sense of the unique performer-developer creative context has led me to adopt a self-reflexive approach to my research. The three 'wayfinding' chapters trace the major developments in my creative practice, detailing formative programming experiments as well as the creation more substantial creative projects. Each chapter concludes with reflections on issues, concerns and interests that have arisen throughout this creative trajectory.

4.2 Formative development approaches

As is common in practice-based doctoral projects in the creative arts (Scrivener 2000), the initial stage of this research project was characterised by an extensive period of creative experimentation. In the context of my programming practice, this resulted in the initiation of numerous Max projects that explored the central creative concerns identified in Chapter 1. From the beginning of 2010 I began working on a series of techniques for use in interactive music systems that have since found their way into more substantial and long-lasting projects. These in turn revealed some fundamental questions about the forms of interactivity and generativity that are both achievable and desirable in such computer music contexts.

During this early period, some 142 individual Max patches were developed as a part of my creative work. These software patches ranged from small programming exercises to evolving creative projects that integrated a variety of sketched programming ideas into their code-base. In parallel to these practical experiments, reflective memos were kept in hand-written notebooks, and numerous audio recordings were made of the results of the software programs developed. In what follows I triangulate these various forms of data to uncover nascent research themes that have emerged from this project. The section will

conclude with a reflection on these emergent themes, situating these in context both with the creative work undertaken during this period, and with the wider field at large.

Before beginning this doctoral project, much of my programming in both the Pure Data and Max programming environments had been focused upon the creation of custom software instruments such as polyphonic samplers, sound file granulators, software synthesisers and other sound generation methods common to computer music practice. In addition, as a trained interpretive performer with developing creative programming skills, I had gained some experience in the creation of deterministic software systems for the performance of 'mixed' music compositions for instrument and electronics.[18] However, up until this point any meaningful connection between the two creative practices that I was engaged in – namely instrumental performance and computer music programming – remained lacking. This was due in part to the level of my technical expertise as a self-taught programmer, and also by my limited exposure the potentials of both interactive approaches in computer music practice. Considering this, it was during this early period of my research that I began to focus intently upon achieving novel forms of interactivity between instrumental sources and computer music software. In particular, it was in this period that a concern for the process of live sampling as a generative strategy was first explored in any depth. In addition, a focus upon various forms of input analysis and representation dominated the programming experiments at this time, leading to the development of a series of approaches to the analysis and use of sampled performance data.

In addition to the promising development trajectories pursued at this time, there were also a number of programming experiments that, whilst having potential as self-contained technical exercises, were not followed through into more substantial software systems. This was sometimes due to their incompatibility with the creative trajectories favoured in other experiments, and at other times more promising methods of achieving similar outcomes were pursued, rendering such approaches redundant. Although these experiments may appear to be irrelevant to the larger creative trajectory of my practice,

[18] One such system was built to facilitate the performance of *Ost-Atem* (1992), a piece for saxophone and electronics composed by French composer François Rossé. A Pure Data patch was developed to replace the obsolete hardware equipment required by this composition. The new software version was performed in concert by saxophonist Joshua Hyde in June 2009 at the Conservatoire de Bordeaux.

each of these programming experiments do highlight the emergent potential of such creative programming practices, and the potential for failure to inspire new strands of creative enquiry.

Reviewing the source code developed in the Max environment during this period, the programming experiments undertaken at this time may be roughly categorised into the following areas of concern:

- Input analysis and segmentation
- Temporal pattern recognition
- Data sampling techniques
- Probabilistic methods for musical generativity

The remainder of this section will outline the programming projects undertaken in each of these areas, and their significance to my developing creative trajectory.

4.2.1 *Input analysis and segmentation*

As an instrumental performer, one of my first concerns in this project was to developing efficient and useful ways of analysing my instrumental signal for use by interactive and generative processes. Although there are numerous means by which an acoustic signal may be analysed for use in such systems, I began by focusing on the integration of fundamental frequency analysis in order to expand upon previous experiments with both live sampling and event-based experiments. As an end-user programmer making use of such algorithms, my approach to using real-time pitch analysis was linked specifically to the creative tasks I set out to achieve in the Max environment. My approach to pitch-tracking was therefore highly idiosyncratic and based upon personal heuristic methods, informed in large part by approaches taken by other artist-programmers (Dobrian 2004; Winkler 2001). Having begun experimenting with MIDI-based generative approaches in Max, my pitch-tracking experiments were borne out of a desire to represent, as accurately as possible, an acoustic signal into something manageable within such a MIDI-based system. Unfortunately however, this optimisation process was not as simple as I had initially anticipated, and by necessity pushed the creative development process towards a

development style usually associated with problem-solving, software engineering disciplines.

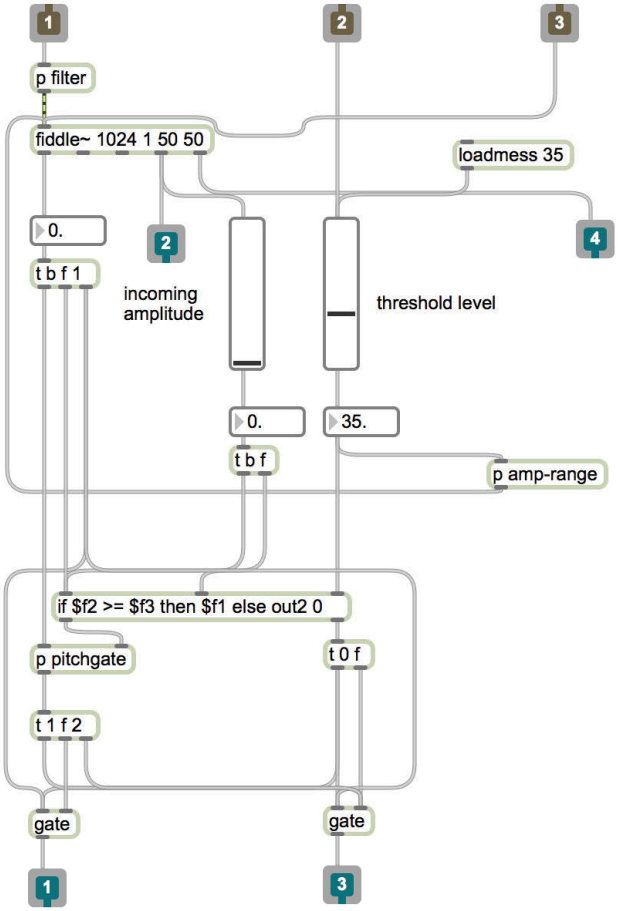


Figure 3: *thresholdpitch* subpatcher from *newaudiotracking.maxpat*

Based initially upon Miller Puckette's *fiddle*~ external object (Puckette, Apel & Zicarelli 1998) (and subsequently using the more recent *sigmund*~ algorithm[19]), the abstraction *newaudiotracking.maxpat* was programmed in order to convert fundamental frequency analysis streamed from the *fiddle*~ object into MIDI note-on and note-off messages. The purpose of analysing and representing a monophonic instrumental source in this way was an attempt to extend the capabilities of fundamental frequency analysis available from such external objects. The rationale being that such a standardised representation of the acoustic signal would then enable an expansion upon my nascent, event-based generative approaches by substituting MIDI input for real-time, instrumental input. With access to

[19] Developed Miller Puckette for Pure Data, a port to Max iby Puckette, Lippe and Apel is available from http://vud.org/max/

high level objects such as *fiddle~* and *sigmund~,* the work of a Max programmer wishing to use fundamental frequency as a control signal within Max is on the surface ostensibly simple. However, there are a number of delicate aspects to this process that make efficient, accurate and reliable pitch to MIDI conversion a difficult task. Whilst both of these objects provide adjustable parameters for the user wishing to optimise pitch-tracking accuracy, in order to effectively represent an instrumental signal as MIDI data some further programming is often needed.

The work completed on this pitch-following abstraction was informed by the work of Dobrian, who has illustrated techniques in the Max environment for discretising and representing analysed pitch and amplitude data streams from an instrumental signal as MIDI data (Dobrian 2004). The approach taken in *newaudiotracking.maxpat* made use of a simple amplitude threshold technique to filter the pitch output of the *fiddle~* object; only allowing note-on messages to be generated after the input signal crossed a user-specified amplitude threshold (see Figure 3). This technique was developed to work with the internally defined amplitude thresholds of the *fiddle~* and *sigmund~* objects, which only output so-called 'cooked' pitches above a specified amplitude. Making use of a user-specified amplitude threshold, the abstraction reported a note-on after the peak amplitude had exceeded the threshold level, packing the most recently analysed pitch with its peak amplitude scaled to MIDI velocity range (0-127).

The amplitude threshold described above was most useful however in determining the all-important note-off data needed for generating MIDI sequences. This was achieved by combining the detection of a negative threshold crossing with a timing mechanism to generate a 'note-off' from the input signal after a specified time delay. This enforced a sensitivity threshold for the amplitude input stream, mitigating spurious reports of MIDI notes in quick succession that could be created by rapid fluctuations in amplitude from the input signal. With regards to the pitch output of the *fiddle~* object, another sensitivity mechanism was included by way of a timing threshold, ensuring that only the most stable pitch estimations were output as note-on messages. Finally, a 'legato filter' was also included to force a note-off for stable changes in pitch from an input that were not accompanied by a negative crossing of the amplitude threshold. From this basic MIDI data the abstraction also took care of some initial timing analysis, calculating and outputting both the duration of each note from note-on to note-off – the note's *delta time*

– and the period between successive note-on messages – or *inter-onset-intervals* (IOIs) – enabling a rudimentary representation of the rhythmic distribution of MIDI note data.

This amplitude-driven approach, recognising the limited stability of such pitch-tracking algorithms, attempted to filter analysed pitch data to be represented as useable MIDI information. The various filtering, thresholding and gating approaches deployed in the abstraction mitigated hung MIDI notes and spurious pitch estimations with reasonable accuracy. The abstraction was developed and tested using monophonic sources (namely soprano, alto and tenor saxophones), and for my purposes proved a relatively reliable means of representing traditional, note-based instrumental performance as MIDI input for use in an interactive or generative system.

4.2.2 *Temporal pattern recognition*

As the previous abstraction demonstrates, my programming endeavours at this time were focused upon representing musical input in a computer music system as a series of 'events' (such as MIDI note-ons and note-offs). Such event-based methods, in addition to providing a consistent link between the current analytical and generative aspects of my programming work, also aided greatly in facilitating the analysis of temporal patterns from an input source. Consequently, this period gave rise to some experiments with the analysis, storage and recognition of various aspects of musical timing from an input. Having implemented some basic schemes for analysing both inter-onset-intervals and delta times from an acoustic signal, I began to make use of such basic timing information to glean more information about a live performance, seeking to recognise basic temporal patterns from an input. Such an analytical approach was initiated by a desire to siphon as much information as possible from the performance of an improvising instrumentalist, knowledge that might later be used in the generation of new musical material by the computer.

In these experiments with temporal and rhythmic analysis, I sought to make use of the preliminary analyses of IOIs in order to track higher-level rhythmic trends, facilitating the recognition of common temporal gestures such as accelerations and decelerations, as well as specific rhythmic sequences. These approaches, in contrast to the more representational focus of the above-described pitch-tracking abstraction, required a shift

towards a higher-level, analytical approach in my programming practice. However, for reasons that will become clear towards the end of this section, this foray into higher-level analysis was a relatively short-lived part of my development trajectory. In spite of this, the technical achievements of these experiments did help to reveal some conceptual boundaries in my approach to the use of various forms of analysis in the development of interactive music systems.

As a starting point for the recognition of temporal patterns, the abstraction *aceldeceldetect.maxpat* sought to recognise specific acceleration or deceleration gestures analysed from an input source. The choice to track accelerations and decelerations from an improviser was directly influenced by the kinds of rhythmic gestures that had become common in my saxophone playing during this period. My improvisational style was often characterised by short and often pointilistic gestures without a stable underlying pulse, and I was therefore interested to see if the recognition of such temporal gestures might be useful in developing the generative capabilities of an interactive music system. The algorithm worked by comparing successive IOIs analysed from an input source, and reporting the appearance of 'absolute' acceleration and deceleration gestures; i.e. temporal gestures represented by consistently increasing or decreasing inter-onset-intervals.

In the algorithm, an acceleration gesture was recognised and reported if each successive IOI was shorter than the previously analysed IOI across a finite sequence of impulses. Conversely, a deceleration gesture would be recognised and reported if each IOI became progressively longer (see Figure 4). The abstraction therefore provided some basic conditions to which any input sequence must conform in order for such a gesture to be recognised. The two independent algorithms tested input IOIs concurrently over a user-specified number of impulses, and would output a message indicating whether or not a gesture of either type had been found. Using this simple form of iterative comparison, the algorithm was therefore able to detect these two specific types of temporal gesture from an instrumental input of either an acoustic or digital origin.

Figure 4: Impulses forming an absolute deceleration gesture

However, whilst testing this abstraction during development it became clear that human performances of these types of gestures did not always conform to such absolute temporal relationships. In my iterative testing of the abstraction using both digital (keyboard/mouse and MIDI keyboard) and instrumental (saxophone/electric guitar) inputs, I learned that such an exact measure of acceleration or deceleration was not necessarily correlated to a perception of these gestures in a musical sequence. As the algorithm left no room for error when searching for the recognition of these temporal gestures, it would often not detect the inexact and slightly 'noisy' nature of human performance. In addition, because the focus of the abstraction was concerned with pursuing efficient means of recognising input patterns, not enough thought had been put into the integration of such an algorithm into an eventual working interactive system. Beyond proving useful as a technical exercise, the output of the algorithm remained less than inspiring due to the focus upon simply reporting the recognition of these gestures.

As a result of these issues, two revised abstractions (*acceldetectthreshold.maxpat* and *deceldetectthreshold.maxpat*) were developed that sought to give the recognition algorithm more flexibility with respect to the inexact nature of human timing, and further outputs were added to enhance the usefulness of such an algorithm as a component within an interactive system. As the names suggest, these new abstractions incorporated flexibility in their recognition strategy by means of a thresholding mechanism. A *detection threshold* incorporated into each abstraction enforced a degree of tolerance when analysing successive inter-onset-intervals to determine whether or not an acceleration or deceleration gesture had occurred in the input. This tolerance allowed for some degree of error in the recognition of these temporal gestures, by enabling a percentage of IOIs that did not conform to an exact accel/decel trajectory to pass through without affecting the overall recognition of the gesture.

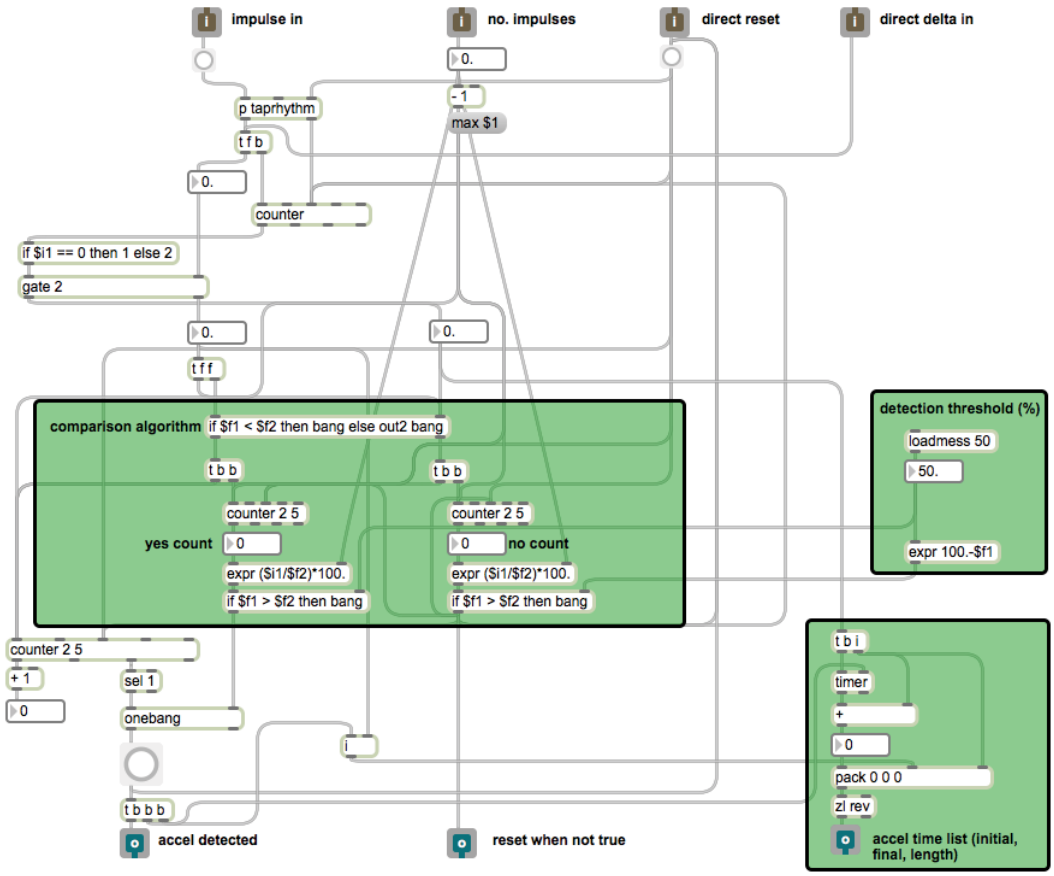


Figure 5: Inside the *acceldetectthreshold* abstraction

As illustrated in Figure 5 above, throughout the duration of a user-specified number of impulses, the acceleration detection algorithm maintained a 'yes count' of the total number of occurrences of decreasing IOIs received from an input (i.e. those that conformed to an acceleration trajectory), in addition to the number of IOIs that did not conform, a 'no count'. The user-specified detection threshold, represented as a percentage, was used to specify exactly what percentage of analysed IOIs must conform to the expectation of a progression of decreasing IOIs in order for an acceleration to be recognised. This augmentation of the original algorithm represented a form of statistical filtering that, whilst relatively simple, enabled a degree of freedom between individual impulses whilst still requiring a sequence of IOIs to satisfy either a decreasing or increasing trend, i.e. an acceleration or a deceleration. Such a tolerance measure ensured that despite the possibility of a select few errors occurring in an input sequence, an overall acceleration of deceleration trajectory could still be recognised from the input.

Secondly, as alluded to above, it was decided that taken alone, the simple reporting an accel/decel gesture (represented by the output of a *bang* message in the Max environment) would not prove very useful as a data type to be used into a developing interactive system. This type of temporal pattern recognition was quite specific, and the simple recognition of such patterns was not seen as especially useful in the development of machine behaviours in response to an improviser's performance. Although useful as a technical exercise, this approach began to represent part of a 'crisis of relevance' to the overall goals of my programming practice at the time. After reflecting upon this, in addition to the momentary reporting of these gestures, a skeletal reduction of each gesture recognised was also collated for output alongside the reporting of a recognised gesture. As is shown in the bottom right of Figure 5, the third outlet of each abstraction was reserved for the output of a list of analysed values from each of the gestures recognised from an instrumental input. This list was comprised of three values stored during the comparison process, namely the first IOI analysed from the gesture, the gesture's final IOI, and the total elapsed time in which the recognised gesture took place.

In addition to simply recognising a gesture therefore, the algorithm now acted as a reductive mechanism for sampling performance data from an instrumentalist. By outputting a reduced version of the gesture itself, the abstraction facilitated the approximate recreation of temporal gestures analysed from the live performance of an instrumentalist. In practice, such a recreation required only a limited amount of further programming to achieve, an example of which is illustrated in Figure 6. This figure illustrates how two captured lists of timing information (the first representing an acceleration and the second a deceleration), combined with a timed ramp of control data (the line object) and a variable rate metronome (the metro object), facilitate the approximation of the original contents of gesture analysed from an input. Whilst not recreating the exact temporal relationships between the impulses analysed in the original gesture, this reduced form was flexible enough to enable an interactive system the capacity to dynamically store, perform and alter specific rhythmic gestures siphoned from an instrumentalist during performance.

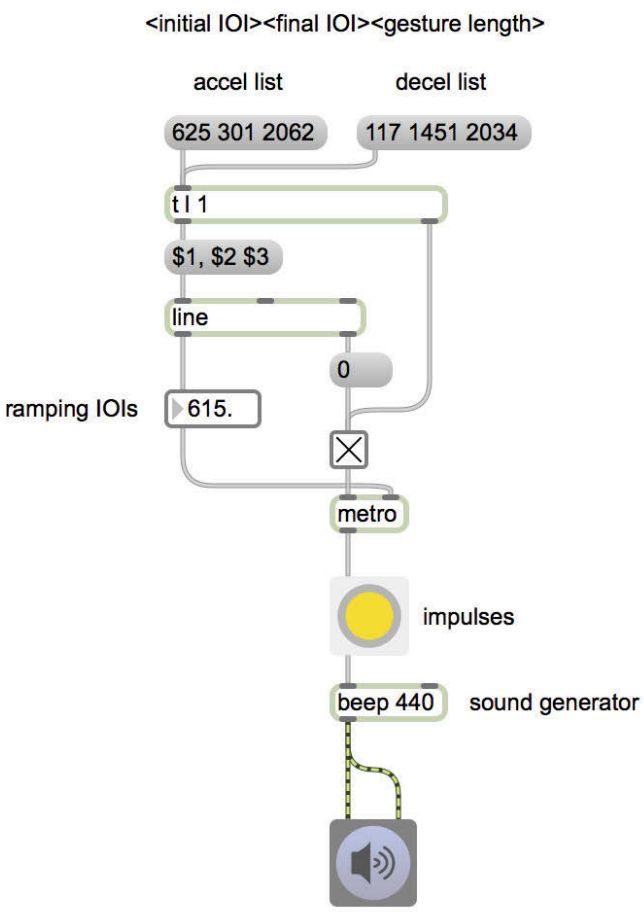


Figure 6: Using accel/decel lists to generate rhythmic gestures

By intentionally reducing analysed gestures so that they may be output as complimentary data, this algorithm had clearly strayed from its original purpose. What had begun as a technical exercise in pattern recognition had been augmented to directly facilitate the generation of rhythmic patterns as derived from an input. Acceleration and deceleration gestures, at first the subject of a directed form of pattern recognition, had become a direct link to an interactive system's generative potential. Through this simple form of reductionism, a strictly analytical algorithm had therefore been linked directly to the development of a repertoire of rhythmic gestures for a developing computer music system.

In parallel with the development of the above algorithms I also began experiments that sought to further these pattern-matching techniques into a more deterministic space. The most notable of these experiments was the abstraction titled *findthatrhythm2.maxpat*, a program that compared impulse sequences from an input source with pre-defined

rhythmic patterns stored on disk. The purpose of *findthatrhythm2.maxpat* was to experiment with the possibility of using such rhythm recognition techniques to provide high-level, analysis-based control triggers to an interactive music system. Given that this abstraction was focused upon the recognition of patterns from an input, its predominant output (the report of a recognised rhythm) remained largely similar to that which I had identified as undesirable in the pattern recognition work described-above. This program therefore suffered from a similar 'crisis of relevance' within my programming practice. Despite these misgivings however, I used the development of this abstraction to further expand upon some of the thresholding techniques I had been developing, as well as to integrate the acceleration and deceleration detection algorithms I had developed into a more specific, pattern recognition context.

The abstraction was tasked with the following objectives:

- to compare a sequence of impulses from an input with a stored rhythmic pattern
- to report that a rhythm has been recognised
- to provide a 'performance score' to the user showing how similar the input sequence is to the stored ideal

In this abstraction the process of comparing input sequences to stored rhythms was achieved by representing rhythmic patterns as simple temporal ratios. These ratios were related neither to either a central pulse, absolute inter-onset-interval values nor to an overall time signature, but rather to the initial relationship between the first two impulses in a sequence.

As an example, a rhythm such as the rhythmic sequence shown in Figure 7 would be expressed as the following sequence of ratios: *1. 0.5 0.5 1. 2. 1. 2.*

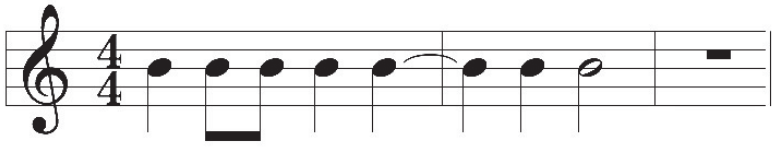


Figure 7: Example rhythmic sequence in *findthatrhythm2.maxpat*

By representing rhythmic patterns as ratios, the successive comparison of incoming IOIs with sequences of stored ratios became a relatively trivial task. In order for the algorithm to compare live input to stored sequences, the abstraction first represented the incoming IOIs as ratios to the first IOI analysed from the input, and then compared subsequent incoming ratios with those stored in a sequence chosen by the user.[20] As discovered in the initial pattern recognition algorithm discussed above, it was important however to provide a measure of flexibility for the real-time comparison of the input source, as human performance of even the simplest of rhythmic patterns could never conform to the exact ratios expected by the algorithm. Consequently, a threshold percentage similar to that discussed above was employed, enabling a degree of error when comparing incoming ratios with those stored on disk. This threshold mechanism enabled analysed IOI ratios to pass above or below the ideal ratio *r* by a user-specified tolerance factor *t*, represented as a percentage of the originally stored ratio. Each successive incoming ratio would be counted as a match between the ranges of $r + tr$ and $r - tr$.

For example, in a performance seeking to match the rhythm illustrated in Figure 7, with a tolerance factor of $t = 0$ the algorithm could only accept incoming rhythmic ratios that exactly matched the sequence *1. 0.5 0.5 1. 2. 1. 2.* However, with a tolerance factor of $t = 0.3$, the following ranges of values would be accepted as representing the stored ideal rhythm: (*0.7 – 1.3*) (*0.35 – 0.65*) (*0.35 – 0.65*) (*0.7 – 1.3*) (*1.4 – 2.6*) (*0.7 – 1.3*) (*1.4 – 2.6*). Such a metric therefore made the recognition of stored rhythmic ratios more tolerant to noisier, human temporal fluctuations. In addition to the above recognition algorithm, a metric for reporting a simple 'performance score' for these input sequences was also developed. The metric first calculated the absolute difference between a list of performed ratios with the 'ideal' sequence stored on disk. If the performed sequence satisfied the recognition algorithm described above, the algorithm then computed and output an average accuracy of the performance as a percentage across the entire sequence of ratios (see Figure 9). This performance score was output alongside the momentary *bang* message indicating the recognition of a stored sequence.

[20] The abstraction was designed to test a performance against a specific stored rhythm chosen by the user. It is conceivable that a more sophisticated design could match an incoming performance to a rhythm stored in corpus of rhythms, however this kind of matching was not implemented in my work at this time.

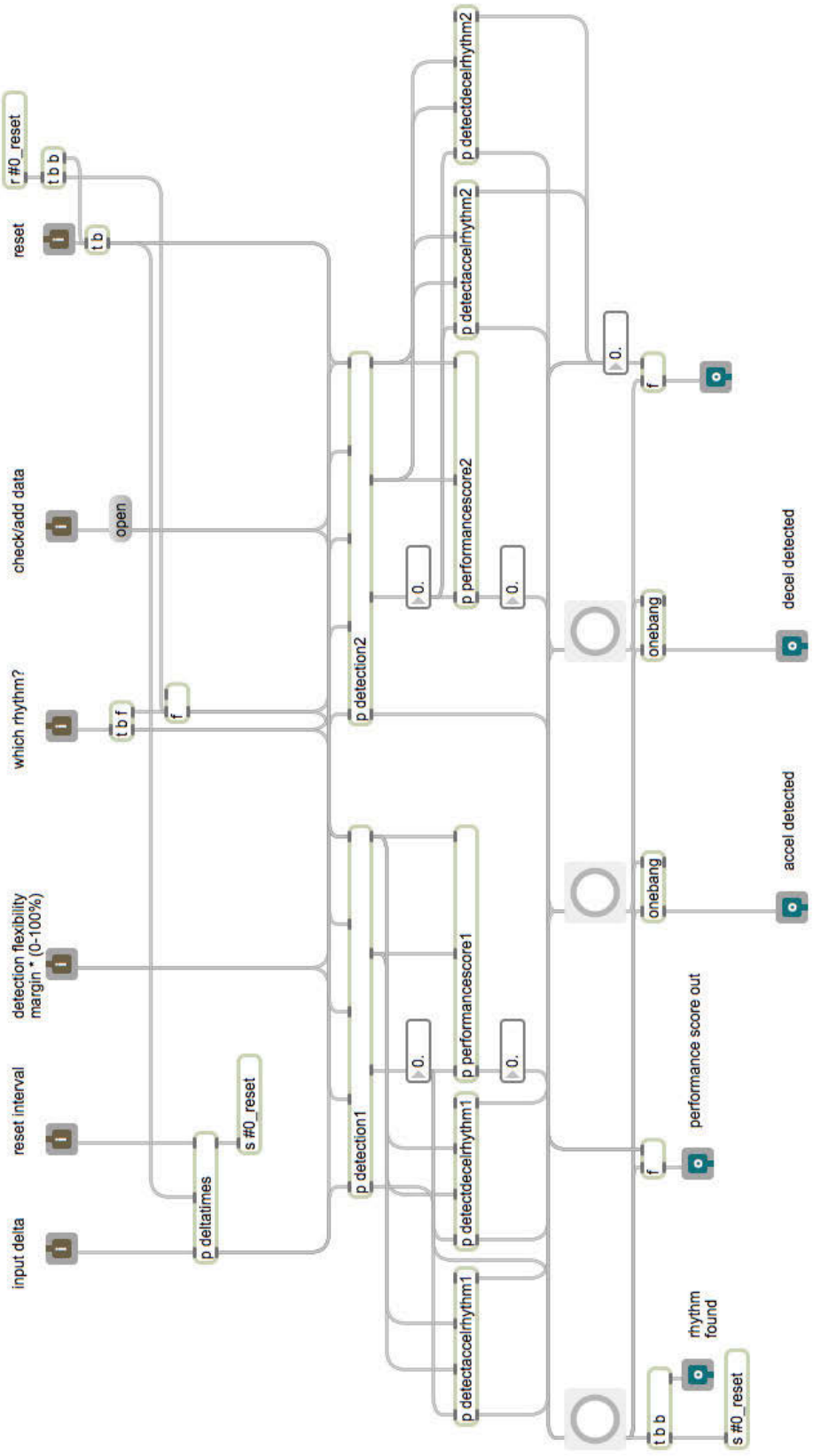


Figure 8: Inside the *findthatrhtythm2.maxpat* abstraction

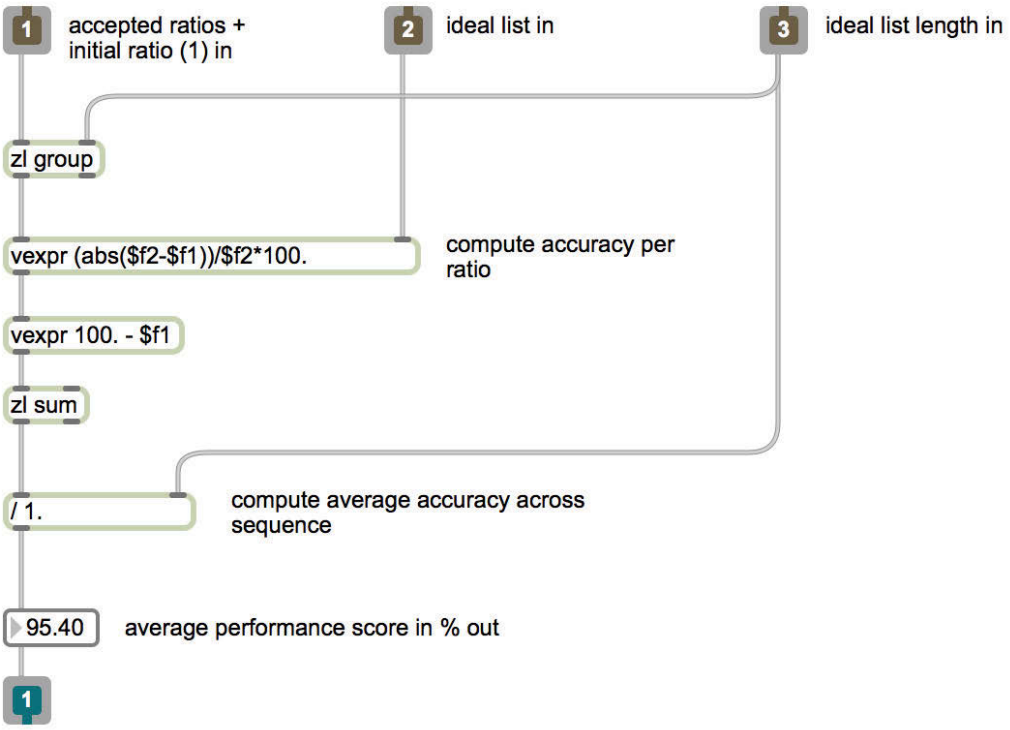


Figure 9: Performance score algorithm from *findthatrhythm2.maxpat*

As mentioned above, *findthatrhythm2.maxpat* also integrated acceleration and deceleration detection as a core element of its design. Making use of the above-described *acceldetectthresold.maxpat* and *deceldetectthreshold.maxpat,* the abstraction gave an indication of the overall tendency of the rhythmic performance to either accelerate or decelerate across the course of the pattern being analysed. Due to their specific focus upon acceleration and deceleration detection, these algorithms were re-purposed to aid in the detection of such temporal trajectories in relation to stored sequences, as opposed to the recognition of the specific temporal gestures of acceleration or deceleration.

To achieve this, the abstraction computed the difference between each incoming performance ratio and those of the stored sequence, passing these differences as direct inputs to both the acceleration and deceleration detection algorithms. The length of the ideal sequence was chosen as the length to test for an accel/decel trajectory, allowing these algorithms to report whether or not the analysed performance tended to accelerate or decelerate across a the length of a stored sequence. In addition to the momentary reporting of an accel/decel tendency from the performance, the abstraction also output a

value indicating exactly how much faster or slower the performance was to the stored ideal, as tested using analysed ratios.

Although *findthatrhythm2.maxpat* had been designed as an analysis abstraction for use in larger musical systems, it remained largely a proof of concept system that was not pursued further after its initial development. The shift from input analysis and representation to pattern recognition had been a surprising development in my programming practice at this time. Having begun by searching for interesting methods of informing generative processes through analysis, my trajectory had led me towards a deterministic approach that, whilst technically interesting, found no real place in my developing creative trajectory. It had become clear to me that whilst the potential for detailed, idiosyncratic pattern recognition techniques was technical feasible in my practice, their potential use for informing the generation of musical material in performance remained limited. Whilst I had been comfortable developing idiosyncratic pattern recognition techniques that reflected my musical pre-occupations, soon after their implementation I began to question the purpose of pattern recognition as an interactive strategy in the context of non-idiomatic, improvised computer music performance.

A concern that emerged from this period of development was related to the relative usefulness of specific levels of analysis used to inform a generative algorithm of the context of a live performance. Such was the experimental nature of my programming practice at this time that, despite my initial goal of providing a system a rich array of data analysed from an instrumentalist, the specificity of the knowledge gained from analysing a musical input had reached a threshold for my practical purposes. I believe that moving from FFT analysis to MIDI representation helped nourish a desire for higher-level analytical knowledge about an improviser's performance, which in turn led to a level of determinism that I was uncomfortable with as an interactive mechanism in computer music performance. Beginning with the analysis of acceleration and deceleration trajectories, it soon became clear that the practical use of detecting patterns in an input would require of a system a level of prior knowledge about the possible range of musical gestures of a performer, which in turn would limit a system's ability to respond to a greater diversity of musical material without developing an extensive array of such recognition algorithms.

After exploring these avenues in some depth, it was decided that these ideas were not worth pursuing any further in my work, as they had strayed far from the original goal of developing novel interactive and generative techniques for improvised human-computer performance. These recognition experiments, whilst novel, remained sufficiently abstracted from this goal that my attempts at this time to steer them towards this purpose remained somewhat cumbersome and deterministic. However, the approach that remained the most useful from these experiments was the choice to output and store reduced versions of detected accelerations and decelerations. An example of the implementation of these ideas into larger systems can be seen in the patches *audiotomiditest.maxpat* and *interactiveoptions.maxpat*, where the detection of such gestures was used to build databases of analysed temporal gestures to be triggered for output. Such an approach, originating in experiments in pattern recognition, came to dominate my programming practice at this time, leading to further exploration of such 'data sampling' techniques for use in the development of autonomous system responses to live instrumental input.

4.2.3 *Data sampling techniques*

After implementing the above-described method for reducing and storing temporal input gestures, I began experimenting with methods of using such data for creative purposes. In addition, it was also during this period that I began to further explore methods for the sampling and storage of other types of performance data that could be later called upon for use by the computer in an interactive environment. The reduced gestures of these accel/decel sequences, comprising of a small amount of data, became a useful starting point for a variety of methods that sought to gather information from the instrumentalist for later use. The patches alluded to above, *audiotomiditest.maxpat* and *interacrtiveoptions.maxpat,* were the first to include this sampling and storage method as a part of larger interactive environments, alongside other nascent methods for capturing performance data from a live instrumental performance. As is clear from their titles, these two patches were not however developed as integrated musical systems, but rather as collections of *options* being explored in the development and testing process. As many of these options had arisen from programming experiments developed in isolation from the approaches discussed previously, these two patches provided a locus for a heuristic approach to developing larger, interactive musical systems.

Audiotomiditest.maxpat and *interacrtiveoptions.maxpat* explored a combination of triggered processes, MIDI transformations, delayed and re-mapped amplitude envelopes and well as direct *one-to-one* mapping of analysed pitch to parameters of synthesis and processing modules such as additive synthesis and sound file granulation.

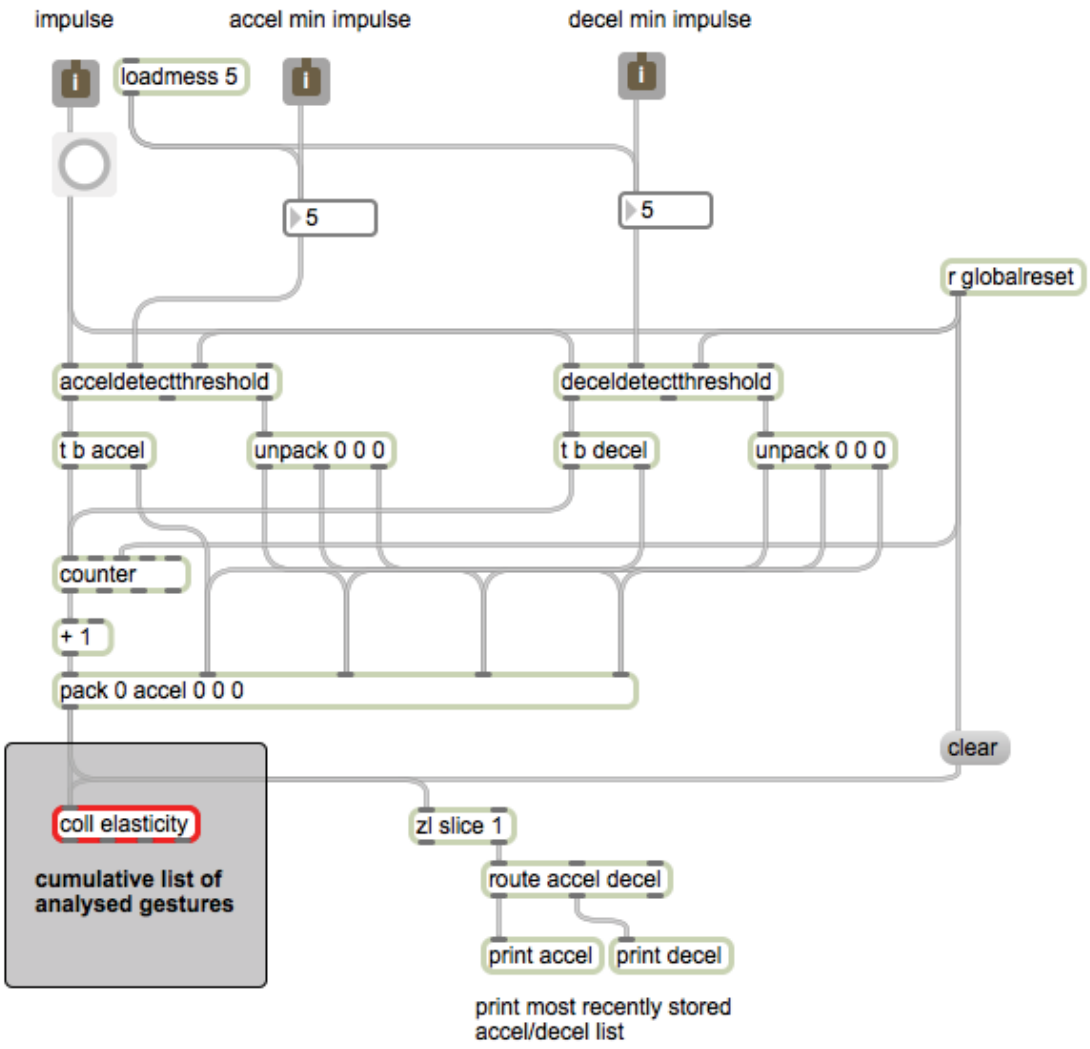


Figure 10: ***elasticitystorage.maxpat***

Importantly however, both of these patches made use of complimentary algorithms that implemented accel/decel detection for the sole purpose of gesture storage and performance. These abstractions repurposed the original momentary recognition algorithm towards its direct implementation in an interactive environment. In both *audiotomiditest.maxpat* and *interactiveoptions.maxpat*, the output of the note-on detection from *newaudiotracking.maxpat* was used as an event input to *elasticitystorage.maxpat,* which contained a data collection that incrementally stored detected acceleration and

deceleration trajectories analysed from this input (see Figure 10). *Elasticitystorage.maxpat* was designed to store these trajectories in a cumulative data collection, so that their contents could be accessed for output by any algorithm seeking to use these rhythmic gestures in a larger interactive system. In the case of *audiotomiditest.maxpat*, a basic second abstraction was used to trigger output of these stored gestures by an additive synthesis module playing exponentially decaying bell tones. The mapping of this abstraction however remained limited both with respect to the choice of gesture to output (a randomised choice), and the way in which the gestures were triggered (following a time threshold linked to the duration of an incoming MIDI note-on and off data).

In *interactiveoptions.maxpat*, the *elasticityread.maxpat* abstraction was first implemented, an abstraction that enabled a more flexible means of outputting gestures stored in *elasticitystorage.maxpat*. This abstraction enabled the user to choose from three types of output from the gesture data collection, *randomised, sequential,* or *discrete*, the latter allowing the user to specify an exact index of the gesture desired for output. In *interactiveoptions.maxpat* this abstraction was implemented once more with the additive synthesis module described previously. Despite these additions however, the means by which gestures were output remained the same. Relying upon the direct triggering of these pre-analysed gestures, this approach remained limited as a generative strategy as this storage and output approach did not help to provide an autonomous and contextually-aware musical contribution to an improvisation with a live musician.

Another approach taken at this time was to provide a more flexible means with which to facilitate the sampling of time-varying data, and to investigate more permanent methods for capturing and storing such data. The abstraction *dataatintervals.maxpat* was created in order to facilitate the sampling of time varying data from an input source. Originally conceived for the sampling of amplitude data from a live signal, this abstraction was generalised in order for it to be used to capture, store, output and alter any time-varying control data in the Max environment (see Figure 11).

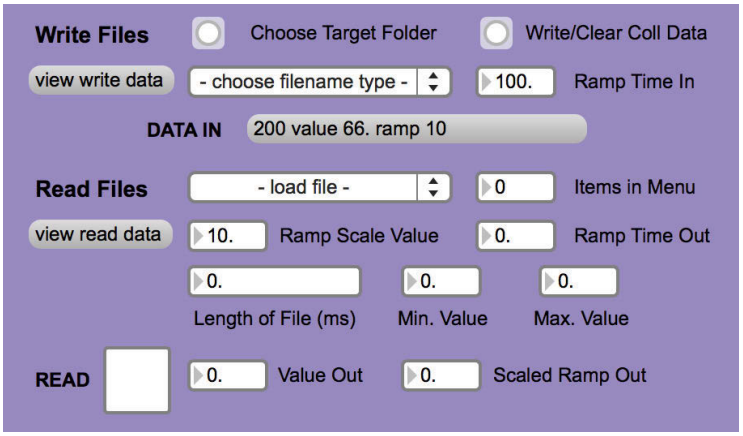


Figure 11: *dataatintervals.maxpat* graphical user interface

In contrast to the reductive mechanism applied in *elasticitystorage.maxpat*, this abstraction was designed to store lists of floating-point numbers pertaining to individually sampled data points captured from a live signal. Each data point captured in a locally stored data collection was accompanied by a *ramp time* value representing the sampling rate at which incoming data had been captured for storage in the collection. As these temporal gestures were stored as discretely sampled points, the *ramp time* value was included so that this information could be later used to aid in the compression and expansion of stored temporal gestures upon output. *Dataatintervals.maxpat* therefore provided a direct link between the sampling rate chosen for the capturing of performance data, and the ability to alter the temporal identity of captured gestures. This separation between the data point and its sampling rate was chosen due to its ability to interface with the commonly used *line* object in the max environment, which provides for a linear interpolation between data points over a specified ramp time.

An important element of this abstraction was its ability to read and write data to and from disk, enabling captured temporal gestures to be made use of beyond the confines of a single interactive session. For data storage, the simple user interface of *dataatintervals.maxpat* asked the user to choose a target folder for their data collection to reside on disk, in conjunction with a unique filename type. After setting the *ramp time*, a user could enter their continuous data into the internal data collection of the abstraction for storage.

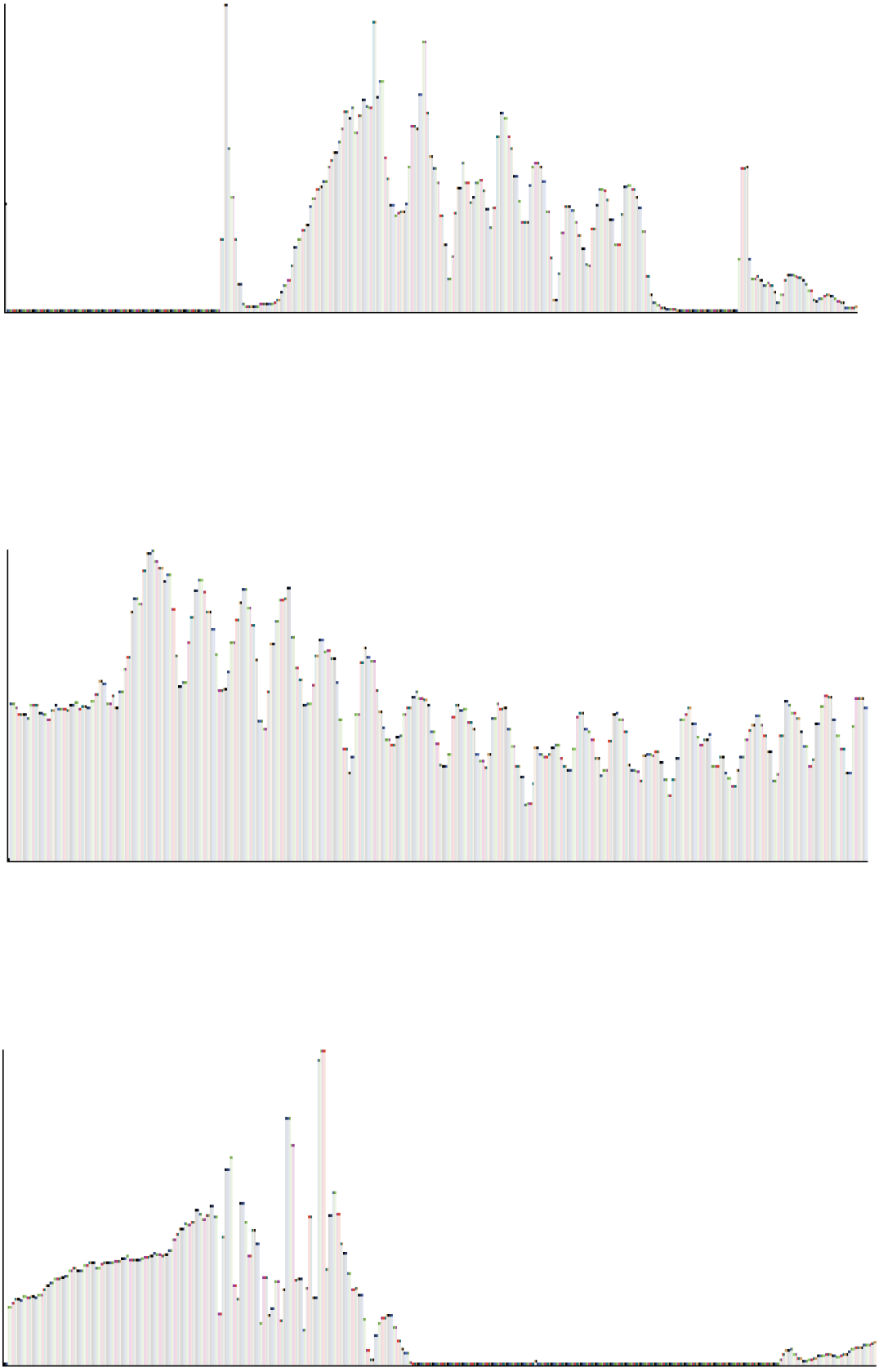


Figure 12: Amplitude curves sampled from the tenor recorder using *dataatintervals.maxpat*

Once a temporal gesture was complete, clicking 'write/clear coll data' would write a new, indexed text file containing the gesture to the target folder, with each subsequent gesture incrementing the numbered index. To read gesture files previously stored on disk, the abstraction automatically populated a menu of indexed text files corresponding to the *filename type* chosen at the top of the interface. Once a gesture had been chosen for output, the interface displayed some meta-data about the temporal gesture useful for scaling of control parameters, including the minimum and maximum value of the gesture, the number of data points in the file, the stored ramp time of the gesture and finally the file's original length in milliseconds (determined by the sum the ramp times values stored alongside each data point). Most importantly, the *ramp scale value* included in this abstraction enabled the real-time adjustment of a gesture's output ramp value, enabling the temporal dimension of these stored gestures to be easily altered in real-time.

This abstraction marked a change in the way in which data sampling was managed in my work. Previously to *dataatintervals.maxpat*, the approach taken towards sampling real-time performance data was local to a specific interactive session. That is, data sampled from a performer was only envisaged for use within the same session, with no prevision for this data to be used in other interactive contexts. *Dataatintervals.maxpat* was specifically designed as a means of collecting real-time, continuous data streams for use outside of any one, specific interactive context. Although the abstraction could be used in this way, its primary goal was to enable the efficient and flexible future use of performance data siphoned from a performer. During this period, I made use of the abstraction to store temporal gestures from a variety of time-varying sources, ranging from sampled amplitude curves (see Figure 12 for an example) to movement and acceleration data sampled via OSC from sensors within smartphone applications. These pre-analysed temporal gestures were made use of to automate a variety of control parameters of complex sampling and synthesis modules such as phase vocoders and sound file granulators. Sampled gestures such as these provided a useful, non-randomised means of automating certain musical parameters in systems either directly controlled by a performer (a sampler), or triggered by event-based analyses of improvised performances.

4.2.4 *Probabilistic Methods*

4.2.4.1 DurationalProb

DurationalProb was the first fully functional system developed in this research that was intended for use as an interactive system for improvised performance. The software represents the culmination of numerous experiments that investigated the relationship between input analysis and generativity in the creation of an interactive music system. As with much of my early programming experiments, this system made use of the MIDI protocol as its main source of both musical input and output. Taking a stream of MIDI note/velocity pairs from an improvising musician (on any MIDI-enabled instrument, or through a pitch to MIDI converter), the software generated new and varied material by referencing a cumulative history of the captured performance data, enabling an interactive dialogue to develop between human and machine in an improvised performance.

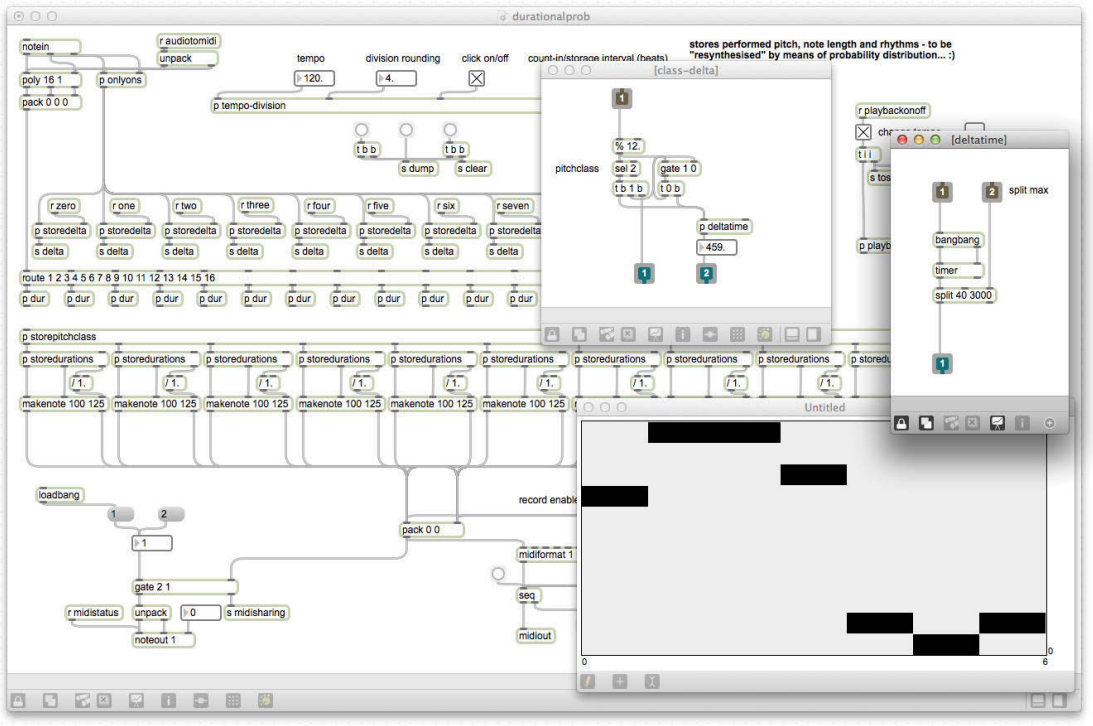


Figure 13: *DurationalProb* Max patcher

Making use of the cumulative history of MIDI data streamed from a musician's performance, *DurationalProb* sought to mine a musician's past performance as a tool for achieving autonomous generativity in a computer music system. The data sampling approach used in this system manifested itself in numerous programming experiments undertaken at this time, and it became central to my thinking about the relationship between interactivity and generativity in computer music performance. The possibility of recognising the genesis of a computer's musical gestures in one's own performance, whilst at the same time being presented with self-generating structures was very appealing to me as a performer. The majority of my work in this area has involved variations on this approach.

DurationalProb makes use of analyses of a musician's performance in order to dynamically create *state spaces* that the system uses to create its own material. The creation of these state spaces is achieved by collating the number of occurrences of the various notes, velocities and timing information captured from a live MIDI performance into a series of performance histograms. These parameter histograms are useful for any number of generative algorithms to mine for patterning and generation, from simple probabilistic generation to more complex data-mining methods. *Durationalprob* made use of probabilistic techniques in developing histograms from MIDI note and velocity information, as well as analysed timing information extrapolated from this MIDI data captured from a performer. The generative strategy treated the histograms as simple probability tables to lookup for note generation, creating a statistical system with multiple, independent musical parameters generated simultaneously from independent histogram data. In this approach, the greater number of occurrences of a particular value analysed from the input, the higher probability weighting given to that value, therefore increasing the likelihood that it would be chosen for output during performance by the software.

Despite the simplicity of the probabilistic methods employed, this approach provided the potential for rich and varied musical generation due to the multi-dimensional nature of the data siphoned from the performer. Although the separation of these musical parameters into independent histograms might be seen as somewhat arbitrary, such a parametric approach to musical composition has historical precedent as a method of

structural generativity in musical composition, from the isorhythmic compositional approaches 13th and 14th century composers such as Philippe de Vitry and Guillaume de Machaut, and later to the parametric techniques of the 20th century integral serialists (Taruskin 2009).

However, although ostensibly useful as a data driven method of musical generativity, an obvious limitation with such a probabilistic approach in interactive performance is that the system's generative capabilities are devoid of any context from which stored data points appear in the analysed input. That is, although the accumulation of statistics on a musician's performance creates a rich state space representing statistics on past performance, a simple probabilistic method of generating new material from this space lacks any connection to the temporal trajectories of the original input sequences. The use of probability in musical composition has a long and rich history, belonging to a family of algorithmic approaches referred to as *stochastic music* (Roads 1996, p. 868). However, in the context of live, interactive performance, such static methods for generating musical material remain musically unconvincing. From melodic and dynamic shapes to timbral changes, understanding musical patterns as occurring both in series and in time is central to achieving convincing musical generativity. A probabilistic approach that took into account the sequential context of musical patterns was sought next in my practice.

4.2.4.2 Tripartite Markovia[21]

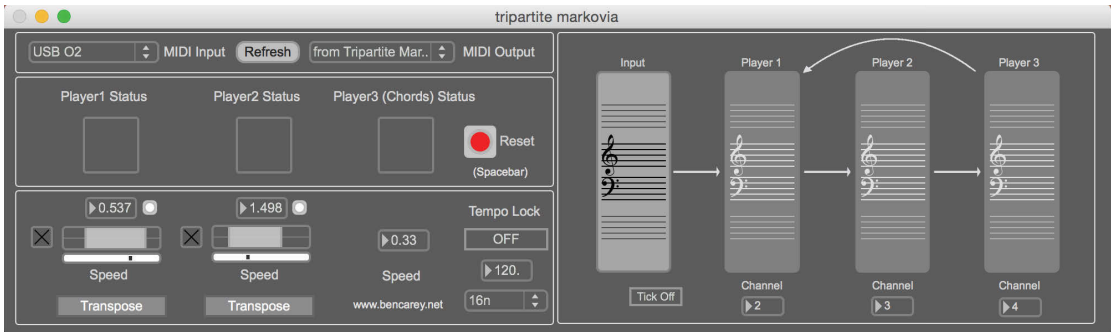


Figure 14: The *Tripartite Markovia* graphical interface

[21] Two demonstration videos of this software can be viewed at the following URLs: https://www.youtube.com/watch?v=byZjyBiehmk and https://vimeo.com/19863192

Tripartite Markovia makes use of the *Markov property* for its generation of new musical material in performance, a mathematical principle that has been used to model random processes using probabilistic methods. Markov chains have been applied in a variety of fields, extending classic probability theory to account for the temporal aspects of randomised events. A *Markov chain* is a probabilistic process that uses the outcome of a previous probability experiment to affect the outcome of future experiments, thereby creating chains of events that are dependent upon the previous output of the process itself (Grinstead & Snell 1997). Markov chains have been used in algorithmic composition since the late 1950s, and have proven to be a useful method of creating coherent musical structures in both compositional and interactive musical systems (Ames 1989; Roads 1996; Zicarelli 1987). Fundamental to the Markov property is what is called the *transition matrix*. The transition matrix is a stochastic matrix that contains probability weightings for the transitions between events in an *n*-dimensional state space. As these matrices hold weightings for the *transitions* between events and not of the individual events themselves, Markov chains generate sequences that are dependent on their own history of decisions. Chosen states are fed back into the Markov process, which are subsequently evaluated to provide the next state, thereby creating a chain of discrete events that are dependent on a previous output state.

One of the most important properties of Markov chains is its *order*, the property of the transition matrix that determines how many previous events in the chain are taken into consideration when determining the next event. For example, a *zeroth-order* Markov chain describes a standard probability distribution, as exploited in the above-described *DurationalProb*. Such distributions are one-dimensional, meaning that a decision about the next state is not informed by the preceding state in the chain, but only by the standard probability weightings of all possible states. Sequences generated by zeroth-order chains are therefore not imbued with any sense of historical context. *First-order* chains however rely on a two-dimensional matrix that describes the probability that the current event will be followed by another event. Such a Markov chain creates a sequence where each event is determined by probability weighting attached to its immediate predecessor (Ames 1989). First-order Markov chains are therefore the simplest instantiation of the Markov process, whereby the output sequence takes into account part of the history of probabilistic decisions. As the order of a Markov chain increases, so too does the specificity of the sequential nature of the chain itself. The output of a *second-order* Markov

chain takes into account the probabilities of two preceding decisions in the chain, whereas a *third-order* chain takes into account the three previous decisions and so on. When used in conjunction with transition matrices derived from an input source – such as MIDI note data siphoned from a live performance – higher-order chains such as these begin to more closely resemble the input sequence itself.

Given my interest in creating generative processes derived from data-sampling driven approaches, the application of Markov chains provided interesting and contextually-aware musical results when applied in tandem with real-time data sampling methods. In *Tripartite Markovia,* the transition matrices themselves are generated interactively from a growing histogram of MIDI data streamed from the performance of an improvising musician. As its generative basis, *Tripartite Markovia* employed four independent first-order Markov chains that combined the data-sampling approach discussed above with a probabilistic process that took into account the historical nature of the data gathered during a musician's performance. Listening to the MIDI input of a live performer, the software streamed and collated four musical parameters into separate histograms for MIDI note number, MIDI velocity number, inter-onset-interval (in milliseconds) and MIDI note duration (in milliseconds). These histograms were then used to build four independent transition matrices informing the state transitions of the Markov chains tasked to generate new material for the improviser to respond to in performance.

In *Tripartite Markovia,* the Markov property is exploited for use in its simplest form, making use of only first-order Markov chains in its design. Given this restriction, the ability of the chosen method to accurately model input sequences was admittedly limited, as the implementation of lower-order chains creates less predictable output than those of a higher order. However, in designing an improvisation system whose generative potential was based solely on material extracted from a performer's performance history, a balance was sought between the generation of recognisable musical gestures, and the creation of novel musical trajectories from this source material. In other words, the Markov property was not chosen in order to accurately model the stylistic tendencies of an improviser, such as in the work of Pachet (2002) or Assyag et. al (2006), but rather it was used as a novel and efficient means of generating complimentary material from a history of the performer's actions.

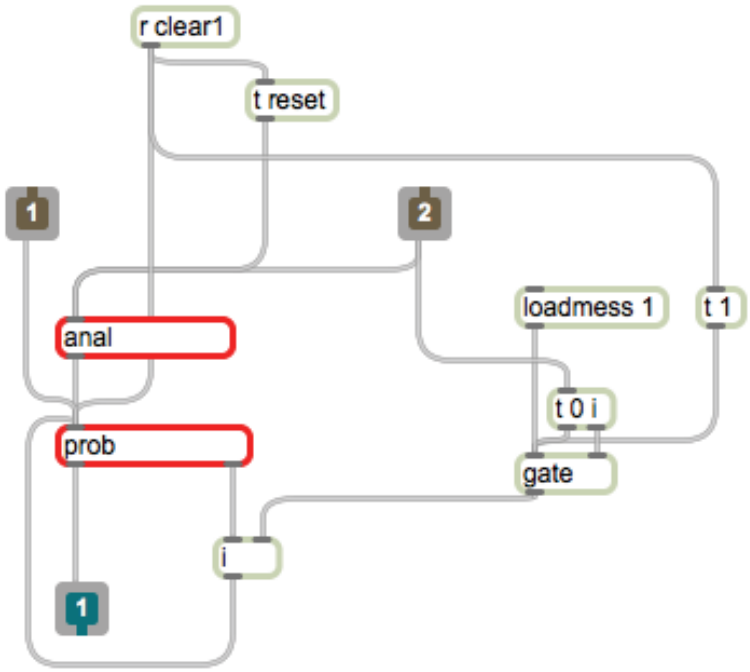


Figure 15: The *pitch_markov* subpatcher contained within *Player 1*. The *anal* object is responsible for building a histogram of incoming MIDI notes, whilst the *prob* object is used to build the necessary transition matrix from this data

As the software made use of these first-order Markov chains for multiple, simultaneous musical parameters, the system was able to generate a complex and nuanced response derived from the musical input of an improviser. This process was aided by the independent nature of the Markov chains assigned to each musical parameter. Given that these Markovian sequences exhibit a great deal of unpredictability, the probabilistic trajectories of these independent parameters displayed a degree of opacity in the resulting musical material. The relationship between rhythmic and pitch sequences can therefore be described as creating a type of parametric counterpoint, again reminiscent of the isorhythmic and serial methods mentioned in relation to the *DurationalProb* system described in Section 4.2.4.1.

In addition, as suggested by the software's namesake, the computer's generative responses to the performer were expanded to encompass three, separate 'Markovian' performers that each made use of independent transition matrices developed during an improvisation. *Tripartite Markovia* was therefore designed to provide the performer with a virtual ensemble with which to improvise in performance. This virtual ensemble, following the sequential nature of the Markov process, was designed to follow a pre-

configured *direction of influence* that affected the development of transition matrices, the output of the resultant Markov chains as well as their temporal synchronisation. Rather than creating three identical players with access to the same transition matrices, a sequential method of passing MIDI information from the live performer was implemented, so that each performer accessed the material siphoned from the improviser at varying degrees of abstraction. This direction of influence is represented in the graphical interface of the software (see Figure 14), and is described as follows:

First, *player one* receives raw MIDI information from the live performer and begins to build a transition matrix based on four elements analysed from the performer (pitch, velocity, IOIs and duration). After waiting for a random number of MIDI events from the performer, *player one* begins generating new material by referencing its internal transition matrices. As *player one* begins to output material, *player two* builds its matrices directly from the data output by *player one*. The transition probability weightings used by *player two* therefore refer only to the analysed output of *player one*, and not the live performer directly. After a randomised number of events collated from *player one*, *player two* begins to generate material by referring to its internal matrices. Subsequently, *player three* builds its transition matrices in a similar fashion, by referring only to the output of *player two*'s material. *Player three* is tasked with generating five-note chords taken from the sequential output of its MIDI pitch transition matrix, modified with an octave transposition value per note. *Player three* therefore generates a harmonic layer to the contrapuntal texture of players one and two, modeled upon the output of *player two*.

The complex and interdependent nature of *Tripartite Markovia's* generative strategy resulted in a highly polyphonic musical texture from the system. The level of complexity exhibited by the software was unsurprising given the independence of the various musical parameters, and the sequential, dependent relationship created between the Markovian players. Such a degree of complexity generated from the input of a sole live musician was enticing and provocative as a performer interacting with the system, however it was specifically in the temporal domain that I felt that the independent nature of the various Markov processes began to crowd the musical texture. The rhythmic generation of the three players was not tied to a metric grid, as it was my intention to maintain the elasticity of the original temporal sequences in the software's output.

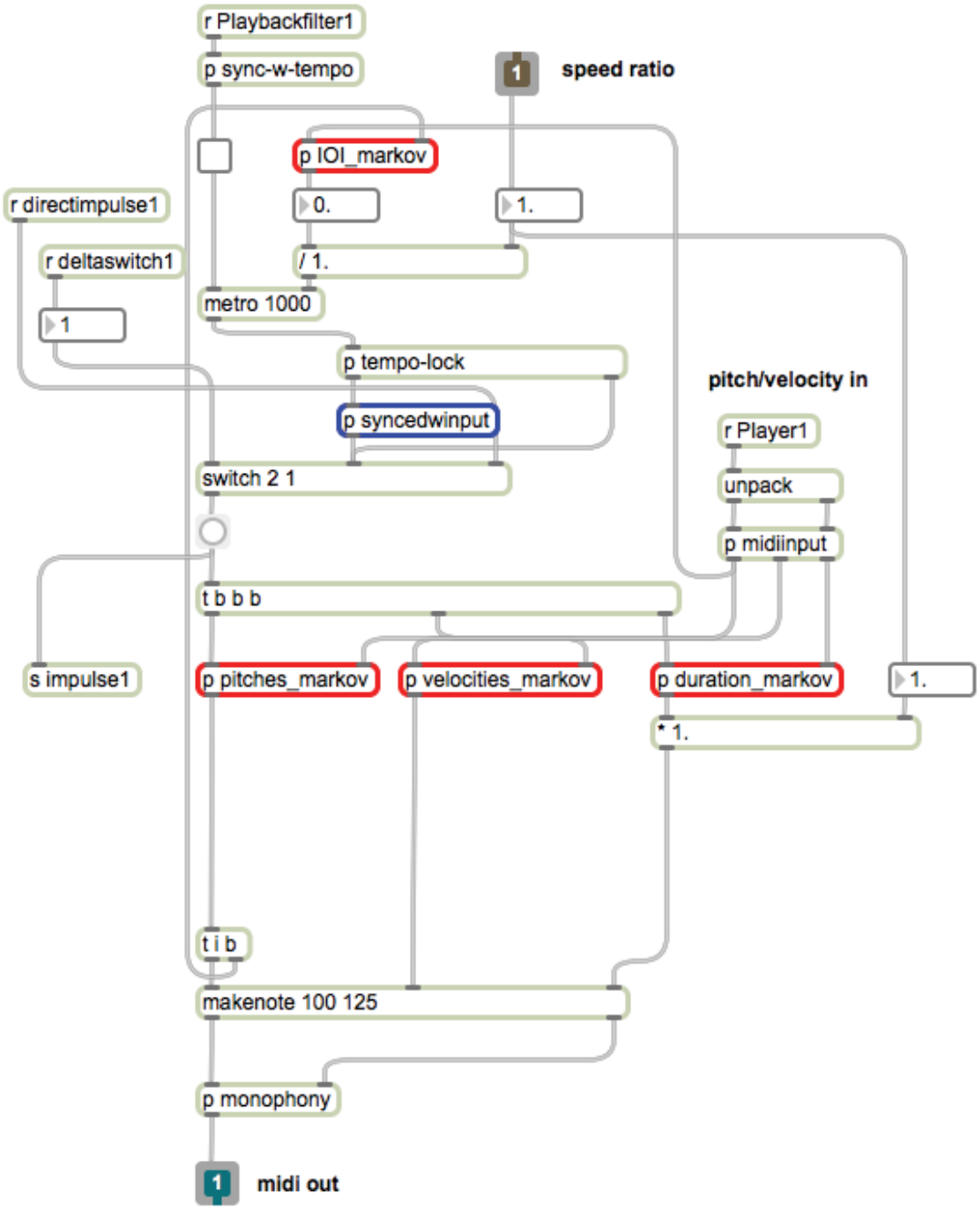


Figure 16: The *probabilityplayer1* subpatch from *Tripartite Markovia*. In red, individual markov chain algorithms per parameter; in blue, the synchronisation algorithm.

Instead, the Markov chains generated from IOI and note duration data were built using absolute values represented in milliseconds, as analysed directly from the input. In comparison to the parameters of pitch and velocity, which exhibit a maximum of 127 possible values, relying upon absolute durational values as input into the system resulted in a limited array of possible transitions between states. Despite their low order, the Markov chains built from this temporal data exhibited more determinism than other parameters in the system.

Although these rhythmic sequences exhibited more determinism, the parametric independence inherent in each player, coupled with the indifference each player displayed to their current musical surroundings, resulted in complete temporal autonomy between the three computer parts and the live input. Given this, it was decided that a balance needed to be struck between this temporal autonomy and maintaining musical coherence between the various parts in the improvisation. This related both to the interaction between the three players themselves, as well as in the interaction between real-time input of the musician and the system as a whole. Searching for this balance, subsequent programming experiments proceeded in a heuristic fashion, trialing different methods of constraining and linking the temporal output of the three players with respect to each other and the live input.

These constraints worked on two primary levels of control. First, by implementing algorithmic control over the global level of musical density, and second on a local, event by event level, by forcing rhythmic convergence between the various parts in the improvising ensemble. Seeking to constrain the density of the system's global output, control over which player was heard at any one time was outsourced to a quasi-random control system known as the *playbackfilter* algorithm, which followed the same direction of influence present in development of the player's transition matrices described previously. In contrast to the autonomous means by which the players generated their material, this control mechanism gave the performer a direct yet opaque form of control over the output of the system. In addition, the same control mechanism gave the three players themselves control over each other's presence in the resulting musical output.

As an improvising musician performs, the *inputplaybackfilter* algorithm counts the number of events performed by the improviser, only enabling the output of *player one*

after a randomised number of note-on messages – its playback threshold – has been reached. After beginning to output material, *player one*'s *playbackfilter* counts the number of events it has performed, subsequently turning on *player two*'s generative output after reaching its own internally generated threshold value. If still playing, *player one* will also turn off *player two*'s output after another randomised number of events has been reached. In turn, *player two* has the same direct control over the output of *player three*, ensuring that this player (performing only chords) is constrained by the output of *player two*, which is itself constrained by the output of *player one*. To close the loop, *player three* has direct influence over the output of *player one*, so that if no live input is being fed into the system, the three players are able to self-regulate the global density of the system as a whole.

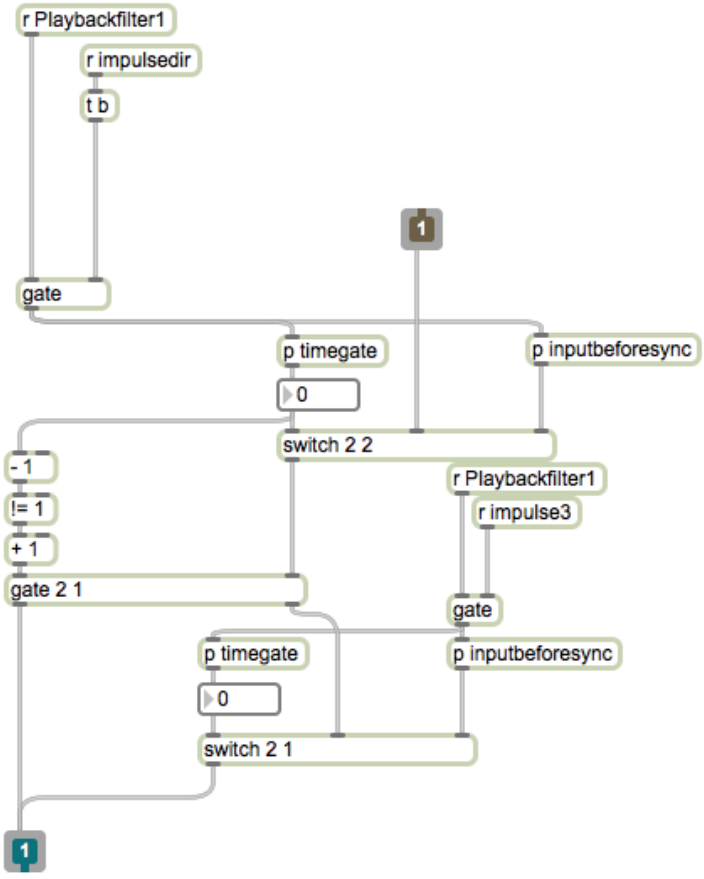


Figure 17: The *syncedwinput* subpatch from *Player 1*. See Figure 16 above for context.

With respect to constraining local rhythmic relationships, a time threshold mechanism was implemented that synchronised the active players in the system in a hierarchical manner (see Figure 17 above). This synchronisation hierarchy also followed the same direction of influence discussed above, proceeding from the live input, to *player one,* then

player two, followed by *player three* and back again to *player one*. A *synchronisation threshold* value (shared amongst the three players) was consulted in order to determine whether or not a player should synchronise with the preceding player in this loop. Beginning with the live input from the performer, passages would be synchronised if their IOIs were smaller than this threshold value (i.e. faster). If true, synchronisation was achieved by momentarily bypassing the output of a player's IOI Markov chain in order to synchronise directly with the incoming events of the player preceding it. The process occurred only if the preceding player was active, and if the player's incoming IOIs fell below the user-defined threshold value in milliseconds. After some initial tests of this approach, it was decided that complete unison rhythm added an undesirable level of predictability to the output of the system as a whole. To improve upon this, a final filtering algorithm was included in order to alternate, in a quasi-random fashion, between complete rhythmic unison on the one hand, and synchronising with every second event received from the preceding player.

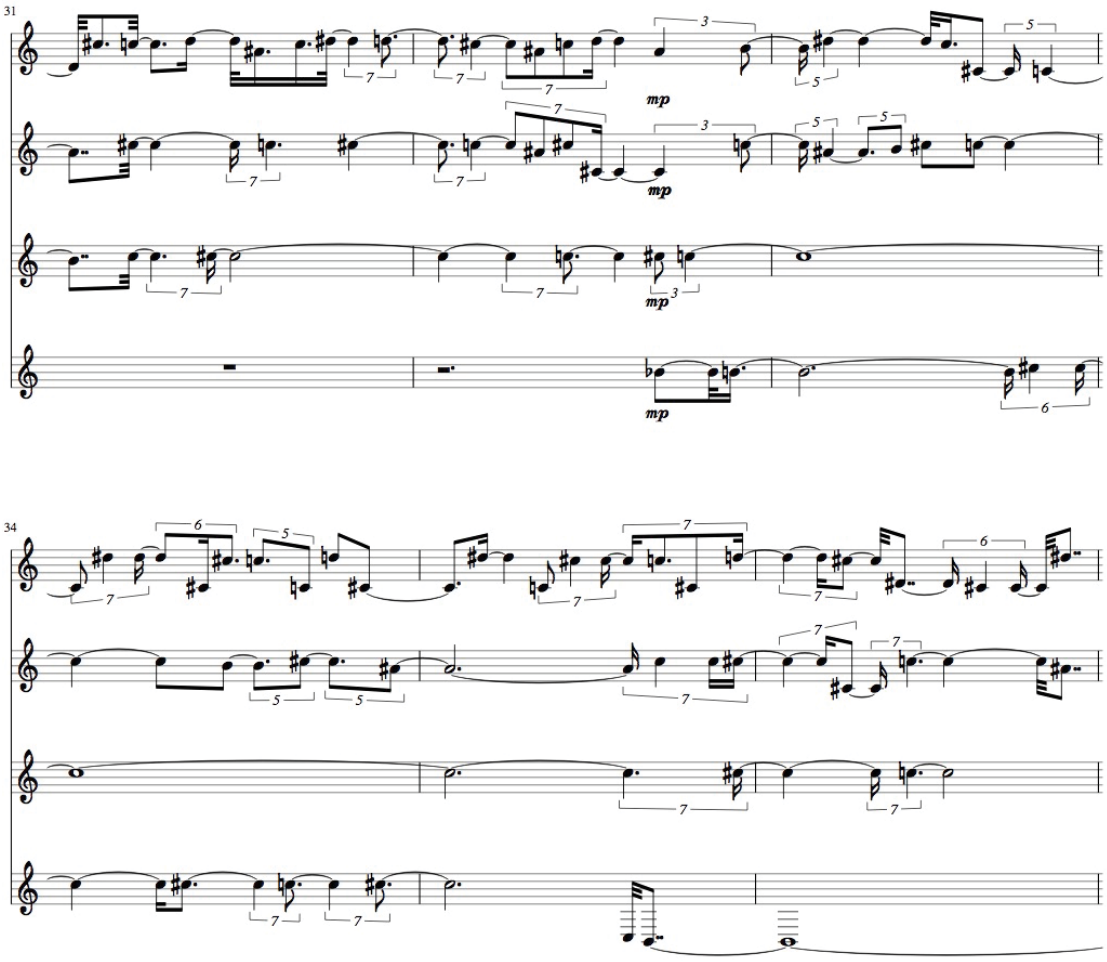


Figure 18: Scored excerpt of a performance using *Tripartite Markovia* – rhythmic values quantised to the nearest 32nd note value.

The constraints on the rhythmic independence of the various parts can be seen in the transcribed excerpt of an improvisation displayed of Figure 18. The top line displays the live, improvised part (performed on a MIDI keyboard), whilst the bottom three layers represent players one, two and three of the software.[22] As can be clearly seen in the score, the rhythmic gestures of the live performer constrain the computer's rhythmic performance. At the recorded tempo of 110 bpm, and with the synchronisation threshold set to the default value of 500ms, all rhythmic values faster than a crotchet beat (545ms) are filtered through the synchronisation algorithm discussed above. Bars 31-34 are a good example of this process. In bar 32, *player one* (second line) performs in complete synchrony with the live performer (top line). Given that the majority of rhythmic values in this bar are shorter than a crotchet in length, it is clear that *player one*'s synchronisation algorithm has been invoked at this point.

Meanwhile, *player two* (third line) carries over a c-sharp from the previous bar, before synchronising with the fifth septuplet pulsation of *player one* inside beat two of bar 32 (the c-sharp). The appearance of this held pitch is due to the fact that the pitches performed by *player one* are longer than one crotchet beat at this point – hence not constraining *player two*. The septuplet synchronisation of *player two* to *player one* in bar 32 displays the periodic synchronisation of the algorithm as described above. This synchronisation with *player one* was invoked by a series of short durations from that player, after which *player two* continues to hold for a longer duration into bar 34. This subsequent change from synchronisation to independence in bar 34 follows the tied low c-sharp player by *player one*, a duration that is longer than the synchronisation threshold once more.

As exhibited by the programming trajectory described above, decisions were made throughout the development of *Tripartite Markovia* in order to constrain the inherently autonomous nature of the Markov generation process. The multi-layered, parametric autonomy exhibited by this system, owing to the use of simple first-order Markov chains, resulted in a form of system autonomy that posed challenges for maximising its potential as an interactive musical system. The probability experiments of this period solidified an approach to sampling performance data as a powerful generative strategy for use in interactive, human-machine performance. However, through working with these

[22] The five note chords of player three have been filtered to the lowest sounding pitch in this particular example.

algorithms in performative testing, it became clear that to maximise a system's interactive potential, autonomous methods such as these needed to be balanced with methods for constraining and controlling the way in which these algorithms functioned in a real-time performance.

4.3 Reflections

After reflecting upon the above practical experiments, a number of emergent conceptual ideas become clear. Some of the key concerns to have surfaced from this period of development are the following:

- Reconciling analysis methods with generation techniques
- Balancing control, influence and derivation in interactive systems
- Hearing vs. listening in computer music systems

4.3.1 *Reconciling analysis with generation*

The programming experiments discussed in chapter relied upon the discretisation and segmentation of acoustic signals into event-based MIDI representations, as well as the sampling of continuous data streams analysed from an improvising musician. The choice to segment the instrumental signal into a standardised MIDI representation (comprised of individual notes with durations and velocities) enabled relatively simple and quantifiable approaches to generating musical sequences and triggering events. Such event-based approaches, whilst useful for certain types of musical generation, did not allow for a nuanced connection between an acoustic instrumentalist's performance and a computer's generative response. What these representations gained through efficient classification of pitch, rhythm and duration, they also lacked with respect continuous changes in dynamics, timbre etc. As a result, the form of analysis chosen dictated the kinds of generative approaches pursued in my work. Whilst some approaches regarding remained promising (probability, Markov modelling etc.), further deterministic analysis such as rhythmic pattern recognition were not easy to reconcile with considered forms of musical generativity. These techniques were therefore not pursued further in my work.

Seeking to provide nuance in conjunction with event-based techniques, the exploration of continuous sampling in *dataatintervals.maxpat* provided an efficient way of making use of temporal gestures performed by the instrumentalist. Similar Dobrian's description of *stealing expressivity* (Dobrian 2004), this approach was conceptually interesting, however my efforts at linking these analyses to other event-based generative designs proved overly cumbersome and somewhat arbitrary. Reconciling the choice of analysis method with the type of generation desired in my system had therefore come into focus. Through considered reflection it became clear that the choice of analysis method largely dictated the form of generation that could be pursued in an interactive system.

4.3.2 *Balancing control, influence and derivation in interactive systems*

With *Tripartite Markovia* and other experiments during this time, I solidified an approach to musical generativity that relied upon data continuously captured from an instrumentalist during performance. Reflecting on my experience in developing and testing the *Tripartite Markovia* system, it became clear to me that I was seeking a delicate balance between control, influence and derivation in an interactive system. As each of the three players in this system could independently generate their own material from analyses of the performer, my largest challenge was to constrain their independence so as to provide a coherent 'ensemble' approach from the software. In developing constraints on *Tripartite Markovia* I followed a *bricolage* programming approach, refining the interactions set up between the three players by iteratively constraining the software's autonomous characteristics. As a result, the constraints on *Tripartite Markovia's* generative autonomy facilitated a more interactive feel to the software. Giving the performer a degree of real-time control over the system enabled a sense of rhythmic coherence in the ensemble as the musician performed. In addition, the same constraints helped maintain a similar coherence without the presence of the performer. This marked a significant turning point for the way in which I approached balancing the relationship between generativity, control and interactivity in such performance systems.

Whilst testing these constraints on the system throughout its development, it became clear that relatively small constraints upon the output autonomous algorithms had a profound effect on the interactive relationship between the system and a performer.

Prior to its synchronisation constraints, *Tripartite Markovia's* players were free to generate material derived from an improviser's performance history. However, they had no sense of contextual awareness to the present improvisatory context. The addition of these constraints served to centralise control over the temporal independence of the system's output to the improvising musician. The immediacy of this addition was noticeable when testing the software, however I was also wary of diminishing the autonomy of the three players through such a form of synchronisation. In recognistion of this fine balance, the addition of a user-defined synchronisation threshold therefore gave the performer nuanced control over the level of synchronisation desireable in any given interaction. Beginning with *Tripartite Markovia,* such 'fine-tuning' controls were provided to the performer for determining the right balance between control and autonomy over the output of an interactive system.

4.3.3 *Hearing vs. Listening*

Throughout this period, I began reflecting upon the purpose of analysis in the creation of interactive software, and the separation between deterministic and non-deterministic approaches to using these analyses. As discussed above, my initial attempts at representing an incoming audio stream as MIDI events, whilst useful, limited the means by which such analyses could be used. Throughout this period, various levels of analyses from the instrumental performer had been used in order to suit separate design goals. The central analysis module, *newaudiotracking.maxpat* was responsible for collecting and segmenting live audio into MIDI representations. The function of this module was to take low-level analysis such as fundamental frequency estimation and peak amplitude tracking, and to apply further analyses to create musically meaningful events (MIDI data). This secondary layer of analysis, as discussed above, was useful for probabilistic approaches as well as and onset/offset detection from the live signal.

In addition to these analyses, higher-level analyses were used to group, classify, segment, store and compare data streamed from *newaudiotracking.maxpat.* With the breadth of analysis approaches I was experimenting with at the time, I began to reflect upon the separation I saw between 'hearing' and 'listening' algorithms in a computer music system. Reflecting upon the differences experiences between approaches, I began drawing a distinction between *passive* data collection and storage - a *hearing* mode - and

active data separation, analysis and storage - a *listening* mode. In the work outlined in this chapter, there is a blend of these two approaches to using analysis data streamed from a live instrumentalist. In the *Tripartite Markovia* system, after pitch-to-MIDI conversion, the software uses *passive* hearing techniques to continuously collect and build histograms of past data. The system is agnostic to the specifics of the data itself, as it is to be used for later generation. However, the synchronisation algorithms within the same software could be classed as *listening* analysis algorithms, actively listening for threshold crossings from the input before changing the synchronisation parameter.

Such listening analyses are most often related to control over the software's output. The software *responds* in some meaningful way once a certain condition is met from the input. Therefore, the system may be conceptualised as *listening for* a particular event. Such control triggers, as is the case in *Tripartite Markovia,* can greatly influence the type of interaction had by a musician with the software. Conversely however, overly proscriptive listening analyses, as was the case in my temporal pattern recognition algorithm, provide exact conditions that need to be met from an input. Such an approach to 'recognition' was not pursed in my work because such *selective listening* becomes deterministic, and therefore more suited to *score-driven* interactive designs (Rowe 1992). The conceptual separation between *hearing* and *listening* in an interactive system is an interesting one. Finding a balance between surprise and constraint is therefore also a consideration in the analysis stage of such an interactive design.

4.4 Conclusion

This first period of my development trajectory was formative both for developing my programming skills, and for discovering emergent themes of my interactive musical practice. An exploratory approach to development that followed a *bricolage* programming style led to both successful musical systems, and redundant technical exercises. The iterative and exploratory practice engaged in during this period revealed concerns, issues and interests in my practice for me to continue pursuing in my creative work, guiding my creative practice towards new areas of interest.

Chapter 5. Wayfinding – Part 2: Synthesis and sampling

5.1 Introduction

As detailed in Chapter 4, formative approaches to interactive system design raised questions for me as a performer-developer designing for improvisatory interaction. What this first period solidified was an approach to creative programming that was exploratory in nature, one that involved non-linear cycles of research, experimentation and evaluation through self-reflection. As discussed previously, such an approach was sometimes prone to an over-development of tangential avenues of enquiry discovered throughout the creative process. This is exemplified in particular by the deterministic approach to temporal pattern matching discussed in Section 4.2.2. However, by following these various lines of enquiry in depth I began to solidify some nascent ideas of about designing for musical autonomy, such as ceding human control over algorithmic processes and balancing autonomy with constraint in interactive computer music practice.

As previously discussed, I had become dissatisfied with forms of generativity that manipulated representations of acoustic instrumental sounds in the form of MIDI data. Instead, I sought forms of sampling-led generativity that made direct use of captured instrumental sound as its raw material. As detailed in the present chapter, this resulted in some interesting yet unforeseen issues and concerns arising in my creative practice as both a programmer and instrumental performer engaging with the systems I was designing. In addition, the exploratory and sometimes tangential nature of my creative programming practice also became responsible for some of the core elements of what would eventually become known as the *_derivations* system. This account details how the present integrated architecture of this software system began as a series of modular components developed independently of each other.

Similarly to the previously discussed period of development, the present period was typified by a confluence of programming approaches and concerns developing simultaneously. Therefore, its chronology as a developmental period of the first part of this period is naturally messy. However, a number of strands of programming enquiry have been identified as forming the backbone of my creative practice during this period.

These areas of programming practice can be broadly categorised into the following two areas:

- Synthesis and processing methods
- Towards integrated systems

The remainder of this chapter details specific projects related to these areas, outlining a trajectory of creative practice and research that eventuated in the *_derivations* interactive performance system, as discussed in detail in the following chapter.

5.2 Synthesis and processing methods

As a saxophonist, I quickly grew dissatisfied with pursuing algorithmic approaches that relied upon only the pitch, rhythm and instantaneous dynamic parameters of my acoustic performance. Although the Markov generation methods employed in the *Tripartite Markovia* system were promising, such a reductive approach to musical generativity neglected the timbral and expressive nuances of instrumental performance. In my work the acoustic instrumental signal had served either as something to be reduced and statistically modeled, or used as a trigger for arbitrary electro-acoustic generation. As discussed in Section 4.2.2, *interactiveoptions.maxpat* and *audiotomiditest.maxpat* made use of synthesis and processing modules to be triggered by audio and temporal analysis. Such modules included frequency modulation, additive synthesis and sound file granulators. However, the implementation of these modules remained simplistic, as these systems were primarily oriented towards the testing of recently devised temporal pattern matching approaches. As discussed previously, these approaches were deterministic in nature, and resulted in the triggering of arbitrary musical events and processes (bell tones, granulation of pre-loaded sound files, etc.). In the context of improvised and interactive musical performance, such approaches situated my work within the boundaries Rowe's definition of controlled, *instrument paradigm* as well as *score-driven* systems (Rowe 1992, pp. 6-7).

Whilst attempts were made to algorithmically control the processing parameters of these modules, the more sophisticated *player paradigm* generative methods possible in the MIDI domain had not yet been implemented in the context of live sampling of my

saxophone performance. Maintaining an interest in sampling-led approaches, it was during this period that I began exploring methods for coherently integrating the sound of the instrumental performer with the output of an interactive system. Searching for flexible and coherent means of using the instrumental signal in my work, I turned to spectral analysis/re-synthesis as a methodology for creating new musical gestures from those sampled from the live performer. Specifically, the techniques of phase vocoding and sinusoidal additive re-synthesis were exploited as complex and nuanced methods for creating this connection between performer and system.

5.2.1 *Four buffer phase vocoder*

The analysis/re-synthesis technique of *phase vocoding* was chosen in order to provide a flexible means of manipulating digital audio materials whilst maintaining the identity of a sampled instrumental source. A phase vocoder (PV) is a form of analysis/re-synthesis that enables independent time and pitch transformations on a digital audio signal. PVs make use of the short-time fourier transform (STFT) to represent digital audio signal in the frequency domain, enabling the independent manipulation of frequency and time (Roads 1996, p. 566). To explore the potentials of this analysis/re-synthesis method in my work, the patch *4-buff-pvoc-test.maxpat* implemented four independent phase vocoder samplers that re-synthesised and transformed sampled and analysed materials. This patch sought to balance control and autonomy in the re-synthesis of materials stored in four separate audio buffers, and to manage polyphony between these multiple playback sources. The patch was designed to sample and analyse short segments of audio from an instrumental source, and for these samples to be re-synthesised and transformed via linked transformation methods. To facilitate testing, the patch was also envisaged as a standalone digital instrument. This instrument allowed for the importation of audio samples into memory from disk, and some of its processing parameters could be controlled directly by a user.

The basic PVs implemented in this patch were developed using the *pfft~* external object, a simple and efficient implementation of the STFT for analysis/re-synthesis purposes. This algorithm required analysed materials to be recorded in buffers of spectral data stored within the *pfft~* abstraction, with each player accessing their own audio buffer housed within a dedicated sub-patch in the object. Each sub-patch enabled the flexible

temporal scrubbing of spectral data, and independent pitch transformations via the *gizmo~* external object. One limitation of this method was the real-time nature of the STFT algorithm. This simple implementation of the PV required audio to be played into the object in real-time in order for their amplitude and phase data to be recorded into memory. In this approach, sound files chosen for analysis must first be played into the *pfft~* before any re-synthesis can occur.[23] In the context of live-sampled performance this initially posed no problems, however an improvement on this method was sought at a later date.

This patch was developed as a controllable digital instrument, allowing a performer to manipulate mixing, playback scrubbing and transposition values of four separate phase vocoder players. The controllable dimension of the software was provided by user interface elements that were mapped to external controllers communicating via the Open Sound Control protocol (Freed & Wright 1997).[24] This direct control over processing parameters was implemented in order to test the possibilities of controlling multiple parameters with a single source of control – a two-dimensional XY pad. Although the phase vocoders were not envisaged for direct control in performance, manual control of these parameters facilitated a gestural understanding of the potential of these parameters during the development process. Four individual phase vocoder players were implemented whose output mix, transposition values and playback scrubbing positions were controlled by three separate mixing algorithms. Controlled by separate XY pads, each two-dimensional space acted as a multi-parameter mixer, simultaneously increasing and decreasing the values of four separate parameters as a pointer traversed the space of the XY pad (see Figure 19). Each of the three multi-parameter spaces could either be controlled manually, or by a randomised traversal of the space, the speed of which could be chosen by the user. In addition, the playback scrubbing could also be algorithmically controlled by four separate instantiations of the *dataatintervals.maxpat* abstraction, enabling more independence between the four scrubbing parameters (see Section 4.2.3 for details on this abstraction).

---

[23] See Dudas and Lippe (2006) for a detailed account of this process.

[24] A screencast performance of this prototype instrument using an Apple iPhone running TouchOSC (Fischer 2008-13) can be viewed at the following URL: https://vimeo.com/16791411

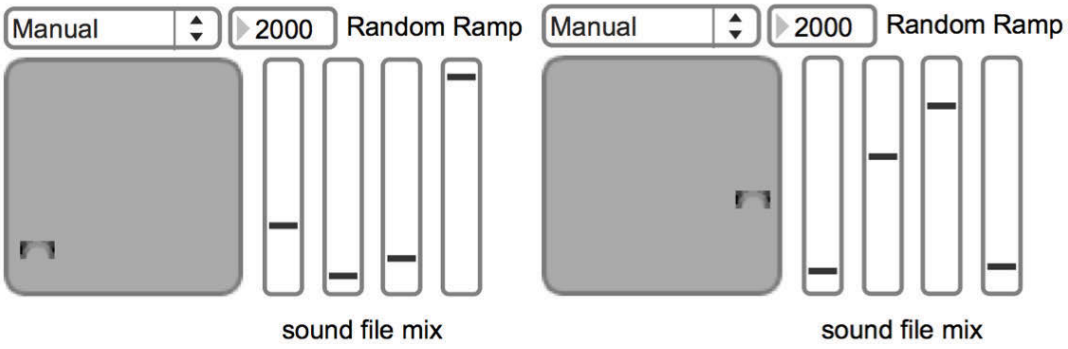


Figure 19: Sound File Mix GUI element in *4-buff-pvoc-test.mapat* – displayed at two separate positions

As a digital instrument, this combination of direct and algorithmic control provided a playful type of 'shared control' between user and system, as defined by Chadabe (1984). For example, by automating playback scrubbing the user was free to manually control output mix and transposition in response to the module's algorithmic playback. The four channel mixing approach enabled a small number of controls to modulate multiple parameters at once. However, this flexibility was also inherently constrained, as each parameter controlled by an XY pad was inversely proportional to the values of the other parameters. For example, as shown in the far left of Figure 19, positioning the cursor in the bottom left corner of the XY pad increased the value of parameter four whilst attenuating the other three values. This patch was an experiment in a new approach towards manipulating sampled audio material in my work. Despite the inherent simplicity of this controlled approach, the combined algorithmic control over sound file mix, transposition and playback position enabled fluid and unpredictable results from the re-synthesis process. However, the patch was not immediately connected to a more holistic interactive environment. In the software, parameters such as record triggers and scrubbing ranges required manual control, and the automation of parameters remained governed by layers of constrained randomisation. These control elements of the patch were therefore treated as placeholders for future autonomous processes.

This aspect of *4-buff-pvoc-test.maxpat* was typical of my development trajectory during this period. My previous focus had been on the analysis and generation side of interactive musical system development, focused upon event-based triggering and MIDI-based generative algorithms. Although much can be achieved in this space, by focusing on these methods it had become harder for me to discover means of connecting analysis, generation and synthesis in my work that satisfied my aesthetic criteria. As a performer-

developer devising my own interactive systems, I wanted to prioritise the sonic quality of the systems being designed for my own performances. My development approach therefore focused upon achieving interesting sonic and musical processes, leaving open the means by which these approaches could then be integrated into more complex interactive environments. The balance between developing sound synthesis/processing and finding autonomous use of sampled material dominated my thinking during this period, as evidenced by my reflective writing at this time:

Reflective memo, November 11th 2010:

One of the biggest issues I'm grappling with is the deferred and algorithmic use of sampled material - both data and audio - and finding ways to influence the output of the system. This is in direct contrast to controlling it directly […]. The decoupling of gesture to sound making is what interests me here. Why? I think it's because as a performer you already have something to control, your instrument - and so this should not get in the way of continually playing the instrument – otherwise it becomes a meta-instrument and not an instrument interacting with abstracted versions of itself.

5.2.2 *Pitch Models*[25]

Having previously explored additive synthesis techniques in my practice, during this period I began exploring sinusoidal additive re-synthesis (SAR) techniques as a form of sampling-led sound generation (Roads 1996, p. 555). Although concerned with achieving a coherent timbral palette between human and machine, given my interest in *player paradigm* interactive approaches I sought abstracted and highly flexible means of achieving this aim. The phase vocoder approach described above, although capable of a variety of transformations, projected re-synthesised signals with a close connection to the original source. The SAR approach enabled the flexibility of using analyses instrumental signal as the timbral basis for a wide range of synthetic transformations. Crucially, these transformations could be at the level of individual sinusoidal partials.

The *pitch models* system implemented an approach to SAR within the context of my recently developed data sampling methods. Using Miller Puckette's *sigmund~* external object, the system extracted a fixed number of partials (frequency-amplitude pairs) from

[25] An example of this early version of *pitch models* can be heard at the following URL: https://soundcloud.com/emeidos/improvisation-tenor-saxophone-and-electronics

the live signal at a constant rate, storing them in an expanding data collection to be used for re-synthesis using additive synthesis techniques. The system made use of the *poly~* object within the Max environment in order to manage the output of multiple overlapping sinusoidal models derived from this sampled spectral data. The *poly~* abstraction developed employed the sinusoidal modeling object *sinusoids~* developed by Adrian Freed of the Centre for New Music and Audio Technologies at the University of California at Berkeley (CNMAT)[26]. This object was used to output and manipulate sinusoidal models of an instrumental input stored from the output of *sigmund~*'s partial tracking and sinusoidal decomposition analysis.

5.2.2.1 Sampling and storage

The storage approach used in *pitch models* was aided by the pitch detection and amplitude thresholding techniques developed in *newaudiotracking.maxpat* (see Section 4.2.1). The system provided three storage types for the sampling of sinusoidal models from the *sigmund~* object: *pitch detection* mode was a momentary sampling method that captured a single model from the input upon the detection of a stable pitch onset; *continuous* mode provided a continuous analysis and storage of sinusoidal models at a constant rate (in the vicinity of 100hz); whilst *threshold* mode made use of a user-defined amplitude threshold to start and stop continuous model sampling. Each model output from *sigmund~* was comprised of fifty frequency-amplitude pairs compiled into a single list. Once collated, these lists were parsed and collated into a format that could be used directly for re-synthesis via the *sinusoids~* external object. Sinusoidal models were then stored incrementally in an indexed database, enabling the system to derive its sonic output from a large variety of captured analyses of an instrumentalist's past performance. Figure 20 shows this storage process at work in the [p storemodels] subpatcher.

Given the large amount of data stored by *pitch models'* analysis and storage mechanism, a synthesis approach was sought that could make nuanced use of these captured models in performance. A common method of implementing SAR is via peak or partial tracking analysis, an analysis technique that aids in maintaining the spectral contours of the

---

[26] These objects are available within CNMAT's large collection of external objects for Max: http://cnmat.berkeley.edu/downloads

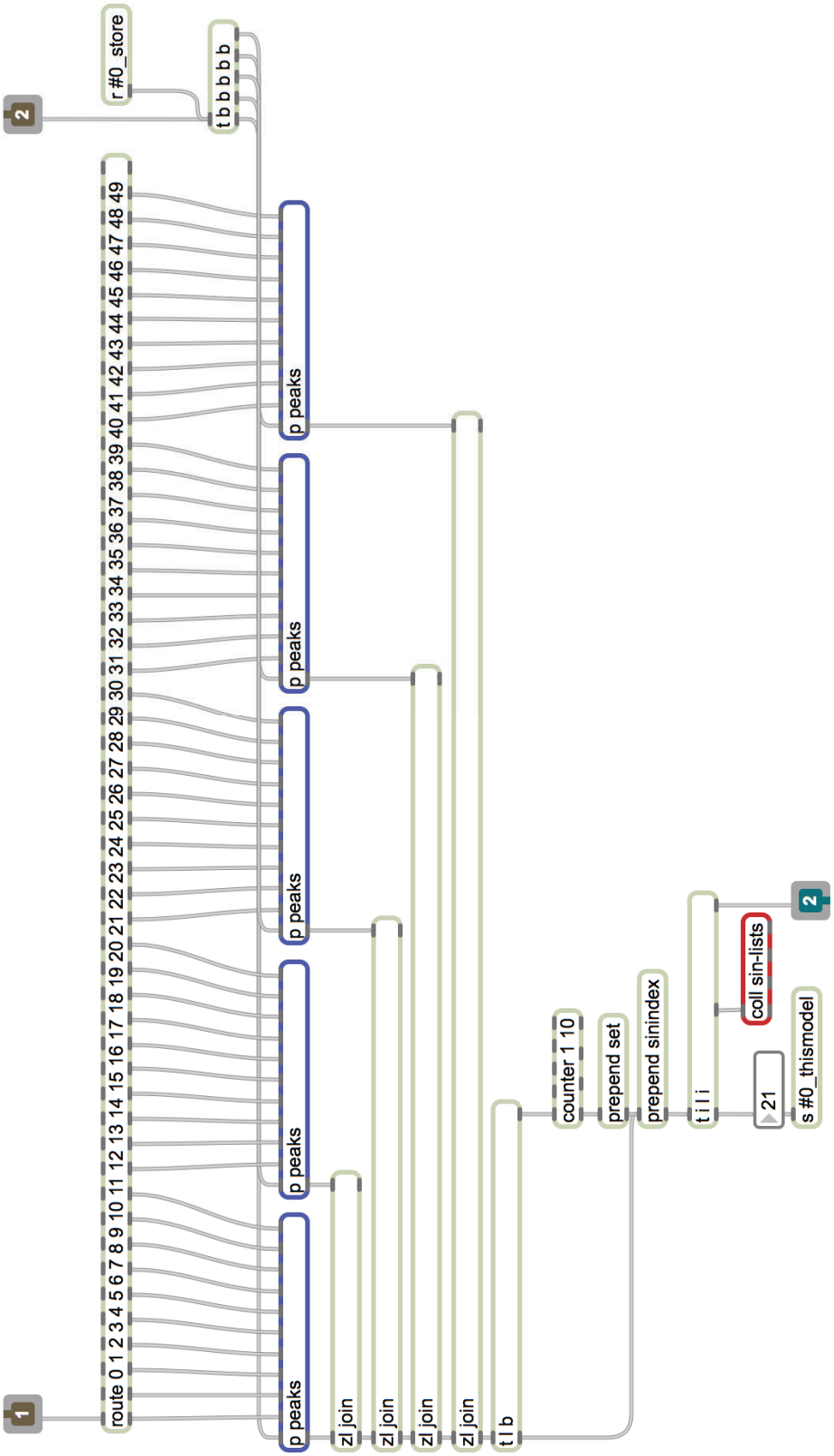


Figure 20: ***storemodels*** **supatcher from** ***pitchmodels.maxpat.*** **Incoming frequency-amplitude pairs are parsed into lists of ten partials in the [p peaks] subpatchers, grouped together into a single list using cascading** ***zl.join*** **objects and stored in the [coll sin-lists] data collection for later output.**

analysed input in the re-synthesis process (Klingbiel 2009; McAulay & Quartieri 1986; Roads 1996, p. 569; Serra 1989).

This method allows for the re-synthesis of sinusoidal components organised into continuous tracks, minimising artefacts caused by abrupt changes in oscillator frequency and amplitude.[27] Whilst useful as a live re-synthesis method, I decided to avoid this continuous approach for reasons of efficiency of storage and flexibility of re-synthesis. Instead, I chose an approach that was less faithful to the spectral contour of adjacently analysed models, yet allowed for flexibility in their automated manipulation and structuring. Rather than seeking to directly re-synthesise sampled sonic materials, this method used captured models as raw spectral data from which to polyphonically synthesise sounds derived from the timbre of the live instrumentalist.

5.2.2.2 Model triggering

Given the complexity of *pitch models'* various processing parameters and its connection to the musician's past performance, it was decided that a simple start/stop control would suffice as a method of controlling the module's output. The algorithmic re-synthesis of *pitch models'* sinusoidal models was either triggered via amplitude threshold crossings of the instrumentalist, or engaged continuously throughout a performance. The output of stored sinusoidal models from *pitch models* was achieved via a dynamic and layered algorithmic process analogous to traditional sample and hold techniques. A looping rhythmic module was developed that generated periodic triggers whose inter-onset-intervals (IOIs) were derived from two dimensional lookup table drawn by the user. This lookup table was known as the module's 'rhythmic envelope', whose range of IOI values was set by the user-definable *impulse range* parameter (see Figure 21). This module launched a ramping control signal that queried the lookup table in series over a specified period in milliseconds. A *metro* object periodically sampled new values from the resultant stream of IOI data, updating its delay time thereby creating dynamic rhythmic patterns. To avoid creating a repetitive rhythmic loop, the period of the ramping control signal itself was modified at the beginning of each cycle of the control signal. This was determined by a random value chosen from between the user-defined *length range* in

[27] An implementation of this re-synthesis method using *sigmund~* is usefully demonstrated in the *sigmund~* Maxhelp patcher.

milliseconds. This created a dynamic expansion and contraction of the table lookup process, facilitating a variety of temporal trajectories from the same range of IOI values.

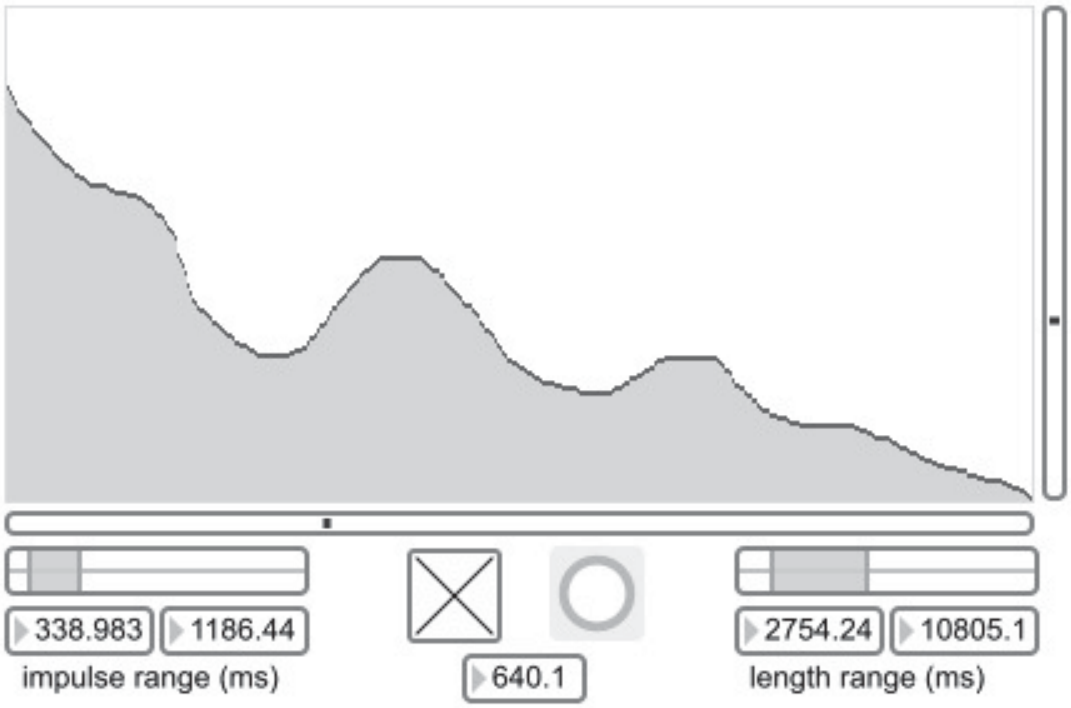


Figure 21: The IOI lookup table in *pitchmodels.maxpat*

5.2.2.3 Model selection

In order to provide flexibility to the automated output of sinusoidal models, three separate options were provided to determine how models stored in the growing database were chosen for re-synthesis. Each trigger received from the rhythmic envelope was sent to one of three model selection algorithms. These algorithms included *random range,* a random choice of a model index from with a specified index range; *this model,* the direct output of the most recent model analysed (triggered from a pitch or amplitude onset); and *random sweep,* an automated data stream controlled by a looping control signal. In my performative tests of this system, the two most common options chosen were either *this model* or *random sweep.* However, both of these options provided very different interactive potential in performance. *This model* enforced a triggered and controlled mode of performance with *pitch models* that did not make use of its sampling capabilities. In this mode, the system acted as a synthetic extension of the performer, shadowing the performer with re-synthesised gestures derived from the most recently analysed performance. The looping control signal approach of *random sweep* was relatively simple,

yet capable of emergent and unpredictable results. The *dataatintervals.maxpat* abstraction was once more employed to create a continuously looping control signal for the automatic navigation of the database of stored models. This control signal was scaled to the current number of models stored in the database, enabling the rhythm module to periodically select models to output from this data stream.

5.2.2.4 Model synthesis

Polyphonic synthesis of models selected for output was achieved using the *poly~* object, an efficient means of developing polyphonic synthesisers in the Max environment. The *poly~* abstraction retrieved stored sinusoidal models and facilitated transformations to the models before, during and after their re-synthesis. Once an individual *poly~* voice has received a chosen model index, this model was retrieved from the central database and sent through a 'partial scrambling' algorithm to manipulate the amplitude values of the output list. The algorithm parses the static model it receives from the database, and 'scrambles' the amplitudes of the first few partials via list interpolation, thereby altering the timbre of the model during its re-synthesis stage. This scrambling process randomly swaps the first *n* amplitude values stored in the static model, interpolating smoothly between amplitude values over the length of a synthesised gesture.[28]

As illustrated in Figure 22, the scrambling algorithm interpolates between the original model (Figure 22a) and the scrambled model (Figure 22b), sending this interpolation continuously through the *sinusoids~* object for re-synthesis.[29] Further transformations to the stored models included model transposition prior to re-synthesis, stereo panning and control over the amplitude envelope of output gestures. In order to further enliven the interpolated models, a final transformation was applied to the final output of each model voice by way of formant filtering. Also developed by Adrian Freed, the *resonators~* object implements a parallel bank of resonant filters from sinusoidal model rather than using additive synthesis techniques (Jehan, Freed & Dudas 1999).

---

[28] The user chooses the number of amplitudes to scramble, and the length of this scrambling process in milliseconds.

[29] An example of this scrambling process is also illustrated in an animated GIF image accessible at the following URL: http://bit.ly/scramble_partials

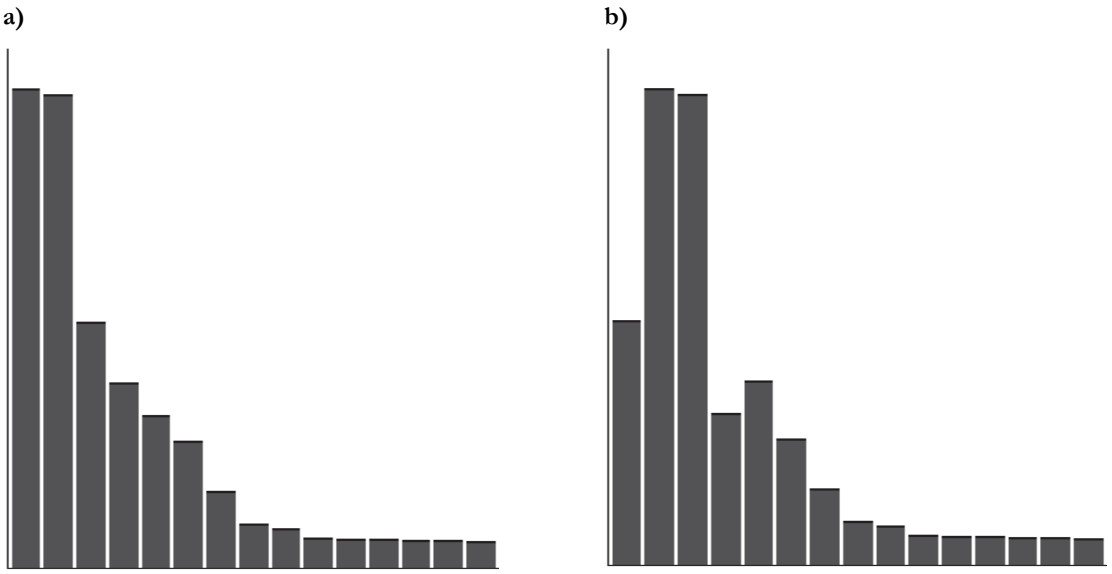


Figure 22: Visual representation of the partial scrambling algorithm used in *pitchmodels.maxpat*. Figure 22a shows the amplitude distribution of a static sinusoidal model, whilst Figure 22b displays the results of the model after the 'scrambling' process. In Figure 4b the amplitude of the fundamental frequency has been replaced with the original amplitude of partial three; the amplitude of partial four is replaced with that of the fifth partial, etc.

Making use of the linked *res-transform* object in 'formant form' mode, the output of the *sinusoids~* object was passed through a series of interpolating vowel formants. This approach was modeled upon the *vowel~* abstraction programmed by Wright and Zbyszynski and distributed with the CNMAT collection of Max objects and abstractions. The formant algorithm randomly interpolates between various formants at a user-specified rate. These formants were provided by the *vowel~* abstraction mentioned above in a text file entitled 'ircam-vowels.txt'.

5.3 Towards integrated systems

As discussed in the previous two synthesis approaches, live sampling of both audio and spectral data was the primary method of driving the computer's sonic contributions in my systems at this time. Siphoning material from the instrumental performer continued to be of interest to me during the period that followed, however I began to approach the use of the sampled material with more nuance and complexity. The following section details two separate systems that built upon my previous work in this area, namely the *Live-processing-1* and *phrase player* systems. In Section 5.3.1 I detail the integration of *4-buff-*

pvoc and *pitch models* into a larger interactive environment, highlighting the specific technical and conceptual challenges arising from this process. Following this, Section 5.3.2 details a further system that addressed some of the shortcomings of momentary approaches to live sampling materials from the performer. This section details how the continuous and cumulative approach of *pitch models* became a model for the way in which audio data was sampled and stored for use in a live sampling based system.

5.3.1 *Live-processing-1*[30]

After further testing and refinement I began integrating the above-described analysis/re-synthesis methods as independent modules within an integrated performance environment. The system *Live-processing-1* combined the SAR techniques explored in *pitch models* with the phase vocoder implementation of *4-buff-phase-vocoder-test.maxpat* (now known as *4-buff-pvoc*). Although both modules used analysis/re-synthesis as their primary methods of sound generation, they did so in contrasting ways. These two modules were therefore deemed useful starting points for the development of a live sampling based human-machine performance environment. In *Live-processing-1,* the live instrumental signal was used to trigger algorithmic processes as well as to provide the source for FFT analysis and re-synthesis. The central analysis module used within the system was *newaudiotracking.maxpat* (discussed in Section 4.2.1), which was used to stream MIDI pitches, amplitude values as well as the frequency-amplitude pairs needed for *pitch models'* analysis.

By seeking to integrate these two modules into an overall interactive environment, I engaged in a form of programming practice that McLean and Wiggins have described as *bricolage programming* (McLean & Wiggins 2010). Although my overall aim was to develop a dynamic human-machine improvisation environment, I chose an exploratory approach that was firmly situated within the musical and interactive potential of these recently developed synthesis modules. By beginning with the development of these sound generation modules, my exploration of interactive methods followed a 'hands-on' approach that was intimately connected with my experiences of human-machine improvisation in practice. In *Live-processing-1* the development of musical interactivity was

[30] A screencast demo of this performance system can be streamed at the following URL: https://vimeo.com/19109988

therefore an emergent property of the interaction between my saxophone performance, the affordances of these existing modules and subsequent programming decisions made in response to my interactions. This was an intuitive and creative task.

As both *pitch models* and *4-buff-pvoc* possessed their own idiosyncratic automated processes, the challenge was to develop plausible means of integrating these two modules within a dynamic and coherent interactive framework. Whilst *pitch models* had been tested as a standalone performance environment, *4-buff-pvoc* had not yet been coherently integrated into a live performance context. Importantly, neither of the modules had been used in conjunction with other synthesis/processing modules in a broader interactive context. I began by considering the way in which the live instrumental signal could be used to control and influence the output of these modules during performance. Given the nuanced and complex automated capacities of the existing modules, decisions about performer control and influence over these processes were shaped by the inherent affordances of their combined musical agency. Proceeding in a *bricolage* fashion, I began by repurposing previous event-based approaches to trigger these pre-defined automated processes. This included the use of pitch and amplitude onset and offset detection as facilitated by the *newaudiotracking.maxpat* abstraction. During performative testing, the interactive implications of these programming decisions prompted revisions to my initial algorithms, and to the automated processes of the modules themselves. *Live-processing-1's* development therefore exhibited a process of *interactive stabilisation* between performance and algorithmic refinement (Pickering 1995). The details of this process are outlined below.

5.3.1.1 Control and automation of 4-buff-pvoc

To trigger the live sampling and FFT analysis of *4-buff-pvoc*, a simple time threshold was employed to filter incoming pitch detection and amplitude threshold crossings through to control sampling/analysis triggers. A drop-down menu enabled the user to choose either pitch or amplitude detection as the control signal, and a time threshold value in milliseconds was chosen to filter detected events. Once the time threshold had been reached, a gate was opened to allow the following event detected to trigger sampling of

the input audio into one of four fixed-length audio buffers.[31] This constituted a simple form of event filtering; after an event passed through the gate it restarted the time threshold, limiting the frequency at which input from the musician was sampled and analysed for re-synthesis. As the performer continued to improvise, *4-buff-pvoc* therefore had access to up to four individual samples at any one time, representing the most recent history of the instrumentalist's performance.

In contrast to this onset detection approach to triggering, *4-buff-pvoc's* sonic output made use of an offset detection method by way of a silence threshold algorithm similar to that described in the work of Cuifo and Hsu (2005; 2006) (see also Section 2.3.1). This threshold tracked the amplitude of the instrumental signal, starting a timer when the amplitude dipped below the threshold set in *newaudiotracking.maxpat*. If the input remained below this threshold for a user-specified length of time, the threshold algorithm sent a trigger to engage scrubbing algorithms that facilitated the re-synthesis of samples captured from the input. A 'performance time' value in milliseconds also constrained the length of time in which the algorithmic re-synthesis took place. Once the end of this time threshold was reached (a random value chosen between 10,000 and 20,000ms), the scrubbing algorithms were turned off and the silence threshold mechanism was again re-engaged.

Re-synthesis of sampled and analysed audio in *4-buff-pvoc* was automated in a similar fashion to the original *4-buff-pvoc-test.maxpat* (see Section 5.2.1), with transposition, output mix and playback scrubbing following independent automated processes. Whilst both transposition and sound file mix maintained the interdependent controls provided by the four channel mixing algorithm, for playback scrubbing both the XY pad and *dataatintervals* approaches were replaced with a randomised scrubbing algorithm aligned to each phase vocoder. Seeking more independence between the four phase vocoder players, the XY control mechanism was abandoned for the scrubbing parameters due to the interdependence it enforced between the four players. Although *dataatintervals.maxpat* had initially provided this independence, it was decided that this abstraction was too arbitrary and cumbersome as a means of controlling processing parameters in my systems.

---

[31] The buffers were recorded sequentially, and buffer lengths were user-definable and usually set between 4000-8000ms.

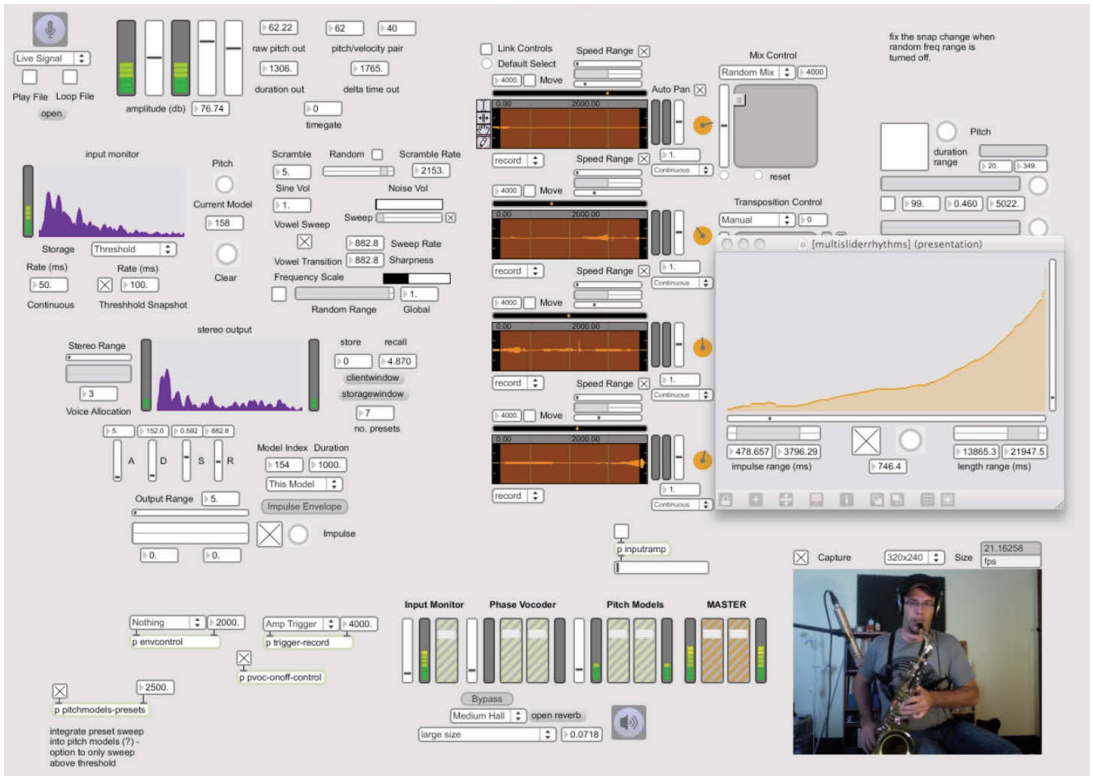


Figure 23: Screenshot of a performance with *Live-processing-1*. The left hand side of the figure features *pitch models* many output parameters, whilst the audio buffers in the centre represent the sampled and analysed material used in *4-buff-pvoc*.

As the polyphony of both audio and control data increased in my systems, such randomised data streams were found to provide a suitable level of complexity for the control of individual parameters. Given this observation, it was decided at this time that sampled data would only be used if directly siphoned from the current performance context. However, such automated control signals in my systems were fine-tuned over time in order to achieve a balance between novel and musically desirable output gestures. Adjustments were also made to account for the constraints provided by other interdependent parameters. This approach was a natural part of the performer-developer dynamic, as performative testing informed fine-grained programming decisions to improve the musical potential of the modules being developed.

5.3.1.2 Control and automation of pitch models

Whilst *pitch models* had been tested as a standalone performance system, its direct integration into *Live-processing-1* highlighted the module's complexity as a synthesis instrument, and the need to develop efficient methods for automating changes to its

many and varied parameters. Prior to its inclusion in this system, various static preset configurations aided in enabling rapid access to chosen configurations of these numerous parameters. To achieve further flexibility and variety in the module's automated performance, an interpolating preset system was developed using the *pattr* family of Max objects. These objects (*autopattr, pattrstorage,* etc.) provided a flexible method of storing, recalling and interpolating between presets containing a large number of parameters. In all, more than twenty parameters were included in the *pattrstorage* data collection in the *pitch models* module. A user could easily store snapshots of these parameters as new presets to be saved and exported as a standard JavaScript Object Notation (JSON) file to disk (see Figure 24).

	Name	1	2	3	4	5	6	7
	attack	2000.	5.	1000.	5.	5.	1.	5.
	decay	0.	1000.	0.	1000.	25.	50.	15.
	duration	3000.	1000.	1500.	1000.	1000.	50.	100.
	freqrange	0 0	0 24	0 22	2 11	0 24	11 12	11 12
	freqrangeonoff	0	0	0	1	0	1	1
	freqrangeonoff[1]		0	0				
	globalscale	1.	1.	1.	1.	1.	1.	1.
	release	1000.	700.	1000.	100.	1000.	50.	50.
	scramble	5						
	scramble-on-off	0						
	scramblerange	0 49	0 49	0 49	0 23	44 49	0 49	0 6
	scramblerate	4000.	2000.	4000.	1100.	2000.	1000.	200.
	sharpness	2.011765	1.235...	1.129...	1.129...	1.129...	2.011765	2.611765
	sustain	1.	0.4	1.	0.5	0.6	0.3	0.555...
	sweeprate	1000.	1000.	1000.	100.	1000.	1000.	2000.
	vowelsweeponoff	1	1	1	1	1	1	1
	voweltrans	1000.	1000.	1000.	100.	1000.	1000.	2000.
	function-rhythms							
	impulse-max	1949.	847.	677.	889.	974.	125.	200.
	impulse-min	974.	338.	296.	338.	423.	75.	90.
	length-max	10345.	8932.	11228.	5296.	6496.	3000.	3000.
	length-min	3777.	4142.	5508.	1483.	2683.	1000.	1000.
	rhythmic-function	2.538...	0. 7.6...	2.538...	100. 9...	89.34...	0. 0.2...	10.15...

Figure 24: A screenshot of the presets and associated parameter values within *pitch models*. The current interpolation value of this algorithm is 4.8, as visually represented in this figure by the relative shading of columns four and five.

This preset system facilitated a great degree of flexibility in the sonic character of the *pitch models* system. The system was designed so that a single control signal could control the interpolation between different preset states of this large array of individual parameters. The *pattrstorage* object facilitated the interpolation process by accepting both integer and

floating-point values, with floating-point values determining the linear interpolation between parameters stored in two adjacent presets. In addition to such adjacent interpolation, *pitch models* made use of the 'recall' message to *pattrstorage*, a method facilitating the interpolation between non-adjacent presets. Each preset saved in *pitch models* could be therefore be interpolated with any other preset in the database, significantly broadening the scope of possible sonic gestures to be generated from this synthesis module.

As a form of high-level control in *pitch models*, the interpolation parameter was treated in a similar way to other streams of control data in my systems – through the use of constrained randomisation. A random walk algorithm was implemented to automate the adjacent interpolation process, whilst non-duplicating random algorithm controlled non-adjacent preset interpolation. This was achieved by cycling through each preset in the collection in a random order. Both algorithms were automatically scaled to the number of presets stored in the database, and the rate of the morphing process determined by a fixed interpolation interval chosen by the user. The user could also choose which type of preset automation was to be used (adjacent or non-adjacent), and if engaged this preset morphing process was continuous during the output of stored sinusoidal models.

5.3.1.3 Control and interactivity in Live-processing-1

By placing these two existing modules into an integrated interactive environment, I contended with the complexity and limitations of live sampling and re-synthesis as methods for musical interactivity in human-machine performance. The aim of this exercise was to assemble a system from existing components, one that was functional enough to facilitate interesting musical interactions during performance. As a performer-developer, I was first and foremost concerned with creating working systems that could be tested in rehearsal with a live instrumental input. My programming approach therefore relied upon a 'code-and-fix' methodology that sought to build working prototypes without much concern for their further integration with other systems.[32] This approach to development was focused upon quick results, and of integrating previously developed algorithms to incrementally improve upon an evolving yet functional artefact.

[32] In the context of formalised software development methodologies, this programming approach is often derided as 'cowboy coding' (Boehm 2006; Lindell 2012).

Given the scope and complexity of the various moving parts already present within *4-buff-pvoc* and *pitch models*, previously explored techniques were often absorbed into my systems without immediate concern for their revision or expansion. The implementation of event-based triggering and randomised automation, whilst initially considered as placeholders for future algorithmic methods, became part of the software's internal dynamics, and therefore its material agency. As a developer, a large part of this form of development was facilitated by the partial externalisation of my decision-making process. By beginning with these two existing modules, further development relied upon my perception of the combined agencies of these systems to suggest new avenues for exploration. As a form of design, this process allowed the 'back-talk' of the materials to be foregrounded as an integral part of the decision-making process (Schön 1983, p. 79). Rather than following a premeditated development trajectory, *Live-processing-1* evolved as a function of the combined potential of its constituent parts, as revealed and navigated through interactive human-machine performance.

By considering the environment as a whole, the internal dynamics of both *4-buff-pvoc* and *pitch models* were altered to suit the overall interactive context. In addition, a balance was sought between direct control and automation in *Live-processing-1,* which was achieved through the implementation of simple performer-controlled triggering and streamlined software automation. Although the triggering methods described above enforced a degree of performer control over various musical processes, this control remained high-level and the exact results any direct triggering remained opaque to the performer. In contrast to the deterministic pattern matching techniques described in Section 4.2.2, these simple algorithms did not require further layers of analysis, nor did they seek to recognise specific patterns from the input in order to trigger musical events. Designed to be unobtrusive, these methods facilitated the high-level control over sampling and playback to be a function of the instrumentalist's continued performance. Nonetheless, the directly triggered nature of both sampling and system output was a noticeable performance dynamic enforced by the software, a factor I considered less than ideal in a musical system of this type.

In both of these independent modules, a tension existed between sampling-led methods of sound generation, and the growing complexity of the many and varied parameters used to control re-synthesis. As discussed above, these parameters were

largely automated through layered forms of constrained randomisation, not through nuanced and sophisticated generative and machine learning algorithms. Although I was interested in employing such approaches in my work, I was also conscious of the potential for such methods to obfuscate the relationship between the sampled material and its origins. Given the complexity of this issue, I sought to simplify the relationship between instrumental performer and *Live-processing-1* by maintaining independence between the algorithmic processes themselves and the analysis of the human performer. Although the performer provided the material with which the system generated its sonic materials, the improviser had no further influence over their structuring or processing. Real-time analysis of the instrumentalist was used purely for event-based triggers and for *pitch models'* sinusoidal decomposition.

Through the design of *Live-processing-1,* balancing the conceptual clarity of live sampling and the novelty of autonomous generativity had revealed itself as a significant creative challenge in my work. Although relatively simple, the above-described layered approach to randomised automation did provide a degree of complexity that provided interesting results in interactive performance. However, having acknowledged this tension between sampling and autonomy in my work, I began searching for alternative that could further enhance my interactive performance systems. As I had arrived at this issue in my work as a result of my *bricolage* approach to development, it became clear than a more considered and planned approach to addressing this challenge was needed from my practice.

5.3.2 *Phrase Player*[33]

After implementing triggered and momentary methods of live sampling and processing in *Live-processing-1*, I sought to expand my approach to enable a wider array of sampled material for use in performance. I also sought an approach that displayed less determinism and more potential for autonomy in its sampling and processing capabilities. In the *4-buff-pvoc* module of *Live-processing-1*, amplitude onsets analysed from the audio stream served as triggers for sampling and processing using fixed-length audio buffers. Although the content of these buffers were available for a wide variety of transformations, the momentary nature of this sampling method limited the range of

[33] A short simulation of the output of this system is documented at the following URL: https://vimeo.com/18285721

possible musical material to be used at any one time. Performing using this approach, I quickly became dissatisfied with the structural limitations of such momentary sampling. Given previous experiences with the use of histograms, Markov chain processes (see Section 4.2.4) and the cumulative sampling achieved in *pitch models,* this momentary method lacked flexibility. The sampled material that could be held for processing at any one time was limited, and these methods suffered from some similar structural limitations as the *live looping* practices discussed in Chapter 2 (Section 2.3.3.1).

Phrase player was subsequently developed as a proof of concept system to deal with the deficiencies encountered with these previous approaches. This new system involved the segmentation and storage of continuously sampled material from a live audio stream. This required the management of a more rigorous and continuous approach to sampling live audio, one in which the segmentation and storage of captured musical material could be automated in a meaningful fashion. Using a similar approach to techniques employed in *Live-processing-1*, a form of silence thresholding was used to capture and store timing information from an incoming audio stream, as well as to efficiently start and stop recording of the audio input to disk, rather than to an audio buffer in memory. This segmentation approach was designed to capture small units from the input delineated by 'phrase boundaries'. This simple event-based conception of what constituted a phrase was used to efficiently chunk the audio stream into manageable units for later output, an approach directly inspired by the silence thresholding techniques of Ciufo and Hsu (2005; 2006) (see also Sections 2.3.1 and 5.3.3.1).[34]

Given the choice to record the live input directly to disk, a decision was made to limit the number of audio files written during a session so as to make the recording/writing process more efficient. Two separate silence thresholds were used as segmentation triggers acting upon the recording on the live input. The first user-definable silence threshold was relatively short (in the vicinity of 100ms), and was used trigger the storage of a growing list of time-stamped 'cue-points' referring to the segmented phrases. These cue-points were cumulatively stored in a format commonly used by the *sfplay~* object for locating cue-points in audio files. A second silence threshold (fixed to 1000ms in length)

[34] Despite these units being dubbed 'phrases', it was not my intention to classify the content between these segmentation boundaries as constituting a musical phrase by any analytical measure. Such an approach would have necessitated a detailed form of analysis that was beyond the scope of the current approach.

was used to trigger the writing of this file to disk, and to send the list of cues associated with each file to a group of four *sfplay~* objects. Figure 25 displays the logic of this cue storage process. Inputs one and three receive timestamp data (in milliseconds) derived from a central clock triggered from the output of the first silence threshold. These timestamps are then grouped together, prepended with an index number and stored temporarily in the [coll preloadinfo] data collection object.

Upon the crossing of the second silence threshold, the accumulated cue-points are dumped out of the coll object and its memory is cleared (received via the [r #0_dump] receive object). Each cue-point is then prepended with the preload symbol and send via the [s #0_storecues] send to the *sfplay~* objects for later output and processing. This final process coincides with the writing of the file currently being recorded to disk. The *sfplay~* objects were then able to access the file stored on disk and the various segmented phrases indexed to this file. In addition to storing this information, the length of each phrase is also stored in the data collection [coll phraselengths] for later use in processing.

The generation and processing methods used relied heavily upon both data sampling and constrained random generation techniques similar to those described in *Live-processing-1*. In an effort to provide variety to the output of the phrases captured from the live signal, a polyphonic approach was implemented that saw each playback device accessing the list of stored phrases by following their own independent generative trajectory. In the software, the growing list of recorded cue-points was used to constrain the range of phrases available for output by four separate *sfplay~* objects. Each playback device referenced an independent data stream continuously outputting potential phrase indexes. Using an identical method to *pitch models,* these data streams were generated by four independent instantiations of the *dataatintervals.maxpat* abstraction, each referring to a different stored list of sampled data curves for its output. Working independently, each data stream provided a scaled control signal from which each playback device would sample its next phrase index for output. The specific curve used could be chosen by the user and once enabled the curve would simply loop throughout the duration of a performance. As the data streams were scaled to match the growing range of phrases captured from the live input, the space of possible materials each player grew in direct proportion to the current performance history.

In addition to the output logic, another live sampling method was used to process the polyphonic output of the system. Throughout a performance, amplitude curves were analysed from the instrumental signal and stored in an expanding database. These curves were scaled for use as control signals to modulate the delay time of individual feedback delay lines processing each playback device. Figures 26 and 27 show both the sampling and storage logic, as well as the scaled output of the various curves respectively. As shown in Figure 26, during the recording of a phrase the amplitude values of the incoming signal are collected in the subpatcher and stored in the [coll ampstream] temporary data collection. Once the phrase boundary (indicated by the silence threshold trigger) has been reached, the phrase on/off toggle is turned off forcing the output of the current stream. This stream is collated into a list, prepended with an index number and stored in the data collection [coll amplists] for later use. These amplitude curves are then accessed for use as control data for modulating feedback delay lines. The scaled output logic for these amplitude curves is displayed to the right of Figure 26. The identical subpatchers [p-ampstream-output(1-4)] control the output algorithms used to output the individual curves.

Figure 27 displays the inside of the first of these subpatchers. As is clear from this figure, a random number generator is used to access the curves stored in the [coll amplists] data collection. Upon the output of a phrase, each playback device outputs the phrase length stored in the [coll phraselength] data collection to the corresponding phraselength send object. Upon the output of a phrase by the playback device, a random amplitude curve stored in the data collection is chosen for output. Data received via the receive object [r #0_phraselength1] initiates this process. As can be seen in Figure 27, the phrase length received is then used to calculate the ramp time between each individually stored data point in the amplitude curve. This method ensured that each sampled amplitude curve used as a control signal would be output over the exact length of the chosen phrase. Finally, the output of each buffer was further automated with a stereo panning mechanism for each playback device, and the inclusion of an automated mix between the four elements. The panning automation was achieved using a crude one-to-one mapping from the output of each of the phrase choice data streams, and automated mix between the four players made use of the same 4-channel mixing device used in *4-buff-pvoc.maxpat* (see Section 5.2.1).

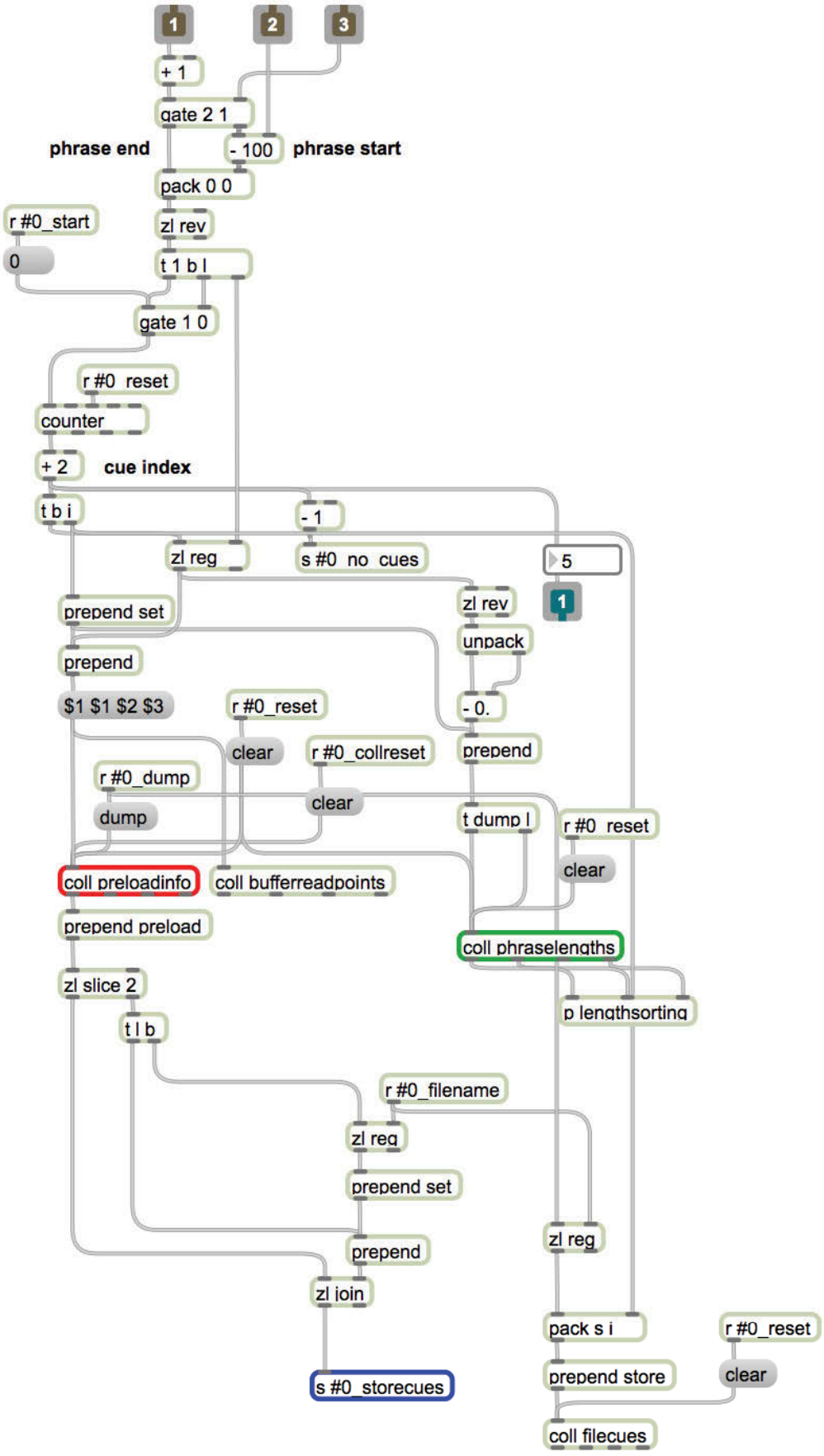


Figure 25: *storecues* subpatcher from *phraseplayer-GUI.maxpat*

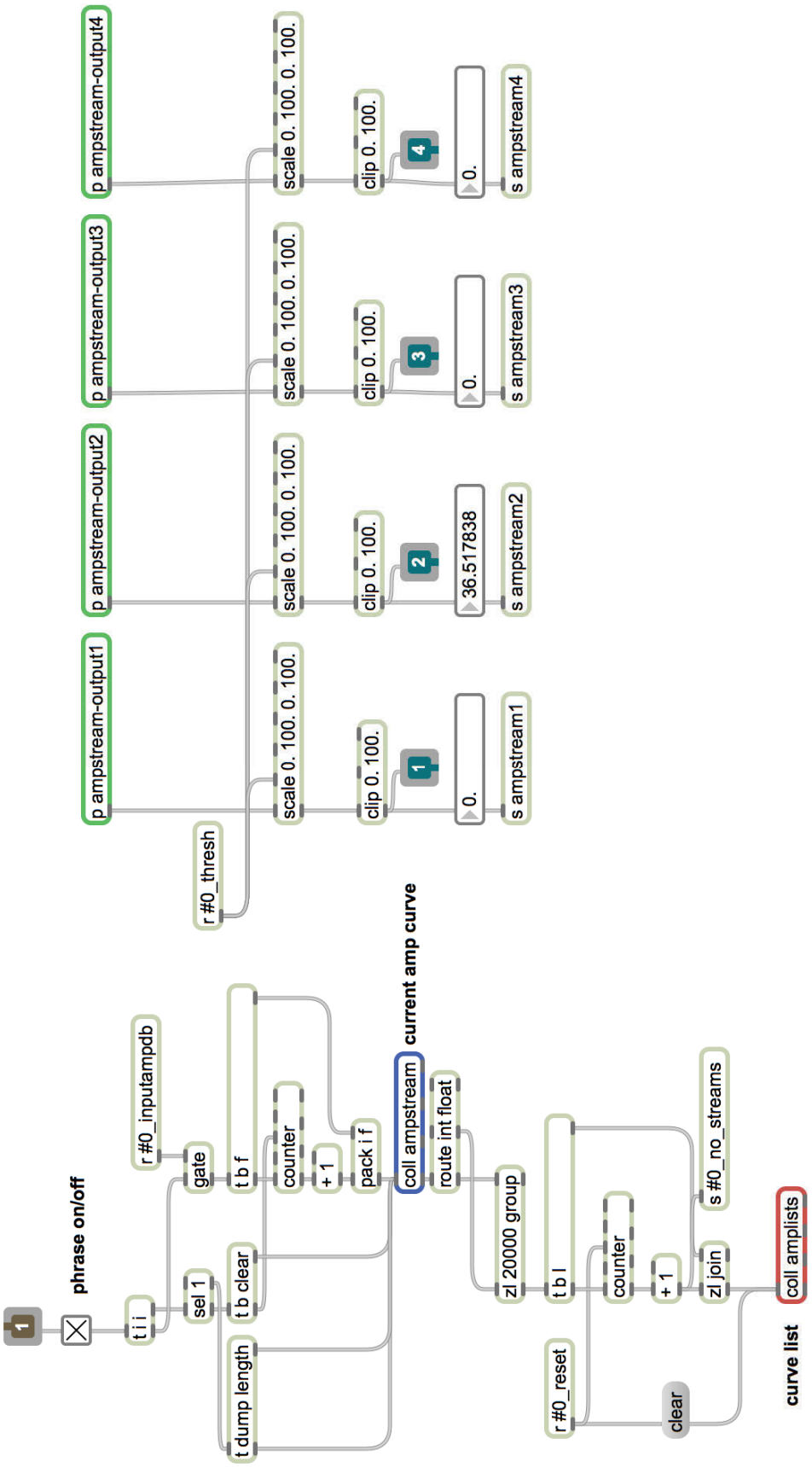


Figure 26: ***amplists*** **subpatcher from** ***phrase-player-GUI.maxpat***

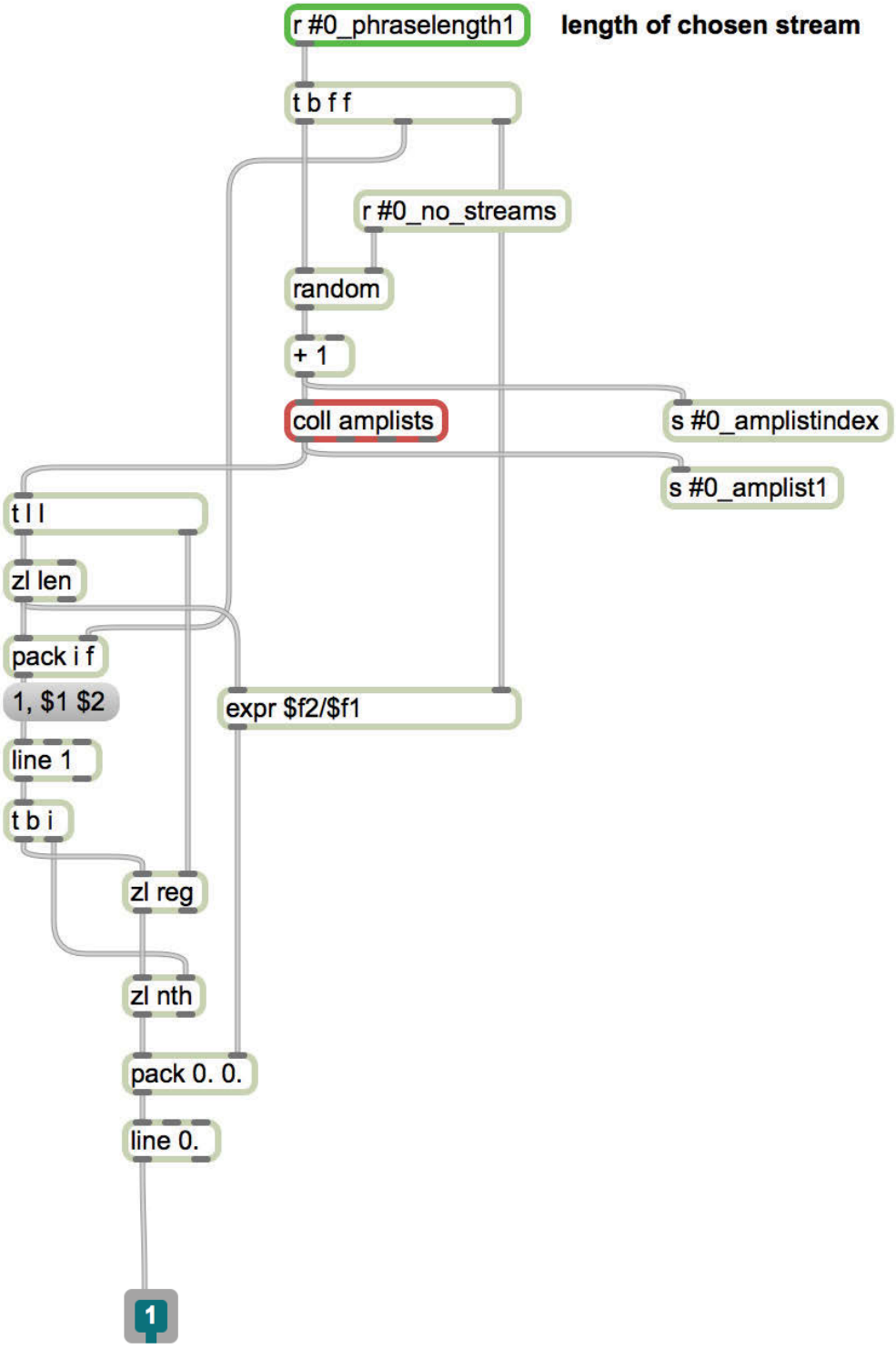


Figure 27: ***ampstream-output1*** **subpatcher inside the** ***amplists*** **subpatcher**

By focusing upon efficient methods of storage and recall, the *phrase player* system opened up new avenues of creative speculation in my work. The approach pursued in this system was guided by dissatisfaction with deterministic sampling techniques, as well as the desire to further enhance the autonomous capabilities of my previous interactive designs. However, besides the decision to change the sampling approach in my work, the coherent use of these sampled materials had yet to be considered in any depth. *Phrase player's* design was entirely contingent upon decisions made in relation to the sampling of live material, not planned as an integrated part of the initial design process. The generative structuring techniques implemented in this system, although capable of emergent behaviours, remained a somewhat arbitrary solution to the structuring of live sampled musical materials. The use of previously sampled data curves using *dataatintervals.maxpat*, whilst efficient and unpredictable, remained convenient rather than genuinely novel as a generative method. This was due to both the conceptual separation of these streams from the present interactive context, as well as their fixed and deterministic nature. In addition, the playback and processing methods explored were not as complex sonically or algorithmically as previously explored synthesis and processing approaches. The direct one-to-one mapping of delay line modulation remained too rigid and simplistic to be considered for further development, an approach that was also triggered by a rather arbitrary aleatoric process.

Given the fully automated nature of the generative experiments discussed above, this system was yet to achieve the same kind of balance between influence and autonomy that had been previously achieved in the constrained Markov approach taken in *Tripartite Markovia,* nor the sonic and interactive nuances of *Live-processing-1*. In the former, the generative patterning mechanism of the Markov chain process enabled musically relevant extrapolations from sampled MIDI data. In addition, the syncronisation constraints also ensured these materials were imbued with a direct relationship with the current context provided by the improviser. In the latter, although *4-buff-pvoc* was driven by a simplistic form of live sampling, its larger range of sonic transformations and combination with *pitch models* made *Live-processing-1* a superior holistic environment. However, the automated sampling and storage mechanism enabled me to speculate upon further avenues for making use of this captured material for the computer's output.

5.4 Reflections

Throughout this second period of development I solidified an approach to live sampling in my electro-acoustic performance practice. The creative trajectory I have outlined displays an exploratory form of development, with a strong focus on developing independent synthesis and processing modules designed for interactive performance. Through a *bricolage* programming approach, my creative practice engaged deeply with select synthesis and processing methods, namely *4-buff-pvoc* and *pitch models* (see Sections 5.2.1 and 5.2.2). Through iterative development and performative testing, this developmental approach uncovered inherent challenges and opportunities in using a live instrumental input as the primary means of sonic and algorithmic derivation in interactive performance systems.

5.4.1 *Co-evolving systems with practices*

By focusing upon automating the internal dynamics of *4-buff-pvoc* and *pitch models,* I was able to evolve the musical character of the individual components before their incorporation into a larger system architecture. This bottom-up approach was highly dependent upon the feedback of these two synthesis and processing modules in performance, making performative testing an invaluable creative space for both modules. Equally, this space began to influence my conceptions of interactive performance and subsequently my own performance practice. By interacting with the musical potential these two evolving modules I began to learn what I wanted from an interactive system of this type as a performer. The creation of *Live-processing-1* was an attempt at discovering a performance practice through the intersection of two independently designed musical systems. Whilst the two systems were evolved separately, they were also left open enough to enable flexibility in their integration with each other in a larger system.

However, whilst *Live-processing-1* pointed towards a new form of interactive performance in my practice, it also highlighted the limitations of some of the 'placeholder' triggering approaches implemented in each module. Amplitude threshold-based triggering appeared an arbitrary connection between a musical performer and the nuanced generation provided by both *4-buff-pvoc* and *pitch models.* In addition, the approach to live sampling used in *4-buff-pvoc* immediately revealed itself as inflexible

alongside the more sophisticated continuous data sampling approach of *pitch models*. This final aspect led towards the creation of the *phrase player* system discussed above, enabling the efficient recall of indexed *phrases* segmented and stored on disk. Given that *phrase player* was designed to address the sampling issues encountered previously, I had not yet however made meaningful use of the recorded segments, leaning once more to tried and tested methods of triggering individual phrases.

These issues became a matter of perspective and priority in the development process. By focusing my programming efforts on the internal dynamism of each synthesis and processing module, the global organisation of an integrated interactive system became a challenging task. The shift towards a planned, methodical approach to live sampling following *Live-processing-1* not only addressed the immediate creative needs of my software, it also signalled a shift in the type of programming engaged in throughout my work. Following a period of considered focus upon the details of individual working modules, a designed, *problem-solving* approach was needed. Finding a balance between a *bricolage* approach to building such systems, such a designed approach would later surface in my practice in the early stages of the creation of the *_derivations' phrase matching* algorithm, as discussed in the following chapter.

5.4.2 *Connections between data and generativity*

The placeholder approach taken in my practice was also beginning to reveal more fundamental issues with sampling-led generativity in human-machine performance. By focusing upon segmenting and storing continuously sampled data, the question arose as to how an interactive system might organise sampled material in a meaningful fashion. Until this point in my creative trajectory, I had not taken pause to ask this question of my developing practice. In the *Tripartite Markovia* system, continually sampled MIDI events were automatically organised by virtue of the Markov generation process (see Section 4.2.4.2). In previous data sampling experiments, sampled data curves were mapped directly to synthesis parameters, or used to create sampling-led, aleatoric control data (see 4.2.3, 5.2.2.3 and 5.3.2). However, with such a wealth of audio data arising from *phrase player's* continuous sampling approach, a more sophisticated form of organisation was required. This emerging theme in my practice would become the next challenge of my programming practice.

5.5 Conclusion

Throughout this second period of my creative trajectory I solidified a number of techniques that would become part of my ongoing practice in the *_derivations* system. By focusing my programming endeavours on individual modules, I prioritised the internal dynamics of specific, automated modules over a global approach to designing for interactivity. During this period my creative practice evolved from initially exploratory practices, to a more focused forms of creative development. This process allowed me to iteratively refine both my specific creative designs, and my desired mode of performance with these modules. Hence, the creative practice itself was part of a *dance of agencies* between human and material, revealing the *mangling* inherent in such a performer-developer context.

Chapter 6. Wayfinding – Part 3: *_derivations*

6.1 Introduction

The present chapter traces the development of the *_derivations* interactive performance system through a series of advancements in my programming practice. Throughout the previous two chapters I have detailed incremental advances in my creative work, and my developing conceptions of interactivity and generativity in improvised human-machine performance. In Chapter 4 a series of idiosyncratic, event-based approaches coalesced to create the *Tripartite Markovia* system, a Markov-chain based system based with multiple players interacting with each other in a self-referential feedback loop. With *Tripartite Markovia* I had the advantage of focusing solely upon the generative design, given the MIDI paradigm the system was working in. As a result, this system was complex and detailed in its generativity, yet lacking without a specific sonic identity.

Throughout Chapter 5 I described an alternative development approach, focused upon the creation of specific synthesis and processing modules from which larger interactive systems could be fabricated. The interactive systems *Live-processing-1* and *phrase player* provided fertile ground for the creation of an integrated interactive environment using live-sampling techniques. The *bricolage* approach initiated by *Live-processing-1* remained appealing given the intuitive means by which such complexity could be achieved. However, the issue encountered with the limited sampling history available to *4-buff-pvoc* needed addressing using a planned approach to managing sampled material. The *phrase player* system, whilst not of immediate generative interest, provided a useful technique for segmenting and storing continuously sampled data from an audio stream. Therefore, the desire to combine the methodical, cumulative sampling of *phrase player* with the complexity of sonic generativity of *Live-processing-1* eventuated in the first incarnation of a new hybrid system: the *_derivations* interactive performance system.

The name *_derivations* was chosen to reflect the system's goal of deriving its sonic vocabulary directly from the history of the live performer. Whilst the previously described systems made partial use of this history, it was only through the combination of systematic sampling and storage with processing that this began to take shape as the core generative method in my creative work. As alluded to at the end of the previous

chapter, the sampling and storage capabilities of both *Live-sampling-1* and *phrase player* lacked a coherent approach towards the organisation of continuously live-sampled materials. In the *_derivations* system, I sought to better organise the musical material sampled throughout an improvisation, and also to make sophisticated use the sonic content of this material to drive the system's generative capabilities. As described in Section 6.5 below, this approach led to the design of an approach to a form of content-based music information retrieval (Casey et al. 2008) that I define here as *phrase matching*. The *phrase matching* approach used spectral analyses of live-sampled material to drive a self-referential generative system, allowing the content of previously performed material to influence the future direction of the system's contribution to an improvisation.

The *_derivations* system therefore integrated a number of strands in my creative practice into a single interactive system. Reflecting upon past approaches through my development trajectory, this system emerged both as a response to the initially identified issues, concerns and interests of this research (see Section 1.4), and as a result of an iterative process of trial and error, performative testing and reflection-*in*-action that formed the core of my creative practice. By bolstering the autonomous capabilities of my sampling-led approaches using *phrase matching,* this system maintained the physical performative freedom of a human improviser, whilst relinquishing direct control over the organisation of musical materials to a sophisticated, algorithmic process. As will become clear throughout this chapter, this approach to organisation and self-generativity allowed an instrumentalist to interact with coherently organised material from their previous performance(s). Thus, an explorative and 'hands-free' approach to live-sampling was created that suggested novel avenues for performative exploration in an improvised human-machine performance. In addition, integrating the linked synthesis and processing modules described in Chapter 5 with this generative approach achieved a variety of timbral extensions to material captured from the improviser.

Throughout the remainder of this chapter I chronicle the iterative development of *_derivations* towards its stabilisation as a useable performance artefact.[35] Major technical developments are outlined alongside reflections upon the emergent trajectory of my

[35] The three most recent distributions of the *_derivations* system are detailed in Appendix A, and have been supplied in the submission materials alongside this thesis. Video documentation of the software is also detailed in Appendix E.

programming and performance practice. The chapter concludes with reflections upon the use of *_derivations* in performance with reference to audio and video recordings supplied in the submission materials.

6.2 The phrase database

Building upon the templates provided by both *Live-processing-1* and *phrase player*, a central database was developed to manage the segmentation and storage of live-sampled materials from a live input signal. Given the complexity and novelty provided by the processing algorithms explored in *Live-processing-1*, this database sought to harness the entire history of a musician's performance in a systemised fashion, to be accessed for re-generation and processing by linked synthesis and processing modules. This central database – named the *phrase database* – was an abstraction residing in the newly titled *_derivations* interactive system. In contrast to the approach trialed in the *phrase player* system, the *phrase database* did not contain references to segmented phrases written as individual files to disk, but instead to specific phrase points within one single audio buffer stored in memory. Discovering the original cue-point method overly cumbersome and prone to disk read errors, it was decided that all audio material captured from the live audio stream would be stored in a single audio buffer residing in memory. This required some slight modifications to the indexing approach implemented in the *phrase player* system. In the modified version, the algorithm accessed each phrase in the database with reference to its position in the buffer (in milliseconds), rather than by its filename on disk. When a segment of audio was chosen by the *phrase matching* algorithm for output, the algorithm simply consulted the time stamp information stored alongside the phrase in order to select the correct segment of audio to use for the system's output.[36]

[36] As discussed in Chapter 2, other systems make use of long buffers for such live sampling and retrieval work, with notable examples in the work of Assayag et al. (2006), Ciufo (2005) and Casey (2009).

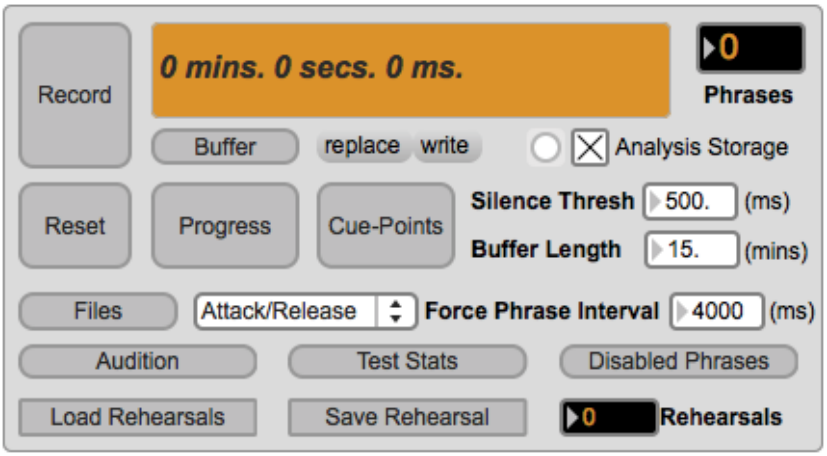


Figure 28: *Phrase Database* module within the *_derivations* system

Given the advancements of the previous two systems, implementing the live sampling and segmentation approach of this software was relatively trivial. From the commencement of an improvised encounter with the software, an audio buffer of a fixed user-defined length recorded the input stream of the instrumentalist. As the improviser continues to perform, the silence threshold mechanism indexed a growing list of phrase points in milliseconds in the central phrase database. The four phase vocoders contained within *4-buff-pvoc* could then use this data collection (stored in a central [coll] object) to point their internal audio buffers to sections of the central audio buffer containing the relevant phrase index. This central sampling process (relying upon a single internal audio buffer) solved a fundamental structural issue I had encountered when working with *Live-processing-1*. As described previously, the momentary and triggered sampling approach used for *4-buff-pvoc* limited the breadth of musical material accessible for processing by the four phase vocoder samplers. With this new approach, each PV could reference any segment of the continually expanding musical material stored in the central audio buffer. The internal buffer thus contained the entire history of the improviser, much the same as *pitch models'* approach to cumulatively building a central database of sinusoidal models. This flexibility completely changed the interactive and musical potential of my software system with respect to live sampled audio.

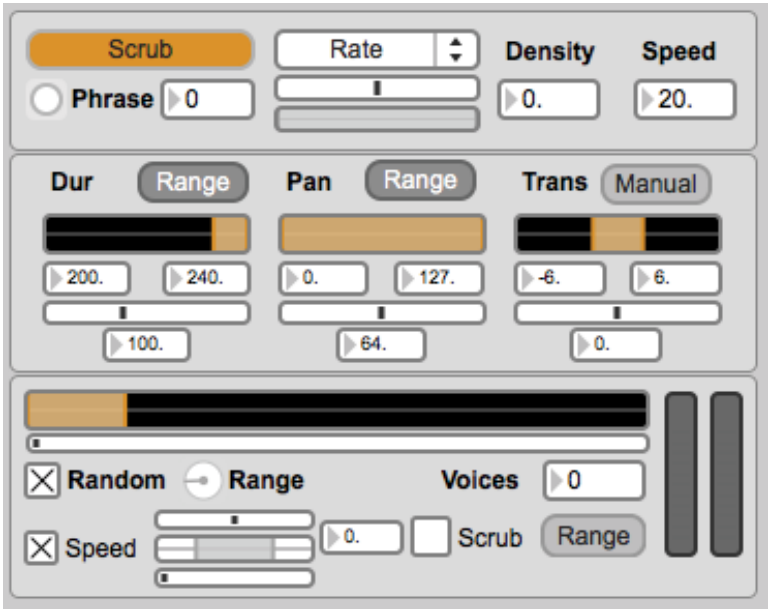


Figure 29: *Granulator* processing module included in the *_derivations* system

As the above process solidified as a core generative method, a third output module was added to further enhance the textural complexity of the *_derivations* system. A previously developed sound file granulator was adapted to be included as a final output module in the system, giving *_derivations* three separate and contrasting synthesis and processing modules with which to express a global sonic vocabulary (see Figure 29). As with the *4-buff-pvoc* module, the *granulator* accessed the central *phrase database* for its material, loading the relevant section of the central audio buffer into its internal buffer for use in processing. The output of the *granulator* was based upon an automated 'scrubbing' mechanism. A randomised control signal was used to dynamically automate both the scrubbing position within the audio buffer, and the density of the output grains. Once triggered for output, the *granulator* could heavily obscure the original source material, creating sonic gestures through the superimposition of randomised parameter curves. The user could preset the remaining parameters of the granulator such as stereo spread, grain duration and transposition in advance of a performance.

In addition to the addition of the *granulator* module, the original *pfft~* based phase vocoder algorithm implemented in *4-buff-pvoc* was upgraded due to my dissatisfaction with its efficiency and sound quality. As noted in Section 5.2.1, the STFT implemented using the *pfft~* object limited the use of the phase vocoder in real-time, live sampled

performance. This was due to the requirement that spectral data be recorded into the *pfft~* object in real-time before being made available for re-synthesis. In the context of the recently developed *phrase database,* this limitation made it difficult for *4-buff-pvoc* to dynamically access phrases from the database for re-synthesis. A new algorithm was subsequently implemented based on a Max-based phase vocoder described by Dudas and Lippe (2006, 2007). The advantage of Dudas and Lippe's PV was primarily in its capacity to load reference phrases from the database without the delay of real-time spectral recording. After testing this PV for some time, a higher quality solution was eventually sought from the SuperVP library of external objects released by the Institute de Recherche et Coordination Acoustique/Musique (IRCAM), available from the 'SuperVP for Max' collection from the 'Forumnet' subscription service (IRCAM 2015). In particular, the *supervp.scrub~* external had the added advantage of enabling the preservation of transients on time stretched material, allowing for a less 'smeared' sound quality from particularly percussive source material. Finally, *pitch models'* analysis object was also updated at this time. Having made use of *sigmund~* for sinusoidal decomposition, IRCAM's *iana~* external (from the 'Max Sound Box' collection) was implemented in its place to improve the efficiency with which sinusoidal lists were output and stored (Todoroff, Daubresse & Fineberg 1995).[37]

6.4 Phrase triggering and selection

Given the focus upon expanding the use of live-sampled audio materials, *4buff-pvoc* and *granulator* now required a mechanism for phrase selection in addition to output triggering. Whilst in *Live-processing-1* the internal buffers of each phase vocoder were filled with recently sampled material, *_derivations'* generative paradigm required a selection to be made from the central *phrase database.* This new method of accessing source material necessitated a form of phrase selection at the point of activation of each of the output modules accessing the *phrase database.* As with many of the advancements in my software systems throughout this project, such a major shift in the structure of a working system forced reconsideration of the means by which musical generativity was to be achieved in performance. However, as has been demonstrated in other projects, rather than initially

[37] These proprietary objects from IRCAM's Forumnet are only included in the 'SuperVP' distribution of *_derivations.* As outlined in Appendix A, both the 'standard' and 'standalone' distributions implement Dudas and Lippe's PV, and the *sigmund~* object.

dealing with such change through considered planning, I preferred enabling a direct performative experience with the changes in software through recourse to simple, tried and tested forms of generative design. By falling back on known methods of both triggering and automated phrase selection, I was able to discover what was required in the system through performative engagement with my developing software algorithms.

Given the added complexity of the *granulator* module, a balance was now sought in automating the output triggering of these three processing modules in the context of a live performance with an instrumentalist. In *Live-processing-1*, a simple time threshold triggering mechanism had been used to trigger the generation of both *4-buff-pvoc* and *pitch models* (see Sections 5.3.1.1 and 5.3.1.2). This direct triggering approach of *Live-processing-1* initially provided the basis for *_derivations'* output triggering, with the various time thresholds carefully manipulated so as to achieve a balanced polyphonic output between these three modules in performance. Using this method, three separate time thresholds filtered the incoming amplitude threshold crossings, feeding the triggers to the relevant output module.

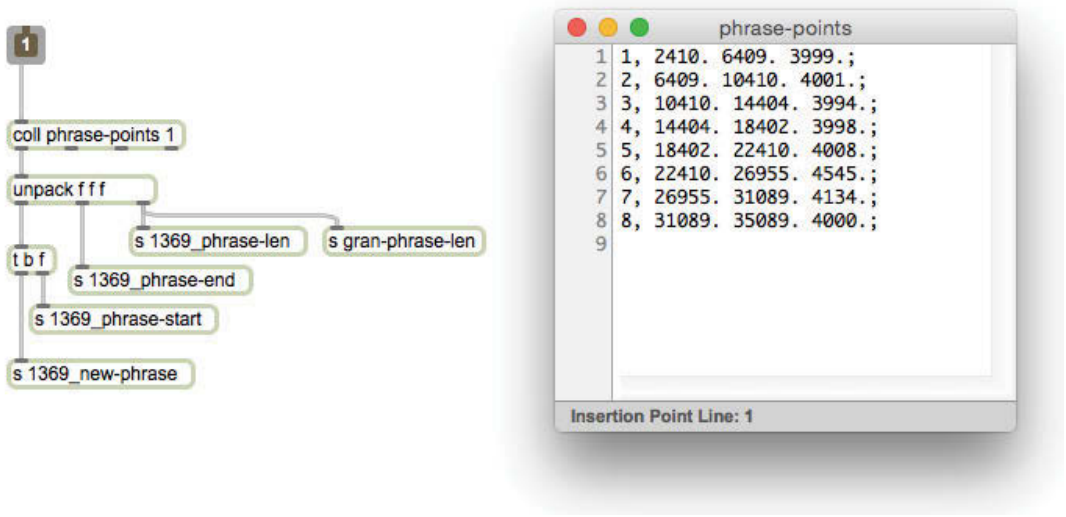


Figure 30: Parsing phrase point data in the *granulator*. The floating window displays the data contained within the [coll phrase-points 1] object. Each indexed list refers to the position of an individual *phrase* as it appears in the internal audio buffer. Values for each index take the following format: <*phrase start*><*phrase end*><*phrase length*>. The named *send* objects in the left of the figure send this data to the granulator before processing.

In order to choose material for processing/re-synthesis from the growing *phrase database,* these performer-controlled triggers were also appropriated to trigger the selection of a phrase for output from the database. Although an initially arbitrary approach to

achieving musical generativity, the first implementation of this process saw phrase selection delegated to a random selection algorithm akin to the approach implemented within *phrase player* (see Section 5.3.2). Upon the triggering of either a phase vocoder or the *granulator* for output, a random algorithm chose a phrase index from the database containing indexed phrase cues stored within the central audio buffer. Once chosen, these indexed portions of the central audio buffer could then be used as a reference for processing via granulation or phase vocoding (see Figure 30).

With respect to *pitch models*, whilst the original implementation of this module's triggering remained in tact, with the advent of the *phrase database* a fundamental change was made to the model selection algorithm to respond to indexes received from the *phrase database*. Up until this point, *pitch models'* re-synthesised output followed an aleatoric trajectory that was not bound by phrase boundaries analysed from the input. The challenge at the point was to connect *pitch models* with the segmented approach to phrase selection enforced by the central *phrase database*. Whilst this module did not make use of the recorded audio referred to by the *phrase database's* time stamp data, the data was instead used to group together model indexes associated with each phrase index in the database.

To make use of these phrase boundaries, a further data collection was created listing the range of model indexes for each time range. This additional data collection was then referenced when a phrase index was received from the *phrase matching* algorithm. To output models, the *random sweep* algorithm described in Section 5.2.2.3 was replaced with a new *one shot* mode that facilitated the sequential output of models within the range specified by the incoming phrase index. Once *pitch models* received a phrase index for output, the [coll thresh-phrases] data collection was consulted to retrieve the first and last model in the range. For output, the length of the phrase stored in the database was used as the basis for a timed control signal that swept across the range model indexes associated with that phrase index. Similarly to the *random sweep* algorithm, it was from this control signal that *pitch models'* IOI lookup table would then choose models for output.

With the addition of the *phrase database,* a cumulative approach to musical generativity was now being pursued consistently throughout the *_derivations* system. The sampling-led approaches employed throughout previous projects had coalesced into an integrated

system whose sonic vocabulary was derived entirely from data and audio captured from a live musician. Yet, whilst this performance system was flexible due to this cumulative approach, a challenge remained as to how the system might make nuanced and relevant use of the rich history of sonic material. Whilst the above approach provided an expansive array of materials for use in an improvisation, their triggering and selection remained somewhat arbitrary due to the choice of simplistic output and selection methods. As with the approach implemented in the *phrase player* system, selecting segmented phrases using a random algorithm was not designed to be a long-term solution for musical interactivity in my work. In addition, the means by which the three output modules were triggered during performance was also problematic, due to the overly reactive relationship this created between human performer and musical system. These two issues of musical interactivity and generativity were now clearly highlighted as areas of concern as the *_derivations* system began to stabilise as an integrated interactive performance system.

6.5 Phrase matching in _derivations

Whilst the advances in sample storage and retrieval were of great benefit to *_derivations'* ability to organise material from an improviser's performance, the means by which this material was used in an improvisation remained somewhat arbitrary. At this point in my creative trajectory, my sampling-led approach generativity lacked a coherent, content-based approach to their use in performance. Without referencing the sonic *content* of the material being stored for later processing and re-synthesis, I decided that it would not be possible for my system to be genuinely interactive or autonomous in performance. By relinquishing control over electronic processes to a computer music system, I was hoping to engage the performer in a meaningful interactive dialogue with the computer. To achieve this aim, it was therefore decided that *_derivations* must display some form of awareness of the current and past sonic context of an improvisation. Given that the system's sonic vocabulary was derived entirely from an improviser's past sonic gestures, any such awareness must therefore express itself in a *relational* fashion. In other words, *_derivations* should be able to understand the performance of a human instrumentalist by making *comparisons* between what it hears, and what it has already stored in memory.

The following section details the incremental implementation of the analysis, storage and matching approaches developed to address the selection of sampled material for processing in the *_derivations* software. In addition to providing technical details about the specific machine listening and music information retrieval (MIR) methods used, this section illustrates how elements of technical problem solving became instrumental in solidifying some of the most fundamental artistic concerns of the developmental project. It was in the exploration of this area of music technology that I began to become acquainted with areas of computer music research that were initially beyond the scope of my technical expertise. As I continued to pursue the creation of my software artefact in an exploratory fashion, a narrowing focus upon spectral analysis as the basis for musical generativity forced a consideration of the potential pitfalls of 'getting it wrong' on a technical level in my practice, something I had yet to seriously encounter in my work at this time. Consequently, the developmental approach of the *_derivations* software shifted gears by approaching these tasks from a problem-solving and solutions-focused development perspective.

Whilst initial experiments with sampling and storage of live audio had begun what I have termed *sampling-led* approaches to musical generativity, the manner in which such a growing database of material was to be used in an interaction had not yet been sufficiently explored. As detailed in previous sections, at this stage I had managed to find perceptually meaningful methods of segmenting and storing musical phrases from a live input using silence threshholding methods. However, the choice of how to access and use this stored information was limited to the referencing of some basic meta-data of the stored data points. Such meta-data included the actual index numbers associated with each phrase, the total length of a phrase or the relative temporal position of the phrase in the database as a whole. In other words, although these systems demonstrated the possibilities of segmentation for building large databases of live sampled materials, the methods of generativity that such meta-data facilitated remained lacking. Given that this meta-data was devoid of any contextually significant information about the contents of each entry in the database, the generative capabilities of these early systems relied heavily upon aleatoric methods of selecting materials for output and processing.

In light of these challenges and limitations, the next step in my developmental trajectory was to investigate methods that used the content of segmented and stored data

as an integral part of the software's generative strategy. The method that I began to explore in my work at this time I define here as *phrase matching*, a technique that may be broadly described as a form of content-based music information retrieval, as defined by Casey et al. (2008).[38] *Phrase matching* is a content-based audio similarity search method that makes use of sound descriptor analyses streamed from a live audio signal to query a database of segmented and pre-analysed audio materials. These audio materials, or *phrases*, are segmented in real-time from the instrumentalist and time-stamped using the silence threshold algorithm discussed in Section 5.3.1.1.[39] In the various iterations of this technique that were implemented in the *_derivations* software, sound descriptor analyses of an improvising musician were used both to build this cumulative database – the *phrase database* – as well as to query the database in order to find similar, stored materials for playback, processing and re-synthesis.

6.5.1 *Multi-descriptor phrase matching using analyzer~*

Thus, the *phrase matching* approach began as a means towards enabling the contents of the data being stored in these databases to lead the musical generation process. As discussed in relation to the *pitch models* module however, I had already begun in this project using FFT-driven spectral analysis as a means towards integrating the sound of a computer music system with a collaborating instrumentalist (see Section 5.2.2 for details). However, the present *phrase matching* method required a higher-level approach to the use of sound descriptor analyses streamed from the live audio signal. Rather than collecting and storing streams of analysis data for re-synthesis, the purpose of the *phrase matching* algorithm was to build a statistical representation of each audio segment to be stored within the database. This statistical representation could then be used to make contextually significant decisions about which stored phrases were to be used for playback, processing or re-synthesis in a live performance with the system.

---

[38] In this survey paper on content-based music information retrieval, Casey et al. refer to real-time, performative efforts in this area collectively as 'soundspotting'. However, there are some fundamental differences in the present approach than those outlined in this paper and others by Casey. For more detail on Casey's own *soundspotting* system, the reader is referred to Section 2.3.3.2 of the Literature Review.

[39] This segmentation approach is also discussed in Carey (2012).

In early iterations of the *phrase matching* algorithm, the system response in *_derivations* relied upon referencing *phrase vectors* that contained statistical representations of four sound descriptors captured throughout the duration of each phrase. These *phrase vectors,* grouped and analysed after the conclusion of each phrase, were indexed alongside each phrase contained within the database, creating a perceptually significant extension to the existing meta-data contained within the original database. The vectors would then be used to select phrases stored in the phrase database that best matched the most recently analysed phrase performed by the instrumentalist. This process is described below.

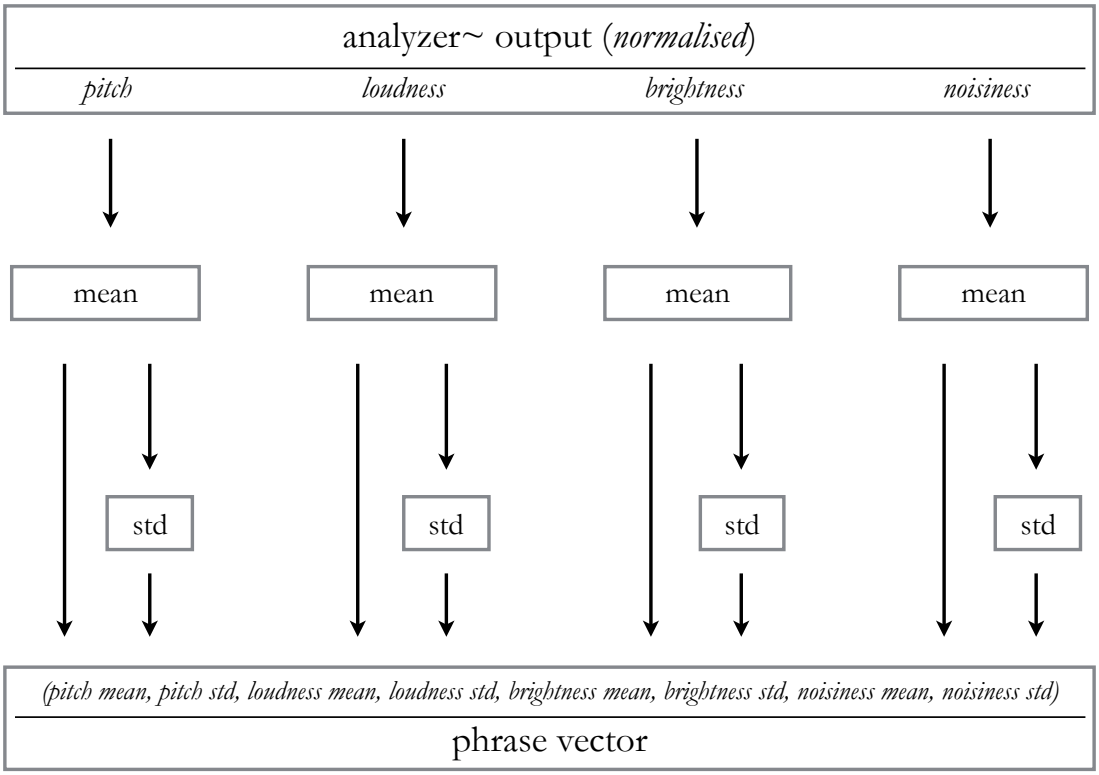


Figure 31: Building a phrase vector

Making use of Tristan Jehan's *analyzer~* external object for Max (Jehan & Schoner 2001), the *phrase matching* algorithm collected the continuous output of four sound descriptors from the live signal throughout the length of each phrase. The descriptors streamed from *analyzer~* were *pitch*, *loudness*, *brightness* (spectral centroid) and *noisiness* (a measure of spectral inharmonicity). Upon the completion of a phrase, the system computed both the mean and standard deviation of the accumulated data for each descriptor, storing the resultant eight statistical values in the database as a *phrase vector* indexed to each

individual phrase (see Figure 31). In order to find the closest matching phrase contained within the phrase database, the matching algorithm compared the most recently formed phrase vector against the database accumulated from all previously stored vectors. This process was executed once the statistics had been grouped together at a phrase boundary, immediately prior to being stored in the phrase database as a new phrase vector. This querying process was not, however, completed on the vector as a whole, but by separating out the statistics stored for each descriptor so that they may by queried individually against the accumulated databases of each of the separate descriptors.

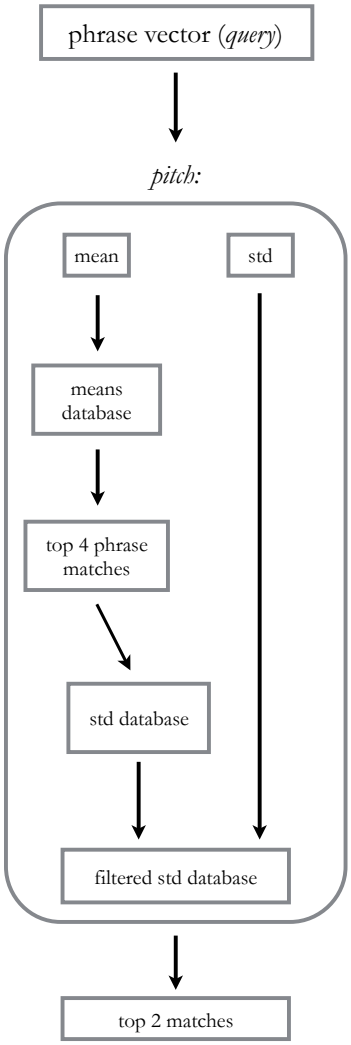


Figure 32: Querying the phrase database – *pitch* comparison

This querying process can be defined as follows: the mean value of each sound descriptor in the current vector is compared against the database of all of the means

calculated for that descriptor (the mean database). To do so, the algorithm takes the Euclidean distance between the mean of the input query and the means contained in the database, ranking the values returned from the database from smallest to largest distance from the input query. The algorithm then returns the four closest matching phrase indexes to this input. These phrase indexes were then used to filter the database of stored standard deviation values for each descriptor, and a further Euclidean distance was computed between the standard deviation value of the input phrase and these four phrase indexes (the filtered standard deviation database). This process was executed simultaneously across all four sound descriptors, returning a total of eight phrase indexes as potential matches to an input phrase (two per descriptor). This process is illustrated for the *pitch* descriptor in Figure 32.

Given that this matching approach dealt with multiple sound descriptors as well as inconsistent overall phrase lengths (i.e. analysis windows), a choice was made to focus only upon these basic statistics of each descriptor (mean and standard deviation). By surveying first the mean and subsequently the standard deviations values across the individual descriptor databases, the algorithm attempted to determine, in steps of increasing specificity, the relationship between the input representation and the statistical values stored in the *phrase database*. By itself, the mean value was not considered satisfactory as a similarity measure for descriptor comparisons, given that the temporal nature of the analysed descriptor data would not be contained within such an analysis. By continuing the similarity search by computing the standard deviation value as a second step in the process, the algorithm was able to compare the relationship between the input statistics and the phrase database, by focusing on the degree of variance between the input and the database of descriptor data.

6.5.2 *The common match algorithm*

Once the above-described two-step similarity search was complete, the final step in the matching process, defined as the *common match* algorithm, sought to find common matches amongst the top ranking phrase indexes returned by the four descriptor similarity searches. In this process, the individual matches returned from each descriptor are compared against every other descriptor in order to find like matches amongst the eight total phrase indexes returned, as illustrated in Figure 33. Following the individual

querying of descriptor databases outlined above, the returned phrase indexes were tallied in order to find agreement between descriptors, with the highest number of occurrences of a phrase index chosen for selection. In cases where descriptor matches were tied between two agreeing phrase index matches, such ties were broken at random. By contrast, where no common indexes were found amongst the descriptor matches, a random choice was made between the between top ranking indexes chosen by the search algorithms.

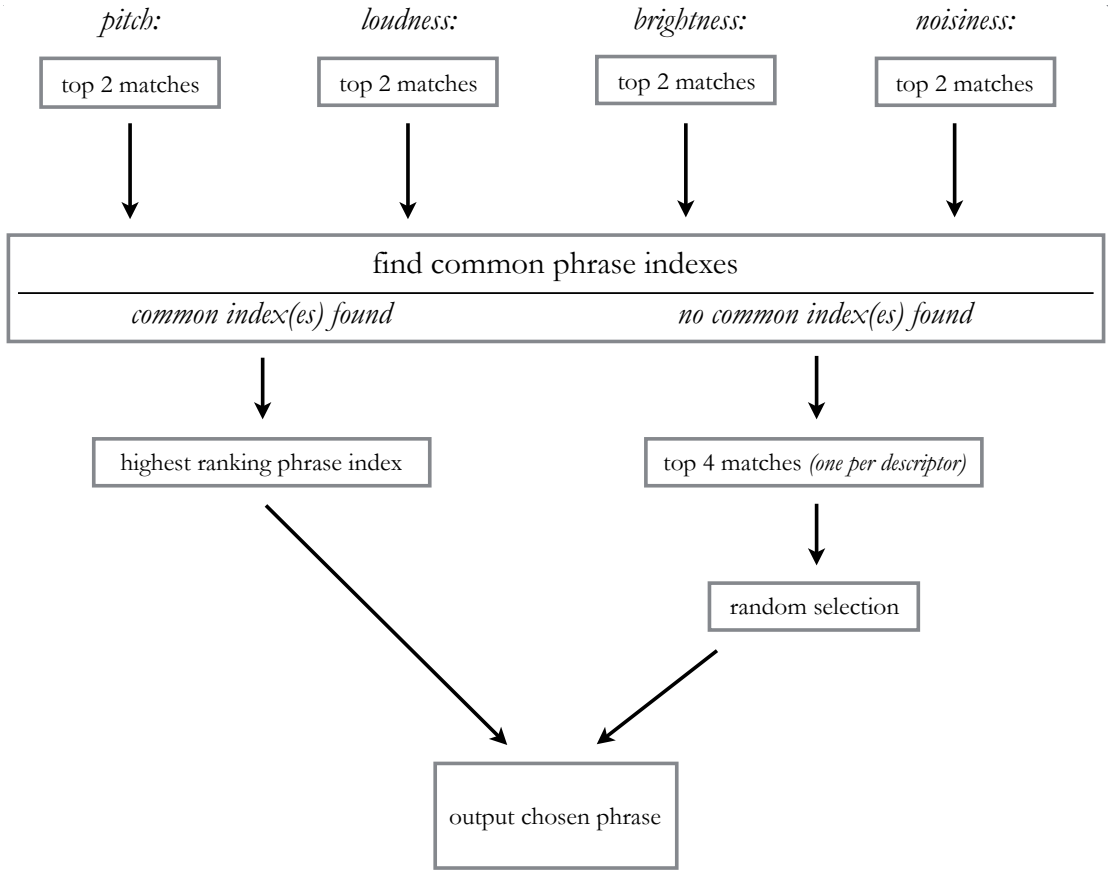


Figure 33: Comparing descriptor matches – the *common match* algorithm

6.5.3 *Limitations of the multi-descriptor and common match approaches*

At the time of development, the *common match* algorithm represented a solution to the discretisation inherent in building a multi-dimensional representation of recorded audio segments from a live signal. This approach privileged agreement between the four descriptors in order to find the closest match to an input phrase. In addition, a choice

was made to compute descriptor comparisons separately, as opposed to as a complete feature vector of the input. As illustrated above, the multi-descriptor approach treated the mean and standard deviations of each descriptor individually before computing a matching phrase. Comparisons between matched phrases were therefore made only after each descriptor had queried its own area of the phrase database. Although each of the descriptors queried the database in parallel, the comparison of both mean and standard deviations with the database therefore occurred in series, with each descriptor querying the mean and subsequently the standard deviation database.

Whilst proving useful in providing a basic matching scheme for use in the *_derivations* software, the above approach did prove overly cumbersome, as it required multiple steps of analysis and comparison, as well as prioritising agreement between descriptors that in practice may occasionally not share any matching phrase indexes. As specified by the *common match* algorithm, it could often be a random decision that determined which descriptor's top ranking phrase index would be chosen for output. This in turn provided a limitation to the algorithm that would sometimes result in the output of perceptually unexpected results. Given that the ultimate goal of this analysis and matching scheme was to return perceptually significant matches to an input phrase, the separation between descriptors in the matching process did not always prove consistent from one input phrase to the next. This was especially true for situations where an instrumentalist either a) played sonic materials that did not have a great deal of precedent stored in the database or b) improvised with a multi-source database, where there is an inherent variety of materials present in its design (more information on multi-source databases see Section 6.8).

By way of example, let us consider an input phrase to this algorithm that returns no common phrase indexes across the statistics databases queried for each descriptor. In such a case, although the statistics databases for each descriptor would return their top two closest phrase indexes, none of the four descriptors share matching indexes. In this example, the algorithm randomly chooses between one of the top four matching phrase indexes to output. According to the above-described matching algorithm, there would be an equal chance that the closest matches returned by either *pitch, loudness, brightness* or *noisiness* could be chosen for output. Although the algorithm would faithfully choose a phrase index for output that closely matches the input phrase query, the choice of

descriptor may not accurately reflect the most *perceptually relevant feature* of the input to the improviser. In addition, this random prioritisation of one descriptor over another also made no account of any history of previous decisions made by the algorithm in previous matches. In light of this, the choice to output a phrase index returned from the *loudness* statistics of a phrase might not be considered perceptually similar enough to a musician's recent performance history with the software. This would be particularly true if the musician was expecting matches that closely related to features of their sound other than the dynamic profile of their most recent performance (as expressed by the statistical representation provided by the *loudness* database).

Having tested the above-described approach extensively in rehearsal and performance, it was decided that the matching scheme suffered from problems of consistency. Often such inconsistencies manifested themselves in the system making perceptually inconsistent jumps between stored materials, when compared with the musical variety exhibited by a human improviser. Although these inconsistencies could not be technically classed as errors of the matching algorithm itself, their occurrences did conflict with the overall purpose of employing such a matching algorithm in an interactive music system. That is, if an algorithm for comparing the sound of a live improviser is to be implemented in such a system, it should be able to be relied upon to make perceptual sense in performance, and for it to be somewhat predictable and reliable. Of course, in an improvised performance any unpredictable output of this kind would often become the catalyst for new musical explorations. This is because it was very hard to tell the cause of a perceptual jump in sonic material by the algorithm in performance, and indeed in practice this knowledge would be completely irrelevant to the ongoing musical discourse. Instead of viewing these surprises as erroneous behaviour of the algorithm, they would often be viewed as digital provocations by the software that a musician might choose to react.

At this point however, it may be useful to reconsider my creative aims in the development of interactive performance systems, as outlined in Section 1.4. As with the majority of the approaches outlined in this research, *_derivations* was developed to provide playful exchange between a computer and a human improviser. By relinquishing control over the system to algorithmic processes, *_derivations* sought to encourage an exploratory approach from the improviser in performance, by ensuring that both musical coherence

and surprise was evident in the system's output. In particular, this system sought to display genuinely emergent properties from the recombination and processing of musical materials familiar to the human improviser (i.e. recorded segments of the musician's past performance(s)). However, given that the type of analysis and generation relied heavily on specific music information retrieval techniques, this analysis and matching component – the heart of the generative system – did need to prove as consistent as possible. As discussed above, if an analysis and matching algorithm is to be used as the core generational element in such an interactive system, reliability, repeatability and predictability of the algorithm would need to be maximised. Therefore, it became increasingly clear at this stage of the development process that the place for variety and unpredictability in a system of this type should be found elsewhere in the design of the system. The unpredictable and surprising parts of such systems should be included as intentionally programmed elements, not as a fortuitous byproduct of internal inconsistencies. It was therefore decided that the ability of such an algorithm to provide perceptually relevant matches to a live signal must rely upon the algorithm being consistent and predictable in nature.

To account for the inconsistencies outlined above, three extensions to the phrase matching algorithm were developed and tested in the rehearsal process that were intended to provide more reliable and predictable output from the matching process. Two of these extensions to the matching approach required user decision-making in rehearsal, whilst the other extended the algorithm to automatically choose the most perceptually significant descriptor from the input phrase. These extensions are described below, followed by a critique of their effectiveness.

6.5.4 *User-defined descriptor weighting*

The first approach that tested to improve matching consistency made use of a set of user-defined descriptor weights. This was included as an option in the software, allowing the user to choose between the weighting scheme and the original *common match* approach as outlined above. In the weighting scheme, four sliders allowed the user to weight the importance of each descriptor in matching the input to the phrase database. These sliders were mapped to weighting values in a probability table consulted by the matching algorithm before the output of a matched phrase (see Figure 34). Thus, instead of relying

upon descriptor matches reaching agreement (or picking a random match in the case of non-agreement), the weighting scheme enabled the user to define the likelihood that the matches returned by particular descriptors would be chosen for output.

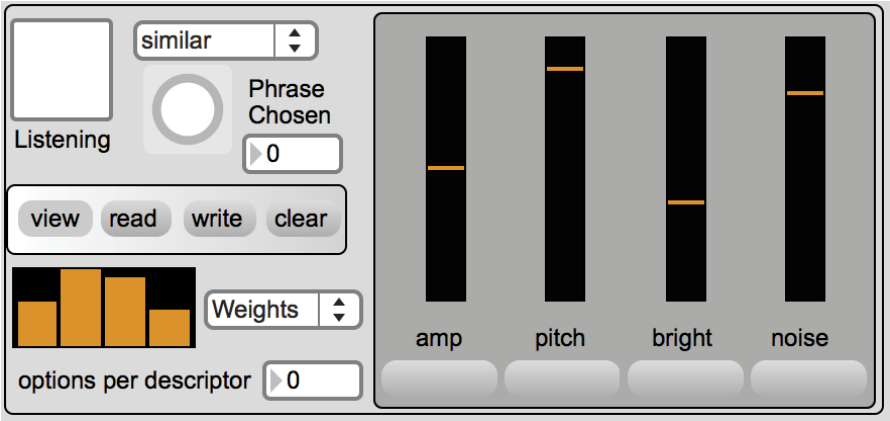


Figure 34: User-defined descriptor weighting

The rationale for this approach was that allowing a user to weight the importance given to various aspects of the internal analysis/matching of the software, the more likely the algorithm would return phrase indexes that matched their input according to their own criteria. In addition to this, an *options per descriptor* parameter was added that enabled the user to specify how many options each descriptor would return as possible matches. This parameter served to broaden the scope of the *common match* algorithm when searching for agreement between descriptors, allowing a larger pool of possible phrase indexes to be consulted by the *descriptor weights* algorithm.

6.5.5 *User-defined descriptor filtering*

Secondly, a more brute-force approach was developed that allowed users to choose a subset of descriptors they deemed relevant for perceptually significant matching of their input. This option required the user to decide upon which single descriptor, or combination of two descriptors, they would prefer the system prioritised from their input to obtain meaningful matches in their improvisation, to the exclusion of all other descriptor matches (see Figure 35). The choice of a single descriptor simply forced the algorithm to pick the top ranking phrase index returned by the chosen descriptor, whilst the choice of a subset of two descriptors caused the software to implement a stripped-down version of the *common match* algorithm, seeking agreement only on the matches

returned by the two chosen descriptors. As well as reducing the potential for abrupt changes in descriptor focus during matching, this approach also limited the matching process to at most half of the original similarity search in previous approaches.

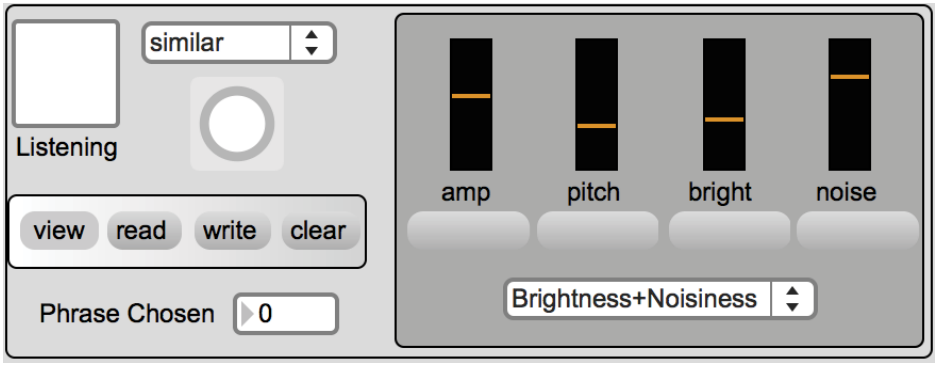


Figure 35: User-defined descriptor filtering

6.5.6 *Automatic similarity metric*

The final approach to improving consistency comprised of an algorithm that filtered the eight chosen phrase indexes according to an automatic similarity metric. Instead of relying upon the user to determine the most salient descriptor to be prioritised, this metric filtered the top four indexes and compared their standard deviation values in order to choose the most similar phrase for output, based on the smallest Euclidean distance. As the values contained within each of the descriptor databases were normalised between 0 and 1, directly comparing the standard deviation values calculated from the analysed data points was deemed a useful measure of similarity. As described above, the matches returned by each of the descriptor databases were calculated by choosing the smallest distance between the standard deviation value of the input and the top ranking phrase indexes. Given that these standard deviation values were calculated from the mean of all data points analysed during the length of a phrase, this algorithm required only a small amount of extra programming as it was the statistical values themselves that were being compared and not agreement amongst the various descriptors.

6.5.7 *Evaluating multi-descriptor phrase matching*

As is shown by the above extensions to the *phrase matching* algorithm, initial attempts at using content-based techniques as a generative strategy revealed some fundamental problems with using separate sound descriptors for this task. Although the individual matching algorithms of each descriptor worked consistently and predictably, there remained great difficulty in integrating this matching approach into a coherent scheme across the four descriptors chosen to represent the sonic content of each phrase. Making use of the *analyzer~* external as the basis for a multi-descriptor analysis approach, the multi-descriptor matching scheme relied heavily on agreement between descriptor matches. As critiqued above, unforeseen problems of consistency in the *common match* algorithm stemmed directly from the simplicity of this approach.

The first two extensions of the *phrase matching* algorithm attemped to address inconsistencies by giving the user more control over the outcome of the final decision made by the matching algorithm. What these extensions acknowledged was that the problems of consistency in the algorithm stemmed from the jump in descriptor focus that could occur if this multi-descriptor approach could not agree internally on the best matching phrase index. Both of these approaches sought to reduce the likelihood that the algorithm would make random decisions, instead calling upon the user to provide contextual information to the algorithm to aid in finding consistent matches during performance.

Comparing these two user-defined approaches in rehearsal and performance, it was found that the simpler, brute-force *descriptor filtering* approach improved the consistency of the matching algorithm, whilst the *descriptor weights* extension of the matching algorithm proved to be an ineffective means of improving consistency. In my own performances and rehearsals, the *descriptor filtering* approach suited my desired interaction style with the software. Given that I was primarily interested in engaging with *_derivations* using timbre as main analytical focus, when using this algorithm at this time I consistently opted for a combination of *brightness* and *noisiness* as the descriptors to be queried from my input (eliminating both pitch and loudness from the similarity search). By contrast, although enabling users to prioritise certain descriptors over others, the nature of the probability table used in the *descriptor weights* algorithm did nothing to

eliminate the possibility that inconsistent jumps between materials would occur. This is because such a probabilistic approach did not take into account transitions from one descriptor to another, hence leaving open the possibility of inconsistent matching decisions occurring (albeit with less frequency).

Finally, from my perspective as both a performer and developer of the software, a non-trivial disadvantage I saw in both of the above extensions was the increase in options provided to an improvising musician working with the software. Given the complexity of the software as a whole, I was also concerned to avoid over-complicating the software for potential users that may not be as interested in making such fine-grained decisions. The *_derivations* system, whilst providing improvising musicians with a musical environment that could be customisable to their musical inclinations, should also work 'out of the box'. That is, the software must be able to facilitate meaningful performative engagement without requiring users to delve any deeper than they deemed necessary. Giving a user choice over relatively low-level parameters risked being both superfluous as well as potentially confusing for the average user interacting with *_derivations*.

It was a desire to remedy this issue that prompted the creation of the *automatic similiarity metric* discussed above. In principle, enabling a more autonomous means of establishing consistency in a similarity algorithm was ideal when compared with the previously discussed heuristics. However, the discretised analysis method made the problem a difficult one to solve. In the *automatic similarity* metric, the normalised descriptor data streamed from the *analyzer~* external presented very levels of variation for each descriptor. Here the process of normalising mean and standard deviation values to ensure correct comparison relied upon making assumptions about the range of the various data streams in advance of their analysis. The problem was therefore one in which the process of normalisation itself came into question. The algorithm as it stood made assumptions about the consistency between the four descriptors, and it was decided that improving upon its effectiveness was not an ideal means of improving consistency in the matching algorithm.

As evident in the various attempts at taming the *multi-descriptor* matching approach, this process relied on a great deal of trial and error in both development and testing. Although the original *common match* algorithm did prove somewhat sufficient in enabling a

content-based approach to sampling-led generativity, the few inconsistencies encountered during repeat performances with the software led me to question to merits of the approach, and the methods by which such a core component of my software was developed. The 'code-and-fix' methodology employed in the *phrase matching* algorithm provided a detailed and exploratory approach to fine-tuning the core matching algorithm. However, given the complexity of the interlocking parts within the *_derivations* system, this approach to development proved inefficient at solving such a fundamental design problem. As the *_derivations* project progressed, it became clear that the conceptual interest in the matching approach required a stable and reliable algorithm, one that could remain hidden from the user and trusted to provide predictable results. The paradoxical situation uncovered through this process was that without a deterministic and predictable generative process underpinning the design, the layered aleatoric methods of processing and generation could not be relied upon to generate interest. *Phrase matching* therefore needed to be as foolproof as possible in order for the complexity to reveal itself as the driver of the system's novelty in performance.

The fundamental issue with the *common match* algorithm was the difficulty in using four separate descriptors to determine similarity. Given my inclination for timbral matching, in the period that followed I implemented a matching algorithm based on a single analytical measure of the musician's live performance, Mel Frequency Cepstral Coefficients (MFCCs). Compared with the *multi-descriptor* method of analysis in *_derivations*, the use of MFCCs proved a more consistent and streamlined way of recording perceptually significant information about a recorded audio segment. The details of this matching approach are outlined below.

6.5.8 *MFCCs in _derivations*

Employed heavily in the speech recognition community, Mel-frequency Cepstral Coefficients (MFCCs) have been shown to be useful for music information retrieval tasks (Casey et al. 2008; Logan 2000). The mel-frequency cepstrum is considered a 'spectrum of a spectrum', representing the power or magnitude spectrum of the spectrum obtained from an FFT. MFCC feature vectors are obtained by taking the logarithm of the amplitude spectrum of a windowed audio signal, spacing the resultant audio bins along the Mel frequency scale before decorrelating the values using the

discrete cosine transfer (DCT). The Mel scale is a perceptual scale of pitches that more accurately reflects the nonlinear frequency perception of human hearing. MFCCs are therefore more likely to result in perceptually significant representation of incoming audio for musical applications than an FFT representation. Figure 36 outlines the process undertaken to create vectors of MFCC features for use in the analysis of instrumental timbre.

In *_derivations,* MFCC analysis replaced the *multi-descriptor* analysis using the *analyzer~* object. MFCC feature vectors are extracted from the input audio signal using the *zsa.mfcc~* external object for Max, developed by Emmanuel Jourdan and Mikhail Malt and released as part of the *zsa.descriptors* library (Malt & Jourdan 2008, 2009).[40] Using the same segmentation method based upon amplitude thresholding, the segmented audio signal is analysed via an FFT with a window size of 1024 samples (approximately once every 23ms), which is then fed through the *zsa.mfcc~* object to obtain twelve cepstral coefficients for each analysis frame. This feature vector is streamed and collated throughout the duration of each phrase, and then subjected to further statistical analysis for use in phrase matching once a phrase boundary has been reached.

Upon reaching a phrase boundary, the collected data streamed for each of the twelve coefficients is analysed for mean and standard deviation over the duration of the analysed phrase (see Figure 37 below). The resultant values are collected into a 24-dimensional feature vector representing timbral content of the phrase, indexed by phrase number in the database. In contrast to the previous multi-descriptor approach, however, the MFCC phrase matching implementation used the entire feature vector for comparison with the phrase database. In order to calculate a match, the Euclidean distance was calculated between the input feature vector and the database of stored vectors. This process was achieved using the *zsa.dist* external object from the *zsa.descriptors* library. This method has proven more efficient, as is has not relied upon the agreement between the output of multiple features in determining a match, and the statistical data points are easily compared as single, multi-dimensional feature vectors.

[40] This library is accessible from the following URL: http://www.e--j.com/index.php/what-is-zsa-descriptors/

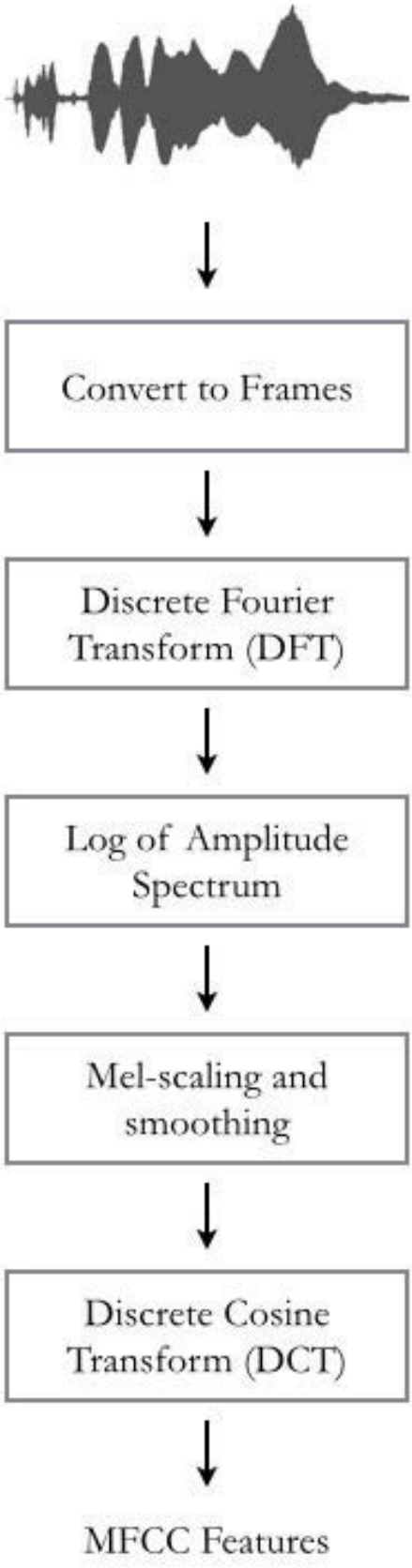


Figure 36: Creating MFCC features from an audio signal (Logan 2000)

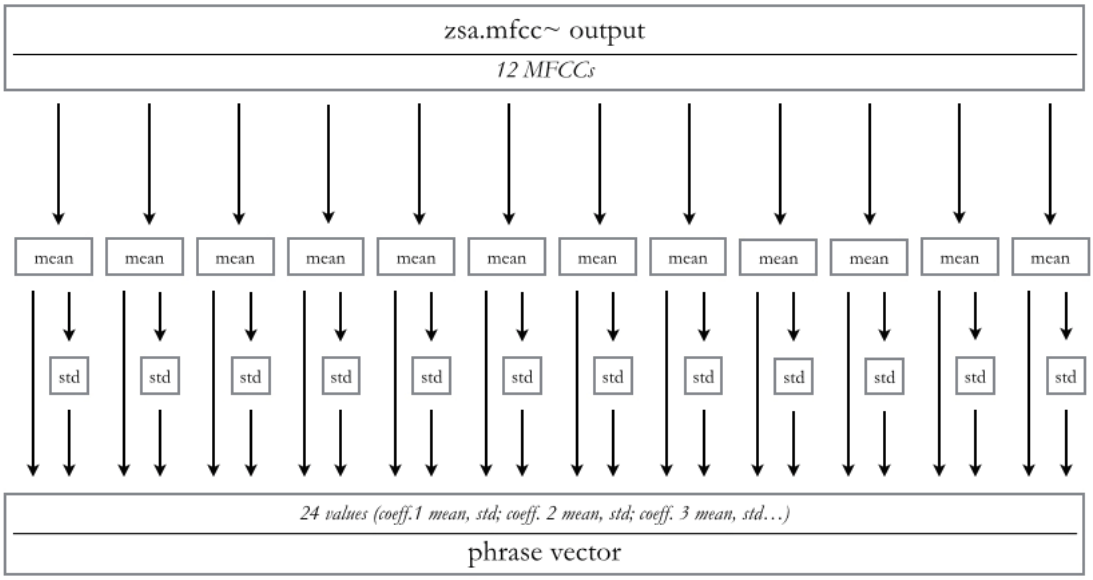


Figure 37: Collating MFCC phrase vector statistics

When tested using simulation performances, the MFCC matching approach produced perceptually strong matches from pre-analysed inputs. The *zsa.dist* external object enabled incoming phrase vectors to be compared amongst the growing list of vectors stored in its growing database. Given that the phrase vectors contained both mean and standard deviation for each of the twelve coefficients streamed through the length of each phrase, this technique enabled an efficient and simple and reliable matching method.

6.6 Self-referencing

Having developed, tested and refined the above-discussed *phrase matching* approach, this idiosyncratic form of content-based music information retrieval had become central to *_derivations'* internal dynamics. The trajectory that the software followed towards the final MFCC implementation, whilst clear in retrospect, could not have been predicted in advance. As has been discussed above, the initial choice to make use of *multi-descriptor* analysis for this task spawned a series of experiments aimed at refining the original matching technique. These experiments, on the whole, were designed to tame a largely inefficient and unreliable algorithm to achieve more consistent results. As has been previously discussed, given my focus on provoking surprise and unpredictability in the design of the system, a need for reliability and consistency in this generative algorithm

was not immediately revealed until such problems were encountered during performative testing of the software. As design criteria therefore, consistency and reliability in the matching process were revealed only as the software and my performance practice evolved.

Similarly, it was the evolution of the *phrase matching* approach and experience with this in performance that led to a revision of the overall architecture of triggering and control in the *_derivations* system. As discussed previously, when the various pre-existing modules were assembled to create *_derivations* as an integrated system, the approaches towards both module triggering and phrase selection were co-opted from previously implemented systems. Whilst the subsequent *phrase matching* approach had significantly changed the software's approach to phrase selection, the triggering of the three output modules during performance was in need of considerable re-design. The three output modules within *_derivations* (*4-buff-pvoc, pitch models* and the *granulator*) were still being triggered based upon amplitude threshold approaches (as discussed in Section 5.3.1.1). Amplitude threshold crossings from the live performer were filtered through a series of time thresholds, allowing for a degree of control of the relative density of the software's output. Whilst both phrase selection and the internal dynamics of each output module were automated, the live performer was still in direct control of the temporal output of the modules themselves.

Given the recent advances in automated phrase selection, I began to scrutinise this method of control in the *_derivations* system. In its current form, this threshold triggering represented a high-level form of control over both the sophisticated *phrase matching* and the aleatoric automation of each output module. As the improviser performed, the latest amplitude threshold crossing passed through the threshold gate would trigger both the matching process and the output of the module itself. In addition, due to the nature of the matching process, each output trigger received from the performer would trigger the index returned from the previous query to the database. This was a necessary restriction of the software, given that a phrase index could not be returned from the *phrase matching* algorithm until the end of a phrase boundary (see Figure 38 below). As a result of this triggering process, the performer was in control of triggering the response of the output modules via the amplitude threshold mechanism.

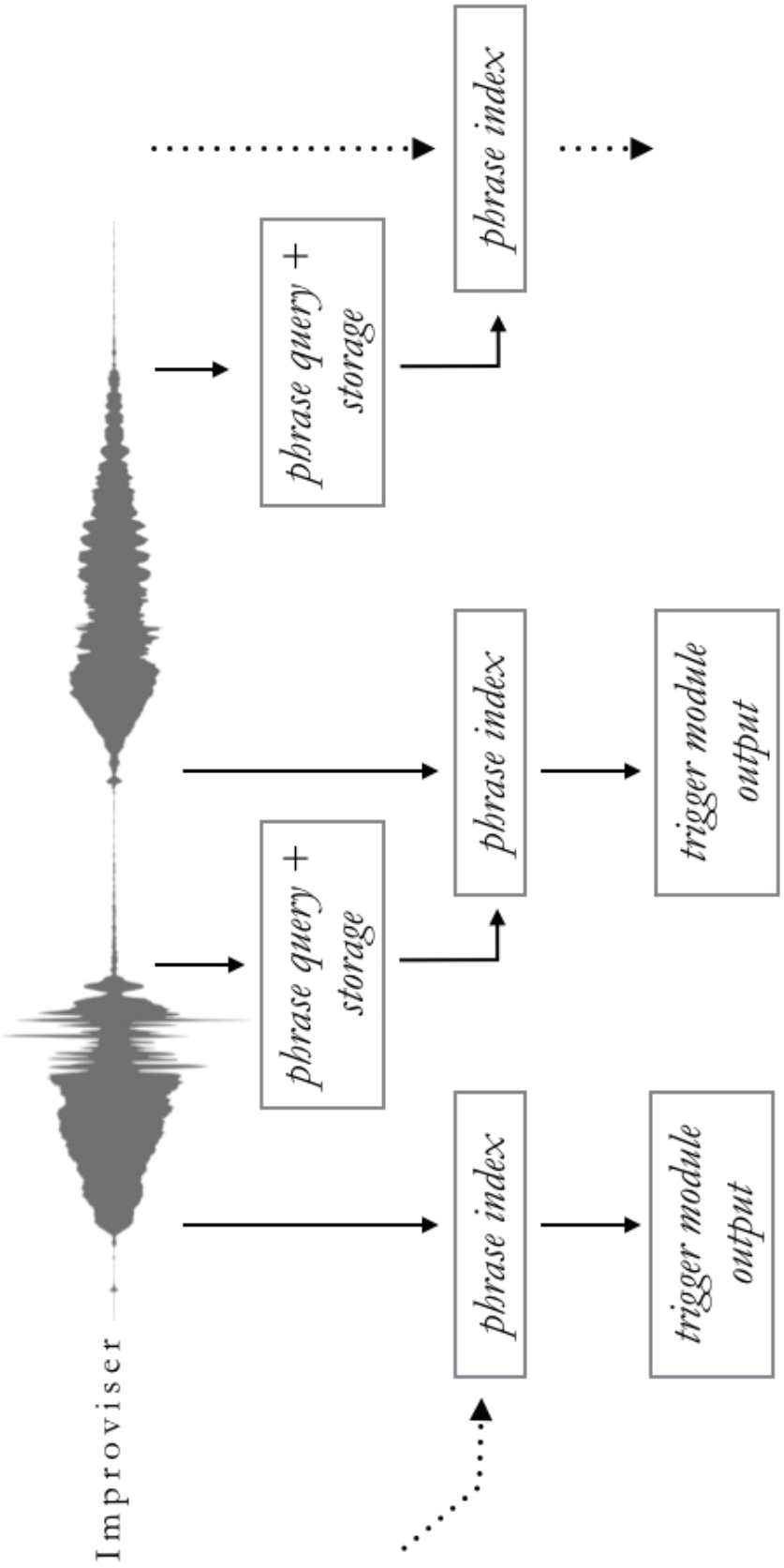


Figure 38: *_derivations'* original phrase storage and triggering logic. The beginning of a phrase boundary sends the previously matched phrase index to an available output module.

After performing for some time with this form of matching and triggering process in *_derivations*, it was decided that the link between performance gesture and automated output was not in keeping with the generative autonomy desired in a system of this type. The coupling of automated phrase analysis with triggering of the output modules no longer made sense in the context of the above-described *phrase matching* algorithm. The reactive nature of this triggering mechanism engendered an ambiguous form of control over the output of the software. That is, although a musician triggered *_derivations'* output directly through amplitude triggering, the precise phrase index chosen for output, the module chosen to process/re-synthesise this phrase and the specifics of the processing/re-synthesis were largely unpredictable to the performer. In other words, a deterministic trigger was being used to initiate an unpredictable process. Whilst this indeterminacy did form a large part of the software's performative agency, it was decided that such a mechanism should no longer be controlled by the performer, but must be automated globally within the software itself.

To achieve this aim, I sought to distance *_derivations'* output modules from the amplitude triggers necessary for the software's analysis. This approach saw a fundamental shift in the way in which *_derivations'* output modules were conceptualised, facilitating the creation of an autonomous dynamical system that controlled the software's overall generative output. Until this time, *_derivations'* three output modules (*4buff-pvoc*, *pitch models* and the *granulator)* had been treated separately with respect to their triggering and internal control. Besides the sonic coherence brought about by the *phrase matching* algorithm, to this point the modules were solely contingent upon the continued performance of the human improviser for their sonic material and output triggering. If the performer ceased playing, the system quickly came to rest given the lack of amplitude triggers received from the live signal. In this new approach, each of *_derivations'* output modules was considered an individual *player* that could have influence over the subsequent output of the system itself. By splitting *4-buff-pvoc's* phase vocoders into four individual players, *_derivations* was now comprised of six separate players with the potential to influence the dynamics of the system as a whole: *Pvoc-1, Pvoc-2, Pvoc-3, Pvoc-4, granulator* and *pitch models*. This new approach was referred to as the *self-referencing* algorithm, and saw an expanded

role of the *phrase matching* algorithm in determining the sequential output of individual processing/re-synthesis modules.[41]

In the *self-referencing* algorithm, input from the live performer is injected into the system as 'seed' material from which the algorithm generates its internal dynamics. In contrast to the previous triggering approach, here each 'player' is triggered sequentially by the previous player to perform, after a given time threshold. These time thresholds are a function of the length of the current phrase chosen for output, allowing the system to maintain a coherent level of polyphony between the six output modules. To create a sense of variety in the output duration of each chosen phrase, the density of the output gestures is governed by a parameter modulating the output length of each phrase slowly within a user-specified range during an improvised session. *_derivations'* temporal output is therefore continuous and contingent upon the output of each individual player, not solely the live performer.

Most importantly, the *self-referencing* algorithm contains six individual instantiations of the *phrase matching* algorithm itself. Once a player receives a phrase index for output, it passes this index to the next available player to find a suitable match from the database, ready for subsequent triggering. The algorithm keeps track of which players are currently available for output, allowing for the continuous output of sonic materials from the *phrase database.* This process can be conceptualised as a digital form of the game *chinese whispers* (also known as *telephone*), whereby the previously matched phrase is queried against the remaining database to find the next closest match. This process is illustrated in Figures 39 and 40.

[41] The *self-referencing* module in the interface displays itself as a pop-up window accessible via the 'Triggering' tab in the main *_derivations* interface.

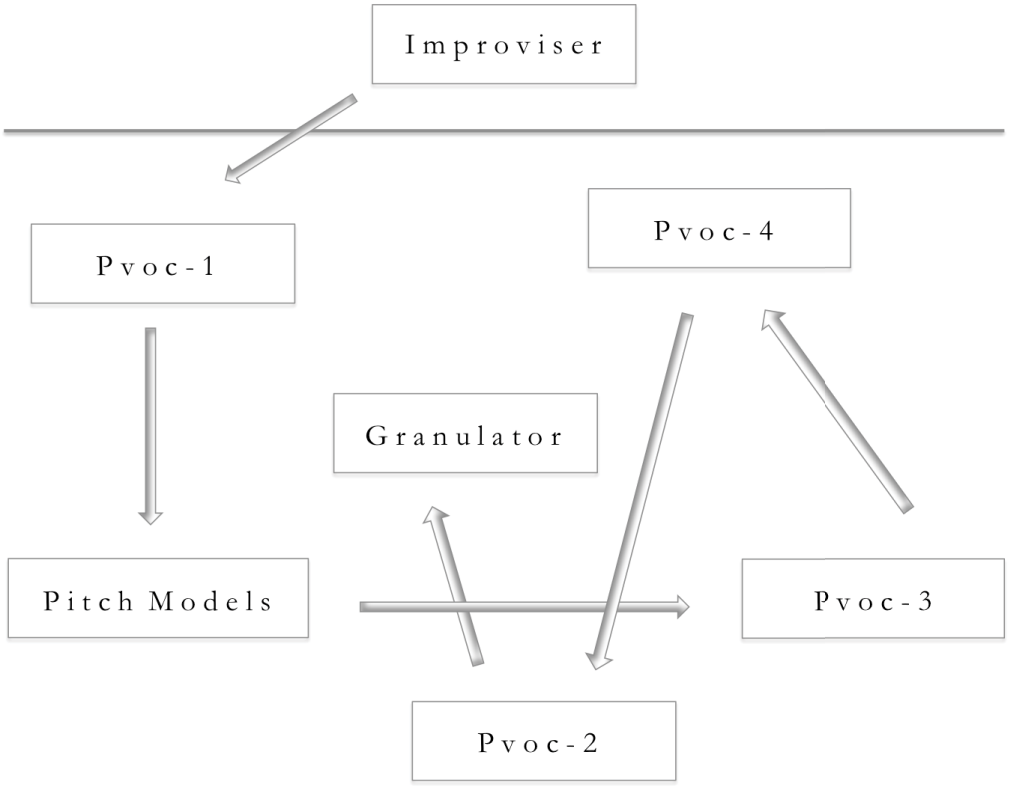


Figure 39: *_derivations' self-referencing* algorithm. This figure displays a cycle of output triggering and phrase comparison that occurs from input provided by an improvising musician.

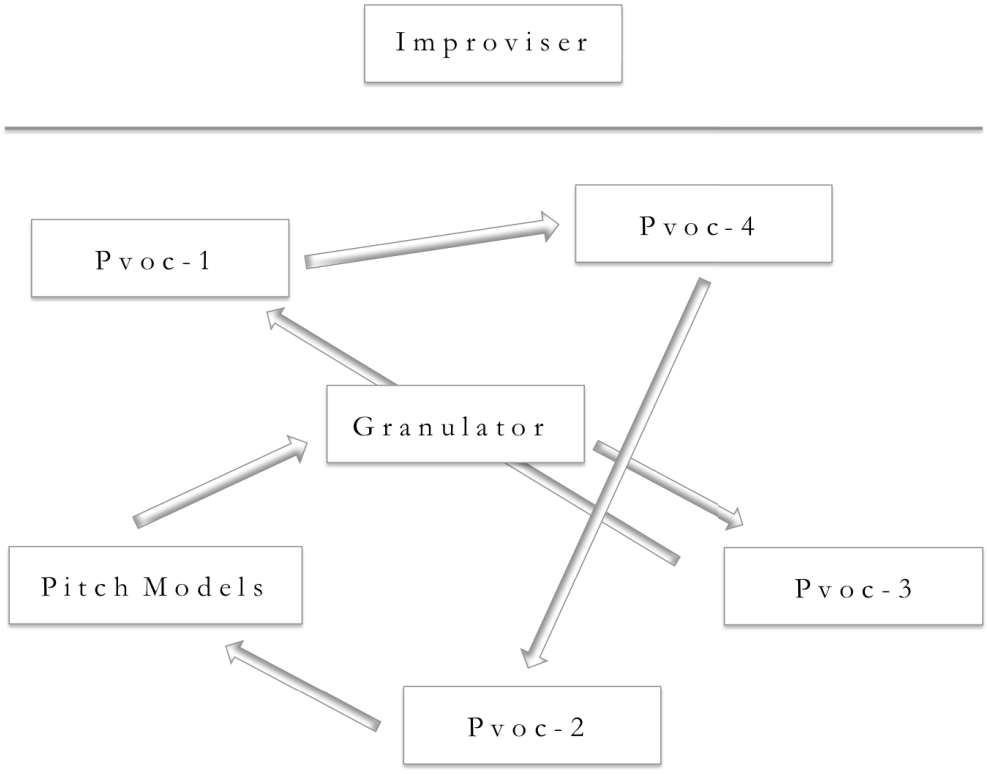


Figure 40: The *self-referencing* algorithm can continue generating material without continued input from the performer.

An important aspect of the *self-referencing* algorithm is the decentralisation of the *phrase matching* algorithm, and the lack of a hierarchy resulting from this continuous form of phrase comparison. As illustrated in Figure 39, the input provided by the improviser represents a single link in a continuous chain of phrase comparisons. As the improviser performs with the system, the input *phrase matching* algorithm is queried upon phrase boundaries to find the closest match from the database. Once found, this match is passed to the first available output module in the *self-referencing* algorithm for output (*Pvoc-1* in Figure 39). Once this phrase index is received by *Pvoc-1*, a subsequent comparison is made with this phrase index and the database, excluding the current phrase, with the next match retrieved and sent to the next module (*pitch models*) for output. If the live performer ceases playing, the algorithm continues passing phrase indexes from module to module, creating a cyclical pattern of comparisons, as illustrated in Figure 40. Consequently, the algorithm does not prioritise the live input over its internal comparisons; it simply passes any information received through the network in a continuous fashion.

The advent of the *self-referencing* algorithm greatly improved *_derivations'* sense of autonomy as a performative system. Whilst the material passed from player to player originated directly from the live performer, the internal dynamics of the algorithm's comparison and triggering created a level of opacity in its approach. Such opacity helped to separate the live performance of the instrumentalist from the generative grammar of the software itself. The conceptual interest of this enhanced form of software autonomy was the solution to a problem encountered in my performances with the software. Finding the software overly reactive in performance, this intentional distancing of *_derivations'* generative grammar from its analytical capacities helped to shape the future direction of the project, along with my conceptions of both performance practice and the design of interactive systems.

6.7 Evaluating live sampling and generation in *_derivations*

As a performer-developer, I evaluate my systems first and foremost for their capacity to contribute to an improvised encounter in a musically interesting fashion. As has been demonstrated throughout this thesis, incremental advancements to my systems often fuelled unforeseen areas creative speculation and refinement. Having worked with

sampling-led forms of musical generativity for some time, a series of questions regarding the nature of sampling as both generative method and principle of structural organisation came to light. With respect to *Live-processing-1,* the notion of 'momentary' sampling had been evaluated as a limiting approach to gathering source material for use in an interactive musical system. Through the advancements of *phrase player* and the above-discussed matching algorithms in *_derivations*, this limited form of live sampling was expanded to enable *_derivations* to 'mine' continuously sampled musical material using content-based music information retrieval techniques. This approach to sampling and re-generation could not have been predicted from the outset of my creative practice.

After performing with the *_derivations* system and fine-tuning its processing capabilities, I began to question the musical interest of continuous live sampling itself as a generative method in human-machine performance. Although this form of live sampling facilitated a flexible content-based form of musical generativity, such an approach still limited the system's response to the temporal context of the present improvisation. This was due to the fact that the growing size of the system's 'vocabulary', although accessed in a sophisticated manner through *phrase matching*, displayed a consistent linear trajectory throughout an improvisation. That is, the relative length of a live improvisation ultimately determined the size of the *phrase database* used by the matching algorithm to develop new musical material in performance.

This aspect of the *_derivations* system raised interesting questions regarding musical form and interactivity between human and machine performances. As critiqued in Chapter 2, the momentary sampling approach of *live looping* (see Section 2.3.3.1) can suffer from an additive and linear form of musical structuring. Although a more sophisticated and automated technique, the continuous live sampling employed in *_derivations* also suffered from its own structural limitations by being tied to the temporal context of the presently unfolding musical scenario. The additive and linear nature of such a sampling-led form of generativity, whilst providing a coherent approach based on the recent past, followed a pre-determined trajectory that was directly related to the growing possibility space provided by the *phrase database.* Therefore, as a performance with the system unfolds in time, the richness of the musical material available to the system grows.

As discovered through performative testing, this facet of the system encouraged certain modes of interactive performance. With the knowledge that the system's database (its sonic *vocabulary*) was growing over time, a performer may take advantage of the cumulative nature of the system's sonic vocabulary as a performance strategy. In my own performances, this growing space of possibilities was exploited to develop musical structures based upon elaborations of initial seed materials. The *phrase matching* method ensured that each musical gesture performed by the instrumentalist was treated both as a query to the *phrase database,* and as a new musical phrase for later use. Due to this aspect of the system's design, performing with *_derivations* became as much an exercise in developing the system's vocabulary, as it was an interaction with the system itself. Though interesting form of musical interaction, the consistently cumulative nature of this interactive relationship began to feel limited as a structural device in improvised musical performance. The connection to the musician's recent performance limited the complexity and unpredictability of the system's output, and therefore its potential for creating a novel interactive relationship between human and machine in performance. If the system's sonic vocabulary was completely contingent upon the sounds siphoned from the performer in real-time, then the ability of the system itself to surprise and provoke – a trait that Young has described *strong* interactivity (see Section 2.2) – was subsequently diluted.

After considering these issues, it was determined that *_derivations* lacked the ability to express a *system-specific* sonic vocabulary. The system's reliance on musical material siphoned from the current performance promoted specific structural constraints on an improvised encounter. It was therefore decided that the form of live sampling in use in this system should be expanded, in order to enable the pre-definition *_derivations* vocabulary in advance of a musical encounter. It also occurred to me at this time that the cumulative nature of *_derivations'* real-time sampling did not take full advantage of the potential of the *phrase matching* approach to musical generativity. As a form of content-based music information retrieval, the application of *phrase matching* need not be limited to the most recent past of a musical improvisation. Hence, the size of the queried database was of no direct consequence to the matching algorithm, as the technique is decidedly agnostic as to the temporal context from which the musical materials were derived. Therefore, it was the potential of this newly developed approach to musical generativity that forced an expansion of my approach to sampling as a method for

gathering musical materials, and therefore a considerable modification of *_derivations'* musical and interactive capabilities.

6.8 Session databases

The sampling and storage architecture of *_derivations' phrase database* was based upon the cumulative storage of musical phrases analysed from a live improvising musician. As discussed previously, this architecture was based around a central data collection containing precise timing information relating to a single audio buffer recorded during performance. The statistics module used in both the above versions of the *phrase matching* approach (the initial *analyzer~* and later *MFCC*-based approaches*),* also contained a list of spectral data indexed to individual phrases contained within the phrase database. Whilst the *granulator* and *4-buff-pvoc* modules used this data for their generation, the *pitch models* module made use of its own internal data collection for storing and recalling sinusoidal models analysed from the live input (see Section 5.2.2.1). In any given performance with the *_derivations* system a series of data collections were consulted in order for the software to generate new musical materials. It was from these data collections that the software made its contribution to an improvised encounter.

Importantly however, at the end of an improvised session all of this data was discarded to make way for new material captured in a subsequent improvisation. Whilst *_derivations* worked in a cumulative fashion within an individual improvised performance, each performance with the system began with an empty phrase database; the system had not been designed to retain audio nor analysis data from session to session. Given the detailed and robust nature of the software's data storage and recall facilities*,* it was decided that a performer should be given the option to save such data collections for later use. Such an approach subsequently enabled the real-time capabilities of *_derivations'* data storage to be re-appropriated for future interactive encounters. Making use of the *coll* object in Max, the above data collections could be easily stored and recalled from disk in the form of simple text files. All audio and analysis data collected during an improvisation with *_derivations* could be later used to define the system's sonic vocabulary in advance of a performance.

To facilitate this, the *phrase database* was amended to allow all relevant data collections to be named and saved in a central database directory on disk. Initially referred to as a *Rehearsal Database*, this central database contained relevant analysis data, the contents of the audio buffer recorded during the session as well as a high-level 'master' file used as a reference to all files pertaining to the exported session. Figure 41 shows the internal mechanism used for saving data stored during an improvisation with *_derivations*. Once a performer has finished an improvised session with the software, clicking the 'Save Rehearsal' button in the *phrase database* launched a save dialog enabling the user to write the data collected during the improvisation (this UI element is visible in Figure 28). Naming the database and clicking 'Save' subsequently wrote the contents of each *coll* object to disk with a unique filename prepended with the user-chosen database name. In the initial iteration of this approach, the user was expected save the database in a dedicated folder of their choice on disk. In later versions of the software, this functionality was replaced with use of the *shell* object and its *mkdir* command, enabling the simplified saving and creation of named directories in a centralised location in the Max application folder.

Once a location on disk was chosen to save the database, the files stored by *_derivations* included the following:

- the '_derivations Master' file (e.g. *databasename*_derivations-MASTER.txt)
- a file referencing the audio files stored in the database (e.g. *databasename*-audiofiles.txt)
- the recorded audio files themselves (e.g. *databasename*-reh-1.aiff)
- a 'phrases' file containing cue points relevant to the audio files (e.g. *databasename*-phrases.txt)
- a 'statistics' file containing descriptor/MFCC data pertaining to the segmented phrases (e.g. *databasename*-stats.txt)
- a 'models' file containing *pitch models* sinusoidal models (e.g. *databasename*-models.txt)
- a file grouping models ranges within phrase boundaries (e.g. *databasename*-model-phrases.txt)
- a file containing indexed silence lengths analysed from the input (e.g. *databasename*-model-silences.txt)

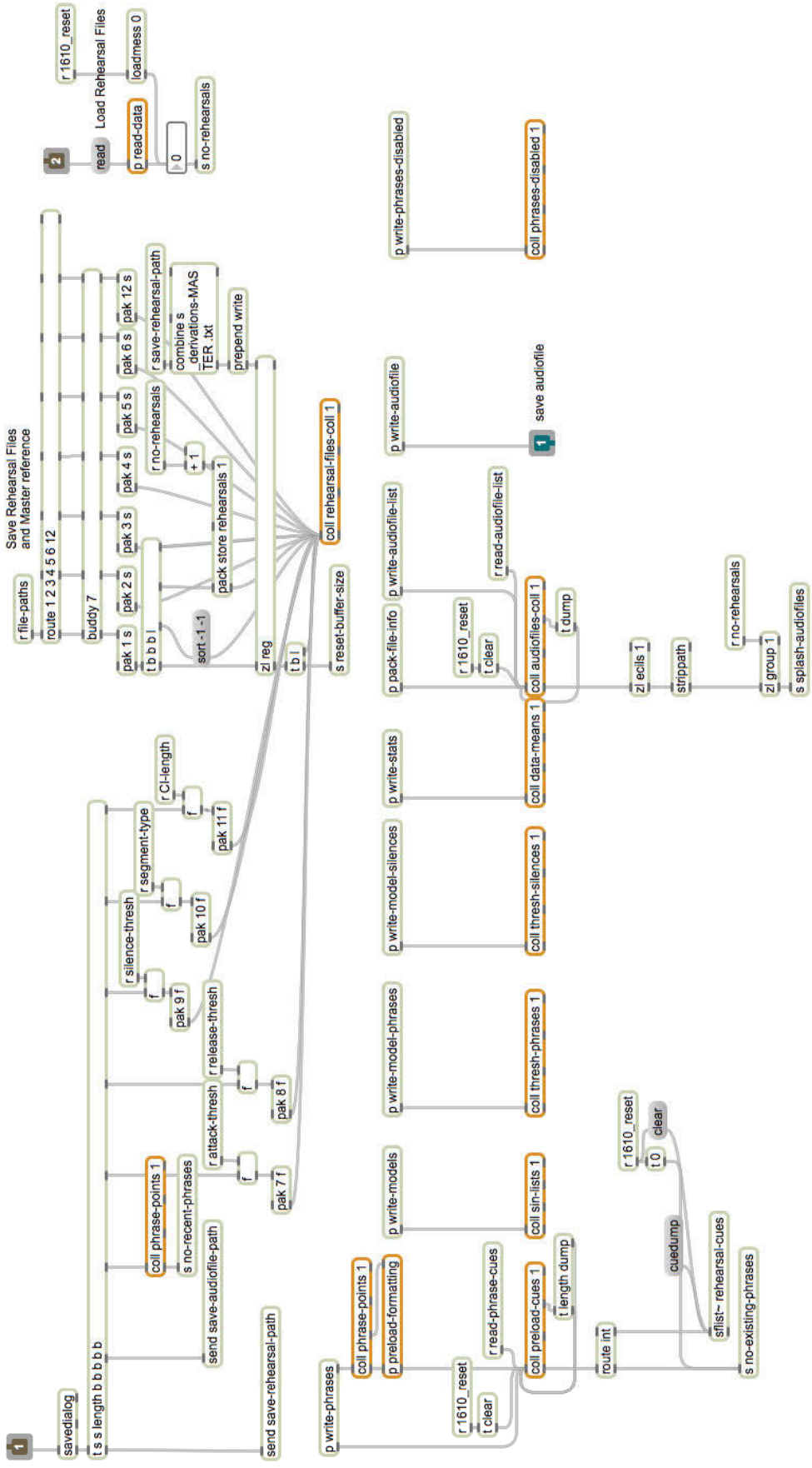


Figure 41: Inside the [p save/load-rehearsals] subpatcher in the *phrase database* module

- a file containing reference to 'disabled' phrases from the database (e.g. *databasename*-phrases-disabled)

Having saved the totality of the software's data collections to disk, the performer was now given the option to engage with *_derivations* as an interactive environment possessing a pre-defined sonic vocabulary. By selecting 'Load Rehearsal' in the *phrase database* module, the performer could navigate to a previously stored rehearsal database on disk to load prior to an improvisation with the software. Selecting the '_derivations-MASTER.txt' file from the chosen directory, the software subsequently loaded the relevant data files from disk back into their original locations in the software. This process had the effect of populating *_derivations'* internal databases with data analysed from a previous performance with the software. Once loaded, a performer could either engage with *_derivations'* loaded session database alone, or cumulatively build upon the loaded database during performance.

6.8.1 *Cumulative databases*

This new approach to musical generativity in *_derivations* was also designed to be cumulative, facilitating the session-to-session development of layered and complex databases of pre-analysed material. By maintaining the *phrase database's* recording and storage capabilities, the software now enabled a hybrid approach that could combine a loaded database with new material recorded and stored throughout the current improvised session. After an improvised session with the software, the performer was given the option to save the current session as a cumulative 'multi-session' database. Re-saving the session with an identical name appended the most recently recorded analysis data to that already stored on disk. In addition, the newly recorded audio was saved to disk alongside the pre-existing audio, with the *databasename*-audiofiles.txt file updated to contain reference to the newly saved file. In this approach, multi-session databases would therefore contain reference to multiple audio files pertaining to the number of sessions used for its creation. Figure 42 displays an example of such a database on disk, whilst Figure 43 shows the splashscreen displayed in the software once this database is loaded.

6.8.2 *Merged databases*

In a further development of this approach, the performer was also given the choice to 'merge' existing databases stored on disk to create hybrid databases of materials. With the assistance of the *shell* object, the software facilitated the copying and re-formatting of database files on disk into new hybrid databases. Whilst a relatively simple extension of the database concept, the facility to merge previously unrelated databases became an inspiring development in the *_derivations* software. In contrast to building *_derivations'* vocabulary cumulatively from session to session, potentially disparate musical materials from contrasting instrumental sources could be combined into a single multi-session database. This form of curatorial authorship over the system's vocabulary became an important feature of the software, enabling great flexibility in the character of the system in performance. In my own work, I began making use of *_derivations'* analysis and storage capabilities to develop large multi-session databases containing a variety of instrumental sources. These ranged from my own performances to those of other musicians using the system, to carefully prepared sessions containing prepared sound design elements, percussion and other sonic materials.

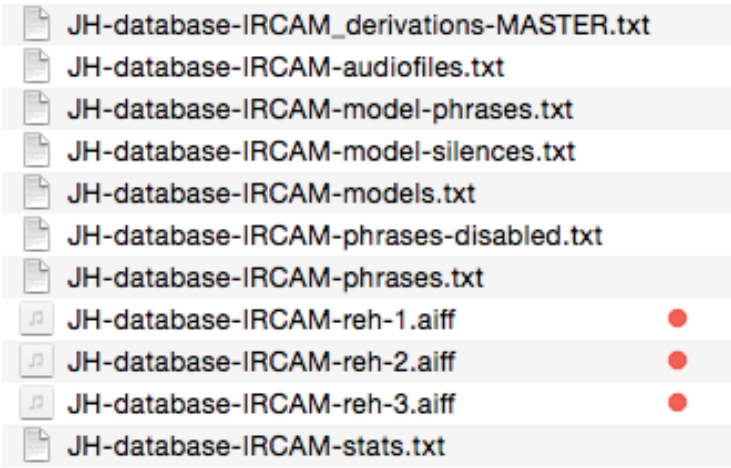


Figure 42: Example multi-session database on disk containing three 'rehearsals' or 'sessions'

6.8.3 *Phrase disabling*

As the session database concept matured, a final function was added to the *phrase database* module that gave the user detailed control over the makeup of the database chosen for use in performance. An 'Audition' window allowed the performer to audition individual

phrases contained within the loaded database, and also to disable both individual phrases and entire sessions from *_derivations'* matching algorithm (see Figure 44). This option was designed to give the user the flexibility to quarantine undesirable sonic materials from output during a future improvisation. This was achieved by making the chosen phrase indexes invisible to the *phrase matching* algorithm.

Without deleting these phrase indexes from the database, this approach simply filtered the chosen indexes from the list of matched phrases in the *phrase matching* process. Given this filtering process, disabled phrases could also be easily re-enabled from within the interface. Phrase indexes listed as disabled were written to the phrase database to be loaded upon a subsequent performance, as stored in the '*databasename*-phrases-disabled.txt' file. This fine-grained control over the software's capabilities allowed for the performer to curate the software's vocabulary with a great degree of control.

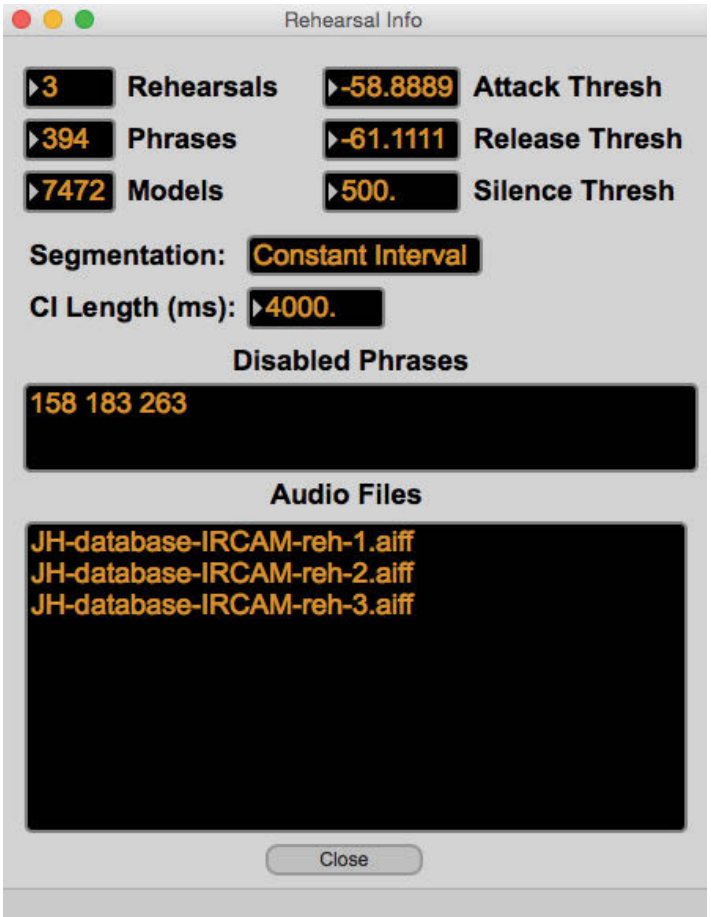


Figure 43: *Rehearsal Info* splashscreen that appears after loading a session. The above database contains three 'rehearsals' or 'sessions' containing a total of 394 phrases. Three of these phrases have also been 'disabled' from use by the *phrase matching* algorithm

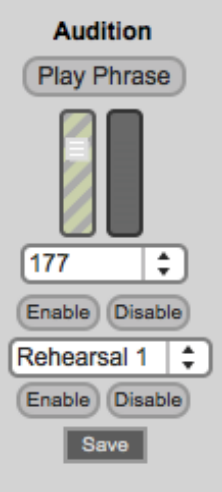


Figure 44: The audition window. The first drop down menu allows the user to audition individual phrase indexes, and enable/disable their use in *phrase matching*. The Rehearsals dropdown menu allows for per-rehearsal enabling/disabling of phrase indexes.

6.8.4 *Performing with multi-session databases*

As discussed in Section 6.5, *_derivations' phrase matching* algorithm was designed to simultaneously query and store new analysis data into the *phrase database* module. As an improviser performs with *_derivations,* each phrase used to query the database is also stored as a new phrase index for future matching. In this approach, a direct link between continuous live sampling and phrase matching was formed that engendered a cumulative approach to musical generativity. With the advent of *session databases*, it was no longer a necessity for *_derivations* to simultaneously build and query the *phrase database* during performance. Once a database had been loaded into the software, *_derivations' phrase database* contained sufficient data to be immediately queried from the analysis of a live performer. That is, the *phrase matching* algorithm no longer needed to be linked to the storage of new material.

This new scenario therefore gave the performer the option of either engaging solely with a loaded database, or with both the loaded database in addition material recorded throughout the current session. Although the cumulative approach was an interesting means of developing coherent multi-session databases, it was found that this approach was most interesting for defining *_derivations'* vocabulary during rehearsal, not necessarily in live performance. The ability to cease sampling new material from the improviser in

favour of direct interaction with a pre-defined database became essential. In order to engage solely with a loaded database, the performer was presented with a simple toggle to disable 'Analysis Storage' (see again Figure 28). This option disabled the recording of audio into the central audio buffer, as well as the cumulative storage of analysis data captured from the improvising musician. Here the output of the MFCC analysis was now used solely to query the pre-defined *phrase database* during performance, and not for the continued development of the database itself.

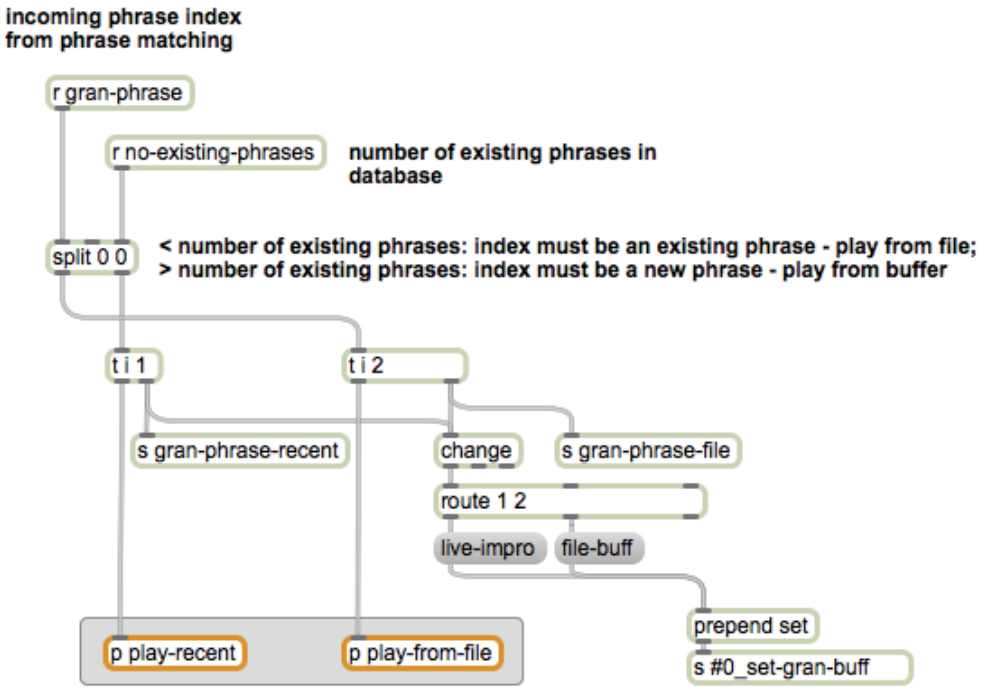


Figure 45: A simple algorithm in the *granulator* for determining which source to access for its audio content

To make use of both current and pre-defined databases in performance, *_derivations'* data storage and recall mechanism was redesigned to facilitate the use of both current and previously analysed audio and analysis data. In the case of analysis data, new analyses captured during a performance were appended to the list of data imported from a previous session. Each newly analysed phrase index and associated analysis data was cumulatively added to the data already contained in the *phrase database*. For audio content, the use of both current and past live-sampled audio materials required an approach that combined access to both the central audio buffer, and the audiofile(s) stored on disk. To achieve this, the two audio processing modules in *_derivations* (*4-buff-pvoc* and *granulator*)

were modified to account for these two different audio sources. By way of example, Figure 45 displays the simple algorithm used in the *granulator* module to determine which source the granulator should access its content for processing.

In this example, if the incoming phrase index received from the *phrase matching* module is greater than the range of indexes present in the loaded database, the index must refer to a newly recorded phrase. In this case, the *granulator* accesses the internal audio buffer as normal, sending the phrase index to the [p play-recent] subpatcher. If however the index received falls within this range of existing phrases, this index must refer to a phrase indexed to a file stored on disk. Here the received index is sent to the [p play-from-file] subpatcher in order to dynamically access the relevant phrase from the audio file stored on disk by referencing the [coll preload-cues] data collection. Given that the audio file(s) stored on disk are likely to be of considerable length (they are the full length of previous improvisations), this dynamic process is aided by the *buffer~* object's *read* and *size* messages, enabling the dynamic loading of specific portions of the audio file into the *buffer~* object, not the whole file.

6.9 Reflections

6.9.1 *Evaluating session databases*

With the complete audio and analysis data captured during a session saved to disk, *_derivations* could now be engaged with as an interactive and generative system containing a pre-defined corpus of sonic materials. This shift in focus in the system initiated a new form of sampling-led interactive performance. Engaging with session databases cumulatively, each performance with the software contributed to the development of a library of material for use in future improvisations. The more a musician performed with the software, the more material the software would have in its vocabulary. In the case of merged session databases, a user of *_derivations* now had a great deal of curatorial authorship over the software's subsequent contributions to a live performance. In this respect, a performer's interaction with *_derivations* was now split between the dual roles as 'user' and performer, with the software acting as both a pre-compositional tool and a live performance system.

This cumulative and curatorial aspect of the software became a defining feature in my own performances with *_derivations*. As a performer, I began to think of *_derivations* as a vessel that could contain a personalised sonic vocabulary specifically designed for each interaction. Although every performance with the software is unique, the added ability to curate the software's vocabulary gives the performer a great deal of freedom to define the space in which an interaction takes place. The potential for building upon previous encounters made the rehearsal space an invaluable place for developing the software's sonic vocabulary for an eventual performance. From the advances of the cumulative approach to developing session databases, I developed an interest in the unique role that rehearsal could play in an interaction between a human performer and a machine interlocutor. Developing a database from one rehearsal to the next was an interesting development, as it broadened the scope in which *sampling-led* approach to interactivity could be conceptualised. In such an approach, cumulative interactivity was present both inside and outside the boundaries of a single performative encounter with the software. In this respect, *_derivations* now took into consideration Paine's concept of interaction as representing the 'cumulative experience of interrelationship' (Paine 2002) (see also Section 2.3.5). Each performance with the software, whether live or in the studio, was also a means of developing the complexity of the software itself. This approach to musical interactivity shone a light on rehearsal as an integral part of the development of an improvisatory practice. It was this view of *_derivations'* new capacities led me to first name databases stored on disk 'Rehearsal Databases'.

The capabilities of merging session databases allowed the performer to breathe new life into sonic material used in previous musical contexts. Large databases of disparate musical materials could be used to diversify *_derivations'* sonic vocabulary, and to enlarge the possible sound palette available to the software in an improvised performance. In my practice, curating such databases became a form of pre-compositional authorship, as I worked to define the possibility space in which I would interact with the software during performance. A combination of contrasting sonic materials allowed me to explore different timbral spaces within an improvisation, effectively guiding the software through a large corpus of materials. In addition, the larger this space became, the more possibilities existed for genuine surprise and unpredictability from the software's output.

6.9.2 *Performing with a stabilised artefact*

In its various iterations, the *_derivations* interactive performance system has been a part of my live performance practice for the past three years. Since mid-2013, the software has been made feely available online via the dedicated *_derivations* website since mid-2013, and has been downloaded approximately 2200 times.[42] Since this time, *_derivations* has been used in numerous performances throughout the world by myself and by a variety of instrumental improvisers. Performances with *_derivations* have been invaluable in informing the continued development of the software, and for evolving my personal conceptions of interactive performance practice. Each performance with the software is unique, and every performer working with the software approaches the improvised encounter differently. These encounters have included performances by improvisers with whom I have had direct collaborative involvement[43], and events curated by third parties with no direct involvement of myself in their production.[44] In February 2014, a collection of improvisations with the *_derivations* system was released as a 6-track EP on the Integrated Records label. Along with the *_derivations* software itself [45] and other performance documentation provided in the submission materials, this album is presented as a significant outcome of this research.

In each of the performances documented for this research, improvisations with the *_derivations* system reveal the particular interactive traits of the software as provoked by the individuality each performer. In my own performances I have explored a breadth of sonic materials with the system, as well as ways of working with the software. The track *Chelmsford* on the *_derivations* EP is the earliest recorded performance with the system itself, showcasing the use of an atonal/microtonal yet melodic form of improvisation. Making use of the software's core live sampling mode, the layering of transposed and processed soprano saxophone materials provides a dense, contrapuntal texture. In my

---

[42] See Appendix D for more information on the website, which is available at the following URL: http://derivations.net

[43] See Appendices B and C and the event timeline outline in Appendix F

[44] See Appendix H for details on third party produced releases.

[45] See Appendix A for details on the *_derivations* distributions included in the submission materials of this thesis.

live performance at the Musical Metacreation Weekend (documented in Appendix C)[46], my improvisation focuses on percussive effects and extended saxophone techniques. A very physical form of action and reaction is evident in this performance, displaying a playful sense of interactivity with processed percussive sounds originating from the saxophone. This form of improvisation has evolved alongside the software, and is similarly evident in the track *Oblique* from the above-mentioned *_derivations* EP.

Close collaborations with saxophonist Joshua Hyde and pianist Zubin Kanga highlighted the way in which the software can be exploited for use in quite different musical contexts. Joshua Hyde's continued use of the software in his solo and duo performances attests to the flexibility of the software as a performer-driven interactive system.[47] Hyde's work has involved heavy use of idiosyncratic *session databases*, using percussive and found sound materials recorded specifically for use in his saxophone improvisations with the software. This is clearly evident in the performance with *_derivations* on his debut solo disc, in which the tenor saxophone is paired with a database comprised of tubax, prepared piano, bowed percussion instruments and various found sounds.[48] In addition to his solo performances, Hyde has performed with *_derivations* in duo and ensemble contexts. Most recently, Hyde made use of *_derivations* in performances with percussionist Noam Bierstone of *scapegoat* duo, before introducing the system to an improvisation workshop with six free improvisers in Detroit, Michigan.[49] His interest in expanding the performative context of the system, and taking curatorial control of its features has been very encouraging.

Pianist Zubin Kanga recently included *_derivations* in a tour program for solo piano and live electronics.[50] In these performances, Kanga chose to interact with the software in its original live sampling incarnation, without recourse to the use of *session databases*. In this mode, the software grows its pool of resources throughout a performance, allowing the

---

[46] This performance can be streamed here: https://www.youtube.com/watch?v=GHxHumlCZOQ

[47] Hyde's performances are documented in two musical releases outlined in Appendix B, as well as a live performance in Appendix C

[48] See Appendix B

[49] See Appendix F for a detailed chronology of recent performances.

[50] A sample performance in outlined in Appendix C, and recent reviews of these performances can be found in Appendix I

pianist more material to improvise with as the performance progresses. This performance was placed in the context of Kanga's tour program 'Dark Twin', a program of works seeing the pianist's performance shadowed and augmented by various forms of audio-visual processing. The piano's natural resonance and dense polyphony allowed Kanga to exploit the system using a layering technique. Using this cumulative format gave the impression of a tangled web of materials that the performer enmeshes with throughout a performance. In this mode, the performer remains in control of the structural layering of the musical materials, guiding the software through a cumulative database of previously performed phrases. This mode of performance suited the musician's pianistic style, as well as the overall concept of the tour program.

6.10 Conclusion

Throughout this final 'wayfinding' chapter I have detailed the advancements in my programming practice that led to the stabilisation of the *_derivations* system. Reflecting upon the development of this idiosyncratic software artefact has revealed a deep *mangling* of my personal performance practice along with the development of the artefact itself. Having developed individual modules before integrating them into a larger interactive system, the trajectory of the *_derivations* system could not have been predicted in advance. The internal dynamism of both the *pitch models* and *4-buff-pvoc* modules discussed in Chapter 5 contributed greatly to the trajectory of the *_derivations* system, and to the decisions made throughout the stabilisation of this software artefact. However, whilst this *bricolage* approach to programming undertaken in this research has been formative to the creation of this software, this chapter has also highlighted the change in my programming practice to find stable and reliable *accommodations* to unforeseen *resistances* encountered throughout my practice.

As discussed in Section 6.5, the *phrase matching* approach devised in this system sought to provide a coherent, generative strategy that could unite the three, independent processing modules contained within this system. Focused upon matching the timbral content of a live improviser with a growing database, this form of generativity required a robust and reliable algorithm to compliment the nuanced and unpredictable automations of *pitch models, 4-buff-pvoc* and the *granulator* module. Following considered testing and refinement, MFCC vectors were chosen as the best possible means for achieving this

aim. The *self-referencing* algorithm in *_derivations* was developed to address the triggered nature of the software's output modules. This advancement in the software altered the system's output from a reactive to an autonomous and emergent form of generativity. This addition enabled the software to generate material with reference to a database without prioritising input from the live improviser. Consequently, the performer's real-time input into the system is treated as just one of several players in a self-referential game of *chinese whispers,* passing phrase comparisons from one module to another. With the use of MFCC feature vectors for phrase comparisons, this form of continual phrase comparisons has proven effective in enabling the software to successfully generate material independent of human input.[51]

The addition of *_derivations' session databases* concept changed the focus of the system from one based entirely on real-time live sampling, to one enabling the performer to make curatorial decisions about the system's overall sonic vocabulary. Given the *phrase matching* algorithm's agnosticism to the temporal context of its phrase indexes, the use of cumulative and merged session databases allowed the system to exploit this algorithm's full potential as a content-based MIR technique. As discussed above, this aspect of the system has been a feature of my own performances with the software, and performers such as Joshua Hyde have made extensive use of this feature to curate their own sonic material for use with the software.

In summary, the previous three 'wayfinding' chapters have outlined the iterative and exploratory approach of my development trajectory throughout this research project. Tracing the evolution of my creative practice, I have used self-reflective practice to advance both creative artefacts and conceptualisations of the practical domain itself. In addition to highlighting issues, concerns and interests relevant to advancing practice, the use and refinement of the *_derivations* system has aided in identifying larger research themes regarding the creative practice of human-machine performance. The following chapter presents three sustained reflections upon such research themes, connecting my creative practice to broader issues concerning human-machine performance practice that are relevant to the interactive computer music community.

---

[51] An example of a 'system only' performance using a large session database can be heard here: https://soundcloud.com/emeidos/derivations-alone

Chapter 7. Findings: Reflections of a performer-developer

7.1 Introduction

The current chapter presents three sustained reflections upon research concerns emerging from the creative work undertaken as a part of this doctoral project. As discussed in Chapter 3, as a *creative-production* research project I have used the creative practices of interactive music system development and use to investigate complex themes arising from these burgeoning practices. The software artefacts, recordings and documentation developed throughout the research, whilst also presented as significant research contributions, must be understood as the end point of a considered search for understandings of the domain in question. As noted by Scrivener and Gray, the multiple issues, concerns and interests of creative production researchers need to be foregrounded in such approaches in order to highlight researchable problems raised in practice (Gray 1998; Scrivener 2000). This chapter therefore presents the culmination of my reflections on such issues, concerns and interests arising throughout this creative practice research project.

Performer-developer devised creative artefacts are naturally bound to the specific artistic, technical and social contexts surrounding their production. The tightly woven relationship that emerges between development and use in such creative contexts presents a challenge for research methods seeking to understand such practices purely from their creative outputs alone. My role as a performer-developer creating and performing with interactive performance systems has given me a unique perspective on the relationship between design and use in such artistic practices. This chapter reflects on personal experiences in order to highlight and analyse relevant concerns emerging from my practice. Importantly, these reflections are both situated within the lived experience of the creative practice itself and contextualised with reference to relevant theory.

The chapter is divided into three sections interrogating separate concerns and interests related to interactive system development and performance practice. Section 7.2 engages with the complex and interrelated topics of machine agency and authorship in the development and use of interactive performance systems. With reference to concepts derived from actor-network theory, this section seeks to make sense of the relationship

that exists between development and use of these systems. Section 7.3 considers the concept of musical interpretation as it applies to improvised human-machine performance practice. Despite the inherent improvisatory nature of the performance practice, here it is suggested that the development of such systems is akin to the creation of a musical text, and therefore can be understood as extending traditional conceptions of interpretation in musical performance. Finally, Section 7.4 interrogates the concept of autonomy and interaction in the use of interactive musical systems. Here *symbiosis* is proposed as a metaphor for musical interactivity, a concept that foregrounds the reciprocal relationship that exists between both performer and system in the practice arising from the use of the *_derivations* interactive performance system.

7.2 Artefact scripts and the performer-developer

Performer-developer devised creative artefacts provide a unique focus of study for the field of human-computer interaction. The role of the designer as both developer and user raises some interesting questions about the intersections of human and machine agency through both the development and use of such artefacts. Here I am interested in the way in which both of the roles present in this creative practice mutually influence each other, and how we may make sense of this process with reference to theories from science and technology studies. In my creative practice, my role as both designer and user of interactive software systems is complex and multi-layered. My software systems are not intended to act as transparent 'tools' for use by the performer, but as quasi-autonomous performance partners exhibiting emergent and dynamical properties. In my practice the relationship between design and use has therefore been focused upon balancing surprise and familiarity for myself as a performer engaged in human-machine interactive performance.

Within this creative practice a reciprocal relationship emerges between the performer-developer and their emerging technology. This development process involves setting the conditions for a future interaction between oneself and an interactive musical system. Despite the naturally close relationship between the development and usage contexts, unpredictable and emergent algorithmic processes lead to a direct consideration of the role of material agency in shaping both creative practices. In this space, the development of the software artefact is an emergent process that relies upon an interplay between the

developer's assumptions of the interactive context, and their direct experience of the software through use. Assumptions of a system's capabilities made in development do not always match experiences in performance, and modes of performance also evolve over time in relation to cumulative experiences with a system over multiple rehearsals and performances. The processes of development and performance practice greatly influence each other through this cyclical form of performance-led software development. This relationship provides fertile ground for both the development of new musical interfaces and performances, as well as new knowledge about human-computer interaction from inside the performer-developer context.

Throughout my creative practice, one area of interest that has become integral to my understanding of this practice is the role and perception of machine agency in both development and use of such software artefacts. The employment of algorithms that facilitate system autonomy encourages speculation on the role of machine agency in musical performance. In addition to this performance-time consideration, I am also interested in the way in which both the development and use of such artefacts hinges upon the perception of the machine at various stages of the creative process. Because of the intimate knowledge a performer-developer has of their software artefact, issues of machine autonomy, agency and authorship are brought into sharp relief. These issues relate both to how the performer-developer as 'user' synthesises such concepts into an emergent performance practice, and how the performer-developer as 'designer' takes cues from this practice to further design.

In the context of improvised human-machine performance, a central aim of the software development process is to define and harness the material agency of software algorithms to provoke new modes of musical discourse. As discussed in Chapter 2, such approaches have involved musical modeling, varying degrees of sonic and algorithmic derivation and a concern for musical autonomy to achieve these ends. As discussed in Chapters 4, 5 and 6, in my practice I have been concerned with balancing both deterministic and unpredictable algorithmic processes in the creation of my software artefacts. This negotiation between autonomous generation and performer control has led to questions about the inherent agency of the machine in this process. From within the performer-developer context, I am questioning whether such musical systems can truly exhibit agency in such a scenario. If so, I am interested in how might we understand

the role of this material agency in defining the relationship between the design and use of such systems.

7.2.1 *Artefacts as instruments of sociotechnical knowledge*

Actor-network theory (ANT) has much to say about the relationship between humans and technology, as well as the role of technology in society. This analytical perspective treats humans and non-humans as equally capable of exercising agency in a network, positing that 'sociotechnical systems' are developed through negotiations between people, institutions and organisations (Latour 1992). In Akrich and Latour's writings, technical artefacts play an influential role in complex networks of actants that make up societies and cultures. These artefacts are considered central to the way in which organisations, institutions and societies interact due to their non-neutral ability to exert agency within networks. This non-neutrality is due to a variety of factors surrounding both the development and use of such technical objects in sociotechnical settings. ANT theorists propose that artefacts be studied as an integral part of understanding social and cultural phenomena, a position articulated famously by Bruno Latour who stated that 'technology is society made durable' (Latour 1990). In order to understand the agency exerted by technical artefacts in their networks, the analyst may trace the artefact's development through it its 'innovation network', the specific socio-technical context within which the artefact is developed. In addition, by analysing human interactions with stablised artefacts we may also uncover information about the way in which arefacts exert influence over those who interact with them in specific contexts.

In actor-network theory, 'technical objects' (artefacts) are foregrounded as mediating forces within interaction, social networks, culture and society (Akrich 1992; Akrich & Latour 1992; Latour 1992). One concept integral to ANT is that of an artefact's *script*. Akrich has defined the development of an artefact's *script* as involving the projection of a 'virtual user' into and through an artefact (Akrich 1992). According to Akrich, designers inscribe and project roles for 'virtual users' into the workings of an artefact through the design process. This process is called *inscription* and describes how the specific constraints and affordances of an artefact are viewed as the embodiment of usage conditions envisaged by its designer. Conversely, it is in the user's encounters with these designed artefacts that the designer's visions are 'de-scripted' and subsequently realised or not

(Akrich 1992). An artefact's script is a rich and complex way of understanding both the motivations and domain specific assumptions of designers during development, as well as the way in which real-users interact with the affordances expressed through the material agency of the artefact (Akrich & Latour 1992). As discussed in Mattozzi (2012), whilst Akrich's definition of the script focuses upon the designer's projections of a virtual user onto the technical object, for Latour the artefact itself can be analysed without reference to the designer's decisions. There is a difference between the designer's conceptions of the usage scenario and what the artefact itself enables/allows, i.e. its *competences*. Rather than focusing upon the disconnection between design and use, Latour's analysis seeks to discover the script of the artefact itself and how this can be seen as an expression of both human and material agencies. The process of analysing an object is therefore not necessarily to 'uncover' the designer's script as imbedded in the object, but also understand the object's own script as perceived (Mattozzi 2012).

These two perspectives provide complimentary means by which we may understand the agency exerted by an artefact within a network, and from where the artefact may derive this sense of agency. However, for performer-developer devised creative artefacts we must also concern ourselves with the boundaries between the script as designed, and the script as perceived. Unique to this particular context is the emergent means by which such artefacts are developed through experimental creative practice, and the way in which performance practices emerge from interactions with the artefact in performance. These two complimentary processes have been discussed in relation to my own software artefacts in Chapters 4, 5 and 6 of this thesis. In seeking to harness this space as a useful site for practice-based research, in Chapter 3 I introduced Pickering's *mangle of practice* to describe how research outputs must be considered as an emergent interplay between both human and material agencies (Pickering 1995).

As noted by Pickering, through periods of negotiation between human and material agency a process of *interactive stabilisation* emerges between the practitioner and their apparatuses. This dynamic process of stabilisation causes the artefact, the user and the user's understandings of the domain of practice to emerge and redefine themselves through interaction (Pickering 1995). It is from within the negotiation between development and use that a performer-developer uncovers and refines scripts embedded within their software artefacts. For the performer-developer, a direct consideration of

material agency (both in design, testing and performance) extends the frame of reference provided by the designer's original intentions. Because this cultural and historical frame is known intimately (it is the very personal history of the designer), balancing the known and predicted output of the machine with emergent understandings of its capabilities become part of this stabilisation process. *Interactive stabilisation* is therefore an integral part of designing and performing with such software systems, given the ability of the performer-developer to alter the artefact as a direct result of previous interactions.

7.2.2 *Performer-developer context*

As suggested above, the notion of an artefact's *script* is complex when applied to performer-developer devised software artefacts. In this creative context, the practitioner often designs an artefact primarily for himself or herself as user, and one may therefore assume a tight correlation between the 'real' and 'virtual' users of the artefact being designed. Additionally, awareness of an artefact's developmental history is ever-present during performance practice, making it difficult to consider an artefact's inherent affordances as separate from the designer's intentions. The complex interaction between development and performance therefore precludes any understanding of this process through engagement with the artefact alone. In the performer-developer context, any virtual user projected throughout the design process is itself informed by feedback from the real user's experiences. Given the openness of this developmental scenario, it is suggested that both real and virtual users of such artefacts develop entirely in tandem with each other through an integrated process of negotiation.

Expanding upon McLean and Wiggins' conception of *bricolage programming* (McLean & Wiggins 2010), here the software development process is expanded to include feedback from the experiences of the developer interacting with their algorithms in real-time performance. For the performer-developer, performative testing is often inseparable from development. As discussed in Section 3.5, for McLean and Wiggins the process of observation may be considered a creative act in the programming process, as the perception of the result of encoded algorithms is a vital part of the creative feedback loop (McLean & Wiggins 2010). Expanded to include performance, observation becomes an interactive concept that acknowledges the real-time understanding of how a developer's algorithms work in practice.

Related back to inscription, here the real user assimilates both the conceptions of the virtual user as well as their direct experience with these algorithms in performance. Tracing through my creative practice, I believe that the separation between use, design and development aided in the creation of hybrid virtual/real users of my artefacts. The real user is itself an emergent part of this sociotechnical process, open to assimilating possibilities encountered in the design process at the same time as directly influencing this process through use. The design process, as detailed throughout Chapters 4, 5 and 6, therefore becomes a speculative position for the performer-developer. New avenues may be explored in development through quite tangential and accidental means, and specific biases in performance are only revealed during the act of performance itself.

In my practice designing interactive artefacts, the process of passing over from being a developer to user is a distinct and definitive shift in role. Whether testing the artefact in the studio or on stage, as user of my software I distance myself from my design history in order to work with the script revealed by the machine's affordances. This process might be thought as *suspending disbelief*, as the performer succumbs to the material agency exerted by the software during performance. This is a complex space where material agency interfaces with the history of the designer's decisions in the moment of performance. The dual roles played by the performer-developer are also further complicated through the agency exhibited by elements of the software being designed. As discussed previously, the enhancement of material agency is a fundamental guiding principle in the context of improvised human-machine performance. The developing relationship that exists between human and material agency should therefore be acknowledged as fundamental in shaping the way in which such artefacts evolve.

This process has been apparent in my performance practice with the *_derivations* software. Any interaction I have with the software is naturally imbued with an awareness of the software's affordances, given how intimately the various underlying processes driving the system's musical contributions are known to me. However, given the emergent and unpredictable nature of various processing and structuring parameters in the software, my role as a performer is to navigate the agency displayed by the machine in the moment of performance. This agency is naturally evident in the specific algorithmic decisions made by the software at any given point during an improvisation, the majority of which remain opaque to me as a performer. Although the algorithmic

generation methods used in *_derivations* do not exhibit high levels of creative autonomy as defined by Bown and Martin (2012), the precise combination of aleatoric and deterministic processes at work provide enough surprise and unpredictability to allow the performer to focus upon navigating the software's inherent affordances. Thus, the real-time interaction is therefore framed by the opacity sought in the development process. Whilst certain details of the machine's performance may be clearly understood by a performer, the global output also remains complex and unpredictable, ensuring a performance remains dynamic from moment to moment. This performance practice is therefore intimately concerned with blurring the boundaries between the known and the unknown, the predictable and the unpredictable.

7.2.3 *An artefact's 'episteme'*

Finding a balance between determinacy and indeterminacy is a common part of software design in computer music contexts. As suggested above, the continual shift between projection and interaction, between encoding and observation is an integral part of the creative practice of the performer-developer. Hamman has sought to understand the processes at work in the development of software artefacts exhibiting dynamical properties, focusing specifically on the dance of agency that occurs between the user and the machine's inherent affordances (Hamman 1999). For Hamman, an important distinction is to be made between artefacts that enable music making through use, and those that engage a user to contemplate the usage context itself during an interaction. In the former, the user employs the artefact as a transparent means through which to achieve an outcome, whilst in the latter, the artefact itself comes into sharp focus, forcing a consideration and navigation of its affordances. The author details an approach to musical interface design where the composer's role is redefined towards interacting with and navigating the task environment itself, as opposed to composing music *through* a task environment. (Hamman 1999).

Although Hamman describes software artefacts designed for compositional purposes, his ideas are especially relevant to interactive performance systems. The separation between a transparent tool and a dynamical system recalls Rowe's separation between systems conforming to either an *instrumental* or *performance paradigm* (Rowe 1992). Seeking to understand interaction in these scenarios, Hamman distinguishes between two

overlapping dimensions of human performance in interaction with a 'mechanism' (artefact) – that of an *action* and a *description*. An action is that which "can affect change within an environment", when coupled with an artefact. Made by a user, an action is made in order to "alter the state of the mechanism, and thus its outcome" (Hamman 1999, p. 94). A *description*, by contrast, defines how the user understands the relationship between an action and its outcome. A description allows the user to hypothesise a mapping between action and outcome, which Hamman describes as an "internalised framework that determines our actions and observations regarding our use of some mechanism." (Hamman 1999, p. 94)

Hamman explains that descriptions are formed historically, both culturally and through individual experiences with familiar artefacts. For familiar artefacts, a user's understanding of action-outcome relationship has been formed prior to an interaction, whether through personal experience or cultural understanding of the artefact's affordances. For the unfamiliar artefact, it is the user's interaction with the artefact that informs this description through use. Hamman introduces Focault's concept of *episteme* here to situate the description in relation to the unfolding of an epistemological frame. Developed in *The Order of Things*, Focault's concept of the *episteme* denotes the structures that underlie the production of knowledge during a particular epoch, or the grounds upon which a statement can be counted as knowledge (Foucault 2005). In this context, Hamman describes the *episteme* as the way in which "a mechanism, within an interaction, comes to make sense through description." (Hamman 1999, p. 95) Regardless of whether the description has been developed through cultural precedent or individual experience, this historically situated understanding of the action-outcome relationship provides the grounds upon which a user understands the outcomes of an interaction.

Hamman draws a distinction between Focault's *episteme* as either open or closed, seeking to establish a basis for the development of interfaces that challenge the traditional notion of a usable, transparent tool. According to Hamman, a closed *episteme* is "deeply coupled to the cultural/technical program according to which the mechanism is designed." (Hamman 1999, p. 95) Such a 'program' informing design and development has been developed through historical precedent and defines the boundaries of an artefact's affordances. What Hamman suggests is that the usage context of the artefact (defined by the *episteme*) shares this technical and cultural frame of reference. As a result,

the user's expectations of the outcomes of an interaction are in line with the designer's specifications. In other words, the artefact makes sense to its user due to its familiarity; the user's descriptions of the artefact are derived from the same cultural and historical precedents referred to by the designer.

By contrast, an open *episteme* is not wedded to such historical frames of reference. As Hamman suggests, an open *episteme* is one in which the particularities of the domain in question define the usage mode of the artefact, therefore beginning to establish a frame of reference for its use. As Hamman explains, the *episteme* is "*porous,* open to input from the particularised situation." (Hamman 1999, p. 95) Characterised as a 'disruption' of a historical frame of reference, the user is placed in a particularly interpretive position whereby developing understandings of the artefact's affordances are derived from the present interaction. Descriptions are formed through interaction with the artefact itself, as they cannot be intuited prior to an encounter with its affordances. As *open* therefore, the artefact's *episteme* forces a direct engagement by the user with the artefact's particularities, thus encouraging new modes of interaction. As noted by Rose and Jones (2005), any interaction between a human and a machine is a situated process in which the human's personal history of interactions influence subsequent interactions. In the case of the unfamiliar artefact, the lack of personal history frames the initial interactive encounter. Each subsequent interaction builds upon this initial experience, helping to define a developing interactive relationship between human and material agency.

Hamman's understanding of the *episteme* in this context can therefore be conceived of in relation to Akrich and Latour's *script*. As closed, the *episteme* enables a tight correlation between the designer's inscription of the virtual user, and a real user's descriptions of the artefact. Both real and virtual users are aligned as the user approaches the artefact with an understanding of the interactive paradigm that matches the designer's ideal usage context. By contrast, the open *episteme* evolves along with input received from the user's understandings of the new interactive domain. The designer may have inscribed a virtual user into the object that is far removed from the experiences of real users, or the user is unsure as to their role as user at all. In such a context, the user's understanding of the artefact forces stronger engagement with the material agency of the object itself, consequently contributing to the artefact's *episteme* through use.

In the design and use of performer-developer devised artefacts, I would suggest that an artefact's *episteme* moves between being open and closed. To begin with, the developer's awareness of their artefact's internal structure immediately connects the user to the historical frame of reference for its use. This inescapable fact places the artefact's *episteme* as closed in this respect. However, the underlying complexity resulting from the interactions between various generative components in the software ensures that new understandings about the artefact's potential are continually gathered. Importantly, such cumulative understandings require nuanced and repeated interactive encounters with the artefact in performance. This scenario describes an artefact whose *episteme* remains open, as each experience with the software artefact contributes to a working understanding of the affordances of the artefact in performance. The various generative algorithms employed in the software become part of the performer-developer's growing understanding of the artefact's affordances. Previous design decisions regarding the choice of algorithms, processing and synthesis parameters manifest themselves through the agency of the machine in performance. In other words, the space in which a tool is purposefully left open enables new descriptions to be formed. These descriptions contribute to new understandings of the design space, ultimately feeding back into future interactive encounters.

Considering the *episteme* of performer-developer devised artefacts is useful in understanding the role played by material agency in this creative context. Through an intentional thwarting of the natural connection between development and use, genuinely surprising interactions between the performer and their artefacts can occur. In reference to the software designed in this project, these concepts help to explain the tension inherent in my development process. The exploratory nature of my design approach may be understood as an open-ended search for surprise and novelty in performance, achieved by balancing deterministic and indeterminate approaches. The continual preference for sampling-led methods of generativity directly challenged my performative desire for unpredictability and surprise. In *Tripartite Markovia* (see Section 4.2.4.2), the depth of understanding I had about the Markov generation approach was not a hinderance to that system's dense and unpredictable performance dynamic. Indeed, the unpredictability of response of this particular system forced a re-consideration of the notion of performer control, eventually implemented through temporal synchronisation methods. By balancing predictability and unpredictability, complexity and coherence,

design decisions were made that forced part of the artefact's *episteme* closed and predictable, whilst the specifics of the musical generation allowed the *episteme* to remain open.

In the *_derivations* system, a deterministic phrase matching approach was tempered by aleatoric methods of processing and structuring material siphoned from the performer. This dichotomy forced me into a search for equilibrium in both performance approach and design methods in my practice. Paradoxically, by implementing the *session database* concept discussed in Section 6.8, I engaged more fully with the cumulative affordances of the software whilst simultaneously decreasing the possibility of repetition and control mastery from this process. The more the system's sonic vocabulary was pre-defined in advance of a performance, the more possible it was to distance oneself from the system's cumulative potential as exploited in previous approaches. Although a large session database may contain material that is very familiar to the performer, predicting the precise timbral connections between these materials becomes more difficult the larger the database. In addition, by disabling 'Analysis Storage' (see Section 6.8.4) the lack of a cumulative connection to the current performance created further unpredictability between the performer's actions and the system's response. The fixed database therefore represented a familiar yet opaque space of possibilities. This inherent opacity helped to drive interactions further into territories unpredictable outside of an interaction. Another example of this increasing opacity was the creation of the *self-referencing* triggering approach (discussed in Section 6.6). By eliminating a direct connection between the temporal gestures of the performer and the system's output, I successfully distanced myself once more from known affordances of the software in favour of the opacity provided by the self-referencing algorithm.

7.2.4 *Stabilised and non-stabilised artefacts*

Akrich and Latour's ideas concerning the role of technical objects in networks hinges upon the distinction between *stabilised* and *non-stabilised* artefacts. *Non-stabilised* artefacts are those considered to be still within a development and innovation phase, they are artefacts for which meaning is still emerging. These artefacts are continually being adjusted and re-defined according to input from developers, end-user testing and so on. By contrast, *stabilised* artefacts are those artefacts that have exited this innovation network

and entered the real world to be made use of. According to Akrich, stabilised technical objects can be considered 'instruments of knowledge' (Akrich 1992, p. 221). In addition, the author explains that in order for us to access knowledge through these objects, one must be able to engage in a process of black boxing:

> the conversion of sociotechnical facts into facts pure and simple depends on the ability to turn technical objects into black boxes. In other words, as they become indispensable, objects also have to efface themselves.
>
> (Akrich 1992, p. 221)

From Akrich's point of view, stabilised technical objects enable us to learn something about a sociotechnical context as their complex, interrelated inner workings have been made invisible. The author explains that for non-stabilised technologies, we are able to draw links between the technical content and user through a *mediator*, such as the innovator of the artefact. In the case of technological artefacts under development, both the innovator and user's viewpoint aids in moving the technology towards stabilisation. They can also help us glean useful insights into the development of design assumptions and the design process itself. However, for stabilised technologies that have been black boxed, the innovator is no longer present and the ordinary user has already taken on board the general prescriptions implied in interaction with the machine (Akrich 1992, p. 211). In such cases, the black-boxed object therefore serves as an instrument through which knowledge can be accessed about its sociotechnical context.

The author's distinction between *stabilised* and *non-stabilised* technologies references a type of technological development in which the artefact is deployed to *separate* end users. Before becoming stabilised, the innovator is present and an integral part of the process. The artefact is part of the innovation network, identifying the technology as fundamentally open therefore not yet black boxed. In this scenario, the aim of the innovation network is to move the technical object closer to being stabilised and deployed into a defined usage context. For Akrich, black boxing describes the ability of a technology to act as an instrument of knowledge, enabling the transcendence of what is knowable from inside this innovation network itself. In order for the researcher to understand the implications of the technology in its usage context, the inner workings of

the artefact and the complex and heterogeneous elements of the innovation network must fade into the background.

As we have seen, there exists in the performer-developer context a unique connection between the innovation and usage networks of development and performance. The kind of knowledge attainable from this developmental context differs from that which is available by analysing the deployment of stabilised technologies. In my practice, the inscription of an ideal usage context is an interactive and emergent process without a defined end-point or goal. The ideal user is itself emergent, and a performance practice that surrounds a continuously developing artefact influences the conception of design. Here the usage context shifts and changes along with the emergent conception of the ideal user. In addition, the process of black boxing is something that occurs throughout both the innovation and usage network, helping to define and shape the final outcome of the development process. The two networks themselves are not as clearly defined as actor network theory might suggest, as they continuously blur into one another.

However, the concept of black boxing remains relevant to this scenario. As discussed in relation to Hamman's ideas, the relationship between *performer-as-developer* and *developer-as-performer* enforces an artefact's *episteme* to be both open and closed. With respect to black boxing, here I argue that the status of the artefact as a black box depends entirely upon the context in which the artefact is used. Following on from Akrich's description, the performer-developer engages in a process of black boxing during performance with their artefact in order to *acquire knowledge*. Black boxing the specifics of their designs is a way of engaging in a performative dialogue with the results of material agency. In my practice, interaction with *_derivations* through both testing and live performance can therefore be conceived of as a form of knowledge acquisition. Although the broader socio-technical context may not be directly interrogated, the act of black boxing a technology in performance enables the performer to ask questions and interrogate accepted norms in interactive and improvised performance. Performance and development can therefore be conceived of as a form of knowledge acquisition that can only be achieved through the process of *intentional* black boxing. It is an intentional hiding of complexity in order to engage with the artefact through use, and to project future hypothetical uses during development.

7.2.5 *Attributing agency*

The complexity of the process of black boxing results not only from the relationship between design and use of technical artefacts, but also from the inherent complexities of these software artefacts themselves. As described above, turning an artefact into a black box enables it to exert agency in a network. In the usage network, an interactive performance system can therefore be said to exhibit agency if it can successfully be black boxed as a *stabilised* artefact. However, in the performer-developer context, this stabilisation Akrich refers to is not permanent. Returning to Hamman's ideas, an artefact's *episteme* is left open for input from the performer's experiences, and the connection between the dual roles occupied by the practitioner preclude the artefact from being completely stabilised. Continually shifting between open and closed, the *episteme* of such artefacts therefore helps to explain an artefact's status as stabilised or non-stabilised.

In actor-network theory, the concept of the *hybrid actor* defines the complex interrelations between the various parts of a network acting as one. For Schulz-Schaeffer, Latour's concept of the *composite* or *hybrid* actor is further complicated by the uneven relationship that exists between the various elements that constitute this entity. The difficulty of understanding the relationship between the human and the artefact is where to attribute *authorship* to an action, and at what point the hybrid actor itself may be said to exhibit agency. According to Schulz-Schaeffer, "the more comprehensive the list of actants becomes which contribute in one way or another to the action in question, the more difficult it is to conceive the association of all these as one actor." (Schulz-Schaeffer 2006, p. 135) A hybrid actor is therefore a product of the interaction between these constituent elements, some of which may be human, and others non-human. In the context of performer-developer designed software artefacts, the concept of a hybrid actor necessarily takes into account the developer's role in the artefact's creation, the desired response of the system and other elements relevant to the design of the artefact.

Designing a plausible interactive space for machine agency to exert itself is a core challenge of this creative endeavour. In order to maintain a dynamic and unpredictable musical interaction, the performer must *attribute* agency to the machine's actions. In the design of interactive systems, the hybridity of the software artefact is displayed through

the various analysis, generation and processing modules in play as well as the developer's role as programmer of the system. In the context of performer-developer devised artefacts, the hybridity of the artefact can increase the artefact's perceived agency in a performance. Rose and Jones (2005) have noted that the process of agency attribution is unique to human agency in a human-machine interaction. Humans attribute actions, causes and outcomes to actors within an interaction, even if this attribution is erroneous. Given this, the design of interactive systems may involve a certain level of obfuscation and trickery. For the performer-developer, this trickery helps to obscure the underlying processes at work in their software. In the *_derivations* system, the various processing and structuring algorithms at work obscure the deterministic process of *phrase matching*. Whilst the broad timbral context of the musician's current performance is predictable, the generative processes of each processing module manipulate the temporal profile of the retrieved phrases that the musician then interacts with. Whilst the process by which the machine decides upon its core material may be transparent, the means by which this material is used opaque and unpredictable. Acting as an automaton, the machine is therefore able to invoke unpredictable temporal gestures from a somewhat predictable analytical process.

7.2.6 *Models of 'invisibilisation'*

For Schulz-Schaeffer, black boxing means "that the complex interrelatedness of the many actants contributing to the overall programme of action becomes invisible" (Schulz-Schaeffer 2006, p. 140). This concept of *invisibilisation* is at the heart of black boxing, and the ability of actors to navigate and make sense of technical objects and networks. The two different models of invisibilisation proposed by Schulz-Schaeffer are the *encasing model*, and the *outshining model*. Schulz-Schaeffer explains that the *encasing model* describes components of the artefact that are made invisible, as they are not relevant to the interaction between a human actor and the hybrid actor. The examples of the inner workings of a bank as a corporate hybrid actor, as well as the insides of a piece of technical equipment are used to define this model. In the *encasing model*, only when there is a break down in the system do the various parts of a hybrid actor reveal themselves, as they have until now been hidden from view (Schulz-Schaeffer 2006).

By contrast, the author describes the *outshining model* as a situation whereby one actant is perceived to outshine the other actants that make up the hybrid actor, whilst still maintaining their presence as perceived and understood parts of the network. As Schulz-Schaeffer explains, the contributions of most of the actants "become invisible because the actant to which actorhood is attributed diverts attention away from them." (Schulz-Schaeffer 2006, p. 141) In contrast to the encasing model, the hybrid actor is engaged with as a black box not through the irrelevance of the other actants in the network, but because their status as actants need not be brought to the foreground. They are therefore intentionally made invisible.

With respect to *_derivations*, there are various levels of this invisibilsation process that occur both for myself as performer-developer, but also from the perspective of the uninitiated performer interacting with the software. This brings forward the notion of authorship in such performative encounters (a concept I explore in more depth in Section 7.3). There is a wide network available to be in engaged with *outside* of performance time, however in performance time this recedes in favour of the performance-time encounter. I would argue that there are elements of the *_derivations* system that are black boxed by way of Schulz-Schaeffer's encasing model, and others that follow his outshining model. Importantly, both of these understandings of *_derivations* as a hybrid actor vary in relevance depending upon who is interacting with the system. For myself as a performer-developer, I am familiar with the vast majority of software code that contributes to the software's autonomous potential. Consequently, the type of black boxing I engage in is an example of the *outshining model*, as I am fully aware and cognisant of a great deal of the actants that make up the performance and development networks. When on stage interacting with *_derivations*, although I am aware of the various interrelated actants that make up the software, I allow these to recede into the background in order for the machine's agency to come to the fore. The machine becomes a single hereogeneous actant with which I engage.

For the uninitiated performer interacting with *_derivations*, the *outshining* model of invisibilisation may also explain some of their interaction with the machine. There are indeed certain elements in the performance network that the performer is aware of that fade into the background as irrelevant at the time of performance. These might include my personal role as the author of the software, the microphone amplifying and feeding

an analysis of the sounds of the instrument, the computer's role of storing the information it hears for later use, the system's capabilities of combining previous performances together, the distribution of the software via the website, etc. In addition however, these performers also interact with *_derivations* as a hybrid actor with many actants made invisible through the *encasing* model. This type of black boxing describes a typical understanding of human-machine interaction in which many details of an artefact are irrelevant to the user, and are therefore not communicated and made invisible. For *_derivations* this list would be exhaustive, however such details would include the source code of the software itself, the inner workings of the output modules, the type and rate of the audio analysis that is taking place, the manner in which files are stored and accessed on disk, along with many other things.

As argued above, interaction with technological artefacts necessitates the user to place some known details about the artefact to one side, and to engage with those that can most easily be attributed *actorhood* and *authorship*. In addition, I have argued that the process of black boxing is present to varying degrees depending upon who is interacting with the software, and with which network the user is interacting. Considering black boxing as *invisibilisation* allows us to understand the role of the performer in navigating the inherent affordances of an interactive software system of their own design. Schulz-Schaeffer's *outshining* model is most relevant in describing the intentional process that occurs in this creative practice. The agency of a software artefact in this scenario is contingent upon the ability of the performer as developer to *suspend disbelief*, but also to allow the various components of the software to recede into the background during an interaction.

7.2.7 *Conclusion*

Throughout the present section I have sought to understand the entangled relationship that exists between development and performance in the context of performer-developer devised software artefacts. By interrogating relevant theoretical concepts as they relate to this practice, I have highlighted a number of specific challenges to understanding the role of material agency in both development and performance. The ability of a software system to exhibit agency is dependent on a variety of context-specific factors. The concepts of *script* and *black boxing* derived from actor-network theory go some way to

explaining the ability of a software artefact to exhibit agency, however these concepts are challenged by the unique and hybrid nature of artefacts produced by performer-developers.

Pickering's concept of *interactive stabilisation* provides a useful understanding of the emergent nature of both development and performance. As highlighted earlier in this thesis, understanding creative practice as a *dance of agency* became a specific methodological consideration in this research. In addition, Pickering's ideas have also helped to further an understanding of software agency in human-machine interactive performance. The process of black boxing has been outlined as forming an integral part of understanding the agency exerted by software artefacts in this context. As suggested above, black boxing as *invisibilisation* is a necessary and intentional act of the performer-developer interacting with their systems, allowing for the suspension of disbelief in interactive performance practice. McLean and Wiggins' concept of *bricolage programming* also acknowledges this intentional shift in role between developer and observer of an algorithm that takes place in such creative contexts.

7.3 Interpretation in improvised human-machine performance[52]

The development of interactive performance systems is an active area of research in the field of live electronic music. Whilst the various models and metaphors discussed in Chapter 2 help to define the boundaries of this practice, the engagement of these systems in improvised performance remains somewhat under researched. As is the case for many emerging artistic practices, sustained critical insight on practice is often carried out by practitioner-researchers directly engaged in the artistic domain in question. These practitioners are at once developing artistic works, engaging collaborators in their creative practices and publishing about their specific technical, artistic and theoretical concerns. New knowledge derived from such practice-based approaches must be understood as arising from this entanglement between research and practice. In order to understand performance practices emerging from the development of interactive software artefacts, the specific relationships that exist between developer, performer and software must be taken into account.

In the previous section, the performer-developer context has been interrogated to further understand the role of material agency in shaping both development and performance practice. However, by expanding the theoretical lens to include interactions between artefacts and performers from outside of the innovation network, a new set of conceptual challenges arises. Placing an improviser in performance with an autonomous or responsive software system provokes significant questions about the notion of artistic authorship in this practice. In these scenarios, improvising musicians are often invited to participate in a performance by the system designer, who is invariably present for rehearsals and performances with the software artefact. The possibility for a performer to *suspend disbelief* in performance is therefore influenced both by their interactions with the software artefact and system's designer. Given this important aspect of the creative practice, here I argue that the term 'musical interpretation' warrants discussion as it relates to the performance practices that emerge from the development of these

---

[52] The ideas presented in the current section have been galvanised by some fruitful discussions with a number of fantastic musicians and researchers. My thanks go to Owen Green, Oliver Bown, Bill Hsu, Mark Summers, Aengus Martin, Arne Eigenfeldt and Pierre-Alexandre Tremblay for taking the time to discuss this topic.

artefacts. Throughout a consideration of the notion of the musical *text,* in the sections that follow I situate such creative artefacts within the context of interpretive musical practice.

7.3.1 *Free-improvisation and interpretative performance*

In a freely improvised performance, improvising musicians develop musical structure in real-time without the presence of an overarching compositional framework. The real-time interactions between one or more musicians dictate both the materials and trajectory of the musical encounter. Also dubbed *non-idiomatic* improvisation, this type of performance distances itself from rules of melody, harmony and form in favour of free expression (Bailey 1993). The most fundamental and defining feature of this type of improvised performance is the contribution of the individual performers to the musical outcome. From a freely improvised perspective, improvising musicians are free to make their own decisions on how to affect the musical discourse, without having to defer to a set of instructions that might constrain these decisions. Saxophonist Evan Parker outlines this type of musical activity as his 'ideal music' (Bailey 1993, p. 81), a form of music making that not only eschews the presence of a pre-defined musical text, but also places specific emphasis on the reliance upon the interpersonal relationships between like minded musicians. These musicians improvise "freely in relation to the precise emotional, acoustic, psychological and other less tangible atmospheric conditions in effect at the time the music is played." (Bailey 1993, p. 81)

This practice contrasts with interpretive practices in which the instructions provided by a composer govern music making, mediated through a musical *text*. In interpretive practices, the musical ideas of the composer are codified in a musical score, stated in a textual description or transmitted orally to the performer. Although these practices require a degree of improvisation in the performative realisation of a musical work, the performance itself partly exists as a *rendering* of the musical ideas communicated by this text. Even in traditionally improvised musical contexts such as jazz, the presence of a musical text or *referent* structures the improvisational activity, along with the melodic and harmonic conventions of the musical style (Pressing 1988). Given this dichotomy, the term 'interpretation' may be considered antithetical to the goals of freely improvised musical practice. In this context, the notion of interpretation as requiring something to

interpret – that is, a pre-existing 'work' communicated via a musical text – does not exist in any traditional sense. However, by introducing autonomous and responsive musical systems into a previously human-only performance scenario, the nature of both improvisational practice and the identity of these systems come into question.

It is important therefore to establish exactly what is meant by interpretation in this context. The Oxford English Dictionary defines musical interpretation in part as "…the rendering of a musical composition, according to one's conception of the author's idea." (Oxford English Dictionary 2015a) In Davies and Sadie's discussion of the term, interpretation refers to "the understanding of a piece of music made manifest in the way in which it is performed." (Stephen & Stanley 2015) For the purposes of the present discussion, here I define interpretation as a 'bringing to life' of the ideas communicated by one or more authors in the form of a real-time performance. This form of practice is common to all performing arts, including dance and its relationship to choreographers, theatre and film's playwrights and screenwriters, and finally to music and its composers. These definitions consider the presence of an author, the existence of a musical work and the decisions made in relation to the author's ideas as communicated in this work.

For music as for other art forms, central to the notion of interpretation is the medium in which artistic ideas are transmitted, the musical *text*. In musical interpretation, the performer must make decisions about how to successfully bring to life a composer's ideas within the constraints communicated by this text. Musical texts take a variety of forms, from traditional western notation to graphic notation, textual instructions as well as verbal and gestural instructions. A musical text might be fastidiously prescriptive by offering detailed symbolic instructions for the performer to follow, such as the complex musical notation of Brian Ferneyhough or the tablature notation of Aaron Cassidy (Rutherford-Johnson 2011). Other types of text outline a framework that guides a performer's musical decisions, such as in the textual scores by La Monte-Young: "Draw a straight line and follow it" (Young 1963), or the instructions found in the works of Christian Wolff: "Make a sound in a middle place, in some respect, of the sounds around it." (Wolff 1964) In addition, in so-called *game pieces* such as John Zorn's *Cobra*, the musical text is manifest as a set of specific interactional rules followed by a conductor and a group of improvising musicians (Zorn 1984).

Regardless of the medium in which they are communicated, musical texts contain not only instructions for the performer to follow; they also express context-specific assumptions about the role of the performer in bringing the work to life. Let us consider a few examples to illustrate this point. The *basso continuo* line in music of the 17th and 18th centuries assumed a level of contextual understanding about harmony on the part of the performer, as the musician was required to improvise a chordal accompaniment to a melodic line. *Tablature notation* is a form of notation that directs the performer towards the execution of physical action as opposed to sounding result. This form of notation implies strict adherence to the physical instructions of the notation, in contrast to the symbolic representation afforded by traditional western staff notation. *Graphic* notational practices often rely heavily upon the subjective interpretive faculties of musicians, leading to a large space of possible renderings of a musical work. Finally, *textual* scores like those of La Monte Young and Wolff focus upon providing performers with a musical and interactive framework with which to construct the musical work. Such texts may direct the performer in the course of some pre-defined action, but may also leave out the specific means by which the action is achieved. These texts may also be relational and interactive, provoking the performer to respond to the unexpected results of previous actions, or to their environment.[53]

7.3.2 *Freedom and constraint*

Despite their disparate manifestations, the above forms of musical text all share the same core attributes. All of these types of musical text articulate the desired boundaries of a musical performance by implying various levels of musical *freedom* and *constraint* for performer(s) interpreting the work. They all involve some combination of explicit and implicit constraints from within which the musician makes decisions as to the best rendering of the work. For Coessens (2013), on one extreme the musical score can be seen as a form of direction or command. The author describes instruction as a target-oriented process, in that it "invites action to realise a fixed goal." (Coessens 2013) At this extreme, a score as instruction becomes analogous to a set of commands pointing towards an expected outcome. The completed process is a result of these commands, and as such any deviation will change the expected outcome. On the other hand, the

[53] The recent works of Scott McLaughlin are a good example of this textual, algorithmic approach to the musical score: http://www.lutins.co.uk/

process of artistic expression is only partly target-oriented, as the interpretation of a musical text foregrounds the score as "material for performative action," where value is to be found in the process of a musical narrative rather than a final product (Coessens 2013). This balance between freedom and constraint is therefore evident in the process of interpreting the musical text. Indeed, as Sarath has noted, even in the most prescriptive cases the freedom afforded a performer in interpreting a musical text may be seen as a type of improvisational practice:

> While interpretive performers do not change the pitches or rhythms delineated by the composer, they certainly do deconstruct personal interpretive patterns in seeking spontaneous renditions of pieces they have already played countless times. Interpretive performance might then be seen to involve temporal principles similar to those defining improvisation within a highly detailed referent.
>
> (Sarath 1996, p. 21)

Understanding a musical text as a set of explicit instructions provided to performers, the term interpretation therefore implies a balance between fidelity to these instructions, and the injection of significant performative and stylistic understandings of the composer's intentions. The relationship between *adhering to* and *interpreting* these instructions is dependent upon a variety of factors, which may include input directly from the composer during rehearsals, stylistic norms of the musical period in which the work was written, and the cultural conventions of contemporary performance practices. In all of these cases, interpretation refers to the process by which any gaps in communication between composer and performer are made sense of in light of supplementary knowledge pertaining to the work in question. In this respect, the musical text communicates the *requirements* of the work as articulated by the composer, whilst the musical and cultural context in which the work resides provides further *constraints* on any reasonable and/or correct interpretation of it in performance.

Discussing the role of the score and of the composer in contemporary music-making, Parker posits that a musical score might be considered as either one of two contrasting types of text: in some cases as a representation of an 'ideal' performance, and in others as a "recipe for possible music-making." (Bailey 1993, pp. 80-1) Speaking as an improviser, Parker's opinion on the role of the composer and the prescriptive nature of some musical scores is articulated in his characterisation of the composer's role as *score-maker*:

"… if anyone in the production of a music event is dispensable, it is the score-maker, or the 'composer' as he is often called." (Bailey 1993, p. 81) Although the perspective articulated here is largely negative, Parker notes that the more a musical score resembles a 'recipe for possible music making', the more there is a gap between the score as 'ideal' and any live performance of it.

To Parker, the more unequivocal a text appears as a form of instruction, the more the score resembles the ideal performance envisaged by the composer. In such scenarios, with the addition of a weight of cultural conventions associated with the text, less interpretive flexibility is offered to the performer(s) in creating their own personal *reading* of it. The text is no longer a recipe, but a prescription. By contrast, graphic scores, as more ambiguous forms of musical text, are often provided to performers with no accompanying instructions or performance direction. Although such scores may not communicate explicit performance requirements of the work to the performer, these objects still exist as forms of musical text to be interpreted. Graphic scores range from a set of symbolic instructions allow for varying degrees of freedom within specific musical parameters, to objects of immense interpretive flexibility, such as Cornelius Cardew's *Treatise* (Cardew 1967). In the latter case, the notion of interpretation rests upon the performer developing explicit rules and procedures in order to realise the musical work. Arrangements of graphic symbols such as those found in *Treatise* can be seen as the catalyst for the creation of a personalised musical and sonic grammar in response to the text. Rather than acting as direct instructions, these scores provide sets of constraints that anchor a musician's interpretive decisions; choices that are guided both by their own personal musical vocabulary, as well as the cultural and stylistic context of experimental musical performance.

7.3.3 *Extra-musical constraint*

For Parker, it is within the gap between the explicated ideal and the live performance that the real-time act of musical performance resides, leaving open for consideration further context-specific 'ingredients' of musical performance in addition to the instructions outlined by the composer. Such ingredients may include the choice of specific musicians to perform the work, where the work will be played as well as how the audience might react (Bailey 1993). Speaking of the difference between *requirements* and *constraint*s in the

context of musical performance, Bown has noted that performances of graphic scores may also include forms of extra-musical constraint that should be considered either directly curatorial, or at least requiring a form of interpretation by a group of musicians (Bown 2014, pers. comm., 28 September). Such constraints might include the decisions and wishes of a concert promoter (venue decision and therefore acoustics, performers asked to 'keep it short'), discussions regarding musical/sonic materials between players, to the presence of 'garish paintings' hanging on the walls of the concert space. Such instructions and environmental constraints may require a form of interpretation on the part of the musicians that will directly or indirectly affect the resultant musical outcome of the performance. As Bown implies, such constraints often also apply to freely improvised music, potentially affecting the way in which musicians interact with each other on stage, despite the lack of a unified performance framework as embodied in a score.

The question however is whether such extra-musical constraints, acting as objects of interpretation, should be considered part of an interpretive framework on par with the constraints provided by a graphic score. My answer to this question rests upon an understanding of composition and curation as intentional activities of persons acting outside of the real-time context of musical performance. The framing of a performance context might be achieved through the existence of explicit requirements (notes on a score, imposition of formal boundaries etc.), but equally through placing musicians in a performance context that might constrain or alter their natural performance dynamic in some intentional way. However, the inherent collectivity and self-organisational properties of improvised music making, free of intentional constraints on musical structure and materials, preclude it from being understood as an interpretive activity in this sense. Because of a decided lack of musical text to be interpreted, any extra-musical constraints that might influence an improvised performance should be considered part of the social and cultural context of improvised performance itself, not as intentionally imposed as objects of interpretation.

Musicians engaging in improvised performance bring with them their own personal histories of engagement with the practice, and although the boundaries and constraints of improvised practice invariably change from performance to performance, these form part of the context of improvised performance that has developed over time. This

context both normalises and frames the performance activity itself. Although external constraints may be 'interpreted' by a group of improvising musicians and subsequently affect the musical discourse, these constraints should be seen as manifestations of the context in which an improvisation takes place. They are not part of a set of intentionally curated decisions in which the musicians must engage. The social and cultural context enables improvisatory music making to occur, rather than directing it as such.

7.3.4 *Interactive systems and improvisational performance*

Acknowledging these specifics of improvised performance practice, we turn to the inclusion of interactive and autonomous systems into this performance scenario. The question we must ask here is this: is an improvised musical performance fundamentally different with the added presence of an interactive musical system? To answer this question we must concern ourselves with the ontology of such systems, and the way in which musicians may perceive them in performance. With the ability to generate independent musical material as well as to listen and adapt to their performance environments, it might not be so unusual to think of such systems as 'virtual performers' in their own right. In some respects, one might argue that the presence of such systems in performance, although synthetic, does not change the nature of the musical context given that a successful system may act in the same way as another musician. From this perspective, the presence of such an autonomous machine might form part of any 'less tangible atmospheric conditions' Parker has referred to – its status as a machine contributor resting as a mere technicality in the context of freely improvised performance.

Contrary to this assumption, I would suggest that the presence of such systems alters the way in which the musical context is perceived in performance, thereby changing the nature of the practice itself. The presence of a generative, responsive or autonomous musical system brings into focus the relationship between system designer and any human musician engaged in this practice. Here I take the view that such types of human-machine performance take place as part of a unique form of *sociotechnical curation*. The presence of the autonomous machine constrains and alters the performance practice in some intentional way, due to its existence as a programmed entity. It is precisely because of the specific creation of a non-human actor to be engaged with in performance (i.e. the

software) – a context traditionally reserved for human musicians – that the dual notions of interpretation and constraint in performance become relevant. However, at issue when relating these concepts to human-machine improvisation is the notion of a 'work', the detachment of the developer from the instantiation of the work in performance, and the ambiguity of the software as a form of musical text. Questions we may ask therefore are where the work resides in human-machine performance, whether or not we can attribute the proposed musical framework (inclusive of the software) to a single author, and whether or not the software itself may be considered a musical text.

In human-only improvisational contexts, when musicians encounter surprising musical materials and forms in performance these can be rationalised in terms of the skillset and abilities of their decidedly human counterparts. Human musicians approach an ensemble improvisatory context aware of the both the musical potential and cognitive faculties of their fellow human interlocutors. However, the same cannot reasonably be said of human-machine contexts. Any relationship that develops between a human and a computer system during performance might be characterised as one of navigation, exploration and discovery. During performance-time, the system's interactive and sonic behaviour promotes a mode of interaction whereby the constraints of any given system are revealed during the act of performance. A system's capabilities, as experienced by the musician in performance, manifest themselves as both musical and interactive constraints on improvised performance. The more a musician spends time with such a system, the more such constraints are revealed. In this respect, initially unfamiliar performance paradigms can be conceptualised as a form of material with which the human improviser engages.

In Bown et al., the authors suggest that performers engaging with such systems do so to varying degrees of appropriateness to the inherent capabilities of designed systems (Bown et al. 2013). The disconnection between performer expectation and system response is highlighted as an issue in assessing the effectiveness of such systems in performance. In addition, experiences with musicians also varied regarding their level of interest in the capabilities of the machine prior to a performance. Consequently, the degree to which performers understand the intricacies of the machine they are to be interacting with will vary widely. To avoid over-complicating the performative scenario, the authors suggest asking a performer to play 'naturally', where the machine is not a

"cause for the musician having to perform, interact or behave in novel ways." (Bown et al. 2013) However, even if a musician is advised not to try and provoke a direct response from the software, certain interactive preferences are revealed to the performer prior to an interaction. In the context of freely improvised musical performance, spontaneity and the ability to provoke and surprise are important elements of the performance practice. The choice not to reveal the underlying capabilities of a system can be seen as a manifestation of such an attitude to performance. In addition, a preference for non-reactive and causal modes of performance further communicates to the musician the aesthetic and interactive preferences of the designer. The assumptions and expectations of both improvisation style, as well as a musician's ability to act as if the machine is human are further revealed.

7.3.5 *Development as sociotechnical curation*

As alluded to above, the machine's existence as a programmed entity highlights the role of the human actor responsible for its design. In this performance practice, it is first and foremost the system designer who proposes the musical scenario of human-machine performance to improvising musicians, either explicitly (via invitation) or implicitly (through software distribution). Regardless of how this proposition is made, the system designer in this context, acting as *author*, proposes a musical framework to be navigated in performance. The placement of one or more human improvisers in such a performance context should therefore be understood as a non-trivial act of *curatorial authorship*. The programmer is no longer only the author of a piece of software, they are also responsible for the framing the musical and interactive context in which both the human and machine engage. From this perspective, although a system may act in an unsupervised and autonomous fashion, the programmer shares authorship over the musical discourse through their software's interactive behaviour, acting as a part of this curated performance framework.

Part of the curatorial position of the software designer involves the choice of musician(s) to be engaging with the software, and how much information is given to musicians about the software's affordances. In some cases, software behaviours may have been explained to a musician prior to an interaction, whilst in others they may be left to the improviser to be discovered during performance. In addition, repeated

experiences with the software by a performer are often coupled by informal discussions with the developer about their experiences, adding to their knowledge of the system's affordances. The performer's prior understanding of the precise abilities of the software; whether directly responsive, analytical, generative or otherwise will no doubt affect the perception of their relationship to the machine in performance. Bown et al. note this in their analysis, reflecting on the influential nature of these specific details upon the musician's understanding of the interactive context (Bown et al. 2013, p. 6).

In the case of the *_derivations* software, individual interactions with musical collaborators varied in this respect. As detailed in the appendices to this thesis, performances with the *_derivations* software have included ongoing collaborations, one-off performances as well as third-party organised performances without my direct participation. With respect to ongoing collaborations, improvisers Joshua Hyde (saxophone), Alana Blackburn (tenor recorder) and Zubin Kanga (piano) had all either experienced *_derivations* in performance, or discussed the software with me in advance of their initial performative encounter. [54] The musicians therefore came into the environment with a conception of the performative context informed by my ideas as a developer. The focus on extended instrumental techniques, a lack of concern for melodic and rhythmic contours were therefore evident in each of their initial performances with the system. In addition, all three of these performers are classically trained interpreters of contemporary music with whom I have personal a history of collaboration. Working with these musicians, discussions on the reasoning behind programming decisions, the intricacies of certain processing modules and the inner workings of the matching algorithm were commonplace. As such, direct experiences of the performers with the software were supplemented by contextual knowledge of the system's affordances.

Importantly, each of these collaborations initially took place in the context of a planned upcoming performance. Discussions were coupled with repeated exposure to the system in rehearsal, a process that involved an ongoing dissection of previous performances with a view to obtaining the best possible performance outcome. This type

[54] Performances with Joshua Hyde and Alana Blackburn are included on the accompanying audio CD *_derivations: human-machine improvisations* (Appendix B). An example of Zubin Kanga's work (as detailed in Appendix C) can be seen at the URL https://www.youtube.com/watch?v=PmUCGbcrGuw.

of collaboration bears a distinct resemblance to the composer-performer dynamic observable in new music performance scenarios.[55]

In addition to the above collaborations, two encounters with musicians Evan Dorrian (drums) and Antoine Läng (voice) were characterised by a lack of sustained exposure to the *_derivations* musical system prior to performance.[56] Instead, an in situ discussion on the basic premise of the software framed each interactive encounter. Neither of these musicians had any prior experience with the *_derivations* system, nor any other interactive performance system of this type.[57] These two musicians both came from improvised musical backgrounds, ranging from jazz performance to noise and freely improvised group improvisation. In these contexts, it is commonplace for spontaneous musical encounters to occur without any pre-arranged grouping of personnel or instrumentation. In this respect, the performance-time context of both encounters was not dissimilar to that of a freely improvised duet. Each performer was open to engaging with *_derivations* with only one short rehearsal session with the software, and although this preparatory encounter was discussed prior to the eventual performance, neither the software's specific affordances nor the interaction was dissected in any great detail.

Despite the spontaneity of these encounters, common to both circumstances was the curatorial context in which they engaged with the software. Alongside two other performers, Dorrian was invited to participate as part of a weekend 'hack-together' event that involved numerous developers working with algorithmic and interactive software. This researcher-led event framed the encounter between the Dorrian and *_derivations*. Similarly, Läng performed with *_derivations* as part of an impromptu grouping of instrumental performers and software developers. The event was the performative showcase for the 'Biome Symposium' – an interdisciplinary research symposium surrounding mathematical approaches to various creative practices.[58] Läng's performance

---

[55] For recent research on issues of collaboration in new music performance see Kanga (2014) and Roche (2011).

[56] These live performances are documented on the audio CD *_derivations: human-machine improvisations* (Appendix B).

[57] Evan Dorrian has since performed in numerous ensemble situations with interactive software devised by Oliver Bown.

[58] More information on this symposium can be found at the following URL: http://www.biome.cc/symposium.html

with *_derivations* was a chance grouping decided upon in the afternoon before the event, as were other performances showcased in the concert. For this performance it was primarily the Biome theme that framed the musical context in which Läng improvised with *_derivations*.

Third party performances with *_derivations* have been many and varied. Since the launch of derivations.net in 2013 the software has been available for free download and use, facilitating performances and recordings with the software without any direct input from myself. In the first instance, performances and recordings have been undertaken by contacts I have met either in person or purely online through various social media platforms. These contacts have been aware of the software and its affordances through exposure to live performances, radio and online media and/or electronic communications with myself. In addition, a number of musicians have produced performances with the software without any direct contact with myself. These musicians have downloaded the software and also potentially engaged with media documenting previous performances made available on the site. Live performances and recordings posted online by these musicians have credited the software and myself as its developer, with many linking to derivations.net in their online media.[59]

The case of third-party musicians is very different to those performances in which I have been an active participant. These musicians have engaged with the *_derivations* software on two separate levels; both as musical performers, and as traditional 'end-users' of the software acting outside of performance-time. In order for these performances to take place, these musicians have engaged with the specifics of the software artefact, initialising the graphical interface and making use of the session database capabilities. To facilitate this self-guided user interaction, five detailed documentation videos were designed to familiarise performers with the software. These videos exist both as a communicative tool and a how-to guide for musicians wishing to interact with *_derivations*. When compared with information typically available to musicians engaged in this type of performance practice, these videos explain the underlying musical, conceptual and technical details behind the software artefact in great detail. This 'de-mystification' of the underlying processes at work within the system

[59] Third-party produced releases with *_derivations* are detailed in Appendix H.

further therefore solidifies my role as author of the interactive context in which the musician engages.[60]

7.3.6 *Software as musical text*

System designers hold personal musical goals and stylistic preferences, many of which are subtly or overtly manifested in the behaviour of their systems in performance. Although these musical systems are often capable of surprising yet musically coherent results, their capabilities should be understood as the result of specific, programmed decisions of a human author external to the performance-time interaction. In the *_derivations* system, although the software is agnostic towards sonic content, the generation and processing capabilities of the system enforce certain aesthetic boundaries on the improvised performance. The lack of melodic and rhythmic awareness of its listening algorithms, and the electroacoustic means by which sampled materials are combined inevitably limits the stylistic horizons navigable by musicians interacting with the software. Whilst neither the programmer nor the machine has made explicit requirements of a musician's performance, the programmed dynamics of *_derivations* ultimately contribute to framing the musical and interactive boundaries of any performative encounter. These boundaries are both imbedded in the artefact as programmed behavioural characteristics, and made manifest as the specific sonic context provided by the software's contribution to the performance. In this context, the interplay between the chosen generative grammar and sonic vocabulary affirms the designer's aesthetic intentions in the creation of the musical system.

For Coessens, an important function of a musical text or score is that it organises action in some meaningful way. This type of text also "reveals underlying ideas in a coded format" – the ideas of the author – rather than simply codifying instructions (Coessens 2013). Although not communicating explicit requirements to the performer, the curatorial context proposed by the developer of an interactive performance system clearly frames the context of an improvised encounter. The presence of an autonomous machine in an improvised scenario codifies the developer's broad ideas about human-machine performance. In addition, the choice of algorithmic, sonic and interactional

[60] Video documentation for the _derivations system is available at http://derivations.net/about/video-documentation, and detailed in Appendices E and J.

capabilities of these machines also organises musical action in meaningful ways. The machine's interactive and generative specificities should therefore be seen as forms of musical constraint with which the musician engages, contributing to the creation of an interpretive framework.

It may therefore be argued that the development of the software is akin to the creation a form of musical text. This text is not dissimilar from both the graphic and algorithmic textual scores discussed previously in Section 7.3.2. However, rather than embodied in a graphical or textual representation, the constraints on a musician's performance are *interactively instantiated* through performance with the software. As discussed above, certain modes of performance will ultimately reveal themselves as implicitly more reasonable than others, and may also have been communicated as such by the developer prior to an interaction. Therefore, imbedded in the software as text are the context-specific expectations of the performance practice by the author of the software. Exhibiting agency within a musical performance, these systems therefore provoke, shape and contribute to a musician's improvised trajectory. The programmed nature of these behaviours, regardless of the degree of autonomy displayed during performance, express the compositional concerns of their authors.

7.3.7 *Conclusion*

Musical performances with interactive performance systems may be seen as an instantiation of the combined musical ideas of the system developer, the musician navigating this space of ideas, and the interactively instantiated contributions of a machine to the performance. It is precisely because such systems are imbedded with these subjective attributes that performers in this context necessarily engage in a form of interpretation. Although their moment-to-moment performance may be freely improvised, the framing of the interactive context suggests a curatorial framework requiring interpretation in performance. From this perspective, any consideration of a musical text must take into account the entire performance scenario. The musical text is, in effect, the boundaries and constraints of such a human-machine musical interaction as influenced by the machine's perceived capabilities. Navigating these possibilities in a truly interactive sense is the task laid out for the musician. By navigating this space proposed by the designer, the musician is engaging in an interpretive act.

7.4 Symbiosis in human-machine performance

The analyses of this chapter situate this type of human-machine performance in an interactive frame imbued with design intentions, curatorial decisions, performance actions and material agency. The relationships that develop between designer, performer and code are emergent results of the specific, situated interactive encounters between musicians and designed software artefacts. The previous two sections have examined the relationship between human and machine agency in the development and use of interactive performance systems. In Section 7.2 I highlighted how machine agency is entangled with software development and use. By interrogating the performer-developer context as the site for balancing machine and human agencies, I have articulated some of the inherent complexities of this developmental practice with reference to relevant theory. In Section 7.3, the notion of musical interpretation was proposed to define the broad musical artistic context in which musicians engage with such interactive software artefacts. Here the cultural context of creative software development is acknowledged as the fundamental driver behind the development of new performance practices. Regardless of the autonomy of the musical algorithms being developed, a confluence of factors surrounding performance-time interaction help to define the space of human-machine improvisation as one of *sociotechnical curation.*

This section is concerned with a specific conception of interactive musical practice developing from my creative work. Here I consider the notion of *symbiosis* as a metaphor for the musical, technological and interactive interests that have crystallised throughout my creative practice. As outlined in Chapters 4, 5 and 6, a series of idiosyncratic technological experiments led to the development of the *_derivations* interactive performance system, representing the end result of a search for a unique form of human-machine performance practice. Although primarily concerned with musical interactivity and generativity, the trajectory of my creative work also shows concern for finding intuitive and coherent means of achieving interactivity between instrumental performers and computer music systems. By focusing upon sampling-led methods of generativity, my practice led towards the creation of a piece of software that could also engage the musician as user outside of a performance-time encounter. As discussed previously, this is a form of metacreation that enables the performer to customise their real-time

encounters with the *_derivations* software through the development and use of session databases.

This particularity of the *_derivations* system further complicates the ideas discussed previously in this chapter. The interpretive framework entered into by the improvising performer, it has been argued, is in part defined by curatorial agency exerted by myself as developer of the software. This agency is exerted through the software's inherent affordances, but also through the distribution of the software via derivations.net and the de-mystification provided by the accompanying video documentation. However, *_derivations'* session database concept (discussed in Section 6.8) also places a significant degree of curatorial authorship over the interactive environment in the hands of the performer. The ability to customise the processing and structuring parameters of the software give the performer agency over the machine's contribution to any subsequent real-time encounter. In addition, by pre-defining the sonic vocabulary of the musical system cumulatively from session to session (or by merging disparate musical materials), the musician's agency and authorship over the resulting interactive context is also assured. Interacting with the *_derivations* software in rehearsal is therefore another form of metacreative authorship.

As we have seen in the case of *_derivations*, musicians can substantially influence future encounters with the machine both during and outside of performance. The machine's performative agency is therefore directly dependent upon the input of the musical performer, both for its generative and sonic capabilities. Although the system's autonomous processing and structuring algorithms provide a degree of opacity from the matching algorithm, the system is entirely dependent on human stimulus to generate its contribution to an improvisation. The software's *self-referencing* algorithm, as discussed in Section 6.6, enables it to autonomously generate patterns of musical material free from the interventions of a real-time performer. Crucially however, the sonic materials accessed for use by this algorithm are digital remnants of past interactive encounters with a human musician. Even the most autonomous form of generativity expressed by *_derivations* is reliant upon past human musical gestures.

This is a cumulative approach to developing musical generativity. The interaction between human and machine in performance is defined by their interaction in rehearsal

and performance. Although the underlying organisation of the self-referencing algorithm remains intact, the machine's sonic contribution is dependent upon human performative agency. This historical and cumulative dimension to the *_derivations* software has encouraged speculation on the nature of this unique interactive scenario. Borne out of a sampling-led approach to generativity, it is the dependence inherent in this generative approach that has led to a *symbiotic* conception of musical interactivity. The software needs the human utterances to express its organisation, and in turn the human needs the expression of this organisation in order to engage in meaningful dialogue with the software.

7.4.1 *Metaphors for interactivity*

As discussed in Chapter 2, interactive models and metaphors have been used to outline modes of interaction, and also to explain the traits of designed musical systems and the interactive modes they engender in performance. These have included metaphors related to instrumental performance to delineating levels of control (Chadabe 1984; Rowe 1992; Winkler 2001); biological systems metaphors such as *stigmergy* that describe group dynamics (Blackwell & Young 2004, 2005); categorisations of interactivity into differing modes such as *conversational/ornamental/instrumental* interaction (Johnston 2009); understanding systems and interfaces as *prostheses* (Young 2008) etc. These metaphors are used in two specific ways: to project ideal modes of interaction as an aid to design, and as means of understanding interactive traits perceived when observing humans interacting with machines. For the former, they provide a means by which complex technical projects can be positioned into useful conceptual frameworks, becoming anchor points with which to ground future design aims. For the latter, they enable retrospective understandings of the traits exhibited in the interaction domain itself. In this case, metaphors aid in explaining observed interactive modes without imposing any preconceived notions of interactivity on the domain in question.

Whilst some of these metaphors of human-machine interaction have been developed through observation of real-world settings, others remain speculative understandings of the space in which humans and machines interact, borne from personal artistic experiences and aesthetic leanings. These speculative understandings say as much about the aesthetic preoccupations of the system designer as they do the phenomenon itself.

The choice to develop musical systems that participate in improvised performance is in itself a speculative position, and as such models and metaphors in this space should be understood as situated within the aesthetic context in which they are developed.

By way of example, the composer and researcher Michael Young has outlined his aspiration for musical systems that are able to adapt and contribute to their sonic environments, engaging with human performers with a level of intimacy to the current performance context (Young 2009) (see also Section 2.2.4). Regarding adaptability, the author's has proposed *stigmergy* as a model for collective interaction between entities and their environments, an organisational property observable in ant and termite colonies. These concepts therefore serve to unify the author's concern for avoiding anthropomorphism in interaction design, with a view of collectivity and sonic interaction in an established musical performance context such a free improvisation. Implicit in Young's description of these two notions is an overall concern for designing for musical improvisation as a known and understood musical practice, albeit a complex one. This aspect of Young's approach can therefore be contrasted with the discussion present in this chapter. Although human improvisatory practice is necessarily the foundation of any human-machine improvisation, it has been argued here that such a dynamic is in itself fundamentally different precisely because of the presence of the machine collaborator. Such a contrast in approach serves to highlight the ability of musical and aesthetic perspectives to influence the development of metaphors for interactivity.

As has been discussed in this thesis, the aesthetic concerns of both free improvisation and electro-acoustic composition have grounded the design aims of the *_derivations* system. In addition, concerns for the development of a useable, customisable and sharable creative artefact have strongly influenced the specific decisions of the software's development and dissemination. In Sections 7.2 and 7.3 I articulated how the artefact development is intimately linked to the usage context of performer-developer devised artefacts. This interest in the openness and transparency in the design process has influenced how interactivity is viewed in my practice as related to *_derivations*. It is therefore from this vantage point that I situate my ideas of symbiotic human-machine performance practice.

7.4.2 *Symbiosis in art and technology*

The biological process of *symbiosis* provides an evocative metaphor for the relationship between humans and technology in both artistic and technological practices. Widely documented in the natural world, symbiosis is an "association of two different organisms […] which live attached to each other, or one as a tenant of the other, and contribute to each other's support." (Oxford English Dictionary 2015b) The process of symbiosis is such that two species, as *symbionts*, are intimately connected to each other and often rely on each other for survival. Examples of symbiotic relationships are those of lichens, composed of algae living inside a fungus, the mutualistic relationship between sea anemones and the hermit crabs, as well as the symbiotic relationship between humans and bacteria living inside the body. There are three main types of symbiosis observable in nature: *mutualistic*, *parasitic* and *commensalistic*. Each one of these types describes the relationship that exists between different organisms:

- *Mutualistic Symbiosis* – describes a situation where two organisms are dependent upon each other, with both benefitting from this interdependence.
- *Commensalistic Symbiosis* – where one organism benefits from the other, yet the other is neither harmed nor helped.
- *Parasitic Symbiosis* – is a one-sided relationship where one organism benefits and the other member is harmed by the interaction.

Licklider (1960) was the first to suggest symbiosis as a metaphor for the relationship between humans and computers, proposing that man-computer symbiosis was an "expected development in cooperative interaction" between humans and machines. For Licklider, humans and computers were to develop symbiotic partnerships, whereby computers could enable humans to control complex operations without recourse to inflexible and predetermined programs (Licklider 1960). This conception of the relationship between humans and machines acknowledged the mutually beneficial nature of both entities, rather than the ability of machines to simply extend human abilities. In addition, this notion of cooperation was conceptually separated from the concerns of artificial intelligence, in which the machine is said to dominate as the principle problem-solver (Jacucci et al. 2014).

In the field of human computer interaction, 'Symbiotic Interaction' has gained traction in recent years as an area of research. The term is used to describe a mode of interaction that combines "computation, sensing technology and interaction design to realise deep perception, awareness and understanding between humans and computers." (Jacucci et al. 2014) This new area of research is concerned with enabling computers to implicitly detect user goals and psycho-physiological states without removing control from human users (Blankertz et al. 2015). Taking cues from Licklider's original definition, this emerging field differentiates itself from artificial intelligence research through a focus on merging computation and human goals and abilities. Jancucci et al. (2014) make specific reference to goal and agency independence of humans and machines in symbiotic interaction, highlighting the specific ideal of such interactive scenarios.

Symbiosis has been of interest to artists-researchers interested in emergent forms, artificial life and artificial intelligence. In their 'Symbiotic Art Manifesto', artist Leonel Moura and scientist Henrique Garcia Pereira proclaim that the future of art-making as one in which humans relinquish control over the art-making process to intelligent machines (Moura & Pereira 2004). Declaring that "Art as we know it is dead. This time it is definite and official," the authors define an artistic paradigm in which the human's role is the design and manufacture of artists, and not of art itself. Moura's artworks make use of robotic artists that follow autonomous procedures set out by the artist. Robots interact with and shape their environments and not each other directly through the process of *stigmergy*, resulting in complex and unpredictable forms. Conceiving of this type of creation as 'symbiotic art', Moura's conception of artistic practice is one in which the artist's task is less concerned with practical skills, and more related to setting the conditions for machines to generate artworks autonomously without human intervention (Moura & Pereira 2004, 2011).

Moura's work is essentially a metacreative practice; that is, the creation of machines that may be deemed creative on their own.[61] However, the use of the word 'symbiotic' displays a concern for the reciprocal relationship that exists between the artist and their creative machines. From Moura's perspective, the symbiotic relationship exists between

[61] Metacreation is an emerging field concerned with endowing machines with creative behaviours. More information on the field and a definition of metacreation can be found here: http://metacreation.net/about/

the artist as designer, and the automaton as artist working in real-time to create artworks. In this context, the human *needs* the machine's agency in order to render artworks, and the machine *needs* the developmental guidance of the designer-artist to make artworks. Moura and Pereira's metaphor of symbiosis therefore rests upon the mutual dependencies that exist between the designer-artist and the robot-artist in this practice. The former conceives of and fabricates the latter, specifying its generative properties. The symbiotic relationship between human and machine is therefore abstracted, with finished artworks existing as a result of both human and material agencies as separated in time.

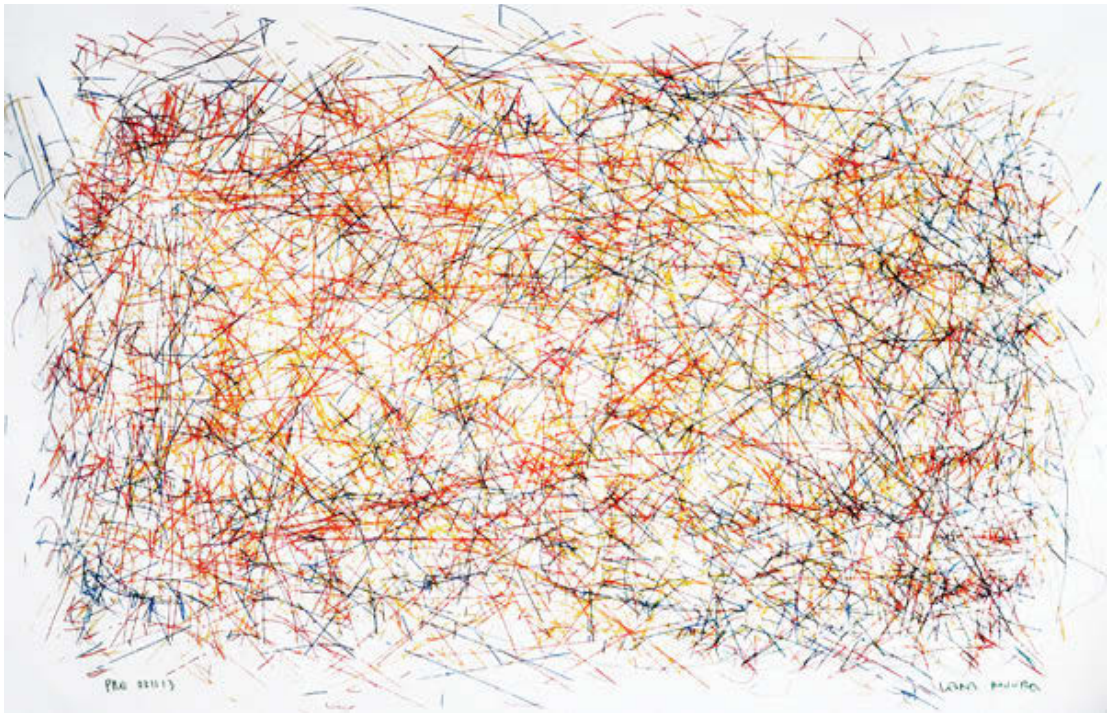


Figure 46: Painting Robots Orchestra (PRO) – *PRO021113* – Leonel Moura (2013). 'PRO is constituted by a series of robots able to detect sound. Each robot is receptive to a different frequency that activates a painting device.'[62]

This symbiotic relationship has provoked Moura to consider how this practice affects the human designer. Considering the development of artists as opposed to art itself, Moura and Pereira ask the question: "what do we become ourselves?" In doing so, the authors acknowledge that this practice puts this mutualistic relationship in sharper focus: "The life of the artist/machine is interlinked to the life of the artist/human." (Moura & Pereira 2004)

[62] http://www.leonelmoura.com/pro.html

American artist Amber Stucke creates drawings that represent symbiotic relationships between imaginary forms and various species of fungi, algae and lichen. In her 'Symbiosis State' project, Stucke's drawings are presented as taxonomies of symbiotic relationships between the real and the imaginary, exploring these complex biological phenomena. Stucke describes her creative drawing practice as an 'embodiment' of symbiosis, a practice in which consideration for the reciprocal relationship between human consciousness, material forms and biological concepts coalesce into an integrated creative practice (Stucke 2011). Through a phenomenological lens, her artistic practice is conceived of as embodying a symbiotic relationship between graphite, paper, flesh and mind. Although the object of Stucke's practice emanates from a concern for this natural phenomenon, the artist has also incorporated and assimilated the concept into an understanding of her own practice. Through a consideration of the potential consciousness of biological forms such as mycelium and lichens, the artists sketches a poetic ideal of reciprocity and dependence in her art making:

> The symbiosis between the paper, the graphite, and me grows slowly in-between the blank spaces like mycelium under the earth feeding on the plant's roots and also becoming a part of its roots–transforming the root system to an evolved dynamic root that is both plant and fungus. This metamorphosis is also a transformation of the mind. It evolves an internal state of mind of becoming symbiotic relationships (symbiosis state).
>
> (Stucke 2011)

In Stucke's work, the symbiotic relationship between human and material is one in which consciousness and performative action is entangled with the results of material agency. Artworks borne out of this negotiation reflect upon the biological process of symbiosis itself, and are a direct consequence of unpredictable relationships. Her works are speculative, engaging the viewer in a consideration of the process of symbiosis as both a biological process and embodied practice. Stucke's conception of interdependence between human and material has parallels with Pickering's conception of the *mangle of practice* (Pickering 1995) (as discussed in Section 3.3). The artist's 'symbiosis state' is conceptualised as a form of knowledge generation, her drawing practice as an integrated and embodied practice enabling new knowledge systems to emerge (Stucke 2011).

Figure 47: ***Mutualistic Relationships No. 5 (Symbiosis State)*** **– Amber Stucke (2013)**

Although the artists discussed above approach symbiosis from contrasting artistic practices, they are both concerned with practice itself as the site for symbiotic interaction between human and material agencies. For Stucke, symbiosis is an embodied process that describes art making, and a philosophy of mind in drawing. Her artworks are the final result of this process of entanglement between human and material. For Moura, symbiosis defines the relationship between human and machine, concept and performative execution. Although the interdependence between the two entities is abstracted in time, the artist's conception of the symbiotic artist reflects a consideration of their reciprocal influence. Returning to the field of human-computer interaction, symbiotic interaction is a term used to describe the ideal of cooperative and reciprocal exchanges between humans and machines. Contrasted with the pursuit of software autonomy for its own ends (artificial intelligence), this conception of practice is concerned with human-machine reciprocity and goal and agency independence.

In Moura's work, the performative actions of autonomous agents create fixed artworks (paintings), and do not engage with humans to create these. The metacreative artistic process is therefore considered symbiotic, not the real-time process in which the automatons themselves create. Stucke's conception of symbiosis is also performative, yet the process of artistic creation also gives rise to fixed artefacts (drawings). For those metacreative works that exist in time (music, animation etc.), it is the *process* of generation and interaction (the performative act) that is presented as an artwork. Furthermore, for those works in which the environment involves the input of a human performer, this performative act becomes more complex, and the notion of a symbiotic relationship must also refer to the act of performance and use of the artefacts by the performer. Mutual dependence between the automaton and the human as both performer and end user must therefore factor into an understanding of symbiosis in such practices.

In light of the above, whilst not subscribing strictly to the definition of symbiotic interaction as outlined by Jacucci et al. (2014), my use of symbiosis as metaphor is concerned with the performative exchange between human and material as asynchronous artistic practice (software development, use of software in rehearsal), as well as with the performative act itself. Human-machine improvisation is a space in which mutual influence must be balanced with concerns for software autonomy. As discussed in Section 7.2 and 7.3, this creative practice should be understood as a complex space in which the relative transparency or opacity of the system is taken into account, along with the cultural context in which the performance practice takes place.

7.4.3 *Defining symbiotic musical interaction*

The development of interactive performance systems, whilst often focused upon maximising the autonomous decision-making capacity of the machine, should be thought of primarily as engaging human musicians in a mutually influential performance context. Human engagement with such systems fundamentally differs from human-human interactive contexts. As articulated throughout this chapter, performers are somewhat dependent upon the interactive capabilities of the musical systems they interact with. Musical systems are developed with specific stylistic constraints in mind, and performers interacting with such systems learn to interpret the nuances of these systems through

repeated exposure to them in rehearsal and performance. They are dependent on the system in a broad conceptual sense, as it is often the case that the musical system's interactive vocabulary (or indeed, sonic vocabulary) defines the range of possible musical territory to be covered in an interaction. Although a musician is clearly an autonomous entity, their musical contribution is shaped and altered by the context provided by the intersection with the musical system.

Symbiosis as an interactive metaphor sits decidedly outside of the purely technological, acknowledging this mutual dependency and influence between player and machine. The metaphor is inherently interactive, with its genesis in the intersection between separate and contrasting organisms. Although describing mutual dependence between human and machine, the metaphor of symbiosis in performance should be understood as encapsulating the complex interaction between two autonomous entities that are both aware but also dependent upon each other in performance.

In light of the above, symbiotic musical interaction may be characterised in the following ways:

- *Mutualism* – mutual dependence between human and machine in performance and from performance to performance
- *Commensalism* – the machine may *feed* off the performer to sustain its own actions, without obscuring or hindering the performer's musical contribution
- That one's actions may affect the evolution of the another's material, though not as a direct reaction to or cause of the other's responses
- Machine autonomy is an emergent property of interactions between humans and software in rehearsal and performance

With respect to system design, it is possible to conceive of such systems as enabling either *mutualistic* or *commensalistic* symbiotic relationships to occur between humans and software systems. The development of these relationships need not seek to maximise the qualities of musical autonomy, however musical autonomy may emerge through cumulative interaction with the system over a period of time. In human-machine performance practice, an interactive environment fundamentally shapes the performance of a human musician throughout an improvised session, as well as from session to

session. Mutual dependence in this instance refers to the change brought about in the human's behaviour towards the system, and the ability for the system to change along with the performer. One way of conceiving of machine dependence in this context is through the material siphoned from the performer and used to further its own autonomous behaviour. Such an approach repurposes the cumulative history of past material as a possibility space, providing the machine with a sense of performative agency that is directly tied to the history of the human musician's performance. In Chapter 2 I suggested the concept of *derivation* for this process in the design of interactive musical systems. Both sonic and algorithmic derivation were highlighted as methods by which system designers make use of the performance of a musician to generate musical contributions of generative software. Such dependency on the musical performer can be seen as displaying qualities of a *symbiotic* musical interaction.

7.4.4 *Symbiosis and _derivations*

My work developing and performing with the *_derivations* performance system has embodied this conception of symbiotic interaction. One of my central concerns for developing musical performance system was that the software's contribution in performance should be heavily influenced by the improviser's playing. Although I was interested in developing a system that could contribute to performance un-assisted, the system's agency as expressed in performance must be emergent and tied to the specific context of the current performance. As discussed throughout Chapters 4, 5 and 6, the work preceding *_derivations* explored the capture and storage both audio and analysis data streams that could then be used to develop the generative capacities of an interactive system. Making use of live sampled phrases of my saxophone performance, I was further drawn to using recorded data as a means of generating complex behaviour using simple underlying rules. Enabling the re-structuring and processing of sampled data streams (including audio recordings) was a metacreative process. In comparison to other approaches, such generative techniques are dependent on outside stimulus. Without having interacted with a human performer, systems of this type have no space of possibilities with which to generate their own material.

The sampling-led approach taken in my work has led to systems and processes that are dependent upon human interaction. The autonomy expressed by the *_derivations* system is

predicated upon connections drawn between materials siphoned from past human performance. In my work, symbiosis is a metaphor for the way in which *_derivations* engenders reciprocity, mutual influence and interdependence between human musician and software system, both inside and outside of a performance. The system is at once independent and dependent, influenced and autonomous. For the performer interacting with *_derivations*, real-time navigation of the system's affordances is entangled with the direct siphoning of one's past musical gestures. The machine is dependent upon these materials to express its underlying organisation. In turn, the musician needs this real-time expression in order to further an ongoing musical dialogue with the machine. *_derivations* feeds off the performer as the performer assimilates these new constructions into their understandings of the immediate performance environment.

In my own performance, the intimate knowledge I have as both developer and performer of the system enables me to shape my interactions in a way that is particularly in tune with the system's character. I have developed a performance practice that is specific to this particular interactive scenario. With the knowledge of *_derivations'* ultimate dependence on my current and past performances, I am able to provoke and predict certain actions from the machine. However, owing to the emergent nature of the software's generative organsiation, I am consistently surprised and provoked by its interactive presence in performance. My performance has evolved with the design of the system, but at the same time, I am dependent on the interactive context that the system provides during performance. This provides a set of constraints for improvisation that I thrive off in performance with the system. It is a composed space of possibilities that enforces a mutually dependent relationship to develop between player and system.

7.4.4.1 Software as symbiont

In the *_derivations* system, the *self-referencing* algorithm is based upon a fixed algorithmic organisation. Each of the system's internal players engages in a game of *chinese whispers* or *telephone*; passing stored analyses from one player to the next (see Section 6.6 for a full description of this process). Although this process is deterministic in nature, left on its own this internal organisation is able to generate emergent behaviours over a period of time without direct influence from a human performer. The individual players in the system maintain the intention of matching analyses received with phrases stored in

memory, and outputting the results for processing and re-synthesis. Crucially however, the machine can only express this underlying organisation with material siphoned from a human performer. This material can either be pre-loaded as a fixed sonic vocabulary, or built cumulatively throughout the current improvised performance. The machine is therefore dependent on the results of human agency in order to evolve through its possibility space. That is, it is dependent on the musician to provide the space with which it expresses its organisation.

In its interactions with a human performer, *_derivations* is therefore a sonic automaton acting as a musical *symbiont*. By entering into a performance with the system, the musician allows the software to feed off its past in order to generate its contributions to the improvised encounter. This type of dependence is unique to the system, and is not pursued in order to approximate human interactive traits. However, this particularity of *_derivations* solidifies its specific function in this symbiotic relationship. The self-referencing algorithm that is at the heart of *_derivations'* generative strategy is deterministic, and also agnostic to the separation between what is self-generated and what is generated externally. Given this, the system may be considered an *autopoeitic* system, as defined by Maturana and Varela (1980). Real-time analyses of the live performer are injected into the system's self-referential feedback loop without any priority given to this analysis. The system takes information from the performer and passes it through its network. However, once this information is in the network itself, the individual components do not distinguish its origins as being from outside or inside. *_derivations* therefore integrates the outside world into its continual self-generative process.

7.4.4.2 Performer as symbiont

Facilitated by the process of *black boxing* (see Section 7.2), a human performer perceives the above-described generative process as a totality, interacting with the system as if it were a single living organism. The trajectory of the human-machine performance is the result of an interactive exchange between the complexity of the self-referential algorithm and the performer's sonic gestures. From outside of this system, the musician engages with *_derivations'* sonic contribution as a whole, expressed by the polyphony inherent in the overlapping of its various players. The inherent unpredictability and emergence of

this process becomes the space in which the musician interacts. It is the expression of the system's self-generative identity. Making use of the system's cumulative live sampling capabilities, the growing complexity of the system's sonic vocabulary benefits the musician by creating a space of possibilities that grows along with the performance trajectory. In addition, the session database facility of the software further structures this symbiotic interaction in performance. The performer's engagement with the software from session to session allows *_derivations* to express its underlying organisation within a larger range of possible source materials.

7.4.4.3 Mutualism in performance

As with mutualistic symbiotic relationships in nature, the needs and goals of the two symbionts differ in this human-machine interactive partnership, yet they remain complimentary. The *_derivations* software maintains a rigid strategy of live sampling and storage, analysis and comparison. This relatively simplistic algorithmic approach is not in any way comparable to the ingenuity, subtlety and novelty achievable by a human musician. However, by relying upon internal and external analyses and randomised mutations of captured materials, *_derivations'* self-referential loop projects novel structures from its fixed organisation. The unpredictability of this process is what makes an interaction with the system dynamic and playful for the musician. Although each musician that interacts with *_derivations* will do so with differing musical and aesthetic aims, it may be suggested that the ultimate goal of a live interactive exchange with the software is a successful performance. As for any performance practice, the relative success of any one performance is purely subjective matter, to be judged by both performer and audience. As discussed previously, adapting to the particular interactive context provided by *_derivations* is a constraint upon a musician's interaction with the software, and therefore the ability to create a successful performance. Each musician will chart the space provided by the system's particularities differently in performance, discovering the most suitable way of co-existing with the emergent structure expressed by the system's persistent organisation.

7.4.5 *Template for the design of a musical symbiont*

Considering mutual dependence as a design principle is sufficiently broad in many respects, but also constraining in others. As suggested previously, the concepts of both sonic and algorithmic *derivation* have proven useful in balancing autonomy and dependence in the design of interactive performance systems. As surveyed in Chapter 2 and articulated with respect to *_derivations* in Chapter 6, the methods available for the generation relationships between captured and analysed performance material are many and varied. However, from my practice I have found that focusing attention on capture and storage, navigation and re-generation has provided an intriguing balance between transparency and opacity of a machine's contribution to a human-machine improvised performance.

To conclude, below I propose some specific traits that may be desirable in a *musical symbiont*. This list is not intended to be exhaustive nor proscriptive. Rather, through retrospective consideration of my own design trajectory and performance practice, this template articulates some of fundamental design considerations that have emerged from the development and use of the *_derivations* interactive performance system.

- The machine and the human form a symbiotic relationship whose musical results are an emergent result of their interactive history
- The machine is an empty vessel – it cannot act without having interacted with a human
- The machine's behaviour and/or content is in some way dependent upon the human's current and past actions
- The impetus for the machine's contribution is tied to some present understanding of the human's actions
- The autonomy of the machine's actions is in some way derived from the interactive history with the human performer
- The machine displays agency by making connections between materials siphoned from human interlocutors
- The machine feeds off the performer, using captured memories of the encounter as fresh material for its contribution

- The machine also listens to itself, depending equally on its own past as an indicator of its future state as it does outside stimulus

7.5 Conclusion

Throughout this chapter I have reflected upon three core concerns arising from the development and use of my interactive and generative designs. Whilst chapters 4, 5 and 6 of this thesis have traced the iterative development of my software, these three reflections have articulated in depth emergent research themes that have persisted throughout my creative practice. Self-reflective practice has been an invaluable tool in identifying these key areas of concern emerging from my work. As my creative practice progressed throughout this research, these broad theoretical concerns developed in tandem with my software and performance practices. Importantly, the reflections on issues, concerns and interests articulated here are made relevant to the wider community through considered analysis and reference to literature relevant to the field at large.

In the first reflection I articulated a concern for understanding creative programming practices as a *dance of agencies* between human and material. Here the context of the performer-developer is outlined as a unique space for investigating the relationship between software and developing performance practices. In this reflection I have articulated how the performer-developer engages in a form of incremental *black boxing,* a process in which the complexities of the developing system recede in favour of the agency displayed by the artefact acting as a whole. In addition, Akrich and Latour's notion of the *script* is appropriated to describe the complex relationship between design and use for the performer-developer. In the second reflection, the concept of musical interpretation is positioned as a framework from which to view improvised human-machine performance. Considering the mediated nature of such practices and the presence of the designer as author, here it is argued that such software systems may be regarded as a type of musical *text.* Understood in this way, improvised human-machine performances can be conceptualised as extending traditional understandings of interpretive performance practice.

In the final reflection I propose *symbiosis* as a useful metaphor for interactivity in human-machine performance practice. Considering the interdependence between both

human and machine agencies in such performance practices, *symbiosis* describes the reciprocal relationship that exists between performer and software. Sitting outside the purely technological, *symbiotic interaction* acknowledges the affect that such software has on the developing performance practices of human musicians. With reference to *_derivations*, *symbiosis* is used to describe the way in which systems of this type can be dependent upon the human musician to act during performance. Contrasted with designing to maximise musical autonomy, such an approach harnesses both sonic and algorithmic *derivation* as the primary means by which such software systems display their material agency.

Chapter 8. Conclusions, Ongoing and Future Work

In this thesis I have undertaken a practice-based approach to research enquiry, positioning artefact development as a site for investigating key issues, concerns and interests encountered in the development of a significant software artefact and related creative outcomes. Throughout the chapters of this thesis I have detailed the development trajectory of the *_derivations* system, revealing the iterative cycles of development that have led to the current working artefact. Seeking to explore novel approaches to both musical software and performance practice, my research has traced the development of personal approaches to designing for musical interactivity. In doing so, I have highlighted technical, aesthetic and theoretical issues encountered from inside my creative practice. The contributions and outcomes of this research relate directly to my practice as a performer-developer. In the remainder of this chapter I summarise the contributions originally outlined in Chapter 1 of this thesis, before outlining the related creative outcomes of this research and my ongoing and future research and creative work.

8.1 Contributions of the research

The development of novel computer music techniques for use in interactive musical performance

As a practice-based research project, the creative outcomes of this research are presented as significant research contributions from which the findings of this thesis may be contextualised. In Chapters 4, 5 and 6 I outlined the various interactive designs developed throughout this research. Presented in the form of a narrative of development, significant creative outcomes from each chapter have been included in the submission materials of this thesis. In Chapter 4 I detailed early work focused upon audio analysis, event-based pattern matching techniques and probabilistic generative methods. From these experiments I detailed the creation of the *Tripartite Markovia* system, an interactive system based around first-order Markov chains (Section 4.2.4.2, Appendix G). Whilst sophisticated Markov modelling approaches are prevalent in the field of interactive computer music (Ames 1989; Martin 2014; Pachet 2002; Roads 1996; Zicarelli 1987), the idiosyncratic approach of *Tripartite Markovia* is presented as an example of the early data sampling approach in my work. Using Markov modelling as a

foundational generative strategy, the trajectory of this system solidified personal heuristics for taming complexity in my designs. The continuous data sampling and modelling approach of this system, whilst based on an event-based MIDI paradigm, also foreshadowed later methods for working with continuously sampled and analysed audio from a live performer.

In Chapter 5 I detailed a series of advancements in my programming practice focused upon developing self-contained modules that could be later appropriated into integrated interactive systems. The creation of *4-buff-pvoc* and *pitch models* solidified a new interactive strategy in my work based upon live sampling and spectral re-synthesis (see Section 5.2.1, 5.2.2 and Appendix G). Given the independence these two modules, their inherent idiosyncrasies served as catalysts for subsequent explorations of musical interactivity in performance. Working in this modular fashion allowed the rapid evolution of an approach to live performance through performative testing and refinement. This bottom-up, *bricolage* approach to development (McLean & Wiggins 2010) led to the creation of *Live-processing-1,* a precursor to the *_derivations* system in its current form. Following these experiments, the *phrase player* system (see Section 5.3.2 and Appendix G) sought to address a fundamental issue with momentary live sampling experienced whilst testing *Live-processing-1*. As a proof of concept system, *phrase player* segmented the live audio signal in a continuous fashion, indexing individual *phrase points* analysed from the live signal using silence threshold techniques. Whilst the system itself was not pursued further, this approach formed the basis of *_derivations'* approach to the segmentation and storage of phrases analysed from a live performer.

In Chapter 6, the *_derivations* interactive performance system is outlined in depth. Beginning with *_derivations' phrase database*, this chapter details incremental advancements to my software as *_derivations* matured into a stabilised software artefact (see Appendix A for the current distributions of the *_derivations* system). The most significant feature of the software is the *phrase matching* algorithm, as described in depth in Section 6.5. This approach to timbral matching solidified into a core generative strategy of the software, and has undergone a significant degree of testing and refinement throughout this project (see sections 6.5.3 – 6.5.7 for details). The current *phrase matching* approach utilises MFCC feature vectors to match target phrases with those stored in the database (as described in Section 6.5.8). This approach is presented as a novel form of content-based music

information retrieval (as defined by Casey (2008)). In addition to *phrase matching,* the development of *session databases* also became a defining feature of the *_derivations* software (see Section 6.8). The ability to save, load and merge pre-analysed databases of material has facilitated a very different form of interactivity than experienced previously, enabling the performer to influence the software from both inside and outside of a performative encounter.

A novel self-reflective study of the development and use of interactive musical performance systems from the perspective of a performer-developer

The development trajectory that led towards the creation of *_derivations*, as articulated in the three central 'wayfinding' chapters, highlights how the creative process has been harnessed to explore emergent research themes arising from my practice. As discussed in Chapter 3, Stephen Scrivener's conception of *creative-production* research is one in which process is foregrounded in practice-based research projects (Scrivener 2000). In contrast to *problem-solving* projects, *creative-production* research projects seek to explore, define and respond to research problems through the plane of practice. Whilst digital arts practices are often concerned with the design and development of unique software artefacts, the criteria by which such artefacts are developed and evaluated differ greatly to engineering disciplines. Throughout the development and use of my various interactive designs, Schön's notion of *self-reflective practice* was used to continually pose questions about musical interactivity, human-machine performance and the relationship between human and material agencies in development and performance (Schön 1983). By investigating these various issues, concerns and interests, this research led to the creation of significant creative outcomes, as well as in-depth engagement with relevant theoretical issues arising from this practice.

My creative practice followed a *bricolage* approach to creative programming. As such, my interactive software was not designed with reference to clear and unchanging design principles, but evolved slowly in response to specific *resistances* and *accommodations* encountered through engagement with my developing software artefacts. By tracing the iterative process of development engaged in throughout my work, I have highlighted the emergence of specific technical and aesthetic challenges that were overcome throughout the development process. Whilst *accommodations* were developed in response to *resistances*

encountered in practice, the process of *reflective practice* was used to surface important theoretical issues related to my practice that could be further be engaged with in extended reflections-*on*-action. These reflections form the core of Chapter 7 of this thesis, and address the final contribution of this research.

Theoretical perspectives on the design and use of interactive musical performance systems

The development and use of interactive and autonomous musical systems is a rich area for practice-based research in the digital arts, provoking questions surrounding human and material agency, authorship and musical performance practice. Throughout iterative cycles of development, testing and performance, reflection-*on*-practice has served as the means by which theoretical understandings have been advanced in this research. By engaging in this form of self-reflexivity, my research has sought to connect these emergent concerns with wider understandings of practice and theory. Chapter 7 of this thesis explored three key theoretical concerns emerging from my creative work, acting as the site for extended reflection-*on*-action. Although addressing specific areas of interest, the three reflections contained within this chapter address related theoretical concerns emanating from my practice. This chapter investigates the relationship between design and use of interactive software artefacts, positions such a burgeoning artistic practice within the spectrum of contemporary musical practices, and proposes alternative models for the development of interactive musical systems.

The chapter begins with an investigation of the reciprocal relationship that exists between human and material agency in the design and use of interactive musical systems. Engaging with the writings of sociologists Bruno Latour, Madeline Akrich and Andrew Pickering, here the unique context of the performer-developer is examined, situating the outcomes of this creative practice within a *dance of agency* between the human (performer-developer) and the material (software/code). Seeking to understand the complex relationship between the performer-developer and their developing artefacts, Akrich's notion of an artefact's *script* and the concept of *black boxing* are used to explain the way in which interactive software artefacts are engaged with in development and performance. In addition, Hamman's conception of an artefact's *episteme* was used to describe the means by which developing artefacts provoke surprise and unpredictability, in spite of the developer's proximity to their developing artefacts.

As my creative practice developed throughout this research, I became interested in the broad performative context of human-machine performance. Whilst my practice revolved around freely improvised performance, the development of software to be used in this context posed fundamental questions about authorship, agency and the notion of musical interpretation in this scenario. The second reflection in Chapter 7 (Section 7.3) engaged with these issues, arguing that any understanding of improvised human-machine performance must contend with its mediated nature. Implicit in such an understanding is the context in which the interaction takes place, the non-human agency exhibited by the machine in performance and the role of the developer as author. By outlining how various forms of musical text embody direct and/or indirect *constraints* upon a performer, I have argued that such software artefacts place the performer within a decidedly interpretive framework. With reference to commonly understood notions of musical interpretation, the development of interactive software is positioned as akin to the creation of a musical *text*. By setting the aesthetic and interactive boundaries of the musical encounter, I argue that such software systems reveal their constraints interactively through engagement with human musicians in performance.

In the final reflection of Chapter 7 (Section 7.4), *symbiosis* is proposed as a metaphor for interactivity and reciprocity in the design and use of interactive musical systems. Sitting outside of the purely technological, *symbiosis* describes the mutually dependent and reciprocal relationship that exists between performer and system. Mutual dependence can be expressed both inside and outside of a performative encounter with such systems, as evidenced by *_derivations'* session database capabilities (see Section 6.8). As illustrated throughout Chapters 4, 5 and 6, there existed in my practice an inherent tension between designing for unpredictability and surprise, and a focus upon sampling-led musical generativity. In developing the *_derivations* system, I sought to foster a mutually interdependent relationship between performer and system. Whilst the improviser does not directly control system in performance, *_derivations* can only express its material agency by having previously interacted with an improviser. Its performative agency is therefore dependent upon the input of a human performer. Conversely, the performative context of improvised human-machine performance places the performer in a somewhat dependent relationship with interactive software. As discussed in Section 7.3, the interactive constraints of such software define the boundaries of an interactive encounter.

8.2 Performances, collaborations and releases

Across its various iterations, the *_derivations* system has been in use in my own practice since 2011, and my performances have evolved along with the software itself. As discussed throughout this thesis, advancements to the software's design have provoked new modes of performative engagement with the software. In my own practice, the development of session databases led to a fundamental shift in the way in which interactivity, authorship and machine agency have been viewed in my work. In the early stages of working with the software, each performance led to new conceptualisations of human-machine performance practice, and new ideas about how to advance the software's design. The feedback from my performances into the refinement of the software was an organic and iterative process, helping to advance the software in a rapid fashion. However, in the past two years the software has coalesced into a workable, flexible yet idiosyncratic system with a fixed structure. Consequently, my recent performances with the software have been focused more on exploiting its capabilities, developing new session databases and being conscious of the developing *symbiotic* relationship between myself and the system I have designed. My most recent performances with the software are detailed in Appendices B and C of this document, and a detailed list of performances and presentations of the software can be found in Appendix F.

Besides my own performances, throughout the life of the software *_derivations* has been engaged with by several improvisers, both through direct collaborations with myself and also by way of 'third party' performances and recordings. In February 2014 a collection of live and studio recordings with the *_derivations* software was curated into an EP titled *_derivations | human-machine improvisations* and released on the independent label Integrated Records[63] (see Appendix B). The collection of recordings groups together six performances recorded between 2011 and 2013, displaying the *_derivations* system from a variety of performative perspectives. In addition to the two performances by myself as performer-developer (on tenor and soprano saxophones), two of the performances are by contemporary classical music specialists Joshua Hyde (alto saxophone) and Alana Blackburn (tenor recorder), both of whom had worked with the software under my

[63] http://interecords.com/

guidance over an extended period of rehearsals. The remaining two musicians, Antoine Läng (voice) and Evan Dorrian (drums), are from freely improvised and jazz musical backgrounds, with neither musician having rehearsed with the software prior to their recorded live performances. Of the six tracks on the EP, half were recorded during live performance and half in the studio, two made use of cumulative databases of pre-recorded material and two more include the addition of multi-source session databases.

8.3 Software distribution and communication

The distribution and communication of the *_derivations* system has been an important part of this project. Although the system reflects my idiosyncratic musical concerns and has evolved as a part of my personal performance practice, I have been interested in finding ways of engaging performing musicians with interactive computer music tools, and making such a complex software system approachable and useable for non-programmers. Since 2013, the *_derivations* software has been made available for free download via the dedicated website http://derivations.net (see Appendix D for more information). Along with the release of the software, a series of instructional videos were developed in order to clearly demonstrate and explain the software's operation to interested users (see Appendices E and J). Emanating from within my personal creative practice, efforts made to distribute the software and to communicate its functionalities reflect a concern for openness, and for sharing of musical and technical ideas. Whilst the software was not designed and tested as a typical 'end-user' software system, it is the software's ambiguous identity as part musical work, part autonomous performance partner and part software tool that has led me to release it to the wider public.

Since the software's release, numerous third-party produced performances and releases have been made with the software. Besides occasional technical questions and troubleshooting requests from users, these third-parties working with *_derivations* have for the most part had no direct contact with myself, engaging with the project entirely online (third-party releases are detailed in Appendix H). To date, the software continues to be used by musicians across the globe. At the time of writing, a musician in Sheffield, UK has recently posted an improvisation with Shakuhachi and *_derivations* to SoundCloud[64],

[64] Multi-instrumentalist Hervé Perez's latest improvisation with *_derivations*: https://soundcloud.com/sndsukinspook/emptysky

whilst another prepares for an improvisation with the system on saxophone for his final Doctoral of Musical Arts recital in Colorado, USA[65]. It is my hope that the *_derivations* system can maintain its presence online as a unique musical artefact, provoking adventurous musicians to develop their own *symbiotic* relationships with the software.

8.4 Ongoing and Future Work

In addition to the practice-based research engaged in throughout this project, my current creative work involves various collaborations and solo compositional endeavours in the area of generative and live electro-acoustic music. Throughout this research I have had the good fortune of meeting and collaborating with a wide range of inspiring and talented musicians and researchers working in the field. I have recently been collaborating with Dr. Oliver Bown (University of NSW, Australia) and Dr. Arne Eigenfeldt (Simon Fraser University, Canada) on a project for collaborative 'musical metacreation'. The *Musebot* project invites musicians and researchers to build algorithmic software systems that can work together to create novel, original music through a networked form of musical generativity.[66] These musical robots, or *musebots*, are tasked with responding to each other and their environment, enabling them to contribute meaningfully to an ensemble interaction without direct control by a human operator. Building upon the skills learned in the creation of the *_derivations* software, my role in this project has been to help build the initial communication protocol for the networked systems, and to program the 'Musebot Conductor' Max application to synchronise and distribute messages from various networked software systems.

My interest in practice-based research methods has led to recent participation in a workshop on practice-based research at the International Conference on New Interfaces for Musical Expression (held in London, July 2014)[67]. The paper I presented during this workshop, 'Artefact Scripts and the Performer-Developer', was selected for further publication in a forthcoming issue of the Leonardo Transactions Journal, due for

---

[65] Saxophonist Paul Zaborac plans a performance with *_derivations* at Boulder College of Music on October 24th, 2015.

[66] For more info on the *Musebot* project see Bown et al. (2015) and Eigenfeldt et a. (2015). Media can be viewed at http://metacreation.net/musebots/

[67] http://www.creativityandcognition.com/NIMEWorkshop/

publication in January 2016. The theoretical ideas presented in this paper, with particular concern for the role of the performer-developer, formed the basis of the reflection contained in Chapter 7 of this thesis (see Section 7.2). Whilst the present research is based upon self-reflective practice, I am interested in pursuing further qualitative research methods for investigating the diverse practices of computer music practitioners identifying as performer-developers. In this future research I intend upon applying the theoretical perspectives outlined by Pickering (1995; 2008), Latour (1990; 1992, 1994) and McLean (2011; 2010) to further understand the unique entanglement that exists between human and machine agencies in the development of human-machine performance practices.

8.5 Final thoughts

In the development of idiosyncratic interactive musical systems, a unique and *symbiotic* relationship develops between a musician and their code. For the system designer, creating a performance artefact is a highly complex process that unites human and material agencies in the pursuit of an emergent yet indefinable end goal. For the researcher interested in the development of new modes of artistic expression, understanding this process further is a complex task. However, it is from within this rich and complex space that I believe some of the most interesting research into new methods, techniques and performance practices will be found. As has been reflected upon in this thesis, the entanglement between human and machine agencies has defined my artistic practice as a performer-developer. In following a self-reflective approach to research enquiry, I have been able to surface and engage with salient research themes related to human-machine performance. In order to understand the significance of new tools and related performance practices, I believe that computer music research must first be articulated from the perspective of artists innovating in this space, giving voice to the valuable insider knowledge accumulated through deeply *mangled* human-machine creative practices. It is my hope that more performer-developers will consider qualitative, self-reflective methodologies in their research, contributing to the diversity of perspectives on the creation and use of today's and most exciting interactive musical artefacts.

Appendix A - _derivations software

The software developed throughout the duration of this doctoral project is presented as a significant contribution to knowledge, and is therefore included in the submission materials alongside this thesis. The main contribution of this work is the *_derivations* interactive performance system, distributed both as a standalone Mac application and as two separate libraries of Max patchers and abstractions. In addition to these three distributions of the *_derivations* software, formative software systems discussed in Chapter 4 have also been included in the submission materials, and are listed in Appendix G as Formative Software.

_derivations distributions:

As detailed throughout much of this thesis, the *_derivations* software has been a major creative focus of this research. Building upon creative programming techniques initiated in 2010/11, the *_derivations* software was made freely available to the public to download upon the launch of the *_derivations* website (http://derivations.net) in June 2013 (for more information about the *_derivations* website see Appendix D). In order to account for the widest possible variety of use cases, *_derivations* has been compiled into three separate distributions. Below I outline the three distributions included in the submission materials and the differences between them.

_derivations 'SuperVP' distribution (Max library)

Version included in submission materials: *_derivations-superVP-v1.08.zip*

As outlined in Chapter 6, a choice was made during the development of the software to employ a select few high quality analysis and processing tools developed for the Max environment by the Institute de Recherche et Coordination Acoustique/Musique (IRCAM). These tools include the *supervp.scrub~* external (from the 'SuperVP for Max' collection), and the *iana~* external (from the 'Max Sound Box' collection) (Todoroff, Daubresse & Fineberg 1995), both of which are available via IRCAM's subscription service 'Forumnet' (IRCAM 2015).

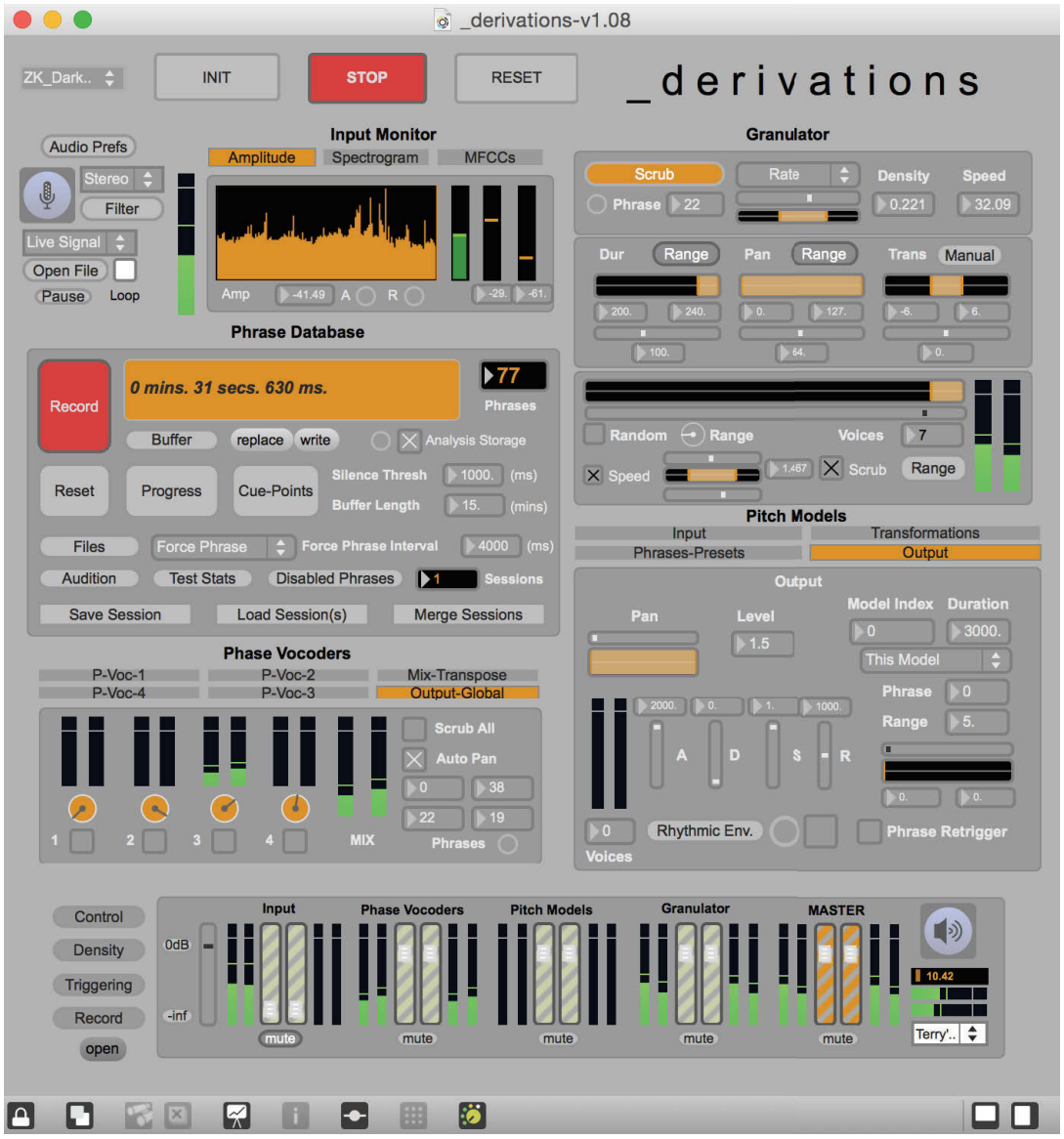


Figure 48: Screenshot of _derivations' graphical user interface (v1.08)

Due to the inherent licencing approach of these two objects, only users with a paid subscription to Forumnet are able to make full use of this *_derivations* distribution. However, this is the software distribution that provides the highest audio quality and system performance. For both my own performances and those in which I am in close collaboration with another performer, this is the distribution that is used. All audiovisual material submitted with this thesis was therefore produced using this distribution.

_derivations 'standard' distribution (Max library)

Version included in submission materials: *_derivations-v1.08.zip*

To counter the limitations of the previous distribution method, a 'standard' distribution of *_derivations* was created to enable any user of the Max environment to download and run a fully functional version of the software. To achieve this, this distribution replaced the analysis and processing tools of the SuperVP distribution with alternative objects and abstractions unhindered by licence restrictions. The *supervp.scrub~* object was replaced with the original phase vocoder abstraction used in my work, an implementation of Dudas and Lippe's phase vocoder for the Max environment (Dudas & Lippe 2006, 2007). Whilst providing an efficient replacement of the *supervp.scrub~* external, this abstraction unfortunately does not match the sound quality of IRCAM's SuperVP algorithms as it is suffers from audible 'transient smearing' artefacts common to other phase vocoders. The *iana~* external was replaced with Miller Puckette's excellent sinusoidal and pitch tracking external object *sigmund~* (ported to Max from Pure Data by Miller Puckette, Cort Lippe and Ted Apel).

_derivations 'standalone' distribution (Mac application)

Version included in submission materials: *_derivations-v1.08-standalone.zip*

Finally, a standalone build of _derivations was made in order to appeal to potential users without access to the Max software, and with no desire to make use of the accompanying source code. Given that the proprietary IRCAM objects discussed previously are rendered unusable in a standalone Max build, this standalone distribution is compiled from the source code of the standard distribution described above.

Figure 49: Screenshot of _derivations 'standalone' disk image (v1.07)

In the Max environment, Max patchers are compiled into standalone applications from within the Max application itself. This process creates a standard Mac application package that includes a non-editable Max 'collective' file (extension .mxf) and a version of the Max runtime software with which to run this collective. The *_derivations* standalone application also includes necessary dependencies within this application package (impulse response files and external objects). The application package itself resides in a parent directory along with other necessary folders, and is built into a mountable disk image for ease of distribution (see Figure 49 above). With over one thousand downloads as of October 2015, the standalone distribution of *_derivations* is the most downloaded distribution of the three made available on the *_derivations* website.

Appendix B - Musical releases

Recorded improvisations with *_derivations* have featured on two discs recently released on the 'Integrated Records' label (Carey & Hyde 2014). These releases are detailed below:

_derivations | human-machine improvisations (February 2014)

Figure 50: *derivations | human-machine improvisations* cover art

6-track EP released on Integrated Records – Sydney/Paris. A collection of live and studio improvisations with the *_derivations* software, recorded between 2011 and 2013.

Ben Carey - saxophone/electronics, Joshua Hyde – alto saxophone, Antoine Läng - voice, Evan Dorrian - drums, Alana Blackburn – tenor recorder.

The physical CD release of this EP is included in the submission materials of this thesis.

http://interecords.com/products/6807406-_derivations-human-machine-improvisations

Joshua Hyde – Berio, Sotelo, Quislant, Parra, Carey (January 2015)

Figure 51 Joshua Hyde – Berio, Sotelo, Quislant, Parra, Carey cover art

6-track released on Integrated Records – Sydney/Paris. Debut album of Paris-based Australian saxophonist Joshua Hyde. *_derivations* features on the final track with tenor saxophone

Joshua Hyde - saxophones, Ben Carey - electronics/production.

A digital audio file of Track 6 from this release, titled *'_derivations'*, is included in the submission materials of this thesis.

http://interecords.com/products/6707299-joshua-hyde-berio-sotelo-parra

Appendix C - Performance documentation

Numerous live performances and demonstrations using *_derivations* and other formative software have been documented throughout the duration of this doctoral project. For a full list of performances and other significant events the reader is referred to the Event Timeline found in Appendix F. The most significant performance documentation has been included in the submission materials and is detailed below.

_derivations | Ben Carey | MuMe 2013 (June 2013)

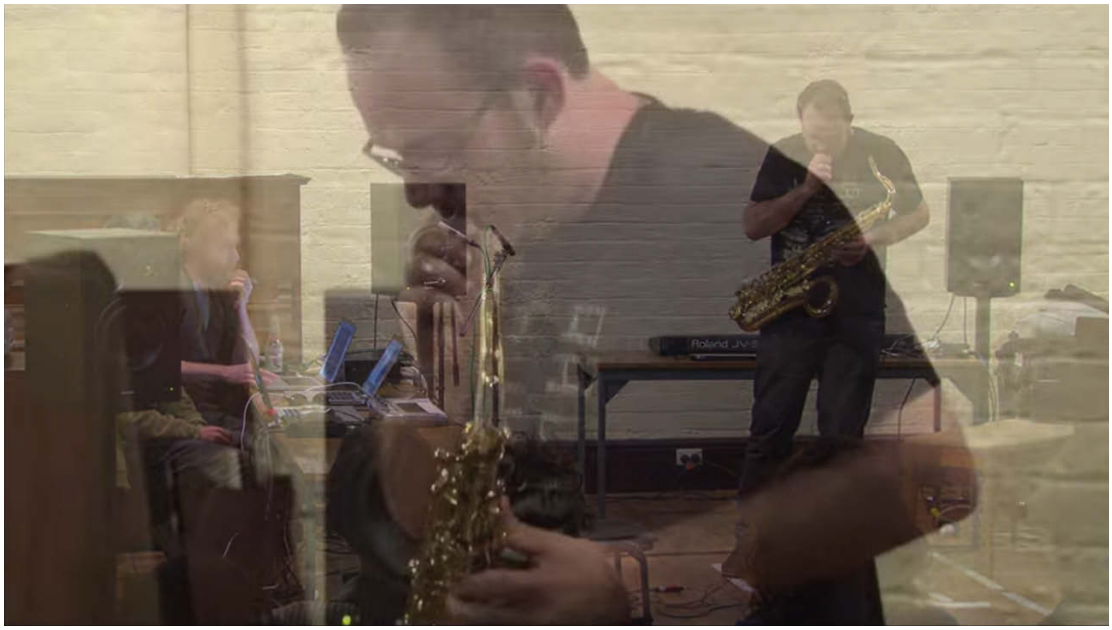


Figure 52: Screenshot from '_derivations | Ben Carey | MuMe 2013'

Live performance footage of myself performing with _derivations on tenor saxophone at the Musical Metacreation Weekend, presented in conjunction with the International Symposium of Electronic Art (ISEA) at the University of Sydney, June 15th 2013. _derivations was programmed in the first concert of the workshop titled 'Improvising Algorithms', one of a series of events curated as a part of this international workshop.

This video can be streamed at https://www.youtube.com/watch?v=GHxHumlCZOQ

Filmed by Paul Gough, recorded by Oliver Bown.

Piano-computer dance: Zubin Kanga & Ben Carey's _derivations (May 2015)

Figure 53: Screenshot from 'Piano-computer dance: Zubin Kanga & Ben Carey's _derivations'

Performance of _derivations by pianist Zubin Kanga for broadcast on ABC Radio National's 'The Music Show' with Andrew Ford, May 2nd, 2015. This performance was filmed and recorded 'as live' at the Australian Broadcasting Corporation (ABC) in Sydney, Australia on May 1st 2015. The broadcast was organised as promotion for Zubin Kanga's piano and electronics program 'Dark Twin' that toured Australia in May 2015.

This video can be streamed at https://www.youtube.com/watch?v=PmUCGbcrGuw

Film and sound recording – Australian Broadcasting Corporation.

Joshua Hyde and _derivations | IRCAM Live @ La Gaité Lyrique (December 2012)

Figure 54: Screenshot of 'Joshua Hyde and _derivations | IRCAM Live @ La Gaité Lyrique'

Live performance footage of Joshua Hyde performing with _derivations on alto saxophone. This performance was given on November 28th, 2012 at the 'IRCAM Live' concert organised during the 2012 IRCAM workshops at La Gaité Lyrique in Paris, France. The performance was presented in 6-channel surround sound, and has been down-mixed to stereo for this video.

This video can be streamed at the following URL: https://vimeo.com/55189188

Technical production - Cyrille Brissot (IRCAM) and Thierry Vitale (La Gaité Lyrique), Sound recording - Jean-Marc Harel (La Gaité Lyrique), Film recording - Alex Boulic (Je&Enjeux), DAFACT.

Appendix D - Website

The *_derivations* website was launched in June 2013 at the URL http://derivations.net. The website is published using Wordpress (Wordpress Foundation 2015). The site includes software downloads, video documentation, links to reading material, recent performances and audio-visual examples of the *_derivations* software in performance.

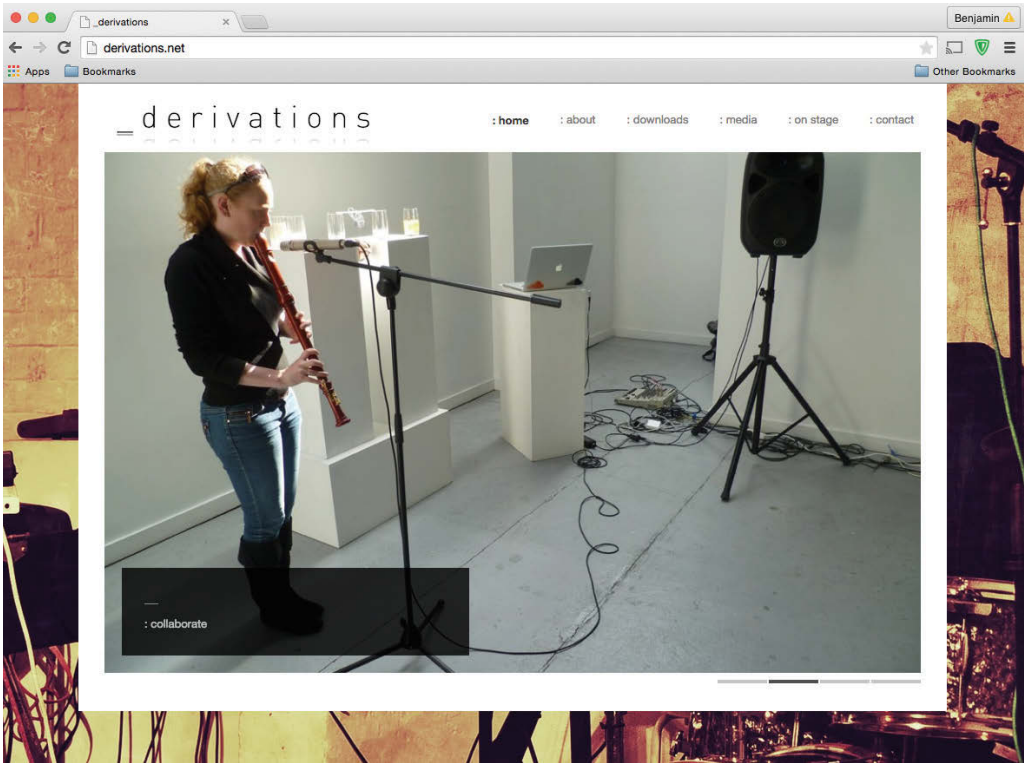


Figure 55: Screenshot of derivations.net landing page featuring Alana Blackburn in rehearsal

To launch of the website, the URL was disseminated via social media (Facebook, Twitter) to my personal and professional networks. Further dissemination of the project and website was achieved by sharing the URL to relevant Facebook groups such as Max/MSP[68], Electroacoustic Composers[69] and Modisti[70]. Furthermore, a short, spoken announcement was recorded and uploaded to the Soundcloud community website and shared to relevant groups[71]. This online service has been of considerable use to me

[68] https://www.facebook.com/groups/2209224391/

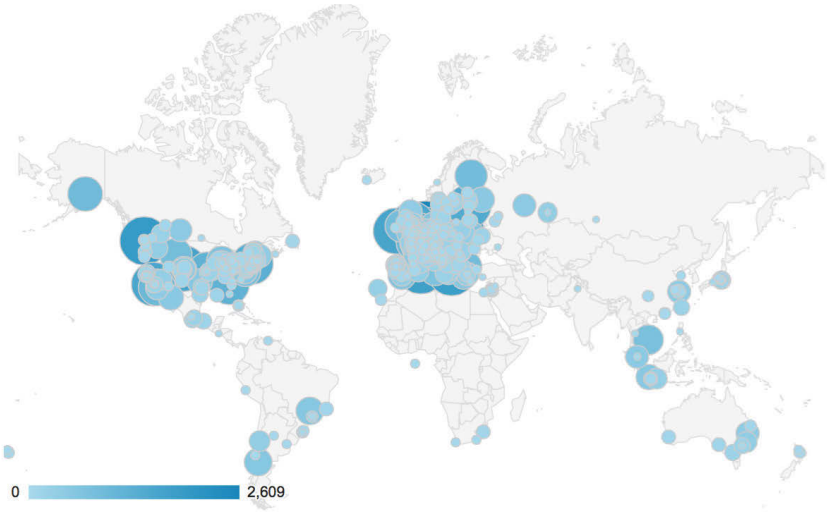
[69] https://www.facebook.com/groups/22370892781/

[70] https://www.facebook.com/groups/modisti/

[71] https://soundcloud.com/emeidos/announcement-derivations

throughout the development of my creative practice. In addition to direct social media promotion, the *_derivations* project and website were disseminated to the Max/MSP community via the 'projects' section provided for registered users of Cycling74.com.[72]

Such targeted dissemination of the project has enabled derivations.net to be accessed by a wide variety of users. According to user data obtained through Google Analytics service (Google Inc 2015), since launching in June 2013 the site has been accessed some 4430 times from ninety-five separate countries. Cumulatively, the three versions of derivations have been downloaded from the site approximately 2200 times. Figure 56 below displays a report from Google Analytics across the lifespan of the site, displaying the top ten cities accessed in the past 24 months. The dots on the map represent average comparative session durations from each location.



City	Sessions	Sessions
	4,430 % of Total: 100.00% (4,430)	4,430 % of Total: 100.00% (4,430)
1. (not set)	419	9.46%
2. Sydney	331	7.47%
3. Paris	149	3.36%
4. London	115	2.60%
5. Berlin	96	2.17%
6. Melbourne	95	2.14%
7. Basel	83	1.87%
8. (not set)	71	1.60%
9. Sao Paulo	70	1.58%
10. Cologne	69	1.56%

Contribution to total: Sessions

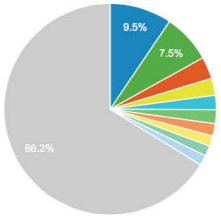


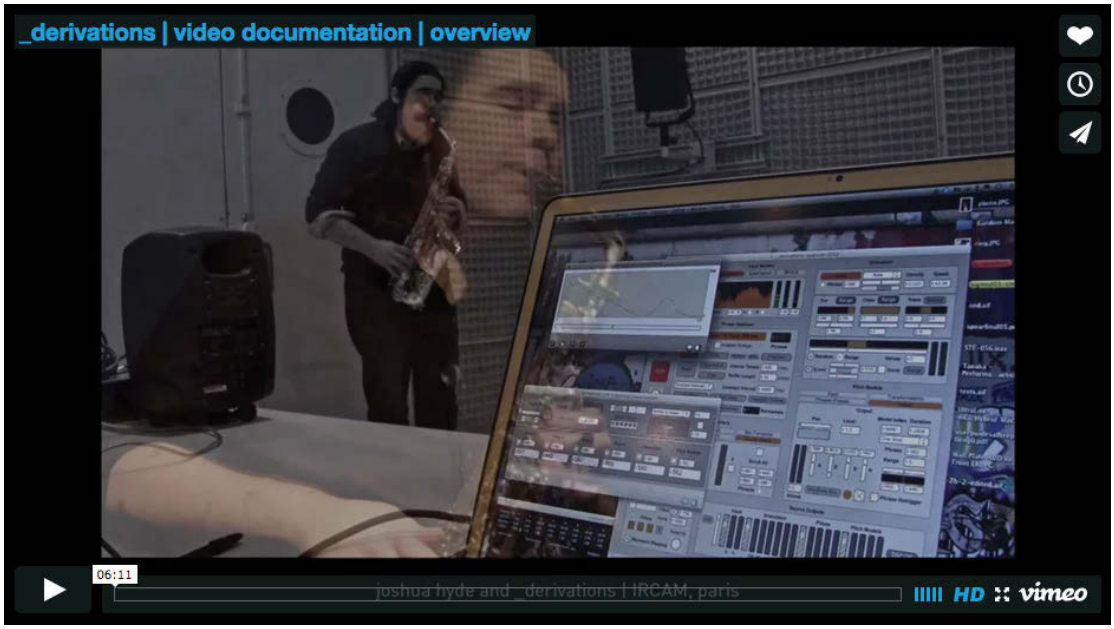
Figure 56: Google Analytics report for derivations.net in the period June 28th, 2013 – June 7th, 2015

[72] https://cycling74.com/project/_derivations/

Appendix E - _derivations Video Documentation[73]

The purpose of the *_derivations* website was to disseminate the *_derivations* software to as wide an audience of potential users as possible. In order to achieve this, an integral part of this distribution process was to effectively communicate the core functionality and creative potential of the software to potential end users. This was therefore considered a functional task for documenting the software, as well as a communications and marketing task. Instead of creating a standard text document or manual, it was decided that a series of screencast videos would be made to allow users to see and hear the various aspects of the software in action. This choice was inspired by some recent approaches to the communication of computer music systems, and especially the approach taken by Rui Penha in the distribution of his *spatium* spatialisation tools (Penha 2013; Penha & Oliveira 2013). These videos were filmed and produced using Telestream™'s Screenflow Software between June 28th and July 22nd 2013 (Telestream 2012). They are hosted on Vimeo.com and embedded on the *_derivations* website at the following URL: http://derivations.net/about/video-documentation/

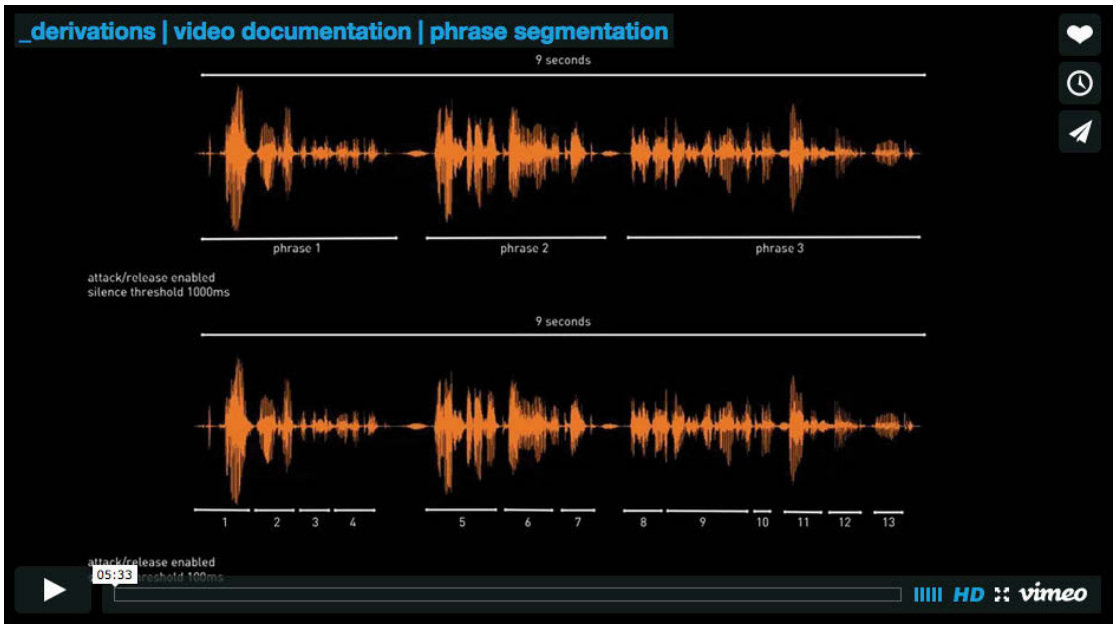
Video 1: Overview



Duration: 6 minutes 11 seconds

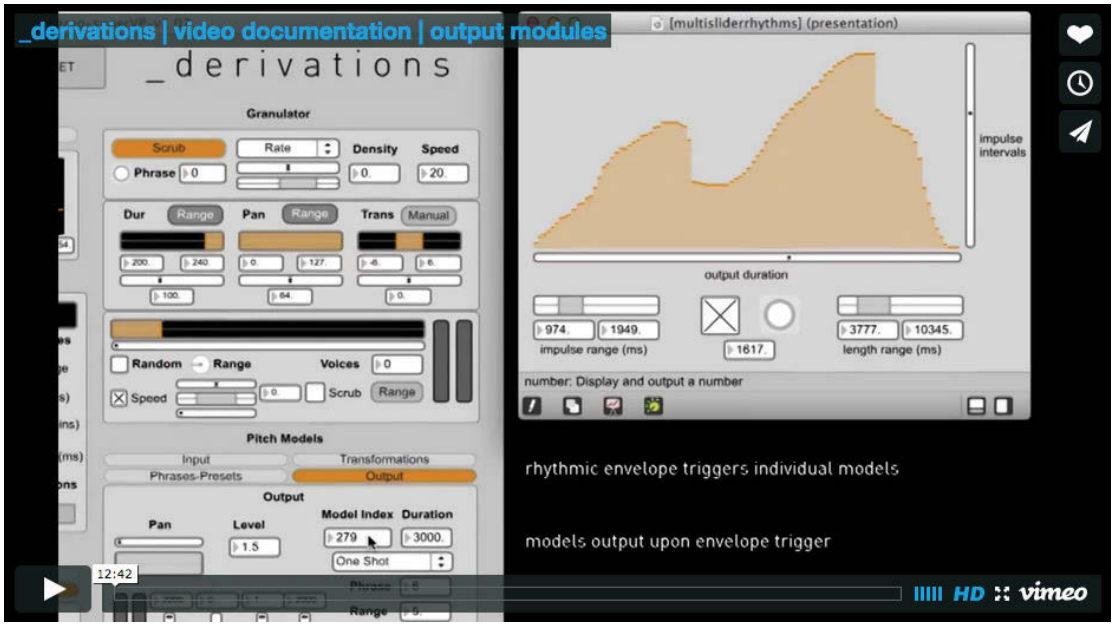
[73] Transcripts of voiceovers from these videos are provided in Appendix J

Video 2: Phrase Segmentation



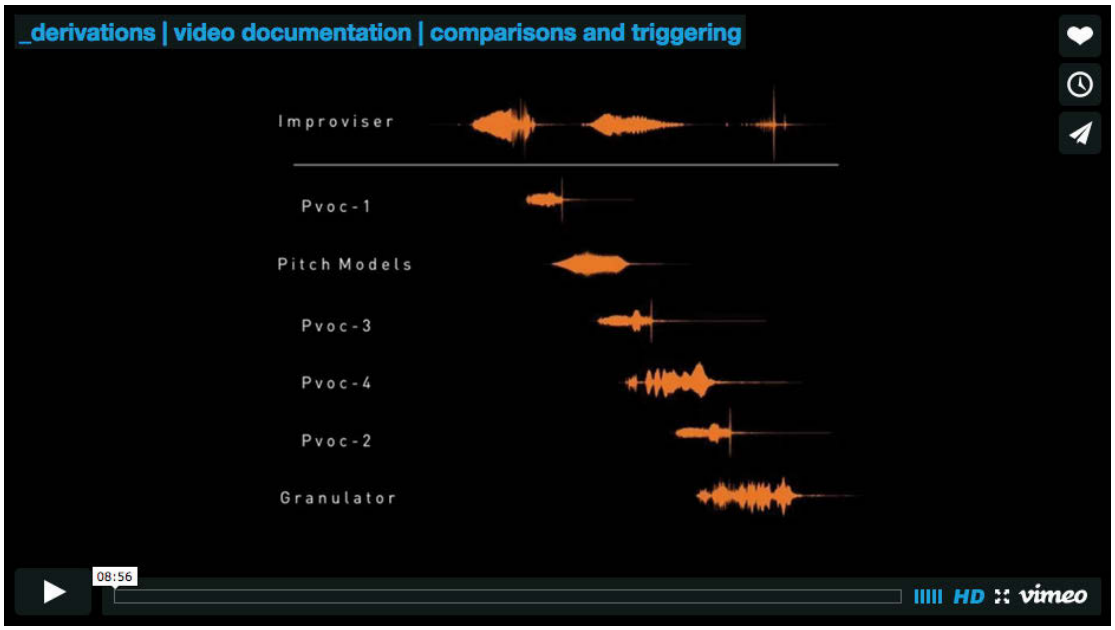
Duration: 5 minutes 33 seconds

Video 3: Output Modules



Duration: 12 minutes 42 seconds

Video 4: Comparisons and Triggering



Duration: 8 minutes 56 seconds

Video 5: Session Databases

Duration: 10 minutes 3 seconds

Appendix F - Event and Patch Timelines

Event Timeline

The following timeline details performances, broadcasts, interviews, recordings and other events that were significant in the development of the creative work discussed in this thesis. The *event* column provides a basic description of the event, including the location, whilst the *activity* column lists the type of event (performance, communication, development). The *significance* column gives some further information about the event, including the personnel involved, and any issues encountered. The *outcome* column is included to assess whether or not the event was deemed successful or not. For example, some performances were abandoned entirely due to technical issues during sound check/rehearsal, whilst others were deemed unsuccessful due to technical failures encountered during the performance. The final *present* column details whether or not I was personally present at each event, given that many of performances were run interstate or overseas and without my direct participation.

Date	Event	Activity	Significance	Outcome	Present
5-May-10	Presentation of Multiple Players (Sydney)	Communication	Presentation of working interactive system - multiple players at USYD. Alto saxophone demo	Success	Yes
22-Feb-11	IRCAM Course (Paris)	Development	Intense early development period, introduction to new approaches and objects	Success	Yes
11-Jun-11	NMN Mini-series performance (Sydney)	Performance (me)	First public performance - spatialisation and gain issues. Tenor saxophone	Failure	Yes

8-Jul-11	ACMC'11 (Auckland)	Performance (me)/ Communication	Second public performance - first research presentation. Tenor saxophone	Success	Yes
1-Oct-11	Tactility – Alana Blackburn (Sydney)	Performance (others)	First performance with another performer/using rehearsal database. Tenor recorder	Success	Yes
18-Nov-11	diffuse @ UTS (Sydney)	Performance (me)	Performance on soprano using a rehearsal database	Success	Yes
8-Mar-12	Prolegomena II – Alexander Berne (NYC)	Performance (others)	NYC Dance performance, Alexander Berne uses customised version of _derivations. Saxophone, custom-made wind instruments	Success	No
21-Apr-12	Algorithmic Hacktogether – Evan Dorrian (Sydney)	Development/ Performance (others)	First performance using self-referencing algorithm. Drum kit	Success	Yes
21-May-12	NIME 2012 (Ann Arbor)	Communication	Paper/Poster - first publication on _derivations	N/A	Yes
1-Jun-12	dBale – Switzerland (Basel)	Performance (me/others)/ Communication	First performance with large multi-corpus database, first duo performance, artist talk. Tenor saxophone	Success	Yes
21-Jun-12	UTS diffuse – with Oliver Bown (Sydney)	Performance (me)	Zamyatin/derivations duo, integrating two software systems in performance. Tenor saxophone/computer	Success	Yes

14-Jul-12	ACMC'12 – Alana Blackburn (Brisbane)	Performance (others)	Performance with multi-database. Tenor recorder	Success	Yes
18-Aug-12	Biome Symposium – Antoine Läng (Sydney)	Performance (others)	Multi-database. Voice.	Success	Yes
23-Aug-12	NMN Mini-series – Alana Blackburn (Sydney)	Performance (others)	Multi-corpus database. Tenor recorder	Success	Yes
1-Nov-12	IRCAM @ Gaité Lyrique – Joshua Hyde (Paris)	Performance (others)/ Communication	Spatialised, multi-database with density trajectory, artist presentation. Alto saxophone	Success	Yes
8-Jun-13	Covalent performance @ Vivid – Zane Banks (Sydney)	Performance (others)	Multi-database performance on banjo, technical failure. Zane Banks – banjo.	Failure	Yes
15-Jun-13	MuMe-WE (Sydney)	Performance (me)	Performance using live sampling, size of space presented challenges for analysis. Tenor saxophone	Success	Yes
28-Jun-13	derivations.net launch	Communication	Launch of derivations.net website	N/A	N/A
12-Jul-13	Australasian Saxophone & Clarinet Conference 2013 – Joshua Hyde	Performance (others)	Performance by Joshua Hyde with a multi-database. Alto saxophone	Success	Present
22-Jul-13	_derivations video documentation	Communication	Creation and distribution of video documentation	N/A	N/A

7-Oct-13	Joshua Hyde/Noam Bierstone Duo at RNCM (scapegoat) (Manchester)	Performance (others)	Duo performance, technical failure due to threshold settings. Performance abandoned. Percussion and saxophone	Failure	No
15-Feb-14	'_derivations | human-machine improvisations' release	Recording	Collection of performances curated into an EP on Integrated Records	N/A	N/A
23-Mar-14	Josh Hyde @ NASA Conference (Urbana)	Performance (others)	Multi-database performance. Alto saxophone	Success	No
26-Mar-14	Josh Hyde @ Eastman School of Music (Rochester)	Performance (others)	Multi-database performance. Alto saxophone	Success	No
26-Apr-14	Interview for CutCommon Mag	Communication	Interview with Sam Gillies for CutCommonmag.com	N/A	N/A
12-Jun-14	Live on Ears Have Ears FBi Radio (Sydney)	Performance (me)/ Communication	Interview/ performance – Brooke Olsen. Performance with database on alto saxophone	Success	Yes
19-Jun-14	C&C 2013 (Sydney)	Performance (me)	Operator error - changed parameters at the last minute. Tenor saxophone	Failure	Yes
25-Jun-14	Interview for Tokafi.com	Communication	Extended interview on aesthetics with Tobias Fischer of tokafi.com	N/A	N/A
2-Jul-14	NIME 2014 (London)	Performance (me)	Technical failure due to room conditions. Alto saxophone	Failure	Yes

1-Aug-14	XU(E) Haiku EP release (Italy)	Recording (others)	Self-directed EP using _derivations software with no input from myself	N/A	N/A
24-Aug-14	Eric Honour @ KCEMA Happy Hour (Kansas)	Performance (others)	Rehearsal database performance, alto saxophone - customised/updated software	Success	No
24-Sep-14	Array Ensemble performance (Sydney)	Performance (me)	Rehearsal database performance - group performance - began with alto saxophone solo, morphed into group improv	Success	Yes
20-Nov-14	scapegoat - dedans/dehors (Paris)	Performance (others)	Rehearsal database performance + crowd noise - percussion and saxophone	Success	No
4-Oct-14	SoundHub Interview (London)	Communication	Interview about _derivations EP on Resonance 104.4 with Elo Masing	Success	Yes
5-Jan-15	Joshua Hyde - CD release featuring _derivations	Recording	Studio recording of _derivations with tenor saxophone with performance database	Success	Yes
11-Jan-15	Joshua Hyde - soundinitiative speakeasy - CD launch (Paris)	Performance (others)	Rehearsal database performance – alto saxophone	Success	No
6-Feb-15	Conference on Artificial Life and Computational Intelligence	Performance (me)	Rehearsal database performance with alto saxophone. Lack of foldback - shorter	Success	Yes

	(Newcastle, Australia)		performance		
14-Apr-15	Tate Carson @ Open Ears (New Orleans)	Performance (others)	Live sampling performance - self directed performance. Solo double bass - (some technical help via email)	Success	No
2-May-15	ABC RN 'The Music Show' performance/ interview – Zubin Kanga	Performance (others)/ Communication	Live sampling performance. Short interview with Andrew Ford on the software. Piano	Success	Yes
8-May-15	Zubin Kanga - Dark Twin Tour (Melbourne)	Performance (others)	Piano performance with live sampling only - first performance with piano	Success	Yes
9-May-15	Zubin Kanga - Dark Twin Tour (Sydney)	Performance (others)	Piano performance with live sampling only - feedback issues in recorded phrases	Success	Yes
15-May-15	scapegoat duo - tour (Toronto)	Performance (others)	Rehearsal database performance with percussion and saxophone. Routing issues. Not performed	Failure	No
16-May-15	scapegoat duo - tour (Toronto)	Performance (others)	Rehearsal database performance with percussion and saxophone.	Success	No
19-May-15	scapegoat duo - tour (Detroit)	Performance (others)	Rehearsal database performance with six local players - bass, cello, percussion, saxophones	Success	No

21-May-15	scapegoat duo - tour (Chicago)	Performance (others)	Rehearsal database performance with percussion and saxophone.	Success	No
22-May-15	scapegoat duo - tour (Chicago)	Performance (others)	Rehearsal database performance with percussion and saxophone.	Success	No
22-May-15	Zubin Kanga - Dark Twin Tour (Perth)	Performance (others)	Piano performance with live sampling only	Success	Yes
24-May-15	scapegoat duo - tour (NYC)	Performance (others)	Unknown technical problem - no sound output from the software in soundcheck - scrapped from the program	Failure	No

Patch Timeline

The following 'patch timeline' details the chronological development of the software developed throughout this doctoral project. This timeline includes the formative software discussed in Chapters 4 and 5 and listed in Appendix G, and complete versions of the *_derivations* software. In addition, it details patches that were not discussed directly in the thesis, but contributed to the overall trajectory of the software discussed throughout Chapters 4, 5 and 6. For these particular patches, notes have been included to outline the functionality of these formative patches. It must also be noted that incremental changes to the *_derivations* software were made between 2012 and 2015 that did not necessitate the creation of a new version number, nor newly developed patches. From 2012 onwards this system was iteratively refined, and no formal version control was developed.

Date	**Patch/Software Version**	**Notes**
Formative/unreleased software		

2010		
January	*newaudiotracking.maxpat*	
	acceldeceldetect.maxpat	
(- September)	*findthatrhythm2.maxpat*	
March	*acceldetectthreshold.maxpat*	
	deceldetectthreshold.maxpat	
April	*elasticitystorage.maxpat*	
	interactiveoptions.maxpat	
	audiotomiditest.maxpat	
May	*dataatintervals.maxpat*	
	durationalprob.maxpat	
	tripartitemarkovia.maxpat	
November	*4buff-pvoc.maxpat*	
December	*Live-processing-1.maxpat*	
2011		
January (-March)	*pitchmodels.maxpat*	
February	*phrase-player-GUI.maxpat*	
(- March)	*big-buff.maxpat, big-buff-phrases.maxpat*	First implementation of long buffer live-sampling
March (-May)	*granulator-phrase-locked.maxpat*	First implementation of the granulator module
March (- April)	*_derivations.maxpat*	First instance of _derivations as integrated system
	yin-follow.maxpat, yin-follow-tabs.maxpat	First dedicated analysis module for derivations
	zsa-running-stats.maxpat	First 'matching' tests
March	*_derivations_SPAT.maxpat*	Porting _derivations to multichannel version
(- July 2012)	*_derivations-CRL.maxpat*	Formalising module control from phrase choice in _derivations
April	*analyzer~tests.maxpat*	Testing Jehan's analyzer~ external
	zsa.keys.test.maxpat, find-in-list.maxpat, list-comparisons.maxpat, list-distance-test.maxpat	Testing list comparisons for *_derivations'* phrase matching algorithm
	analyzer-follow-tabs.maxpat	Further testing of Jehan's analyzer~ external for

		phrase matching
	zsa-descriptor-matching.maxpat	Using zsa.dist for matching lists in the *phrase matching* algorithms
	__derivations-analyzer.maxpat	Final implementation of Jehan's analzyer~ into parent patch
	stat-matching.maxpat, stat-matching-2.maxpat, _derivations-CTRL-stats.maxpat	Implementation of standard deviation and mean for phrase matching
	4buff-pvoc-RT.maxpat	Implmentation of Dudas and Lippe's phase vocoder algorithm
May	*_derivations-rehearsal-buffer.maxpat*	First implementation of rehearsal/session database concept in _derivations.
	4buff-pvoc-superVP.maxpat	Use of IRCAM's superVP.scrub~ external for phase vocoder module
June	*_derivations-analyzer-SPAT-8ch.maxpat*	8 channel version of *_derivations* implemented using IRCAM's SPAT external objects
	_derivations-rehearsal-buffer-2.maxpat	Phrase database improvements. Use of 'constant interval' segmentation, etc.
August	*enable-disable-phrase.maxpat*	Testing 'disabling phrases' concept in _derivations' rehearsal database
	merge-data-tests.maxpat	Testing 'merging' databases
September	*_derivations-info-for-automode.maxpat*	Sketching ideas for a self-referencing algorithm
2012		
January (- Febraury)	*_derivations-DL-stereo.maxpat*	Adaptation of *_derivations* for a performance with Alexander Berne with dancers.
March	*simple-pm-triggering.maxpat*	Testing implementation of

		'One Shot' mode for pitch models module
(- April)	*emergent-phrase-relations.maxpat*	Further test in developing a self-referencing algorithm. First conception of *_derivations* modules as independent 'agents'
May	*_derivations-self-referncing.maxpat*	Implementation of self-referencing algorithm in *_derivations*
September	*_derivations-CTRL-MFCC.maxpat, _derivations-self-referencing-MFCCs.maxpat, _derivations-MFCC-STATS.maxpat, zsa-mfcc-follow-tabs.maxpat, zsa-dist-tests.maxpat, _deri-mfcc-test.maxpat, _derivations-mfccs.maxpat, Euclidean tests.maxpat, __derivations-stereo-zsa-tests.maxpat, zsa-follow-tabs.maxpat, zsa~follow-tabs-source-2.maxpat, zsa-test-deri.maxpat, _derivations-zsa-pfft.maxpat*	Test patches and finalised implementation of the use of MFCCs in *_derivations*
_derivations **software versions**		
(complete packages/standalones)		
2012		
January	v.1.01	Not publically released
	v.1.02	Not publically released
February	v.1.03	Not publically released
	v.1.04	Not publically released
2013		
February	v.1.05	Not publically released
April	v.1.06	First public release (released in June 2013)
June	v.1.07	Public release
2015		
March	v.1.08	Public release

Appendix G - Formative Software

The following software developed in the Max environment is described in Chapters 4 and 5 of this thesis. These software programs are considered formative to the significant work presented in the _derivations software, and are provided in the submission materials as individual Max patchers and Max libraries. Footnotes refer the reader to the relevant sections in this thesis in which these programs are described. Video demonstrations of the software are also listed in footnotes.

Input analysis and segmentation[74]

newaudiotracking.maxpat

Temporal pattern recognition[75]

acceldeceldetect.maxpat
acceldeceldetectthreshold.maxpat
findthatrhythm2.maxpat

Data sampling techniques[76]

elasticitystorage.maxpat
dataatintervals.maxpat
interactiveoptions.maxpat
audiotomiditest.maxpat

Probabilistic methods[77]

DurationalProb.maxpat
tripartitemarkovia.maxpat[78]

---

[74] Described in Section 4.2.1

[75] Described in Section 4.2.2

[76] Described in Section 4.2.3

[77] Described in Section 4.2.4

4-buff-pvoc.maxpat[79]

[78] Demonstration videos of this software can be viewed at https://vimeo.com/19863192 and https://www.youtube.com/watch?v=byZjyBiehmk

[79] Video of this software can be viewed at https://vimeo.com/16791411

Appendix H - Third-party produced releases

Given the free availability of the *_derivations* software via derivations.net, musicians from across the world are free to use the software for their own musical output. The releases below were produced and released online without any input from myself:

August 2014: *Random Thoughts: solo contrabass clarinet improvisations*

8-track album released on Bangsnap Records – Denver, Colorado. _derivations used on various tracks alongside other electro-acoustic processes.

Paul Milmitsch – contrabass clarinet/production.

Available to via Bandcamp: https://bangsnap.bandcamp.com/album/random-thoughts-solo-contrabass-clarinet-june-2014

August 2014: *Flowers Recollect EP*

3-track EP self-released EP – Cremona, Italy. _derivations used to generate source material for electroacoustic productions.

Xu(e): Nicola Fornasari (electronics) and Andrea Poli (electronics)

Available via Bandcamp: https://xu3music.bandcamp.com/album/flowers-recollect-ep

February 2014: *Nepenthes Project*

6-track album released on Bangsnap Records – Denver, Colorado. _derivations used as part of an expanded electroacoustic improvisation setup.

Kurt Bauer – percussion/laptop, Paul Milmitsch – bass clarinet

Available via Bandcamp: https://bangsnap.bandcamp.com/album/nepenthes-project

Appendix I - Online content

Throughout the duration of this project, a variety of audiovisual and textual media has been published online documenting *_derivations* and other formative projects. In addition to self-produced content, below I also detail content disseminated by third parties making use of my software, as well as published interviews, reviews and other media pertaining to the creative work generated throughout this doctoral project.

Audiovisual – self-produced

Self-produced audiovisual content has been largely disseminated via the media-hosting services SoundCloud (SoundCloud 2015) and Vimeo (Vimeo 2015). Much of this content was posted to social media websites such as Facebook and Twitter as well as being embedded on my website at http://bencarey.net and its accompanying blog http://blog.bencarey.net.

I have collated albums/sets of content disseminated relating to my creative work at the following URLs:

Soundcloud: https://soundcloud.com/emeidos/sets/derivations-content
Vimeo: https://vimeo.com/album/3429252

Included in the Vimeo album are five videos developed as 'Video Documentation' for the *_derivations* system as discussed in Appendix E. Transcripts of these videos are provided below in Appendix J.

Content hosted on other media-hosting services can be found below:

October 2014 – audioBoom: Interview with Elo Masing – Soundhub on Resonance FM: https://audioboom.com/boos/2536058-ben-carey-interview-with-elo-masing-soundhub-on-resonance-fm

Audiovisual – third-party produced

May 2015 – ABC Online – ABC Classic FM: Re-stream of live to air concert from Totally Huge New Music Festival, featuring Zubin Kanga performing with _derivations (Perth, Western Australia):
http://www.abc.net.au/radio/programitem/pgZaG5yEeG?play=true

May 2015 – YouTube – Matthew Daher: Preview video of group improvisation with _derivations at Detroit Contemporary (Detroit, MI) featuring Joshua Hyde - tenor saxophone, Noam Bierstone – Percussion, Kim Sutton – cello, Matthew Daher – percussion, Bubba Ayoub – video:
https://www.youtube.com/watch?v=gjMhETPU588&feature=youtu.be

April 2015 – Soundcloud – Tate Carson: Live recording of live performance with _derivations at Open Ears (New Orleans):
https://soundcloud.com/tate-carson/solo-bass-and-electronics?in=tate-carson/sets/open-ears-live-set

April 2015 – Soundcloud – Xu(e): Electroacoustic piece using _derivations for synthetic/textural elements:
https://soundcloud.com/xu-e/thirstyleavesmusic-xue-in-the-midst-of-the-debris

December 2014 – YouTube – Noam Bierstone: scapegoat / Joshua Hyde + Noam Bierstone – selections from dedans/dehors series including derivations improvisation (Paris):
https://www.youtube.com/watch?v=bFnPR1sRxdA

October 2014 – Soundcloud – Hervé Perez: Shakuhachi improvisation with _derivations and field recordings (Sheffield, UK):
https://soundcloud.com/sndsukinspook/amarashak

June 2014 – Soundcloud – Hervé Perez: Soprano saxophone improvisation with _derivations + field recordings and other fixed media (Sheffield, UK):
https://soundcloud.com/sndsukinspook/808fields

August 2014 – YouTube – Eric Honour: Live performance with _derivations at the KCEMA Happy Hour (Kansas):
https://www.youtube.com/watch?v=D6-QIdTawZQ

July 2013 – YouTube – Joshua Hyde: Live performance with _derivations during the Australasian Saxophone & Clarinet Conference – Sydney Conservatorium (Sydney):
https://www.youtube.com/watch?v=V8pJLWnEIFE

Texts – self-produced

The main outlet for self-produced textual documentation of the creative work discussed in this thesis has been via my personal blog:

http://blog.bencarey.net

I have also written about the _derivations project on the blog of Tobias Reber, *100 Quriky Legs:*

http://tobiasreber.tumblr.com/post/42865806477/derivations-guest-post-by-ben-carey

Texts – third-party

Third party textual content related to the _derivations project and other creative work discussed in this thesis has been published in the form of interviews and reviews of performances and recordings. A list of these resources and accompanying URLs can be found below:

May 2015 – Review: Limelight magazine review of Zubin Kanga's *Dark Twin* program featuring _derivations – Lisa McKenney:
http://www.limelightmagazine.com.au/live-reviews/review-dark-twin-zubin-kanga

May 2015 – Review: Blog review of Zubin Kanga's *Dark Twin* program featuring _derivations – Charles MacInnes:
http://bit.ly/charlesmacinnes_review

May 2015 – Review: Partial Durations blog review of Zubin Kanga's *Dark Twin* program featuring _derivations – Matthew Lorenzon:
http://partialdurations.com/2015/05/12/metropolis-zubin-kanga-dark-twin/

May 2015 – Review: Realtime Arts magazine review of Zubin Kanga's *Dark Twin* program featuring _derivations – Laura Halligan:
http://realtimearts.net/feature/Totally_Huge_New_Music_Festival_2015/11940

May 2015 – Review: Realtime Arts magazine review of Zubin Kanga's *Dark Twin* program featuring _derivations – Alex Turley:
http://realtimearts.net/feature/Totally_Huge_New_Music_Festival_2015/11940

June 2014 - Interview: with Tobias Fischer of tokafi magazine:
http://www.tokafi.com/news/coding-aesthetics-ben-carey-derivations/

June 2014 - Review: Benjamin Carey et. al: _derivations - Matthew Lorenzon, Partial Durations/RealTime Arts:
http://partialdurations.com/2014/06/19/benjamin-carey-derivations/

April 2014 - Interview: with Sam Gillies of Cut Common Magazine:
http://www.cutcommonmag.com/ben-carey-exploring-navigating-responding-listening/

August 2012 - Review: Getting more from your instrument - Felicity Clark, RealTime Arts Issue 111, p.48:
http://www.realtimearts.net/article/111/10844

August 2012 - Review: In Varietate Concordia - Paul Nolan, ArtsHub News/Reviews:
http://au.artshub.com/au/news-article/reviews/performing-arts/in-varietate-concordia-191258

July 2012 – Blog post: Musicuratum featured blog post with a variety of embedded audiovisual media:
http://musicuratum.com/2012/07/10/ben-carey/

June 2012 - Feature article: Sound Play - UTS U:Magazine featured event 'Computer Improvisation':
http://newsroom.uts.edu.au/news/2012/06/sound-play

May 2012 - Feature article: Beyond Musical Borders - Sydney Conservatorium of Music 'Conversations' Magazine Feature, pp. 7-12:
http://issuu.com/con-conversation/docs/conversation_issue_4

Appendix J - Video Documentation Transcripts

Transcripts of the Video Documentation outlined in Appendix E are provided below, with permalinks to each video provided in the accompanying footnotes.

Video 1: Overview[80]

Hi and welcome to this overview video of _derivations. My name's Ben Carey, I'm the developer of the software, and the purpose of this video is to get you up and running and interacting with it as quickly as possible. In a nutshell, _derivations is at the same time a musical work, a performance environment and a collaborative tool for use in improvised electro-acoustic performance. The software has been under development for the past two years as a part of my doctoral research at the University of Technology, Sydney.

The system currently exists in three separate versions, one as a standalone piece of software that can be run on any Mac, and two that require Cycling '74's Max/MSP to be installed on your machine. The differences between the two Max/MSP versions relates to the licensing of third party software. I won't bore you with the details here but if you'd like to know more I've explained this on the software downloads page of derivations.net

_derivations works by recording and analysing the input of an improvising musician throughout a performance, and by making relationships between musical gestures stored in an expanding database and those being currently performed by the improviser. The system uses these recorded gestures as source material to bring back and modify live as its contribution to the improvised dialogue with the musician. In future videos I'll go into more depth about the various ways that the software selects and processes these snippets of sound, and how you will be able to customise parts of the system's response to best suit your performative needs during rehearsal.

For now, let's get started with setting up a simple performance session in the software. Once you have installed and launched the software, you will see the main _derivations

[80] Video 1 permalink: https://vimeo.com/69298047

interface. Click on the microphone icon to enable audio, and you can enter audio preferences to double check your settings if needed.
To initialise _derivations parameters to their default settings, choose the Default initialisation file from the drop-down menu in the top left of the window. We will discuss creating your own initialisation files in a future video. Choose your audio input from the drop down menu, which in our case will be 'Live Signal'. Alternatively, choosing 'simulation' is useful if you have a pre-recorded improvisation you would like to use for testing. You can open the simulation file from the 'open file' button.

With Live signal enabled, pay attention to the 'Input Monitor' section of the window to check the level of your input. Much of _derivations analysis relies upon the level of the incoming signal. The two sliders found next to the level indicator represent hi and low thresholds for the input - these help _derivations in determining when the performer is playing or is at rest. Make sure that when you are sending signal to the system the input level rises above the first threshold, and when the input is 'silent', it lies below the right threshold. This will become apparent as we learn more about how the system works.

With this simple calibration set, you're ready to begin improvising. For now, I'll interact with the system as I talk. You will notice that as you play, the system begins to incrementally count 'phrases' found. After a preset number of phrases recorded and analysed - the default is four - the system begins to generate material in response to your continual improvised performance. Let's briefly listen to the system's response to my monologue…

Once you're ready to finish interacting with the software, simply press stop, causing the software to come to a halt after a short amount of time.
Although this was a very short and pretty uninteresting interaction with the software, let's quickly take a look at some of the information _derivations has stored throughout our encounter.

The buffer tab shows us audio from the input that has been recorded throughout our session. As this was a fairly short session, most of the buffer remains empty. Nevertheless, _derivations analysis has found and references to '9' phrases within this

buffer. We can individually audition each of these phrases using the audition tab. For instance, phrase two sounded something like this:
In a future video we will see how auditioning, enabling and disabling of individual phrases and even whole rehearsals becomes useful when the system is being used to create multi-session performance databases. For now, let's simply save this current session to our sessions folder so we can access it at a later date. In the standalone version of the software, this folder is found in the application directory, and for the Max/MSP versions, the sessions folder should have been copied to your Max directory.

That concludes this short overview of the _derivations software. Future videos will delve deeper into the system's various processing modules, and how you can customise the way in which these modules process recorded phrases. We will also look at how to manage multi-session databases, which can allow you to completely pre-define the sonic vocabulary of the system over several rehearsal sessions.

Thanks for watching, and I hope you enjoy working with _derivations.

Video 2: Phrase Segmentation[81]

As discussed in the overview video, _derivations' analysis and decision-making relies heavily on the level of the incoming signal. This is because the system makes its comparisons between live and pre-analysed sounds based on chunks of sound called 'phrases'. In this short video we're going to unpack this concept a little so that you can choose the right settings that work for your performance context.

A 'phrase' in _derivations is the smallest unit of sound used by the software for both analysis and generation of material. Everything that is recorded and analysed during an interaction with _derivations is indexed by phrase number, and the software does this automatically during a performance. Importantly however, determining exactly what _derivations counts as a phrase can be customised by you, depending on the type of input you're putting into the software. Two important settings to help you with this are

[81] Video 2 permalink: https://vimeo.com/69458280

the segmentation type, and the silence threshold. Let's take a look at the phrase database module to understand these further.

There are two types of phrase segmentation in _derivations, firstly Attack/Release segmentation, and secondly, Forced Phrase segmentation.

In Attack/Release mode, we return to our high and low input thresholds discussed in the overview video. In this mode, _derivations counts the start of a phrase from when the input first rises above the left threshold. This threshold crossing is called an 'attack trigger'. Conversely, the end of a phrase is reported by a release trigger linked to a crossing of our low threshold. This release trigger works in a very similar way to our attack, with one notable exception. Once the input dips below our low threshold on the right, _derivations will wait for a specific amount of time before reporting a release trigger - our phrase end. If our signal rises back above our high threshold before the length of this silence threshold, _derivations assumes that the original phrase has not ended.

This waiting period is called the 'silence threshold' and can be set in the phrase database module. You can think of the silence threshold as a way to make sure that a phrase has really ended before the software reports a phrase end. This can be quite important as it will directly affect the number of phrases _derivations will index in its database, and the relative lengths of these phrases.

Let's briefly look at two examples to illustrate how our silence threshold has an effect on _derivations analysis:

In this first example, we can see that _derivations has segmented approximately 9 seconds of audio into three phrases of varying lengths. As our silence threshold in this case is 1000ms - or 1 second - the signal must be silent for a full second before _derivations reports the end of a phrase. As a result, all of the quick dynamic changes in this input are grouped as part of a larger phrases.

In contrast, let's look at the same 9 seconds of audio with a much shorter silence threshold. We can see that the fluctuations in our input, because of our shorter silence

threshold, are counted as individual phrases, meaning that the same 9 seconds of audio is segmented into 13 chunks of audio.

Let's now move to our next segmentation type - forced phrase segmentation. Forced phrase segmentation is useful if your input is often sustained in nature, and therefore rarely dips below the low input threshold. In this case, _derivations provides a 'forced phrase interval' parameter, which sets the maximum amount of time a signal remains active before _derivations forces the reporting of a phrase end. Whilst quite a brute force way of segmenting the input, it does prove useful for a variety of types of instrumental inputs.

The following example shows a fairly sustained signal with less active fluctuations in dynamics. With force phrase enabled, the 24-second sound would be evenly chunked into 6 phrases of equal length. For signals of this type, forcing the indexing of phrases is useful as it enables the system to continually build a database of material, without waiting for the input to fall silent first.

Of course, most sound sources display a combination of both sustained and dynamic material. Choosing force phrase and carefully choosing a maximum interval is therefore a good option, as anything below this maximum will be segmented in the same way as the attack/release mode.

It goes without saying that settling upon a configuration that works best for your input will probably take a good deal of trial and error, however we will see in later videos how decisions made on this input stage can affect the responsive character of the system in performance.

In the next video I will discuss how system response works in _derivations, from phrase comparisons, to autonomous triggering and phrase processing, and how you can tweak various processing parameters to suit your taste. Future videos will discuss managing session databases, and cumulatively building your own performance environments using _derivations.

Thanks again and I hope you enjoy working with _derivations.

Video 3: Output Modules[82]

Hi and welcome to the third installment of our video documentation. In this video we will be discussing each of _derivations' output modules in depth, so that we can understand how the system makes use of our recorded improvisations during performance. We'll also discuss how we can customise various processing options during a rehearsal session with the software. In the next video I'll delve deeper into how _derivations makes it's decisions upon which phrase to process or re-synthesise, and how the autonomous triggering of these modules works during an improvisation.

In _derivations, phrases segmented from our recorded improvisations are used as source material for processing by three separate modules, namely our bank of phase vocoders, the sound file granulator and through spectral re-synthesis in Pitch Models. Each of these modules accesses the global phrase database for its material, which is then processed or re-synthesised automatically following a range of processing options that are preset using the interface.

So, each time a phrase is chosen for output, the chosen output module processes or re-synthesises the original sound - injecting a new musical gesture into the improvised dialogue with the musician. Because our database contains information on the length of each phrase, each new gesture is performed for a length of time that is proportional to the original length of each phrase stored in the database.
Although _derivations is designed to be an autonomous system, the user interface has been designed so that the range of processing capabilities available in each module are accessible for testing and fine-tuning during rehearsals. Each of our output modules contains its own graphical interface with access to a number of settings that will affect the sonic output during performance. The control tab then allows you to set the range of parameters you have settled upon globally, so that they can be stored as a preset setup in an initialisation file. More on this control tab a little later.

Let's turn first to look our phase vocoder module.

---

[82] Video 3 permalink: https://vimeo.com/69728069

Simply put, a phase vocoder enables a sound to be sped-up or down in time, without changing its pitch. Conversely, it enables a sound to be transposed, without changing it's length. In _derivations, four independent phase vocoders are used to access individual phrases from our database for output. Each time a phrase is triggered by one of our 4 PVOCs, the player will scrub through the source file at a constantly varying speed, until the triggered gesture ends. The speed range of this scrubbing can be set individually for each PVOC using the range slider in each tab, or globally by setting the range of PVOC-1.

Transposition of the individual phase vocoders can also be set using a global range, this can be found within the Mix-Transpose tab - but is also found in the control tab. If Transpose On is checked within the control tab, any phrase triggered by the pvocs in performance will be transposed to a fixed random interval within this chosen range. Another global option available in the control tab is 'Glide Probability'. This option determines the probability that transposition will be fixed or whether a Random glissando amount is applied to the triggered phrase upon output. You can test this type of transposition using the transposition drop-down menu.

Next we move across to our sound file granulator.

The granulator creates new gestures from our source phrases by rapidly outputting small grains of sound that overlap to create abstract textures. The granulator scrubs from one side of the file to another in the same way as with our phase vocoders, following a speed range that can be set in the module. As it scrubs through the file, the granulator outputs grains of our source at a given rate and duration. Changing these two parameters can vary a sound from sparse clicks to dense, drones - depending upon the source phrase. Transposition of the grains can be enabled or disabled in the control tab, which also provides a probability for global transposition within the set range, or transposition per grain. The latter option causes the most radical change to the source sound, giving the effect of blurring the pitch centre of the gesture.

Probably the most important parameter to experiment with is the density range, which is also reflected in the control tab. Density in the granulator can be expressed as a function of either the grain duration, or the grain rate. Grain density is also scrubbed throughout a

gesture, which can create a variety of morphing textures throughout the output of a single phrase. Most of the other parameters should be experimented with by hand in rehearsal and then set for performance.

Let's now move on to our final output module, pitch models:

Pitch Models also takes phrases from our database as source material, though unlike the other two output modules, it is not directly reliant upon these recordings to create its gestures. Rather than processing recorded phrases, pitch models makes use of a further layer of analysis to generate purely synthesised sounds derived from the phrases stored in the database. The module does this by continually taking snapshots of the spectrum of the improviser's sound during performance, and then grouping these snapshots together to be indexed alongside each of the phrases stored in the database.

Pitch models uses then these snapshots to re-synthesise new gestures using additive synthesis, which takes the individual frequency and amplitude pairs of a spectrum and then synthesises them using banks of sine wave oscillators. This type of re-synthesis contributes a very different sound to the other two modules, whilst still being derived from the sonic content of our source. It's also a very flexible way of re-interpreting our source phrases, as these snapshots are manipulated in a number of ways before being synthesised.

Taking a look at the Pitch Models interface, we notice that there are a number of control options for this module. Before we discuss the re-synthesis stage, let's begin briefly with two important options that are also available in the control tab - namely the storage type, and the output type. Under the input tab, we are able to choose the way in which pitch models stores our snapshots, whether continuously, upon a pitch detection, or following our input thresholds. This final method is the default option, as we want our models stored alongside our phrases.

The output type controls how the models are output by the module. Let's go through them briefly. Random sweep treats the entire database of models as a space to scrub through, disregarding phrase boundaries; random range will output models within a specified range; this model will output the most recent models collected during an

improvisation, which can be interesting as it effectively shadows the input with synthesised versions of itself; random phrase simply outputs models linked to a random phrase upon triggering of the module, and finally, one shot, which is the default option, will output models according to the phrase chosen by the latest comparison with our database.

Let's quickly take a look some ways in which pitch models re-synthesises our stored snapshots. A snapshot is manipulated by altering the amplitudes of each of the sinusoidal components of that model, changing the timbre of an otherwise static sound over time. This is achieved by swapping the amplitudes of the first few partials over a period of time. Under the transformations tab, the scramble parameter sets the number of these partials to be swapped, and the random range allows you to set a subset of the amplitudes to scramble. The rate parameter then sets the time in milliseconds for this scrambling to occur. In addition, once the models are synthesised they are then sent through a sweeping formant filter, further altering their spectrum. Under this tab the formant filter can be turned on or off, its rate and transition time altered, as well as the sharpness of it's filtering adjusted.

A synthesised gesture triggered from pitch models is in itself polyphonic, as multiple synthesised models overlap to make up the final output gesture. Looking once more at the output module, let's take a look at how the individual components we've just discussed are output. When a gesture is triggered from pitch models, the module activates a rhythmic envelope that triggers the individual models throughout the gesture. In one shot mode, pitch models scans through the models grouped alongside the chosen phrase, outputting a model each time the rhythmic envelope sends a trigger. Each trigger activates the global ADSR envelope, just as in any polyphonic synthesiser. You can modify this envelope, and it's output length and interval range within this tab.

As we can see however, pitch models is the most complex of the three output modules, with many variables combining to control the resulting output gestures. To simplify the many complex variations possible within the module, the majority of these parameters are set and manipulated globally through a bank of presets. Let's briefly look at the Phrase-Presets tab to understand this further. The presets represent a variety of different output states of the module, and _derivations includes 7 of these by default. You can

preview the various presets by choosing a preset number from the recall number box. The preset-params tab displays all available parameters being controlled. If you'd like to change or add a preset, add the preset number you wish to store into the store number box, then click 'write-preset'.

Most importantly however, the presets are designed to be interpolated between, allowing a smooth transition from one preset to the next. During the output of a gesture from pitch models, _derivations automatically interpolates between two of the stored presets. In the control tab, we can select the type of preset interpolation that is triggered upon the output. Adjacent interpolation activates a drunken walk that moves between adjacent presets. So for example, in one gesture we may move between presets 2 and 3, whilst the following gesture will either remove between either presets 3 and two, or choose to move from preset 3 to preset 4. 'Any' mode allows interpolation between any two of these presets, regardless of their position in the preset list.

So, that concludes our round up of the three output modules within the software. Let's finish by looking once more at the control tab. Although not al of the parameters in _derivations are included in the control tab, the most important variables are available to be preset prior to a rehearsal or performance with the software. Once you've decided upon the settings that you would like to use during a performance, you can save these as an initialisation file by clicking Save as INIT, on any of the 4 tabs available in this window. Make sure to save this file in your 'init-files' folder, which should be in either your application directory in the standalone application, or in your Max folder if you're running _derivations in Max. Once you reset the _derivations interface, you should see this new file appear in the drop down menu at the top of the main window. Choosing the file will initialise all of your saved settings.

Make sure to watch the next video, where I'll be discussing how both phrase comparisons and autonomous triggering work within the software.

Thanks again and I hope you enjoy working with _derivations.

Video 4: Comparisons and Triggering[83]

Hi and welcome to the fourth installment of our video documentation for _derivations. In this short video we will be discussing how _derivations' output modules make their decisions about which phrases reference from the database during performance. We'll then go on to unpack the process of autonomous triggering to understand how the these modules become activated during an improvised performance.

In the fifth and final video, we will be looking at the process of saving and loading session databases made using _derivations, which allow you to make use of the software cumulatively from rehearsal to rehearsal, and of course, from rehearsal to performance. This will also include discussion about the merging of previously recorded databases, which enables you to pre-define the sonic vocabulary of _derivations in it's entirety prior a performance.

Without going into the technical details, let's talk briefly about how phrase comparisons work within _derivations.

As we've learned from earlier videos, _derivations decision-making is based upon comparisons between the analysis of an incoming signal and a database of segmented phrases recorded during an improvisation. Looking again at the Input Monitor module, we see that as well as tracking the volume of our signal, _derivations also analyses the spectral content of the improviser's sound.

The system analyses our sound by gathering data called Mel-frequency cepstral coefficients - or MFCCs. This type of analysis is used in a wide variety of signal processing and machine listening tasks, and is often used in speech recognition technology.

_derivations calculates statistics on streams of MFCC's that are captured throughout the length of each phrase, and it is these statistics that are then stored alongside each phrase index in our database. It is from these statistics that the system is able to make

---

[83] Video 4 permalink: https://vimeo.com/69733871

comparisons between one phrase and another, as it can then find phrases in the database that are the most similar or different to the latest input.

Let's now take a look at how this process plays out in the triggering of _derivations' output modules.

The autonomous triggering of _derivations' output modules relies upon a continual cycle of comparison between both the improviser and the phrase database, as well as between the phrases chosen by the modules themselves and this database. In other words, as well as listening outside to the performer's contribution, _derivations also makes decisions by listening inwards - that is, by self-referencing.

Let's take a look at the following graphic to understand this flow of control and influence further:

Analysis of the improviser's latest phrase is picked up for comparison by the first module that is free and available for output. In this first example, the first phase vocoder - pvoc-1 - makes a comparison between the latest analysis of the improviser and the phrase database, and outputs its chosen phrase. This phrase chosen is then picked up for comparison with the database by the pitch models module, whose subsequent output is then evaluated by pvoc-3, continuing the cycle of comparison.

As we can see, this cyclical pattern of comparisons can become quite complex, as the musician's contribution sets off a chain of events of which the outcome is difficult to predict in advance. Because the length of each phrase being output by the modules may vary, the output of the modules will often overlap, creating a complex polyphony of processed and synthesised material.

This also means that the system is unlikely to follow the same direction of influence upon each analysis of the improviser. In addition, once this chain of events is set in motion, the system is able to continue self-referencing without waiting for the input. This enables a fully-autonomous state of performance for the system, allow the performer the freedom to rest and listen to the system as it generates material based upon the current database.

It goes without saying that during a performance with _derivations this process is handled automatically by the software. However, if we want to observe as well as manage some of the high-level decisions of this triggering, this is possible within the 'Triggering' tab. This simple interface displays the 6 interacting components of our output modules, and the phrases they receive for comparison and output during a performance. We can simulate an autonomous interaction between these players in this tab by sending 'seed' phrases to the players to start a chain of comparisons. There are a number of options for choosing these seed phrases in the drop down menu at the top right of the window.

Of course, simulating an interaction in this way is only possible if you have a phrase database from a prior improvised session loaded within the software. In the next video we'll go into depth about managing these databases, however for now we'll work with a multi-session database loaded from disk. To begin testing autonomous triggering, click the auto trigger button in the top left of the window. The system should begin generating phrases on its own, and passing chosen phrases amongst the various components.
As I mentioned previously, the length of the gestures performed by our output modules is tied to the length of the original phrases stored in the database. This doesn't mean however that the two are identical. Instead, each gesture's output length is modulated by a roaming length factor that controls the final output length of a gesture. You can set a range within which the length factor can roam, ranging from 8 times shorter than the original phrase, to 8 times longer.

Finally, I wanted to briefly discuss the concept of density as it applies to the sonic gestures performed by our output modules. As I mentioned earlier, the 6 individual players from our 3 output modules create a polyphony of sonic gestures that often overlap. To enable high-level control of the level this polyphony, _derivations relies upon a density control measure, which can either be fixed, or changed over time throughout a performance.

In this sense, density is the extent to which phrases overlap during a performance. To do this, _derivations evaluates the current lengths of our output gestures, and modifies the time delay between the triggering of one gesture and the next. Essentially, the higher the density, the more modules are active at any one time, and the lower the density, the more space between the triggering of one gesture and another.

The density tab is accessible from within both the triggering tab and the main window of _derivations. The toggle in the top left of this window determines whether a density value is fixed throughout an improvisation, or whether the software follows a preset density trajectory. When set to a fixed value, you can adjust the overall density value using the slider on the right, with 0 being the highest and 999 the highest density value.

Switching the toggle to 'follow curve' tells _derivations to modulate the density level according to the curve drawn in the window. You can also set the length of this curve in minutes, allowing you to pre-determine a density trajectory for your performance. If you find a trajectory that you like, you can save it in the preset box in the top right of the window. Alternatively, 'random on start' will generate a new, random density curve upon each new, improvised session. You can also set and save these settings globally in the control tab under global settings.

That concludes our look at the autonomous interaction of _derivations output modules. Hopefully now you understand a little more about what _derivations is up to during a performance, and you're able to simulate the flow of information from one module to the next during rehearsal. Now that you have a handle on the output modules as well, don't forget that you can simulate an entire performance with the software using the simulation option in the main window, as mentioned in the overview video. In this mode _derivations works identically to how it would with a live input, the only difference being that an improviser is replaced with a sound file.

In the next video I will round up our discussions by explaining session databases in depth, including some tips on how to make the most of your recorded sessions during rehearsal.

Thanks again and I hope you enjoy working with _derivations.

Video 5: Session Databases[84]

Hi and welcome to the fifth and final instalment in our series of videos on _derivations. In this video we'll be discussing how you can make use of the information stored during an improvisation with _derivations to build your own performance environments for future performances using *session databases*.

A session database is a saved collection of both audio and analysis files recorded by _derivations during a performance. The analysis files contain all the information _derivations has captured about the recorded improvisation, including the stored sinusoidal models used in pitch models, as well as the MFCC analysis. After an improvised performance, we can save all of this information to disk and recall this at a later date for use in your next performance, or for merging with other pre-recorded sessions. Let's take a look at how this is managed using the main _derivations interface.

Here I have a full database of material accumulated during an interaction with the software. As we saw in the overview video, clicking 'save session' and naming the database will save this information to your 'session' directory, creating a new session database. Taking a look in our session directory in the finder, we see that _derivations has created a named folder containing an audio file, and a bunch of analysis files. This is our saved database.

After resetting the software, we can re-load this database by choosing load session, and navigating to this folder. Upon load, a splash screen appears to give you some information on the database you're loading. We can see here that this current database contains 117 phrases from one session, which means that there is only one audio file in the database. There is also some stored information about our threshold levels, and an indication of phrases that may have been disabled. More on this in a minute.

So at this point, you may be wondering why such a feature is useful for a real-time interactive system. First off, saving a database is very useful in our testing and fine-tuning of _derivations' output modules, as we saw in our previous video. This allows us to

---

[84] Video 5 permalink: https://vimeo.com/70767558

observe and interact with the output modules by tweaking individual parameters by hand during rehearsal. Recalling saved sessions allows you to get better acquainted with the system's possible range of behaviours, because you're working with material that is familiar to you from a previously improvised session. With a database loaded, we can choose individual phrases to process with the output modules by typing in the phrase index in the number box, or in the case of the granulator and pvocs you can call a random phrase to test by clicking the button next to this number box.

As we saw in our overview video we can also use the audition tab to audition the unprocessed phrases segmented during the saved session. This may be useful in further fine-tuning the type of segmentation you choose to use for your next performance, but it also allows you to disable unwanted phrases, or even entire sessions. After choosing a phrase index from the drop-down menu, we can audition the phrase to hear how it sounds. If we don't like this particular phrase, we can disable it's appearance in the database by selecting 'disable'. Disabled phrases are indicated by a dot next to their index in the menu. So without deleting the phrase, _derivations simply no longer references this phrase when making comparisons during performance. Clicking save will tell _derivations to remember your choice for the next time you load the database. Conversely, you can re-enable a disabled phrase by selecting the phrase and clicking enable.

The most powerful feature of session databases however is their ability to enable both a cumulative and non-linear approach to defining _derivations' sonic vocabulary in advance of a performance. Using a cumulative approach, saved sessions can be recalled and built upon from one rehearsal session to the next, allowing the software to begin an improvised session with an already rich database of material captured from cumulative interactions with the software. So in this way, each performance with the software adds material to the database, expanding the material the software has to reference from one performance to the next.

Working with derivations in this way is identical to that of a standard performance, with the addition of loading a database at the outset. Firstly load a saved database from disk, then initialise the system the way you normally would in a 'fresh' performance, and simply press start. _derivations decisions of which phrases to reference are still made in

the same way, however the phrase could be taken equally from either the current performance, or the phrases already contained in the database loaded at the outset. To save the database, simply press save and name the database. If you would like to replace the original database with this new cumulative database, simply type the name of the original database you loaded. Alternatively you can create a separate cumulative database by typing a new name and hitting save. So looking again in the finder, we see that our database now contains two audio files; the original file, and the most recent recorded improvisation.

You can also pre-define multi-session databases in a non-linear fashion, by merging previously saved sessions into a single, fixed database. Once you have created such a database for use in a performance, you can disable real-time recording of your input, allowing the system to interact solely with what has been pre-defined in advance. You can disable input recording by unchecking the 'Analysis Storage' toggle in the phrase database. This approach treats _derivations sonic vocabulary as fixed, which means that although the system is still feeding off analysis of the performer, it is responding with musical gestures derived from it's own, pre-defined pool of possible phrases. This ability fundamentally changes the interactive paradigm that the system represents, moving from a live-sampling based system, to a system based on a corpus of pre-analysed material.

To merge pre-existing databases, firstly load an initial database using the load session button. After loading, move on to the merge sessions button, where you can choose a second database to merge with the already loaded database. It's worth noting that either database can be a single or multi-session database, allowing us to create large, complex databases in one operation. Here I'm choosing to merge a database containing three sessions with our original single-session database. The result should be a merged database containing 4 sessions. Once you have chosen the database to merge, _derivations asks whether you would like to merge only the loaded session, or the buffer from a current improvisation as well. If you have just finished performing a cumulative session with the software, you can choose the first option to include the current improvisation in the merged session. In our case, we only want to merge two, loaded databases, so we'll choose the second option. Finally, name you merged session, and wait whilst _derivations moves items to place in your sessions folder. A quick look in the finder confirms that our sessions have been correctly merged into a new, multi-session

database. Once we load this database into _derivations our splash screen will further confirm the contents of the database. For a performance with this new database, we can disable input recording, and begin an interaction with our pre-defined corpus of material.

So finally, I'd like to let you know about the session databases page on derivations.net. Because of this flexible database feature of the software, I've opened up the possibility for collaboration between users on the website. On the session databases page of derivations.net you'll find a small repository of pre-recorded sessions that can be downloaded for use in _derivations. The general idea is that users may benefit from sharing and making use of unique session databases contributed by other performers. So, if you'd like to build a unique database for use in performance, you may be interested in making use of material provided by other interested performers using _derivations from around the world. Sessions uploaded to the site are free to download, and contact details of the contributors are provided so you can get in contact with a fellow user if you intend on using their material in your own performance. If you'd like to submit sessions to be used by others in their performances, send me a link to a zipped session database you wish to share and I'll upload it to the site.

So that concludes our series of video documentation on _derivations. Hopefully you now have all the information you need to get going with interacting with the software in rehearsal and performance. If you plan to make use of it in a performance, or have a recording of an interaction with the software on stage or in the studio, please drop me a line as I'd love to hear how the software is used by others.

Thanks again, and I hope you enjoying working with _derivations.

Appendix K - Publications

Refereed publications describing work discussed in this thesis are listed below:

Bown, O., Eigenfelt, A., Martin, A., Carey, B. & Pasquier, P. 2013, 'The Musical Metacreation Weekend: Challenges arising from the live presentation of musically metacreative systems', paper presented to the *Artificial Intelligence and Interactive Digital Entertainment (AIIDE'13) Conference*, Santa Cruz, California.

Carey, B. 2012, 'Designing for Cumulative Interactivity: The _derivations System', paper presented to the *12th International Conference on New Interfaces for Musical Expression*, Ann Arbor.

Carey, B. 2013, '_derivations: Improvisation for Tenor Saxophone and Interactive Performance System,' *Proceedings of the 2013 ACM Conference of Creativity and Cognition*, Sydney Australia

Carey, B. 2014, 'Artefact Scripts and the Performer-Developer, ' *Workshop on Practice-Based Research, Conference on New Interfaces for Musical Expression*, London, UK.

Carey, B. 2016, 'Artefact Scripts and the Performer-Developer, ' *Leonardo*, Vol. 49, No. 1 London, UK, pp. 74-75.

Bibliography

Akrich, M. 1992, 'The De-scription of Technical Objects', in W. Bijker (ed.), *Shaping Technology / Building Society: Studies in Sociotechnical Change (Inside Technology)*, The MIT Press, pp. 205-24.

Akrich, M. & Latour, B. 1992, 'A Summary of a Convenient Vocabulary for the Semiotics of Human and Nonhuman Assemblies', in, *Shaping Technology / Building Society: Studies in Sociotechnical Change (Inside Technology)*, The MIT Press, pp. 259-64.

Allauzen, C., Crochemore, Maxime, Raffinot, Mathieu 1999, 'Factor oracle: a new structure for pattern matching', in, *SOFSEM'99: Theory and Practice of Informatics*

Lecture Notes in Computer Science, vol. 1725, Springer- Verlag, pp. 295-310.

Ames, C. 1989, 'The Markov process as a compositional model: a survey and tutorial', *Leonardo*, pp. 175-87.

Assayag, G., Bloch, G. & Chemillier, M. 2006, 'OMax-Ofon', paper presented to the *Sound and Music Computing Conference*, Marseille.

Assayag, G., Bloch, G., Chemillier, M., Cont, A. & Dubnov, S. 2006, 'Omax brothers: a dynamic yopology of agents for improvization learning', paper presented to the *Proceedings of the 1st ACM workshop on Audio and music computing multimedia.*

Bailey, D. 1993, *Improvisation: Its Nature And Practice In Music*, Da Capo Press, Cambridge, MA.

Bastien, D.T. & Hostager, T.J. 1992, 'Cooperation as communicative accomplishment: A symbolic interaction analysis of an improvised Jazz Concert', *Communication Studies*, vol. 43, no. 2, pp. 92-104.

Beilharz, K., Jakovich, J. & Ferguson, S. 2006, 'Hyper-shaku (Border-crossing): Towards the Multi-modal Gesture-controlled Hyper-Instrument', paper presented to the *Conference on New Interfaces for Musical Expression*, Paris.

Blackwell, T. & Young, M. 2004, 'Self-organised music', *Organised Sound*, vol. 9, no. 2.

Blackwell, T. & Young, M. 2005, 'Live Algorithms', *Artificial Intelligence and Simulation of Behaviour Quarterly*, no. 122, pp. 7-9.

Blankertz, B., Jacucci, G., Gamberini, L., Freeman, J. & Spagnolli, A. 2015, *Symbitoic2015 - International Workshop on Symbiotic Interaction*, viewed July 8th 2015 <http://symbiotic2015.org/>.

Bloch, G., Dubnov, S. & Assayag, G. 2008, 'Introducing video features and spectral descriptors in the omax improvistaion system', paper presented to the *International Computer Music Conference*, Belfast.

Boden, M.A. 2004, *The creative mind : myths and mechanisms*, 2nd edn, Routledge, London ; New York.

Boehm, B. 2006, 'A view of 20th and 21st century software engineering', paper presented to the *International conference on Software engineering (ICSE '06)*, New York, NY.

Bongers, B. 2000, 'Physical Interfaces in the Electronic Arts - Interaction Theory and Interfacing Techniques for Real-time Performance', in M.M. Wanderly & M. Battier (eds), *Trends in Gestural Control of Music*, IRCAM-Centre Pompidou, Paris, pp. 1-30.

Bown, O. 2011, 'Experiments in Modular Design for the Creative Composition of Live Algorithms', *Computer Music Journal*, vol. 35, no. 3, pp. 73-85.

Bown, O., Carey, B. & Eigenfeldt, A. 2015, 'Manifesto for a Musebot Ensemble: A platform for live interactive performance between multiple autonomous musical agents', paper presented to the *International Symposium on Electronic Arts*, Vancouver, Canada.

Bown, O., Eigenfelt, A., Martin, A., Carey, B. & Pasquier, P. 2013, 'The Musical Metacreation Weekend: Challenges arising from the live presentation of musically metacreative systems', paper presented to the *Artificial Intelligence and Interactive Digital Entertainment (AIIDE'13) Conference*, Santa Cruz, California.

Bown, O., Eldridge, A. & McCormack, J. 2009, 'Understanding Interaction in Contemporary Digital Music: from instruments to behavioural objects', *Organised Sound*, vol. 14, no. 02, p. 188.

Bown, O. & Lexer, S. 2006, 'Continuous-Time Recurrent Neural Networks for Generative and Interactive Musical Performance', in, *Applications of Evolutionary Computing, Lecture Notes in Computer Science*, vol. 3907, pp. 652-63.

Bown, O. & Martin, A. 2012, 'Autonomy in Music-Generating Systems', paper presented to the *International Workshop on Musical Metacreation at AIIDE*, Stanford, Palo Alto.

Butcher, J. 2003, CD, *Invisible Ear*, Fringes Recordings.

Butcher, J. & Nakamura, T. 2004, CD, *Cavern With Nightlife*, Weight of Wax.

Candy, L. 2006, 'Practice Based Research: A Guide', pp. 1-19, viewed 17/10/10, <http://www.creativityandcognition.com/research/practice-based-research.html>.

Cardew, C. 1967, *Treatise*, Gallery Upstairs Press, Buffalo, New York.

Carey, B. 2012, 'Designing for Cumulative Interactivity: The _derivations System', paper presented to the *12th International Conference on New Interfaces for Musical Expression*, Ann Arbor.

Carey, B. & Hyde, J. 2014, *Integrated Records*, Sydney, viewed 6th of December 2014, <http://interecords.com>.

Casey, M. 2009, 'Soundspotting: A New Kind of Process', in R.T. Dean (ed.), *The Oxford Handbook of Computer Music*, Oxfrord University Press.

Casey, M., Veltkamp, R., Goto, M., Leman, M., Rhodes, C. & Slanley, M. 2008, 'Content-Based Music Information Retrieval: Current Directions and Future Challenges', paper presented to the *IEEE*, April 2008.

Chadabe, J. 1984, 'Interactive composing: An overview', *Computer Music Journal*.

Chadabe, J. 2005, 'The Meaning of Interaction, a Public Talk Given at the Workshop on Interactive Systems', paper presented to the *HCSNet Conference*, Sydney, Australia.

Ciufo, T. 2005, 'Beginner's mind: an environment for sonic improvisation', paper presented to the *International Computer Music Conference*, Barcelona.

Coessens, K. 2013, 'The Score Beyond Music', in P. de Assis, W. Brooks & K. Coessens (eds), *Sound & Score: Essays on Sound, Score and Notation*, Leuven University Press, pp. 178-81.

Collins, N. 2005, 'Drumtrack: Beat induction from an acoustic drum kit with synchronised scheduling', paper presented to the *International Computer Music Conference*, Barcelona.

Collins, N. & Escrivan Rincón, J.d. 2007, *The Cambridge companion to electronic music*, Cambridge University Press, Cambridge ; New York.

Cycling '74 2014, *Max*, Software, <http://cycling74.com/products/max>.

De Cheveigné, A. & Kawahara, H. 2002, 'YIN, a fundamental frequency estimator for speech and music', *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917-30.

Dobrian, C. 2004, 'Strategies for Continuous Pitch and Amplitude Tracking in Real-Time Interactive Improvisation Software', paper presented to the *Sound and Music Conference*, Paris.

Drummond, J. 2009, 'Understanding Interactive Systems', *Organised Sound*, vol. 14, no. 02, p. 124.

Dudas, R. & Lippe, C. 2006, *The Phase Vocoder - Part I*, viewed 16th April 2011, <https://cycling74.com/2006/11/02/the-phase-vocoder---part-i/>.

Dudas, R. & Lippe, C. 2007, *The Phase Vocoder - Part II*, viewed 16th April 2011, <https://cycling74.com/2007/07/02/the-phase-vocoder-part-ii/>.

Edmonds, E., Muller, L. & Connell, M. 2006, 'On creative engagement', *Visual Communication*, vol. 5, pp. 307-22.

Eigenfeldt, A., Bown, O. & Carey, B. 2015, 'Collaborative Composition with Creative Systems: Reflections on the First Musebot Ensemble', paper presented to the *International Conference on Computational Creativity*, Park City, Utah.

Eldridge, A. 2005, 'Fond Punctions: Generative Processes in Live Improvised Performance', paper presented to the *Generative Arts Practice Symposium*, Sydney.

Eldridge, A. 2008, 'Collaborating with the behaving machine: simple adaptive dynamical systems for generative and interactive music', University of Sussex, Brighton.

Fischer, R. 2008-13, *TouchOSC*, viewed July 17 2015 <http://hexler.net/software/touchosc>.

Foucault, M. 2005, *The Order of Things*, Routledge.

Freed, A. & Wright, M. 1997, 'Open Sound Control: A New Protocol for Communicating with Sound Syntheisizers', paper presented to the *International Computer Music Conference*, Thessaloniki, Greece.

Fricke, D. 1979, 'Electronic Music and Synthesizers', *Synapse Magazine*, Summer 1979.

Gifford, T. & Brown, A. 2008, 'Stochastic Onset Detection: An approach to detecting percussive attacks in complex audio', paper presented to the *Australasian Computer Music Conference*, Sydney.

Gifford, T. & Brown, A. 2009, 'Do androids dream of electric chimera?', paper presented to the *Australasian Computer Music Conference* Brisbane, Australia.

Google Inc 2015, *Google Analytics*, viewed 8th of June 2015, <https://www.google.com/analytics>.

Gray, C. 1998, 'Inquiry through Practice: developing appropriate research strategies', paper presented to the *No Guru No Method Conference Proceedings*, Helsinki.

Gray, C. & Malins, J. 2004, *Visualizing Research: A Guide To The Research Process In Art And Design*, Ashgate Pub Ltd.

Grinstead, C.M. & Snell, J.L. 1997, *Introduction to probability*, 2nd rev. edn, American Mathematical Society, Providence, RI.

Hamman, M. 1999, 'From Symbol to Semiotic: Representation, Signification, and the Composition of Music Interaction', *Journal of New Music Research*, vol. 28, no. 2, pp. 90-104.

Holmes, A. 2006, 'Reconciling Experimentum and Experientia: Ontology for Reflective Practice Research in New Media', *Speculation and Innovation: applying practice led research in the creative industries*.

Hsu, W. 2005, 'Using timbre in a computer-based improvisation system', paper presented to the *International Computer Music Conference*, Barcelona.

Hsu, W. 2006, 'Managing gesture and timbre for analysis and instrument control in an interactive environment', paper presented to the *Conference on New interfaces for Musical Expression*, Paris.

Hsu, W. 2008, 'Two Approaches for Interaction Management in Timbre-Aware Improvisation Systems', paper presented to the *International Computer Music Conference*, Belfast.

Hsu, W. 2010, 'Strategies for Managing Timbre and Interaction in Automatic Improvisation Systems', *Leonardo Music Journal*, vol. 20, pp. 33-9.

Hsu, W. & Sosnick, M. 2009, 'Evaluating Interactive Music Systems: An HCI Approach', paper presented to the *Proceedings of New Interfaces for Musical Expression*, Pittsburgh.

Huberman, A. 2004, 'Artists in Conversation - Kaffew Matthews by Anthony Huberman', *BOMB Magazine*.

IRCAM 2015, *IRCAM Forumnet*, IRCAM, Paris, France, viewed 8th of June 2015, <http://forumnet.ircam.fr/>.

Jacucci, G., Gamberini, L., Freeman, J. & Spagnolli, A. 2014, 'Symbiotic Interaction: A Critical Definition and Comparison to other Human-Computer Paradigms', in, *Symbiotic Interaction*, vol. 8820, Springer International Publishing, Switzerland, pp. 3-20.

Jefferies, J. 2012, 'Mangling practices: Writing reflections', *Journal of Writing in Creative Practice*, vol. 5, no. 1, pp. 73-84.

Jehan, T., Freed, A. & Dudas, R. 1999, 'Musical Applications of New Filter Extensions to Max/MSP', paper presented to the *International Computer Music Conference*, Beijing, China.

Jehan, T. & Schoner, B. 2001, 'An Audio-Driven Perceptually Meaningful Timbre Synthesizer', paper presented to the *International Computer Music Conference*, Havana, Cuba.

Johnston, A. 2009, 'Interfaces for musical expression based on simulated physical models', University of Technology, Sydney.

Johnston, A., Marks, B. & Edmonds, E. 2005, "Spheres of Influence': an interactive musical work', paper presented to the *Proceedings of the second Australasian conference on Interactive entertainment*, Sydney.

Kanga, Z. 2014, 'Inside the Collaborative Process: Realising New Works for Solo Piano', PhD thesis, Royal Academy of Music.

Kimura, M. 1996, 'Computers for Performers: Crossing Boundaries for the Future', *Computer Music Journal*, vol. 20, no. 4, pp. 25-6.

Kimura, M. 2004, 'Creative process and performance practice of interactive computer music: a performer's tale', *Organised Sound*, vol. 8, no. 03, pp. 289-96.

Klingbiel, M. 2009, 'Spectral Analysis, Editing, and Resynthesis: Methods and Applications', PhD thesis, Columbia University.

LaFosse, A. 1996, *Looper's Delight*, viewed 6 June 2015, <http://www.loopers-delight.com/>.

Latour, B. 1990, 'Technology is society made durable', *The Sociological Review*, vol. 38, no. S1, pp. 103-31.

Latour, B. 1992, 'Where Are the Missing Masses? The Sociology of a Few Mundane Artifacts', in W.E. Bijker & J. Law (eds), *Shaping Technology/Building Society: Studies in Sociotechnical Change*, MIT Press, Cambridge, Mass., pp. 225-58.

Latour, B. 1994, 'On Technical Mediation - Philosophy, Sociology, Geneology ', *Common Knolwedge*, vol. 3, no. 2, pp. 29-64.

Lévy, B. 2013, 'Principles and Architectures for an Interactive and Agnostic Music Improvisation System : principes et architectures pour un système interactif et agnostique dédié à l'improvisation musicale', UPMC.

Lewis, G. 2000, 'Too many notes: Computers, complexity and culture in voyager', *Leonardo Music Journal*, vol. 10, pp. 33-9.

Lexer, S. 2010, 'Piano+: An Approach towards a Performance System Used within Free Improvisation', *Leonardo Music Journal*, pp. 41-6.

Licklider, J.C.R. 1960, 'Man-computer symbiosis', *Human Factors in Electronics*.

Lindell, R. 2012, 'Code as Design Material', paper presented to the *Participatory Materialities Invited Workshop at Aarhus University*, Aarhus, Denmark.

Logan, B. 2000, 'Mel Frequency Cepstral Coefficients for Music Modeling', paper presented to the *International Symposium on Music Information Retrieval*.

Malt, M. & Jourdan, E. 2008, 'Zsa.Descriptors: a library for real-time descriptors analysis', paper presented to the *Sound and Music Computing Conference*, Berlin.

Malt, M. & Jourdan, E. 2009, 'Real-Time Uses of Low Level Sound Descriptors as Event Detection Functions Using the Max/MSP Zsa.Descriptors Library', paper presented to the *Brazilian Symposium on Computer Music*, Recife, Brazil.

Martin, A. 2014, 'Methods to Support End User Design of Arrangement-Level Musical Decision Making', University of Sydney, Sydney.

Martin, A., Jin, C., Carey, B. & Bown, O. 2012, 'Creative Experiments Using a System for Learning High-Level Performance Structure in Ableton Live', paper presented to the *Sound and Music Computing Conference*, Copenhagen.

Mattozzi, A. 2012, 'Rewriting the script. A methodological dialogue about the concept of 'script' and how to account for the mediating role of objects', paper presented to the *Department of Philosphy-STePS joint seminar*, University of Twente.

Maturana, H. & Varela, F. 1980, *Autopoiesis. The Realization of the Living*, Springer Netherlands.

McAulay, R.J. & Quartieri, T.F. 1986, 'Speech Analysis/Synthesis Based on A Sinusoidal Representation', *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 4, pp. 744-54.

McLean, A. 2011, 'Artist-Programmers and Programming Languages for the Arts', Department of Computing, Goldsmiths, University of London.

McLean, A. & Wiggins, G. 2010, 'Bricolage programming in the creative arts', paper presented to the *22nd Psychology of Programming Interest Group*, Madrid.

Moura, L. & Pereira, H.G. 2004, *Symbitic Art Manifesto*, viewed 21st of January 2014, <http://www.leonelmoura.com/manifesto.html>.

Moura, L. & Pereira, H.G. 2011, *Istanbul Manifesto*, viewed 21st of January 2014, <http://www.leonelmoura.com/manifesto_istanbul.html>.

Oxford English Dictionary 2015a, - *"interpretation, n."*, *OED Online*, Oxford University Press, viewed June 25th 2015.

Oxford English Dictionary 2015b, - *"symbiosis, n."*, *OED Online*, Oxford University Press, viewed June 25th 2015.

Pachet, F. 2002, 'The continuator: Musical interaction with style', paper presented to the *Proceedings of the International Computer Music Conference*, Gotheborg, Sweden.

Paine, G. 2002, 'Interactivity, where to from here?', *Organised Sound*, vol. 7, no. 03, pp. 295-304.

Penha, R. 2013, *Spatium - tools for sound spatialization*, viewed 10th of October 2013, <http://spatium.ruipenha.pt/>.

Penha, R. & Oliveira, J.P. 2013, 'Spatium, tools for sound spatialization', paper presented to the *Sound and Music Computing Conference, SMC 2013*, Stockholm, Sweden.

Pickering, A. 1995, *The Mangle of Practice: Time, Agency, and Science*, University of Chicago Press.

Pickering, A. & Guzik, K. 2008, *The mangle in practice : science, society, and becoming*, Duke University Press, Durham.

Pressing, J. 1988, 'Cognitive Processes in Improvisation', in W.R. Crozier (ed.), *Cognitive Processes in the Perception of Art*, Elsevier Science Publications, Holland.

Pressing, J. 1990, 'Cybernetic issues in interactive performance systems', *Computer Music Journal*, vol. 14, no. 1, pp. 12-25.

Puckette, M., Apel, T. & Zicarelli, D. 1998, 'Real- time audio analysis tools for Pd and MSP', paper presented to the *International Computer Music Conference*, San Francisco.

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

Robson, C. 2002, *Real world research: a resource for social scientists and practitioner-researchers*, Second edn, Blackwell, Oxford.

Roche, H. 2011, 'Dialogue and Collaboration in the Creation of New Works for Clarinet', PhD thesis, University of Huddersfield.

Rose, J. & Jones, M. 2005, 'The double dance of agency: A socio-theoretic account of how machines and humans interact', *Systems, Signs & Actions*, vol. 1, no. 1, pp. 19-37.

Rowe, R. 1992, *Interactive Music Systems: Machine Listening and Composing*, The MIT Press, Cambridge, MA.

Rutherford-Johnson, T. 2011, *A Journey to Aaron Cassidy's Second String Quartet*, viewed 20 September 2012, <http://www.newmusicbox.org/articles/A-Journey-to-Aaron-Cassidys-Second-String-Quartet/>.

Sarath, E. 1996, 'A New Look at Improvisation', *Journal of Music Theory*, vol. 40, no. 1, pp. 1-38.

Schön, D. 1983, *The Reflective Practitioner: How Professionals Think in Action*, 1995 edn, Ashgate Publishing Ltd, Aldershot.

Schulz-Schaeffer 2006, 'Who Is the Actor and Whose Goals Will Be Pursued? Rethinking Some Concepts of Actor Network Theory', in B. Wieser, S. Karner & W. Berger

(eds), *Prenatal Testing: Individual Decision or Distributed Action?*, Profil, München, pp. 131-58.

Scrivener, S. 2000, 'Reflection in and on action and practice in creative-production doctoral projects in art and design', *Working Papers in art and design*, vol. 1.

Serra, X. 1989, 'A System for Sound Analysis/Transformation/Synthesis based on a Deterministic plus Stochastic Decomposition', PhD thesis, Stanford University, Stanford, CA.

SoundCloud 2015, *SoundCloud*, viewed 8th of June 2015, <http://soundcloud.com>.

Stephen, D. & Stanley, S. 2015, *Interpretation*, *Grove Music Online. Oxford Music Online.*, Oxford University Press, viewed June 26 2015, <http://www.oxfordmusiconline.com/subscriber/article/grove/music/13863>.

Stiegler, B. 1998, *Technics and time*, Stanford University Press, Stanford, Calif.

Stucke, A. 2011, 'Embodying Symbiosis: A Philosophy of Mind in Drawing', MFA thesis, California College of the Arts.

Taruskin, R. 2009, *Music in the Late Twentieth Century: The Oxford History of Western Music*, Oxford University Press, p. 610.

Telestream, Inc., 2012, *ScreenFlow*, viewed 20th of June 2013, <http://www.telestream.net/screenflow/overview.htm>.

Todoroff, T., Daubresse, E. & Fineberg, J. 1995, 'Iana~ (A Real-Time Environment for Analysis and Extraction of Frequency Components of Complex Orchestral Sounds and Its Application within a Musical Context)', *International Computer Music Conference*, International Computer Music Association, San Francisco, pp. 292-3.

Turkle, S. & Papert, S. 1992, *Epistemological pluralism and the revaluation of the concrete*, vol. 11, Journal of Mathematical Behavior.

Vimeo, LLC 2015, *Vimeo*, viewed 8th of June 2015, <http://vimeo.com>.

Wiggins, G.A. 2006, 'A preliminary framework for description, analysis and comparison of creative systems', *Knowledge-Based Systems*, vol. 19, no. 7.

Winkler, T. 2001, *Composing Interactive Music: Techniques and Ideas Using Max*, The MIT Press, Cambridge, MA.

Wolff, C. 1964, *For 1, 2 or 3 people*, Edition Peters, New York.

Wordpress Foundation 2015, *Wordpress*, viewed 8th of June 2015, <http://wordpress.org>.

Young, L.M. (ed.) 1963, *An Anthology of Chance Operations*, La Monte Young and Jackson Mac Low, New York City.

Young, M. 2008, 'NN Music: Improvising with a `Living' Computer', in K.-M. Richard, S, Y. lvi & J. Kristoffer (eds), *Computer Music Modeling and Retrieval. Sense of Sounds*, Springer-Verlag, pp. 337-50.

Young, M. 2009, 'Creative Computers, Improvisation and Intimacy', paper presented to the *Dagstuhl Seminar Proceedings*, Dagstuhl, Germany.

Young, M. & Lexer, S. 2003, 'FFT Analysis as a Creative Tool in Live Performance', paper presented to the *Conference on Digital Audio Effects* London, UK.

Zicarelli, D. 1987, 'M and jam factory', *Computer Music Journal*, pp. 13-29.

Zils, A. & Pachet, F. 2001, 'Musical Mosaicing', paper presented to the *COST G-6 Conference on Digital Audio Effects*, Limerick, Ireland, December 6-8, 2001.

Zorn, J. 1984, *Cobra*, Hathut, Switzerland.