Interaction Design for Digital Musical Instruments

A dissertation

submitted for the degree of

Doctor of Philosophy

by

Patrick McGlynn, BA, MA.

Supervisors: Dr. Victor Lazzarini & Dr. Gordon Delap

Department of Music

National University of Ireland, Maynooth

Ollscoil na hÉireann, Má Nuad

May 2014

Table of contents

Table of contents 2
Table of figures 7
Acknowledgments 8
Abstract 10
Chapter 1. Introduction 13
1.1 Context of research 14
1.2 Summary of hypotheses 16
1.3 Original contribution of thesis 17
Chapter 2. A century of electronic musical controllers 20
2.1 Keyboard based instruments 21
2.1.1 The Musical Telegraph 21
2.1.2 Teleharmonium 21
2.1.3 Optophonic piano 22
2.1.4 Sphaerophone 22
2.1.5 Hammond organ 22
2.1.6 Electronic Sackbut 22
2.1.7 Mellotron 23
2.1.8 Moog modular 23
2.1.9 Optigan 23
2.1.10 Synclavier 24
2.1.11 ADS 100 24
2.1.12 EDP wasp 24
2.1.13 Fairlight CMI 25
2.1.14 Syntar 25
2.1.15 Roland SH-101 25
2.1.16 Yamaha DX7 26
2.2 Buttons and dials 26
2.2.1 Electrophon & Kurbelsphäraphon 26
2.2.2 Dynaphone 26
2.2.3 Voder speech synthesizer 27
2.2.4 Electronium Scott 27
2.2.5 Wurlitzer Side Man 27
2.2.6 Roland TR-808 28
2.2.7 Linn LM-1 28
2.2.8 Dynachord Rhythm Stick 28
2.2.9 Akai MPC60 28
2.2.10 Axis 64 29
2.2.11 Jammer 29
2.2.12 Monome 29
2.2.13 Samchillian 30
2.2.14 Maschine 30
2.2.15 Arc 31
2.2.16 Tenori-on 31
2.2.17 Zendrum 31
2.2.18 Faderfox 32
2.3 Ribbons and strips 32

2.3.1 Ondes Martenot 32
2.3.2 Hellertion & Helliophon 33
2.3.3 Trautonium 33
2.3.4 Theremin cello 33
2.3.5 Buchla Thunder 34
2.3.6 Swarmatron 34
2.4 Gesture based systems 34
2.4.1 Theremin 34
2.4.2 Radio Baton 35
2.4.3 Laser harp 35
2.4.4 Buchla Lightning 36
2.4.5 Very Nervous System 36
2.4.6 Wii Remote 36
2.4.7 Airpiano 37
2.4.8 SoundCatcher 37
2.4.9 Peacock 38
2.5 Wind controllers 38
2.5.1 Lyricon 38
2.5.2 Casio DH range 38
2.5.3 Yamaha WX range 39
2.6 Combined controllers 39
2.6.1 Composer-Tron 39
2.6.2 Buchla 39
2.6.3 VCS3 40
2.6.4 GROOVE system 40
2.6.5 Casio VL-1 40
2.6.6 Synthaxe 41
2.6.7 Ztar 41
2.6.8 You Rock Guitar 41
2.6.9 Reactable 42
2.6.10 Silent drum 42
2.6.11 T-Stick 42
2.6.12 Eigenharp 43
2.6.13 Orbit 43
2.7 Two-dimensional surfaces 43
2.7.1 Misa Kitara 43
2.7.2 Madrona Soundplane 44
2.7.3 Lemur 44
2.7.4 Haken Continuum 45
2.7.5 SLABS 45
2.8 Wearable controllers 45
2.8.1 The Hands 45
2.8.2 Lady’s glove 46
2.8.3 P5 glove 46
2.8.4 Hot hand 46
2.9 Communication protocols 47
2.9.1 MIDI 47
2.9.2 ZIPI 47
2.9.3 OSC 48
2.10 Real-time software 48

2.10.1 Csound 48
2.10.2 Max 49
2.10.3 Pure Data 49
2.10.4 SuperCollider 49
2.10.5 Music Mouse 49
2.10.6 Ableton Live 50
2.11 Discussion 50
2.12 Conclusion 55
Chapter 3. Digital musical instrument design 56
3.1 Core concepts 57
3.2 Classification of digital musical instruments 62
3.2.1 The Hornbostel-Sachs system 62
3.2.2 Atau Tanaka – physical/mechanical 63
3.2.3 Miranda and Wanderley – acoustic similarity 64
3.2.4 Controllerism – ITCH system 66
3.2.5 Human-machine interaction approach 68
3.2.6 Timeline-oriented versus procedural performance 70
3.2.7 Taxonomy of sequencer user-interfaces 71
3.2.8 Thoughts on classification 73
3.3 The instrumental paradigm 74
3.4 Mapping 76
3.5 The design cycle 82
3.6 Evaluation 84
3.7 Conclusion 85
Chapter 4. Interaction design for the digital musician 87
4.1 The importance of a conceptual foundation 89
4.2 Models 91
4.2.1 Predictive models 92
4.2.2 Descriptive models 92
4.3 Describing sensors 93
4.3.1 Degrees of freedom vs. dimensions 93
4.3.2 Resolution 95
4.4 A modular approach 96
4.5 Interaction strategies 98
4.5.1 One DOF sensors with low resolution (on/off) 99
4.5.2 One DOF sensors with high resolution 101
4.6 Independent controllers 103
4.7 Interdependent controllers 105
4.8 Strategies for combining controllers 107
4.8.1 No interdependence 108
4.8.2 Different essential parameters 108
4.8.3 Different non-essential parameters 108
4.8.4 Many-to-one controllers 108
4.8.5 Interactive controllers 109
4.9 Abstract controllers 109
4.9.1 Statistical variables 110
4.9.2 Modal behaviour 112
4.9.3 Automation 117
4.9.4 Saving and recalling settings 119
4.10 Case study: LoopBlender 120

4.10.1 Background 120
4.10.2 Design brief 121
4.10.3 Hardware selection 122
4.10.4 Interface components 122
4.10.5 Discussion 129
4.10.6 Future work 132
4.11 Conclusion 134
Chapter 5. Recontextualising the multi-touch surface 136
5.1 Surface-based interfaces 137
5.1.1 Historical roots 137
5.1.2 XY pads 138
5.1.3 Grid-based interfaces 140
5.1.4 Multi-touch surfaces 141
5.2 Designing multi-touch interfaces 143
5.2.1 Rethinking the GUI 143
5.2.2 Beyond the GUI 145
5.3 SurfacePlayer 149
5.3.1 Aims and objectives 149
5.3.2 Dependencies 150
5.3.3 Implementation 150
5.3.4 Example of use 151
5.3.5 Results 153
5.4 Conclusion 154
Chapter 6. Designing a new multi-touch instrument 156
6.1 Introduction to Oscar 157
6.2 Design objective 158
6.3 Gestural interface 159
6.3.1 Clusters 161
6.3.2 Touches 165
6.3.3 Taps 166
6.3.4 Flicks 168
6.4 Hidden menus 169
6.5 Customisable graphical feedback 170
6.6 Import user programs and audio 171
6.7 Hot-swapping of programs 172
6.8 Csound template 173
6.8.1 CsOptions and global variables 174
6.8.2 UDOs for touch and cluster events 175
6.8.4 Instrument definitions for touch and cluster events 177
6.8.5 Instrument definitions for tap and flick events 178
6.8.6 Reverb, master and accelerometer instruments 180
6.8.7 CsScore 181
6.9 iPad sensors 181
6.9.1 Hard-linking motion data to global variables 182
6.9.2 Combined touch and motion gestures 182
6.9.3 Changing behaviour based on device orientation 183
6.9.4 Purely motion-based gesture recognition 185
6.10 Wireless control 186
6.11 Typical workflow 187
6.12 Case study: DroneTilt 188

6.12.1 Concept 188
6.12.2 Loop parameters 190
6.12.3 Low pass resonant filter 190
6.12.4 Accelerometer 190
6.12.5 Code excerpts 192
6.12.6 Discussion 195
6.13 Linking Oscar to the descriptive model 196
6.14 Assessment of Oscar 198
6.15 Conclusion 199
Chapter 7. Conclusion 201
Bibliography 206
Appendix A: Oscar program template 223
Appendix B: CD contents 229
Appendix C: Papers and publications 230

Table of figures

Figure 3.1 Basic representation of a digital musical instrument 57
Figure 3.2 Adding 'the performer' to the digital musical instrument model 60
Table 3.1 Frequent comments on the positive and negative aspects of acoustic instruments and digital instruments 61
Figure 3.3 Comparing controllers with respect to their resemblance to existing acoustic instruments 64
Figure 3.4 A visualisation of interaction and musical context based on Jens Rasmussen's model of human information processing 68
Figure 4.1 Summary of strategies for single devices and combined controllers 97
Figure 4.2 Summary of abstract control strategies 98
Figure 4.3 T-EMP ensemble performance at Rockheim, Trondheim 120
Figure 4.4 Bimanual division of Korg NanoKontrol2 123
Figure 4.5 Control section for sample group A 124
Figure 4.6 Sample buttons for group A 125
Figure 4.7 Master volume and reverb send for group A 126
Figure 4.8 Loop start and length controls for group A 127
Figure 4.9 Articulation controls 128
Figure 5.1 Comparison of development options for multi-touch musical apps 148
Figure 5.2 SurfacePlayer in use 152
Figure 6.1 *Oscar* running on a 2nd generation iPad 157
Figure 6.2 Two separate clusters 161
Figure 6.3 Individual finger touches represented by grey circles 165
Figure 6.4 Tap event 166
Figure 6.5 Flick event 168
Figure 6.6 Accessing the hidden menu 169
Figure 6.7 Graphics selection menu 170
Figure 6.8 Dropbox menu 171
Figure 6.9 Selecting a hot-swappable program 172
Figure 6.10 csOptions and global variables 175
Figure 6.11 Touch event UDO 176
Figure 6.12 Cluster event UDO 177
Figure 6.13 Instrument 1 – touch event 178
Figure 6.14 Instrument 2 – cluster event 178
Figure 6.15 Tap and flick instrument definitions 180
Figure 6.16 Reverb, master and accelerometer instrument definitions 181
Figure 6.17 CsScore 181
Figure 6.18 OSC message format 187
Figure 6.19 Visualisation of accelerometer axes 191
Figure 6.21 Looping instrument with X/Y auxiliaries 193
Figure 6.22 Except from cluster instrument controlling loop and filter 195

Acknowledgments

I am indebted to Dr. Victor Lazzarini and Dr. Gordon Delap for their support, encouragement and expertise throughout my time here at NUI Maynooth. Their collective influence has enabled me to grow in so many ways as a teacher and student – thank you so much for the countless opportunities that you have opened up for me.

I cannot adequately express the gratitude I feel for my patient, loving and endlessly generous family. Killian, Margaret and James McGlynn have never failed to infuse me with the strength, inspiration and belief that I need to follow my passion in life and keep my feet on the ground – thank you from the bottom of my heart.

I must also acknowledge especially the extraordinary roles played by David O'Connor, Simon Kenny, Nora O'Grady and Momo Furniss. It would take me a lifetime to describe all of the little kindnesses you have shown me, let alone repay them – thank you so, so much.

In no particular order, my sincere gratitude to: Prof. Palmer, Marie Breen, Paul Keegan and the whole staff of the NUI Maynooth Music Department for always making me feel like a welcome member of a dynamic community; Owen Laverty and Lorraine Kane from the Commercialisation Office for your invaluable help and advice; Dr. Charles Markham, Dr. Tom Lysaght and Dr. Joe Timoney from the Computer Science Department for your time and assistance with so many little things; Eilis Murray, Marie Murphy and Melissa Barbier in Graduate Studies for your help and support; Aodhan Coffey, Brian Robson, Alan McCarthy and Shane Byrne for countless fascinating Friday afternoon conversations in the Maker's Club; Jo Mangan and Tom Swift of the Performance Corporation for inviting me into their

delightful and surprising projects; Øyvind Brandtsegg, Bryan Quigley and the T-EMP ensemble for an amazing musical journey; The Ubiquitous Music Group for their inspiring ideas; the wonderful, welcoming people of Ann Arbor; the staff of An Foras Feasa for providing a safe haven to work and grow; Prof. Fischman, Prof. Reilly and John Bradley for a simulating and motivating viva; Jennifer, Barbara, Gráinne, Michael, Feargal, Ciaran and all my fellow Iontas lab dwellers for the good cheer and chats; the security staff of NUI Maynooth for their friendly conversations at odd hours; and all of the beautiful Earthsong folk – Richard Auler, Tom Quinn, Sinead Harte, Jake Quinn, Sarah Ryan, Jack Quinn, Liam Cox, Shane, Samantha Kavanagh, John Harrison, Dave Patterson, Treacy O'Connor, Coralie Mureau, and countless others – for opening my mind and heart.

Finally, I could never have completed this thesis without the unyielding support of the terrific friends who have stood by me over the years: Maghnus Monaghan, Bernard O'Farrell, Steve Kelly, Kevin Coyle, Mark Brennan, Sarah Gill, Eoin Byrne, Simon Greene, Ger Healy, Paul Donnelly, Dr. Mindflip, Edel Doran (and all the Milk & Cookies crew), Luke Kelly, Joe Byrne, Sinead Mawe, Colín ÓhAiseadha, Trina Hanlon, Elaine Hanlon, Lead Balloon, John Power, Mark Farrelly, Joe Hughes, Philip Horan, The Room Appreciation Society, Aidan Guilfoyle, Darryl O'Connell, Mark O'Connell, Brian Connolly, Ben McHugh, James Garvey, Sean Kenny, Eoin Kenny, Lisa Murphy, Thomas O'Boyle, Emma Higgins, Luke Folan, Rory O'Connor, T, Cliffy, Dervil Cody, Dave Byrne, Liz Broderick, Harry Kelly, Bunny Armstrong-Miller, Mary Walton, Denis O'Grady, Brian Carty, Teffia Ki, Donal O'Neil, Michael Gregoire, Fr. O'Gorman and Philip Edmondson.

This work, and any music that arises from it, belongs to all of you. Thank you for sharing this journey with me.

Abstract

The thesis aims to elucidate the process of designing interactive systems for musical performance that combine software and hardware in an intuitive and elegant fashion. The original contribution to knowledge consists of: (1) a critical assessment of recent trends in digital musical instrument design, (2) a descriptive model of interaction design for the digital musician and (3) a highly customisable multi-touch performance system that was designed in accordance with the model.

Digital musical instruments are composed of a separate control interface and a sound generation system that exchange information. When designing the way in which a digital musical instrument responds to the actions of a performer, we are creating a layer of interactive behaviour that is abstracted from the physical controls. Often, the structure of this layer depends heavily upon:

1. The accepted design conventions of the hardware in use
2. Established musical systems, acoustic or digital
3. The physical configuration of the hardware devices and the grouping of controls that such configuration suggests

This thesis proposes an alternate way to approach the design of digital musical instrument behaviour – examining the implicit characteristics of its composite devices. When we separate the *conversational ability* of a particular sensor type from its hardware body, we can look in a new way at the actual communication tools at the heart of the device. We can subsequently combine these separate pieces using a series of generic interaction strategies in order to create rich interactive experiences that are not immediately obvious or directly inspired by the physical properties of the hardware.

This research ultimately aims to enhance and clarify the existing toolkit of interaction design for the digital musician.

"I am standing in a large hall at the sonar festival in Barcelona. On stage is the trio of Christian Fennesz, Jim O'Rourke and Peter 'Pita' Rehberg. All three are playing laptop computers and the movements of their fingertips on trackpads are projected on screens. This assertion of human presence within the improvised evolution of their performance - a dense layering of musical samples and digital processing - adds to the disorientation of the music created in the moment, with minimal physicality and a technology that conceals, rather than reveals. The discomfort of hearing it in a large hall, standing up, surrounded by a half-interested crowd that mills and chatters, leaves me stranded in a mood of ennui. The music sounds wonderful but this is not how I want to hear it."

-David Toop, Haunted Weather: Music, Silence and Memory [185]

Chapter 1. Introduction

"Making music is a process. How well you relate to your tools has an enormous impact on the success of achieving your goals. If you look at the endpoint only and ignore the process, you're depriving yourself of a vital component of the act of creation."

-Stretta, Making music is process [176]

This thesis proposes a descriptive model for digital musical instrument design that focuses upon the individual sensor components of an input device, the data that can be derived from their separate and combined behaviours, and the creation of interaction schemes based upon that data. It is proposed that this model can reveal non-obvious and underused aspects of a physical device. This method is used to design a new generic musical controller - *Oscar* - that reveals fertile and novel interaction modalities specific to the multi-touch surface.

This chapter provides a map of the research project and outlines the evolution of its hypotheses throughout the rest of the thesis. There are three main goals:

- *Situate* the research within the context of live electronic music performance and digital instrument design (1.1). This section highlights some of the most fundamental issues at the heart of the field and distinguishes the questions being addressed within this text from several related, yet separate, topics.
- *Summarise* the concepts central to the research and trace their development throughout the thesis (1.2). This section defines the

central issues under investigation and references the chapters where they are discussed.

- *State* clearly the original contribution this thesis makes to the field of computer music research and identify the future research that it makes possible (1.3).

1.1 Context of research

The foundation for this entire research project can be summarized by the quote that opens this chapter. This remarkable concept has paved the way for unique challenges and inventions alike in the field of music composition and performance. It is clear, especially from the explosive growth of innovation and publication in this area over the past two decades, that the concept of an electronic or digital musical instrument has progressed far beyond the idea of simply interpreting the language of acoustic musicianship using digital equipment [67].

The design of digital musical instruments is no longer a specialized task for a select few who possess arcane equipment and the skills to manipulate it; it has become a legitimate, some would say necessary, aspect of the creative process for any musician who employs digital technology in their live performances. The affordability and flexibility of modern musical interfaces and software coupled with the limitless guidance and inspiration of a dedicated online community have all helped to dissolve the boundaries between performance and creation for the digital musician [67].

This research area is multifaceted and richly influenced by a variety of other fields – including, but not limited to, music performance, human-computer interaction, cognitive psychology, product design, software engineering, interface

design, etc. [78] This thesis draws together aspects of many of these areas of study and aims to synthesise their collective influence in a practical manner.

There are also many sub-categories of research question that are associated with the design of digital musical instruments. This thesis is primarily concerned with investigating the expressive potential of digital interfaces through non-obvious interaction-schemes and design concepts. To ensure a cogent argument throughout this work, it is perhaps wise to identify some of the topics which are influential (inseparable, in some cases) to this discussion but are definitively not the main topic under scrutiny:

- *The classification of digital musical instruments:*

  A detailed overview of the musical applications of sensor technologies can be found in [120], along with a system of classification according to their similarity, or lack thereof, to acoustic instruments. An interesting alternative to this, that focuses more upon the performer's relationship to the equipment as opposed to the functionality of the equipment itself, is the ITCH system. Both of these are described in detail in Chapter 3.

- *The classification of performance gestures:*

  There are many interesting studies available on this topic ([127, 95 and 68] all provide good starting points). A concise summary of various schemes of classification can be found in pp5-18 of [120].

- *Measuring the effectiveness of mapping schemes:*

  This is a rich area of study, with plenty of diverse work taking place, that is beyond the scope of this thesis. Solid foundations for discussion can be found in [69, 70, 120 and 194].

- *Types of sensors and microcontrollers*

  For an overview of the kind of hardware that is typically used for sensing performer input, see the chapters on Sensors &Sensor-to-Computer Interfaces and Biosignal Interfaces in [110, 120 and 142]. A comprehensive summary of more conventional musical controllers can be found in [158].

- *Assessing the suitability of controllers for different tasks*

  The section entitled Comparing Gestural Controllers in [120] provides a clear introduction to this topic. Further discussion can be found in [12, 13, 65, 76 and 202].

1.2 Summary of hypotheses

This section highlights the main points of interest under investigation and identifies the section of the thesis where they are discussed:

- We are operating within an era where the emphasis is firmly upon the development of ergonomic and flexible control devices, as opposed to standalone systems that adhere to the instrumental paradigm (Chapter 2)
- While an analytical language is useful for the design, classification and evaluation of digital musical instruments, the most critical aspect of a system is the relationship between user and interface that arises during performance (Chapter 3)
- A concise and efficient conceptual toolkit for interaction design, compiled especially with the digital musician in mind, has the potential to both expedite and enhance the development process.

Identifying the optimal strategies for combining various control signals can assist the designer in creating more elegant and intuitive interactive systems (Chapter 4)

- The rich expressive capacity of multi-touch interfaces has a tendency to be overlooked due to a widespread over-reliance upon graphics-based interaction paradigms (Chapter 5)
- *Oscar* takes an alternative approach to multi-touch music control and represents a non-obvious, versatile means to develop and perform with interactive audio software on a tablet device. It facilitates the creation of multi-modal interfaces and demonstrates the usefulness of the research concepts that have been defined in previous chapters (Chapter 6)

1.3 Original contribution of thesis

The core contributions of this work are:

- A descriptive model to aid in the design of digital musical instruments independent of the idiosyncrasies of specific devices. This model is presented as a catalogue of useful conceptual tools that can be applied in a wide variety of musical tasks and also provides a comprehensive vocabulary to aid potential developers. It comprises two distinct sections:
    - A micro-level interaction design method that categorises the separate input devices of the digital musical instrument according to the kind of data they generate.

    - A variety of strategies for the interpretation of data generated by these input devices. These strategies accommodate individual devices, devices combined with each other, and abstract controllers such as statistical variables, modes, etc.
- A critical analysis of touch screen music performance applications. This highlights a number of potentially rich control modalities that are underused and describes a proof-of-concept project that investigates their feasibility (Chapter 5).
- A customisable and novel approach to music control using multi-touch surfaces entitled *Oscar*. This is implemented as an iPad / Android application, due for release in late 2014, that embodies many of the design principles discussed throughout the thesis. The software also demonstrates the value of the descriptive model developed throughout Chapter 4 and illustrates how it can be applied to maximize the musical potential of the output generated by any piece of hardware. *Oscar* is discussed in detail in Chapter 6.

The results described in this thesis are derived from intensive practice-based research. This project has been undertaken with the explicit goal of producing a set of concise, useful and versatile concepts that are universally-applicable by the computer music community. The *Oscar* system is also being prepared for general release, subsequent to the completion of this PhD programme, and it will hopefully provide a unique solution for musicians who use multi-touch devices in their live performances.

It is intended that this work will facilitate future research in three different ways. Firstly, as a practical resource for musicians designing interfaces and/or

educators teaching music programming languages – the incremental introduction of various interaction building-blocks on a micro level and their subsequent use on a macro level would form a useful structure for a class, workshop series or course on digital musical instrument design. Secondly, as a starting point for further discussion on design – the strategies described within can doubtlessly be adapted, enhanced or re-framed in a wide variety of contexts. Finally, *Oscar* and its supporting documentation will be released to the general public as the first entirely programmable multi-touch synthesiser app – this will hopefully lead to a wide selection of musicians developing interesting performance techniques using the app and further advance our collective understanding of good practice in multi-touch musical interface design.

To summarise – this research attempts to enhance and clarify the existing vocabulary for interaction design for digital musicians. Future work at post-doctoral level will use *Oscar* to investigate a variety of new approaches to live electronic performance and improvisation.

Chapter 2. A century of electronic musical controllers

"If the process of creating electronic music produces few fine violinists, it nevertheless engenders a new awareness of the nature of sound and our responses to it. In each new experiment, the dynamic between life and its musical reflection is held up to scrutiny…music with new boundaries makes us hear ourselves anew."

-W.A. Mathieu, The Musical Life: Reflections on what it is [112]

This chapter proposes that the most influential changes in electronic music performance technique have been prompted, not by technological progress (as is commonly assumed), but by innovations in design approach. This is supported by a selective catalogue of important developments in electronic musical performance with a particular emphasis placed upon instruments that feature novel control methods and/or interfaces. Informed by this survey, the chapter concludes with a discussion of the relationship between contemporary electronic musicians and their equipment.

Note that the purpose of this chapter is not to provide an exhaustive list of developments in this field (abundant resources are available in [32, 120, NIME.com]). This chapter aims to contextualise the current state of affairs - the *controller era* - where the emphasis is very much upon developing ergonomic and flexible control surfaces, as opposed to standalone systems that follow the *instrumental metaphor* (see 3.1 in [144]).

The goal is not to give an accurate commercial or musical history, but to highlight examples of innovative design or shifts in thinking with regard to the role

of the performer in a live electronic music performance scenario. These examples reveal a tendency towards intelligent repurposing of existing technology, as opposed to entirely new systems created specifically for musical expression.

The following sections categorise electronic instruments in relation to their primary means of control. These categories serve to highlight the most influential developments with regard to a particular input method. However, many of the technologies featured could easily be situated in several categories at once – particularly when it comes to the later 20th Century designs.

2.1 Keyboard based instruments

2.1.1 The Musical Telegraph

The Musical Telegraph was invented by Elisha Gray in 1874 [47]. It consisted of a number of oscillating steel rods and transmitted sound via a telephone line. Later models featured a built-in loudspeaker. The tones were controlled by a two-octave piano-style keyboard – a trend which still prevails in synthesiser design over a century later.

2.1.2 Teleharmonium

Work began on the Teleharmonium in 1898 [26]. Its creator, Thaddeus Cahill, envisaged an electronic musical synthesiser that would broadcast music via telephone lines to public spaces (restaurants, bars, etc.) and private homes alike. The components for this huge device weighed over 200 tons – both installation and maintenance were labour-intensive processes. It featured a complex series of controls that centred upon a touch-sensitive piano keyboard.

2.1.3 Optophonic piano

Invented in 1916 by Futurist painter Vladimir Baranoff Rossiné, the Optophonic piano [11] projected a selection of revolving patterns onto a wall or ceiling using a system of disks, filters and lenses. While it generated no sound, it was conceived as a live performance tool that might be used to accompany a musical performance – a clear forerunner of the modern practice of generating live visuals in response to electronic music (practitioners are often referred-to as VJs or 'visual-jockeys')

2.1.4 Sphaerophone

The Sphaerophone [3] was an expansion of Jörg Mager's Electrophon and Kurbelsphäraphon instruments (see 2.2). Dispensing with the handle mechanisms, the controls featured two small keyboards that could be operated simultaneously, making the Sphaerophone a duophonic instrument.

2.1.5 Hammond organ

Laurens Hammond completed the first design of what would become one of the most enduring and distinctive electronic instruments in April 1935 [32]. The Hammond Organ built upon many of the innovations of the Teleharmonium – with stable intonation and a unique system of timbre control that used drawbars for additive synthesis. Pedal boards and multiple rows of keys, inspired by conventional organs, are common to most instruments inspired by the Hammond.

2.1.6 Electronic Sackbut

Hugh Le Caine built the world's first voltage-controlled synthesizer between 1945 and 1948 [66]. Le Caine had particular ideas about the use of force-sensitive keys for the simultaneous control of volume, pitch and timbre. However, the timbre controls

became more detailed and were designed to be operated by the left hand using a selection of controls separate to the main piano keyboard. These controls allowed the performer to adjust the waveforms and formants of the output. The practice of playing keyboard melodies with one hand while adjusting peripheral controls with the other has become an enduring control technique that is built-into and encouraged by many contemporary keyboard synthesisers.

2.1.7 Mellotron

Produced from 1963 to 1986, the Mellotron was the forerunner of the modern digital sample-based keyboard [160]. Each key triggered a unique strip of magnetic tape for the appropriate pitch, with moveable tape heads permitting the selection and blending of different timbres.

2.1.8 Moog modular

The first production model of Robert Moog's modular synthesiser design was released in 1967. The innovative approach of designing individual modules dedicated to specific signal processing tasks, coupled with the musician-friendly and jargon-free interfaces, rapidly established Moog's influence upon the music world. The inclusion of a touch-sensitive ribbon controller, to complement the standard piano key layout, became a recurrent theme in many of Moog's designs [75].

2.1.9 Optigan

The Optigan was an unusual novelty instrument – a keyboard controlled optical sampler released in 1971 that read a selection of loops from LP-sized discs [63]. Various discs were sold featuring different samples from a variety of genres and musical styles. In addition to the main instrumental sounds that were controlled via

the keyboard interface (with a different loop for each key, analogous to the Mellotron), there were a series of rhythms and sound effects that were triggered via a small matrix of buttons next to the keys.

2.1.10 Synclavier

The first prototype of the Synclavier was developed in 1975 and the basic design continued to advance and improve throughout the 70s and 80s [109]. A highly-influential digital sampling workstation, it was used extensively by composers such as Frank Zappa, Chick Corea and Joel Chadabe. The powerful FM synthesis capabilities were complemented by a user-friendly interface that was designed especially to appeal to the creative process of the working musician, not just the academic research institutes central to its creation. The use of buttons and a large control wheel for editing values was celebrated as an accessible alternative to patching and algorithms.

2.1.11 ADS 100

The Advanced Digital Synthesizer was a high-end analogue device released in 1978 [192]. It featured a dual keyboard design that was microtonally tuneable in addition to a video display for envelopes and disk drive. Only a small number of these synthesisers and their descendants were made, with a retail price of tens of thousands of dollars.

2.1.12 EDP wasp

The Wasp was a budget monosynth released in 1978 that had a 2-octave keyboard with non-moving keys that responded to electro-static touch [192]. Its portability, low cost and quirky design ensured its popularity and later models such as the Wasp

Deluxe and the Wasp Special featured real keys. A modified ‘keytar’ Wasp was also produced by EDP in the early 80s.

2.1.13 Fairlight CMI

The Computer Music Instrument family were a series of powerful sampling/sequencing workstations produced from 1979 to 1985. Aside from the advanced sample manipulation tools built into the software, the CMIs were equipped with a variety of input devices including pitch and modulation wheels, QWERTY keyboard and a graphics tablet for drawing custom waveforms.[158]

2.1.14 Syntar

Created in 1979 by George Mattson, the Syntar was a monophonic performance synthesizer designed to be worn like a guitar in order to free up the musician from behind the keyboard [192]. The neck control featured nine spring-loaded continuous controllers and three switches that allowed the user to perform pitch bends, filter sweeps, modulation and other timbre-shaping effects. These controls in particular allowed experienced players to develop a very smooth and expressive lead style

2.1.15 Roland SH-101

Roland’s 1980 SH-101 was a monophonic bass synthesizer that allowed the performer to alter the VCF, pitch and LFO from the pitch bend control [192]. An optional hand grip controller fitted onto the side of the casing to facilitate keytar-style playing using a shoulder strap and featured an assignable button and modulation wheel.

2.1.16 Yamaha DX7

Released in 1983, the Yamaha DX7 became massively popular as the first affordable digital synthesizer [165]. Its distinctive FM synthesis sounds can be heard in many of the most influential recordings of the 1980s. Some interesting design choices are also worth taking note of – it was one of the first commercial synthesizers to have a full set of MIDI ports and also featured a breath controller input. The lack of a familiar interface of dials (programming was performed using a set of buttons, an LCD screen and a single slider) was a barrier to learning how to customise patches for many users. However, Yamaha would later release a range of expansion cartridges to cater for a range of different sounds and styles.

2.2 Buttons and dials

2.2.1 Electrophon & Kurbelsphäraphon

The Electrophon was developed by Jörg Mager in 1921 for the performance of microtonal music. The performer moves a handle across a semi-circular dial to control a continuous pitch; there was no keyboard control. A later modification, named the Kurbelsphäraphon, featured two switchable tuning handles and a double-pedal mechanism to control volume. [43]

2.2.2 Dynaphone

The Dynaphone was a portable monophonic instrument invented by René Bertrand in 1927 with the support of his friend and collaborator, the composer Edgard Varèse [30]. Like the earlier inventions of Mager, the Dynaphone was operated without a keyboard using a dial. Additional buttons and stops allowed the performer to alter

the timbre and volume of the output – foreshadowing the multi-controller approaches of the latter half of the 20th century.

2.2.3 Voder speech synthesizer

Homer Dudley's 1939 invention was the first device capable of generating human-like speech in realtime [53]. The Voder was operated using a combination of specially-designed keyboards, a wrist bar and a foot pedal. The thirteen keys provide access to a selection of vowel and consonant sounds, the wrist bar alternates between a buzz tone and a hissing noise (for vowels and sibilants, respectively) and the foot pedal controls the intonation. Live demonstrations of the Voder were given by expert operaters, who could clearly hold conversations and even perform simple tunes using the device.

2.2.4 Electronium Scott

The Electronium-Scott, created by Raymond Scott during the 50s, was an instantaneous composition machine that was operated entirely via a series of buttons, patch leads and dials [63]. It had no keyboard controls and is considered to be the first 'self-composing' electronic instrument, using randomness and algorithms to spontaneously generate music – the precursor of automation and generative techniques that would form such an important part of live electronic music in the digital age.

2.2.5 Wurlitzer Side Man

The Side Man was released in 1959 as a percussive accompaniment tool for Wurlitzer's range of organs [164]. It is the first drum machine to use electronically-generated drum sounds. As well as providing a slider to change the tempo of the pre-

arranged beats, the Side Man featured 10 separate buttons which allow the user to trigger individual drum sounds manually.

2.2.6 Roland TR-808

The 808 Rhythm Composer has an instantly-recognisable sound and visual layout – the ubiquitous drum machine of the 80s and a vital tool for hip-hop and electronic producers of the decade and beyond [192]. The simple, unambiguous sequencer layout has been adopted many times in both hardware and software drum machine controls.

2.2.7 Linn LM-1

Roger Linn's classic drum machine used sampled sounds, in contrast to the analog synthesis techniques used by the 808. It was followed in 1982 by the LinnDrum which boasted an improved layout, more samples, live triggering capabilities and the capacity for expanding the onboard samples using upgrade chips [158].

2.2.8 Dynachord Rhythm Stick

Another MIDI controller that is designed to be held like a guitar, the Rhythm Stick (later renamed to The Jamma) was invented by Pete Jones specifically for the live performance of electronic or sampled drum sounds [191]. An intuitive strumming motion and clever fretboard-style design made it possible for performers to develop quite an individual style with the instrument.

2.2.9 Akai MPC60

Designed by Roger Linn and released in 1988, the MPC60 was an integrated drum machine, sequencer and sampling workstation that became very popular in rap/R&B/hip-hop genres [109]. The drum pads, featuring aftertouch and velocity

sensitivity, are still regarded as some of the most ergonomic and highly expressive controllers found in a synthesizer.

2.2.10 Axis 64

Peter Davies' alternative keyboard controller utilises an isomorphic layout based upon the Harmonic Table – replacing the standard manual keyboard design with a honeycomb lattice of hexagonal keys [36]. Standard assignable potentiometers and modulation/pitch bend wheels are also provided. The note assignments of the keyboard itself are highly-customisable and particularly well-suited to alternate tunings, microtonal music and strange keyboard layouts.

2.2.11 Jammer

Invented by Jim Plamondon in 2003, the Jammer is a style of musical interface that is comprised of one (or more) isomorphic keyboard devices and set of thumb-operated expressive controls, similar to those found in contemporary videogame controllers [6]. Jammer-style keyboards utilise the Wicki-Hayden pitch layout which allows players to perform a large variety of scales and patterns quickly and with minimal hand movement. The thumb sticks provide an ergonomic means to control further expressive parameters without compromising finger dexterity.

2.2.12 Monome

The Monome was responsible for igniting a global interest in minimalist generic music controllers [124]. Released in 2006, it consists of a matrix of plain backlit buttons in a wooden casing with a USB outlet. Its abstract appearance and flexibility prompted a growing community of enthusiastic users to write and share software for using the simple device for a huge variety of musical tasks – from the obvious step-

sequencing and pattern writing, to generative systems based upon cellular automata. The Novation Launchpad controller, originally designed for use with Ableton Live, has become a popular alternative for similar interface tasks due to its ready availability (Monome units are produced in small runs and difficult to come by) and significantly lower price [2].

2.2.13 Samchillian

The full name of this unique instrument design is 'Samchillian Tip Tip Tip Cheeepeeeee' – a novel melodic lead instrument invented by Leon Gruenbaum. The physical layout of the Samchillian is a standard QWERTY computer keyboard, but the means of pitch selection is unique to the instrument. Rather than each key triggering its own pre-assigned pitch or sound, the Samchillian layout assigns various positive and negative interval types to each key (referred to as a 'relativistic' keyboard [56]). This facilitates rapid, pattern-based improvisations while making pre-composed melody lines of any reasonable complexity quite difficult to perform indeed. Nevertheless, the unique properties of the instrument make it a most interesting example of the expressive potential of repurposed non-musical hardware.

2.2.14 Maschine

Maschine is a combination of music production and performance software and a complementary hardware controller that offers the immediacy and physicality of a hardware sampler with the added flexibility and depth of software sampling [130]. The device itself features a 4x4 matrix of multi-coloured velocity sensitive buttons alongside a series of buttons, endless potentiometers and an endless rotary encoder. Maschine has a well-supported and large user community and has become a popular alternative to Ableton Live for live sample-based performance using controllers.

2.2.15 Arc

The Arc is a high-quality minimalist controller manufactured by Monome and originally released in 2011 [89]. The continuous equivalent of the Monome grid controllers, the Arc consists of two large endless rotary encoders (with push-button functionality) surrounded by a strip of LEDs.

2.2.16 Tenori-on

Toshio Iwai, Japanese multimedia artist, designed this distinctive grid-based performance tool which was released by Yamaha in 2007 [183]. Iwai explicitly discussed his goal of reuniting the concepts of form and functionality in the age of digital instruments, and the aesthetic qualities of the Tenori-on (frame, LED patterns) are integral to the operation of the device itself. The high price was prohibitive for many musicians, prompting the eventual release of a slightly cheaper variant (TRN-O) and a mobile app (TNR-i) [182].

2.2.17 Zendrum

The Zendrum is a MIDI controller designed specifically for performing rhythmic and percussive material in a live context [226]. The original design was based upon the Drumitar, which was invented by the musician and composer Future Man. Zendrum is distinct from other performance sampler style controllers for a number of reasons – its ergonomic design, guitar shaped body and use of piezo microphones to detect user input (far more sensitive than the force-sensing resistors found in most pad controllers) are all indicative of a serious musician's tool and the rise of 'finger drumming' as a complex and respected form of musical expression.

2.2.18 Faderfox

Faderfox describes a line of generic MIDI controllers that feature a selection of common input devices in various combinations and spatial arrangements [49]. Established by Mathias Fuch in 2004, the range is renowned for its build-quality and versatility. While there is nothing especially novel about the sensors themselves, the variety of potentiometers, buttons, faders and encoders point strongly towards the modern digital musicians' need for robust and non-prescriptive interfaces.

2.3 Ribbons and strips

2.3.1 Ondes Martenot

The inventor of this instrument, Maurice Martenot, was directly inspired to expand upon the potential of the Theremin after meeting its creator in 1923 [26]. There are a variety of control devices used in the more advanced forms of the Ondes Martenot – including a 'floating' keyboard (with moveable keys that controlled vibrato of discrete pitches), a ring attached to a wire (manipulated by the user to control a continuous pitch, much like the Theremin, and forerunner of modern 'ribbon' controllers) and a removable drawer on the left of the instrument body featuring switches that controlled timbre and an articulation key. This key is perhaps the most influential on the playing style itself – whether the performer is playing using the keyboard or the ribbon controller, no sound is generated unless the articulation key is depressed. The key is also touch sensitive, allowing the performer to control the dynamic range and duration of notes simultaneously. This interesting combination of control modalities was inspired by Marenot's own career as a cellist and resulted in an expressive instrument that is still used in specialist cases today.

2.3.2 Hellertion & Helliophon

In 1929, Peter Lertes and Bruno Helberger designed one of the first electronic instruments to use a fingerboard controller [32]. A conductive strip was pressed by a performer, with the horizontal position controlling pitch and the pressure controlling volume. Later models featured several strips arranged in a parallel formation to facilitate polyphonic playing. In 1936, the Helliophon expanded the range of controls to include two piano style keyboards, foot pedals for volume control and a knee lever that controlled vibrato.

2.3.3 Trautonium

Adolf Trautwein first demonstrated his Trautonium in Berlin in 1930 – an electronic instrument controlled by pressing a resistance wire suspended over a metal rail [145]. Switches allowed the user to transpose the instrument and change the combinations of harmonics in the tone, allowing a flexible control of timbre, while a foot pedal changed the output volume. The playing style of the Trautonuim inspired the modern ribbon controller. The composer Oskar Sala composed many pieces specifically for the Trautonium – most famously the soundtrack to Alfred Hitchcock's 1963 movie *The Birds*.

2.3.4 Theremin cello

The Theremin cello (1930) resembles its acoustic counterpart in shape and size, but features a black plastic film fingerboard instead of strings [63]. This allows the performer to select a pitch while volume was controlled using a side-mounted lever. Two rotary dials built into the body of the instrument itself allowed for timbre modifications.

2.3.5 Buchla Thunder

A unique and visually striking controller designed by Donald Buchla in 1991, the Thunder is a pure MIDI controller that consists of a selection of finger strips distributed across a flat, portable surface [19]. The main strips can sense impact velocity, finger position and touch pressure and can also be split into two virtual strips each to double up on the number of control channels. The rest of the strips sense velocity and pressure, but not position. The Thunder remained very much a specialist device, due to its high price tag and complex programming system.

2.3.6 Swarmatron

The Swarmatron was released in 2009 and made famous by its use by Trent Reznor in the soundtrack to the 2010 film The Social Network [129]. The eight built-in oscillators can be tuned and detuned finely using the array of potentiometers, but the main feature is the pair of ribbon controllers that can be used to move the overall pitch centre independent of the other controls. The ability to simultaneously control the relative pitch of all eight oscillators facilitates the creation of dense chordal structures and smooth glissandi.

2.4 Gesture based systems

2.4.1 Theremin

The Theremin was invented in 1917 by Leon Theremin [26] and consisted of a cabinet with two antennae – a vertical pitch antenna and a looped volume antenna protruding from the side of the instrument (although earlier experiments used a foot pedal for volume control). The performer controls a monophonic continuous tone using the proximity of their hands to the antennae. While difficult to play, given the

complete lack of tactile feedback and the resulting need for great physical discipline on behalf of the performer, virtuosic performers such as Clara Rockmore and Lydia Kavina helped to establish the Theremin as more than just a novelty instrument. The Theremin's bimanual, open-handed control system has had particular resonance in recent years for the digital musician, with the advent of portable and affordable motion capture systems such as the Kinect and Leap Motion. One of the first experiments typically undertaken by digital musicians when working with new control devices is to model the behaviour of a Theremin.

2.4.2 Radio Baton

Max Matthews developed this unique control system with Bob Boie in the early 1980s [113]. The instrument uses electric field sensing to track the three-dimensional movements of two batons above a sensing board. A number of foot pedals and potentiometers could also be attached to the system. Matthews performed and lectured with his Radio Baton for many years, demonstrating its usefulness at tasks as diverse as triggering a preset series of notes, moving in three-dimensional parameter spaces and as a percussion instrument.

2.4.3 Laser harp

The Laser Harp is a visually-stunning instrument that was made popular by the composer Jean-Michel Jarre in his live shows [175]. There are a variety of different techniques that can be used to accomplish the effect of a laser harp, all of which depend heavily on the budget and ambient lighting conditions of the performance. There are two main approaches to a laser harp design – framed and unframed (also known as 'infinite') beams. Framed instruments can resemble an acoustic harp design and use photoresistors to detect blocking of the beams, whereas unframed

instruments employ far more powerful lasers that rely upon the light being reflected back for hand detection.

2.4.4 Buchla Lightning

The Buchla Lightning controller consisted of two infra-red light emitting wands that are waved in front of a remote sensor head in front of the performer [20]. Both horizontal and vertical positions are sensed by the Lightning and the wands also feature switches for additional control.

2.4.5 Very Nervous System

David Rokeby's Very Nervous System is a good example of a very popular approach to musical interaction for art installations, galleries, dance, and other art forms featuring camera-based motion tracking instruments [120]. The instrument is markedly diffuse, as opposed to other interfaces that are small and focused, and this characteristic was frequently exploited in the many pieces that Rokeby composed specifically for the system. This immersive contactless style of interface would re-emerge in popularity thanks to the development of affordable webcams and gestural control systems in recent years.

2.4.6 Wii Remote

Launched in late 2006, the Nintendo Wii Remote (or Wiimote, as it is often named) has become a popular tool for musical experimentation due to its low price, portability, ergonomic design and sheer range of sensors [131]. The remote itself features 12 digital buttons, an accelerometer, an infrared camera and can be expanded to include a gyroscope (via the Wii Remote Plus module). The 'remote' label is something of a misnomer when the device is isolated from the gaming

console it was originally designed for – the unit communicates via Bluetooth and therefore does not restrict the user to pointing at any receiving device (with the singular exception of the infrared camera which is used with a peripheral sensor bar or any source of infrared light). Additional features include a small speaker and four LEDs on the body itself and the ability to vibrate. There are a range of accessories that can be attached to expand upon these sensors, the most popular of these being the Nunchuk – an additional handheld controller that features an analog joystick, two trigger buttons and another 3-axis accelerometer. The Wii Remote has been used for a wide variety of musical tasks and experiments, with many popular computer music languages and software featuring dedicated libraries and features designed to work with the controller (see section 7.9.4 for more information).

2.4.7 Airpiano

The Airpiano is a control surface developed in 2007 that consists of 8 infrared proximity sensors mounted in a flat rectangular frame [4]. These sensors are capable of creating up to 24 virtual keys and 8 virtual faders, which are manipulated by the performer moving their hands over the device. A total of 40 LEDs provide visual feedback to the performer. The Airpiano can also take input from an optional footswitch.

2.4.8 SoundCatcher

SoundCatcher is a live-looping and sample manipulating tool from 2009 that uses two ultrasonic sensors to capture open-air gestures from a performer [195]. It was designed primarily with vocalists in mind and can be mounted on a microphone stand for live performances. Vibrotactile and visual feedback cues are used to ensure that the vocalist remains within the active sensing range and a footswitch provides a

further unobtrusive control channel. Typical looping parameters (such as start/end points, crossfade size, etc.) can be controlled in realtime by the performer as they sing.

2.4.9 Peacock

Developed in 2009 as an alternative to wearable and camera-based open handed gestural control systems, Peacock consists of a box shaped interface with 35 infrared proximity sensors facing upwards towards the performer [122]. The instrument can detect three-dimension movements above the surface, without any disruptions from ambient lighting conditions, and send the data to a custom PD patch for musical output.

2.5 Wind controllers

2.5.1 Lyricon

Invented by Bill Bernardi, the Lyricon was the first commercial wind-controlled synthesizer [75]. It paved the way for the Yamaha WX series and Akai's EWI controllers.

2.5.2 Casio DH range

From 1986, Casio released a series of breath-controlled synthesizers known as the Digital Horn range [143]. The instruments were toy-like, resembling a plastic saxophone, but had a number of interesting features that appealed to the more serious musician – chiefly, the ability to use external amplification instead of the built-in speaker and the inclusion of MIDI out.

2.5.3 Yamaha WX range

Yamaha's range of wind controllers hit the consumer market in the mid to late 80s with a variety of interesting interface features – including a choice of mouthpieces, wind and lip control sensors, pitch bend wheel, LED tuning indicators, assignable fingerings and MIDI out [120].

2.6 Combined controllers

2.6.1 Composer-Tron

Osmond Kendall's 1953 invention was, like Scott's Electronium, a step towards making electronic instruments for the composer as well as the performing musician. The unique innovation of the Composer-Tron was its ability to 'read' shapes drawn upon its surface using a special grease pencil [153]. These shapes could be used to represent anything from note envelope shapes to rhythmic passages. The idea of using a graphics tablet or pen as a musical performance tool would be revisited later in the century, both by Iannis Xenakis' Unité Polyagogique Informatique CEMAMu (UPIC) in 1977 and also by the use of Wacom tablet devices as musical interfaces at UC Berkeley's Center for New Music and Audio Technologies (CNMAT).

2.6.2 Buchla

From 1963, Donald Buchla's early synthesiser designs were intended for the performance of experimental music and had some unusual control features – including touch and resistance sensitive plates. Synthesisers such as the Multiple Arbitrary Function Generator and the Source of Uncertainty allowed users to dynamically generate random values for many different parameters. [18]

2.6.3 VCS3

The VCS3 was a unique monosynth with a distinctive appearance that was first released in 1969. Despite its portable design, the VCS3 was in fact a modular synthesiser which permitted a variety of complex patching and signal routing techniques. This was accomplished via a small pin-grid which replaced the more cumbersome wired patch bays common to other modular systems at the time. A joystick was also used to control modulation effects. Later versions produced in the 1970s both expanded the system into larger units (Synthi 100) and packaged it in an ultra-portable case (Synthi A / Synthi AK) [145].

2.6.4 GROOVE system

The GROOVE (Generated Real-time Output Operations on Voltage-controlled Equipment) system was developed by Max Matthews and F. Richard Moore in 1970 [30]. A highly-adaptable system for music composition and live manipulation of pre-composed material, the system was able to store the input actions of a user (keyboard operation, turning dials, etc.) and use the data selectively to control the variables of an analog synthesiser. It represented a large step forward in the development of versatile, continuous parameter control, as distinct from event-based control.

2.6.5 Casio VL-1

Also known as the VL-Tone, this extremely-simple and portable synthesiser enjoyed great popularity throughout its lifespan (1979-1984) [199]. The keys were tiny and soft, with no aftertouch, weight or velocity control, but this led to some interesting and unexpected uses of the hardware – for example, the built-in speaker was commonly cupped with the performer's hand and used to generate a rudimentary filter and/or vibrato effect [222].

2.6.6 Synthaxe

Synthaxe was a guitar-like MIDI controller that was originally produced in 1986 [201]. It generated no sound itself, instead sending performance data to an external synthesizer module. Two sets of strings, nine keyboard keys for note triggering and an assignable tremolo handle made this a powerful controller, but the prohibitive cost (c. £10k) prevented any kind of mainstream use. The Synthaxe did, however, pave the way for more cost effective guitar-to-MIDI solutions in later years, such as those produced by Roland.

2.6.7 Ztar

The Ztar range of MIDI controllers are characterised by their guitar-like bodies and fretboards – unlike a keytar, there are no keyboard style keys present. Instead, the Ztar features separate buttons for each individual fret position and a set of stringlike triggers for actuating notes [173]. Newer models feature additional sensors such as breath controllers, ribbon controllers and joysticks and the overall design approach remains a popular choice for serious guitar synthesizer players.

2.6.8 You Rock Guitar

The You Rock Guitar is an affordable dedicated MIDI guitar controller that uses ‘virtual strings’ on the fretboard as opposed to the buttons common to many similar controllers [101]. The pressure-sensitive frets detect finger positions while a stringlike device on the body of the guitar detects note on events and velocity. The controller also provides a selection of complementary input devices built into the body, including a modulation wheel, tremolo arm and ‘damping bar’ to simulate string muting. Starr Labs also produces a variety of alternate keyboard layout controllers in a similar style. [173]

2.6.9 Reactable

Developed at the Pompeu Fabra University of Barcelona in 2005, the Reactable is a unique tangible tabletop instrument that uses camera-sensing of unique symbols (fiducial markers) mounted on the bottom of plastic blocks to create and modify digital modular synthesizer setups in real time [77]. The Reactable is a visually-striking instrument that is collaborative, intuitive and employs an unambiguous feedback system with no hidden values or confusing menus – making it ideal for galleries, art installations and interactive performances with nonexpert users. The Tangible User Interface Objects protocol (TUIO) [187] and computer vision software ReacTIVision that were both developed as part of the Reactable project have become important tools in their own right, with ports and libraries available for most operating systems and programming languages. A mobile software emulation of the Reactable was released in 2010 [150].

2.6.10 Silent drum

The Silent Drum, developed in 2006 by Jaime Oliver and Matthew Jenkins, was originally designed as a percussion controller [137]. However an alternative approach to using the device without sticks or mallets emerged – the performer manipulates the drum membrane with their hands in order to control the output. This Silent Drum has been made open source and has won numerous prestigious design awards.

2.6.11 T-Stick

The T-Stick instrument has been developed and refined by Joseph Malloch since 2005 [108]. It is a gestural controller comprising numerous sensors installed within a large tube, which the performer manipulates during performance. Touches, taps,

twists, squeezes and shakes are amongst the variety of possible gestures that can be detected by the device, which has been used extensively to perform experimental pieces written especially for the instrument.

2.6.12 Eigenharp

The Eigenharp is a high-end controller designed by John Lambert and launched in 2009 [44]. The instrument encompasses several different types of control surface in an ergonomic stick-like shape – these include a matrix of velocity sensitive keys (that also act as joysticks, with 6 possible directions of movement), a wind controller style mouthpiece, a number of ribbon controllers, and percussion keys on the larger models.

2.6.13 Orbit

The Numark Orbit is a handheld remote MIDI controller whose central feature is a dual-axis accelerometer [135]. In addition to the X and Y data, the controller features a large central jogwheel and 16 assignable buttons split into 4 separate banks. An affordable price and striking tech demos have made the Orbit quite a popular interface during its short lifespan.

2.7 Two-dimensional surfaces

A more detailed discussion of two-dimensional surfaces for music control can be found in Chapter 6. This section lists a number of devices not explicitly covered elsewhere in the thesis.

2.7.1 Misa Kitara

The Misa Kitara was demonstrated in videos online from 2011 – an electronic guitar controller and synthesizer with 24 button frets and an 8-inch touchscreen in place of

the strings on the body itself [121]. A wide variety of synthesizer patches are provided, which can be customised by the user, and played via various multi-touch gestures upon the screen while the other hand selects notes and chords. The device was popularised by Chris Wolstenholme from Muse, who uses it in lieu of a bass guitar in a number of their songs (most notably, the 2012 single ‘Madness’)

2.7.2 Madrona Soundplane

The Soundplane, released in 2011, is a continuous force-sensing membrane set in a rectangular walnut shell [105]. Multiple touches are sensed in three dimensions – horizontal, vertical and pressure values are all available for individual touches. The playing surface can be set up to emulate a fretted string instrument, divided into an arbitrary number of zones, or used in a continuous style analogous to a fretless stringed instrument.

2.7.3 Lemur

The Lemur was first demonstrated in 2004 and released worldwide the following year to great acclaim [74]. A dedicated multi-touch music controller, the Lemur boasted a sturdy metal casing, sleek aesthetic and highly-customisable interface that appealed to professional electronic musicians. The virtual canvas of faders, buttons and other interactive graphical objects could be changed to suit a variety of musical tasks, and the addition of a few hardware buttons ensured that changing layouts during performance was a simple and efficient task. The Lemur was discontinued in light of the competition from tablet applications in later years, but has subsequently been released as an app for iPad and Android devices [99].

2.7.4 Haken Continuum

The Continuum was developed by Lippold Haken and released commercially in 2002 [58]. A large fingerboard controller with a distinctive red playing surface, the Continuum features three dimensions of control – horizontal position, vertical position and pressure. The standard style of setup uses the x-axis for pitch selection (analogous to a manual keyboard layout), y-axis for timbre-shifting effects and the z-axis or pressure to determine amplitude. Jordan Rudess, keyboard player for the band Dream Theatre, is an active user and promoter of the instrument.

2.7.5 SLABS

SLABS consists of two arrays of pressure sensitive touch pads (comprising 24 and 32 pads) designed to be played by separate hands. Each individual pad outputs three values – X, Y and pressure – to a series of programs written in Max/MSP. The creator of SLABS, David Wessel, composes and performs frequently using the instrument [207].

2.8 Wearable controllers

2.8.1 The Hands

Michael Waisvisz created this unique interface in 1984 in an attempt to fully exploit the expressive potential of hand, arm and finger gestures for musical performance [200]. Two ergonomically shaped plates were equipped with various switches and potentiometers while also measuring the hand-tilt angle (using mercury switches) and the distance between both hands (using ultrasound sensing). Apart from the physical interface, The Hands system used a selection of Control Signal Processing Algorithms to introduce randomness and artificial ‘friction’ with a view to enhancing

the expressiveness of the performance. Waisvisz's work was far-reaching and illustrated to many the exciting potential of what he referred to as live composition.

2.8.2 Lady's glove

Sound artist and performer Laetitia Sonami began work on the first version of her Lady's Glove in 1991 – a set of kitchen gloves with Hall effect transducers in the fingertips and a magnet on the right hand allowing varying voltages to be sent to a MIDI convertor and used to control synthesizers. Later versions added microswitches, a pressure pad, resistive strips, wrist-mounted accelerometers and even a miniature microphone. Sonami has performed extensively with her invention, and the successive improvements and modifications of her design are indicative of the important role of feedback, reflection and refinement for the digital musician. Sonami describes her creation:

> The intention in building such a glove was to allow movement without spatial reference, and most importantly to allow for multiple, simultaneous controls. The sounds are now "embodied", the controls intuitive, and the performance fluid. It has become a fine instrument. [170]

2.8.3 P5 glove

The p5 glove was released in 2002 and functioned as a regular mouse as well as a 3D controller for a small number of PC and Mac games. Its wide availability and low cost have made it a popular option for homebrew musical applications [230].

2.8.4 Hot hand

Initially released in combination with a series of specialist effects pedals in 2006, the Source Audio's Hot Hand is a wireless controller that is strapped onto the users'

hand or wrist and sends MIDI control messages to a remote piece of software or hardware [171]. The device contains a three axis accelerometer that can be mapped to, for example, cutoff and resonance of a filter. More unusual applications employ the device as a gestural compliment to standard electronic performance tools [218].

2.9 Communication protocols

2.9.1 MIDI

The MIDI (Musical Instrument Digital Interface) standard was introduced in 1983 to facilitate communication between different synthesizers that supported the protocol [119]. MIDI has had a significant influence upon the development of the commercial synthesizer world and has endured for over three decades in various forms. Its initial goal, communicating between different pieces of hardware, expanded to include sharing of sequences and notation, composition using computers, creating generic control devices to control different synthesizer modules, etc. For all of its usefulness, MIDI retains a narrow focus upon event driven music creation and discrete pitch structures. It was introduced at a time when keyboard interfaces were the norm for electronic music control in the commercial sphere and helped to establish the arguably-limiting predominance of musical controllers modelled upon the traditional piano or organ design. MIDI, despite these shortcomings, is still very much the most widely-supported protocol for communication between electronic music software and hardware.

2.9.2 ZIPI

A (now-defunct) musical protocol that was developed as a collaboration between Zeta Music and the Centre for New Music and Audio Technology at the University

of California, Berkeley (CNMAT) [111]. ZIPI was a comprehensive musical protocol that sought to address the lack of support offered by MIDI in the case of non-keyboard instruments. Additional parameters related to sound generation and timbre were also proposed (e.g. brightness, roughness, etc.). Despite the efforts of its creators to promote it, ZIPI was never adopted by a significant enough number of users to ensure its survival.

2.9.3 OSC

The Open Sound Control specification, or OSC, was designed to address and surpass many of the limitations of the MIDI standard [141]. It uses a system of generic message bundles that, unlike MIDI, can contain multiple types of data and are identified via a customisable hierarchical system of unique descriptive names. OSC data is of a significantly higher resolution than MIDI and can be easily transmitted over the internet and wireless networks, as well as physically linked devices, at high speed. Its flexibility can be somewhat problematic, due to the infinite variety of naming schemes that different users can employ, but it remains the most efficient and flexible communication protocol for interactive digital multimedia systems.

2.10 Real-time software

2.10.1 Csound

Csound is an open source, cross-platform music programming language which evolved from the MUSIC-N series. Csound has an extremely rich catalogue of unit generators (essentially building blocks for synthesis and signal processing algorithms) and an active worldwide community that continues to contribute openly to its development [33].

2.10.2 Max

Max is a visual programming language for music and multimedia applications that was originally designed by Miller Puckette at IRCAM during the 1980s. Max is highly extensible, with a large, ever-growing database of user-generated routines, and remains a popular choice for live signal processing, installations and performance [227].

2.10.3 Pure Data

Pure Data is an open source visual programming language originally developed by Miller Puckette. It has many similarities to Max, most notably its visual patching interface, but also features a wide variety of graphical data structures which can be employed to generate GUIs, graphic scores, etc. The open nature of the software has also led to a large user community that share reusable, modular pieces of code (known as patches) that are often usable as standalone programs [229].

2.10.4 SuperCollider

SuperCollider is a relatively new music programming language (originally released in 1996 by James McCartney) that is largely used for real-time audio processing and algorithmic composition [231]. The environment operates in two parts, scsynth (the server) and sclang (the client), that communicate using Open Sound Control. While it is freely available on most platforms, users generally prefer Mac or Linux operating systems as the Windows development tends to lag behind.

2.10.5 Music Mouse

Music Mouse was a piece of software written by Laurie Spiegel in 1985 for the Mac, which was later ported to the Commodore Amiga, Atari ST and other popular

systems of the era. Essentially a live performance tool, Music Mouse automates many aspects of the generation of musical material – with the ability to use preset chords, melody and rhythmic patterns, harmony etc. The performer is thus free to concentrate on more high-level concerns such as orchestration and articulation. Spiegel championed the use of automation and logic to liberate the creativity of the user, and proposed that such structures can serve to support, extend, and amplify our ability to express and embody the undefinable qualities of aesthetic meaning which we are forever trying to capture [172].

2.10.6 Ableton Live

Since its first release in 2001, Ableton Live has changed the way that musicians interact with and think about computers in live performance [1]. Live is aesthetically and functionally similar to most other Digital Audio Workstations (DAWs), but its particular focus upon live sound manipulation and its myriad features for the preparation of sequences and samples for performance have made it an indispensable tool for a vast number of electronic musicians worldwide. The binary functionality of the Arrangement View and Session View is particularly of note – the former resembles a typical linear audio/MIDI editing program, while the latter has more in common with a hardware sampler or mixing desk and is mostly used for triggering of loops, samples and sequences.

2.11 Discussion

Reviewing the contents of this chapter, we can infer a series of aphorisms that are informed by the trends in electronic music performance tools developed throughout the past century. These observations represent a concise way for us to contextualise our complex relationship with music performance and technology:

1. *Powerful tools for the performance of electronic music are not necessarily reliant upon new technological innovations*

   This statement becomes more apposite as we progress into the latter half of the 20th century. Some of the most radically influential and widely-used control modalities for the performance of electronic music have been based upon a simple repurposing of readily-available hardware. Buttons (Roland TR-808, Linn LM-1, Akai MPC60, Monome, Axis 64), cameras (Very Nervous System, Kinect), and accelerometers (Wii Remote, Hot Hand, Orbit) have all, like the vinyl turntable, been re-appropriated for a wide variety of musical tasks.

2. *Cost-effective digital circuitry and high-capacity storage devices led to a homogenisation of keyboard-based synthesisers*

   Analogue circuitry was used in the 1950s because it was cheap, easy to integrate into new systems and well-understood. The same could be said of subtractive synthesis and the keyboard interface which still dominates the commercial market. The development of cost-effective digital circuitry for FM synthesis, coupled with ROM chips that could accommodate large banks of samples, led to a surge of interest in FM and sample-based synthesis. Once this technology became powerful and popular enough, the distinctions between various synthesiser and keyboard manufacturers started to blur and break down. This prompted a mass homogenisation of digital music workstations in the late 1980s and throughout the 1990s, as manufacturers focused upon packing a wide selection of sampled instruments into their products.

3. *The introduction of standard communications protocols for digital musical instruments, alongside the widespread availability of personal computing systems, has prompted a change in direction*

   MIDI has, for all its limitations, been instrumental in the movement towards more expressive performance equipment for the digital musician. The separation of synthesiser modules and control surfaces in the commercial sphere has made the development and acquisition of high-quality controllers a priority for performers. While MIDI was at the forefront of a wave of keyboard-based controllers, the gradual adoption of OSC seems to have enabled a further shift away from single-purpose instrumental designs.

4. *The development of performance control surfaces has been heavily-influenced by the development of synthesiser hardware*

   Early modular systems relied heavily upon patching for synthesis control, leading to a predominance of cables and sockets on the front panel of the instrument. As the technology developed, common signal flow decisions were allocated switches and knobs that began to dominate the control surfaces of the late 1970s and early 1980s. The digital technology of the 1980s led to a much more minimalist style of interface, with multi-function buttons and LCD displays becoming commonplace. Again, lowering costs and increasing processor power led to the widespread and heavy use of built-in screens on devices, with more expensive models featuring detailed multi-menu operating systems. Hardware controls, such as knobs, buttons and faders, briefly became ancillary devices. The desire for customisation, alongside the rapid rise of the touch screen or tablet computer as a musical controller in its own right, has since ushered in a new level of interest in

generic, assignable control devices – modern controllers typically feature a mixture of knobs, faders, dials and buttons with indeterminate functionality.

5. *The production and/or preparation of musical material has become a more integral part of live performance practice*

   In the past, a combination of ergonomic concerns, processing speeds and the predominance of single-purpose hardware/software tools led to a separation of multiple processes common to digital music creation. Composing, recording, processing, arranging and playing material were, out of necessity, separate activities. The integration of many of these processes into popular performance software, such as Ableton Live, has led to a rise in techniques such as live sample manipulation, pattern editing and loop-based performances. Many musicians using digital technology (particularly in the case of looping and sampling) expose their audience to the creative process by generating their material entirely on stage.

6. *There has been a significant rise in the popularity of contactless gestural sensors*

   The use of nonphysical, nonmechanical gestures (see section 3.2.2) for musical control was, until quite recently, restricted to very specialised contexts due to both the expense of the hardware and the particular ambient conditioners necessary for it to function correctly. The emergence of affordable and powerful camera-based systems such as the Playstation Eyetoy, Microsoft Kinect and the Leap Motion controller has opened up the possibilities of contactless gestural music performance to a widespread community, whose interest in motion control has already been piqued by devices such as the Nintendo Wii Remote.

7. *Electronic instruments can no longer be easily divided into categories of solo or accompaniment instruments – recent designs favour an integrated approach*

   A wide variety of acoustic instruments from around the world can be categorised according to their typical role in a group performance, either as solo or accompaniment instruments. Until quite recently, electronic instruments often followed this pattern. Drum machines, sequencers, chord or pad-based synthesiser patches and samplers could, depending on the context, be allocated an accompanying role in relation to more melodically-focused devices such as the Theremin or monophonic synthesisers. Modern systems can operate equally well in both roles and often permit the performer to accompany themselves using layers of pre-arranged material or similar automated processes.

8. *We are currently operating within the controller era – where generic customisable tools are preferred over single-purpose digital instruments*

   The concurrent and complementary roles of the digital musician – as composer, sound engineer, producer, programmer and performer – are both reflected and enabled by the wide variety of generic control devices that are available today. Contemporary electronic musicians are far more likely to be found using a laptop and USB control device than a keyboard synthesiser – the separation of software and hardware (alongside the variety of generic controllers available) allows musicians to choose the precise set of controls that best suit their live performance requirements. Today, the ability to radically customise a new piece of musical equipment and incorporate it into an existing system is not just highly praised, it is expected.

2.12 Conclusion

We have examined a wide selection of influential tools for the performance of electronic music from the late 19th century to the present day. While there are several strong examples of technological innovation acting as a catalyst for dramatic musical invention (Teleharmonium, Theremin, Moog modular, Reactable) there are many more instances where a novel, apposite juxtaposition of existing technologies has led to the development of radically-experimental instrument designs and performance techniques.

There is clear evidence of this trend in the popular and new interface designs of the past two decades – where some of the most highly respected and coveted interfaces (Monome, Arc, Haaken Contiuum, Zendrum) are designed entirely around a selection of well-established and relatively simple sensors. Informed by this survey, we arrive at a series of design aphorisms that support the concept of a *controller era*. This recent interest in generic controllers is indicative of a worldwide community that is beginning to look beyond the facile features of novel controllers and take a greater interest in the interactive processes at the heart of digital musical instrument design.

Chapter 3. Digital musical instrument design

"Perhaps uniquely in the history of the performance of music, we are able to separate entirely the production of sound from the means used to control it."

-Ross Kirk, in New Digital Musical Instruments: Control and Interaction Beyond the Keyboard [120]

This chapter outlines the key concepts of digital musical instrument design – discussing the literature, definitions and design conventions of modern musical controllers. It contains a comprehensive summary of conceptual models that are useful for understanding the current state of the art – including traditional organology, control dislocation, the digital musical instrument, Miranda & Wanderley's classification system (hyper/extended instruments, instrument-like controllers, alternative controllers, etc.), the controllerist ITCH model (Instrument, Transfiguration, Conjuration, Hybrid), various types of mapping schemes (one-to-many, many-to-one, etc.) and so on.

This chapter also allocates some space to the philosophical discussion that has taken place regarding the difference between a 'controller' and an 'instrument'. This, in turn, introduces the importance of considering the performer's point of view, a vital recurring theme throughout the rest of this thesis. We are also introduced to the argument concerning the separation between the music and the means by which it is performed. The main purpose of this discussion is to map-out the conceptual landscape that is explored, in greater detail, throughout subsequent chapters.

3.1 Core concepts

This section introduces the concept of a digital musical instrument and differentiates it from both electronic instruments and controllers. The model of control dislocation, a vital aspect of any digital musical instrument, is described in detail and the roles of its constituent parts are made explicit.

The term *digital musical instrument* is used to denote any musical system that contains both:

- A control surface (also referred to as a gestural or performance controller, an input device, or a hardware interface)
- A sound generation unit

Both of these modules are independent entities that exchange information via mapping strategies [120] – the diagram below is a basic representation of this model.

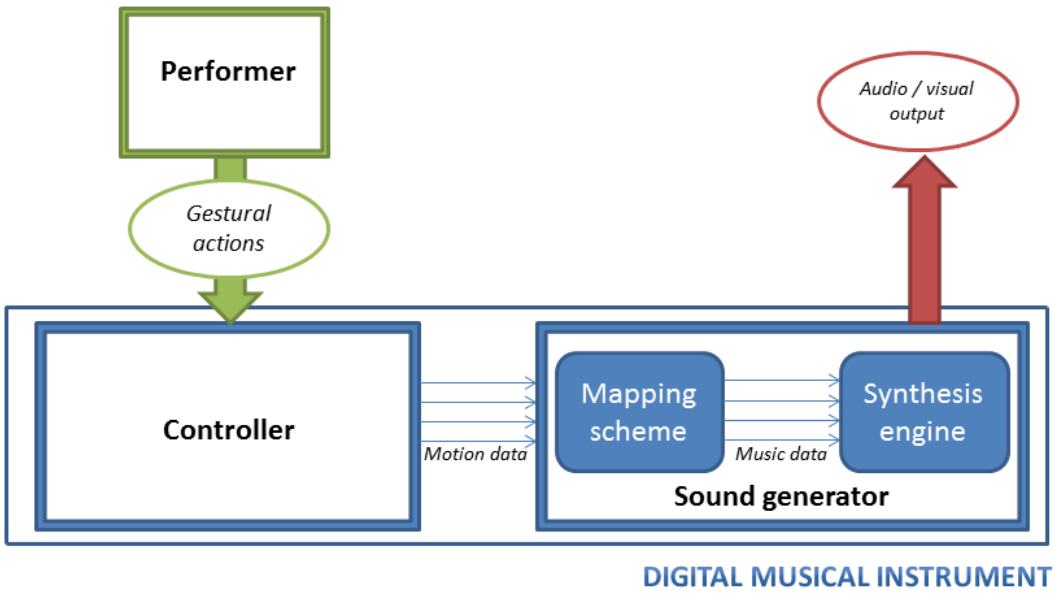

Figure 3.1: Basic representation of a digital musical instrument

The most fundamental aspect of this concept is the separation of the instrument into two distinct units. The gestural controller is where physical

interactions performed by the user are sensed. The sound production unit interprets data from the controller and uses it to drive some kind of synthesis or digital signal processing algorithms. This basic model is shown in Figure 3.1.

The concept of this division of the system into two distinct parts, referred to here as *control dislocation* is critical to understanding the design issues that face musicians using digital musical instruments. The term has been used to describe the difficulties faced by performers using tape/electronics live [45] but for the purposes of this description no pejorative connotations are intended. It is best described via a comparison with traditional (acoustic) instruments, which relied exclusively upon the acoustic properties of tubes, strings and membranes until the invention of the first electronic instruments in the late 19th century [120].

This means, in essence, that the sound being produced and the means of performance were inexorably and intrinsically-linked. In other words, the playing methods imposed by acoustic instruments are determined by their physical construction [78]. This connection is totally-absent in the case of digital musical instruments: the connection between the action of a performer and the resulting sonic behaviour is *completely arbitrary and designed*.

The nature of this connection determines the relationship between the two units and is referred to as the *mapping* layer. This consists of the liaison strategies between the actions of the performer and the sonic behaviour that they cause or influence and will be the subject of much discussion throughout this thesis. Two digital musical instruments that consist of identical pairs of gestural controller and sound production units may behave in any number of entirely different ways according to the mapping strategy in use: it is the very essence of a digital musical

instrument and determines to a large extent the psychological and emotional state of the user during performance [70].

The presence of feedback in this system should also be noted. Contrary to what initial impressions might suggest, primary feedback has little to do with sound output. The primary feedback of any digital musical instrument comes from the physical interactions, if any, that the performer has with the control surface itself. The experience of sound (or any media) that is generated in response to user input is considered secondary feedback, for the purposes of this model, in keeping with the model proposed by Wanderley in [233]. Regardless of the source, the feedback mechanisms employed within a digital musical instrument act as cues-for and reinforcements-of user articulation. In some cases, the feedback mechanism can exert a direct influence upon the control of the instrument itself – audio within a feedback loop can be analysed and used to generate or influence control signals, for example. The model of digital musical instrument presented here will obviously require some modification in special cases like these, as the feedback becomes part of the controller itself, but the vast majority of designs fit into this diagram comfortably.

This diagram serves mainly to illustrate the basic structure common to all digital musical instruments - there are many, many more factors at play that could potentially be incorporated into this view. Research on new digital musical instruments often focuses upon a specific aspect of this model (input devices, mapping, feedback, etc.) and attempts to expand our understanding of it. For example, a more detailed diagram is presented as Figure 3.2 which shows an expanded view of the user experience in addition to the units described above.

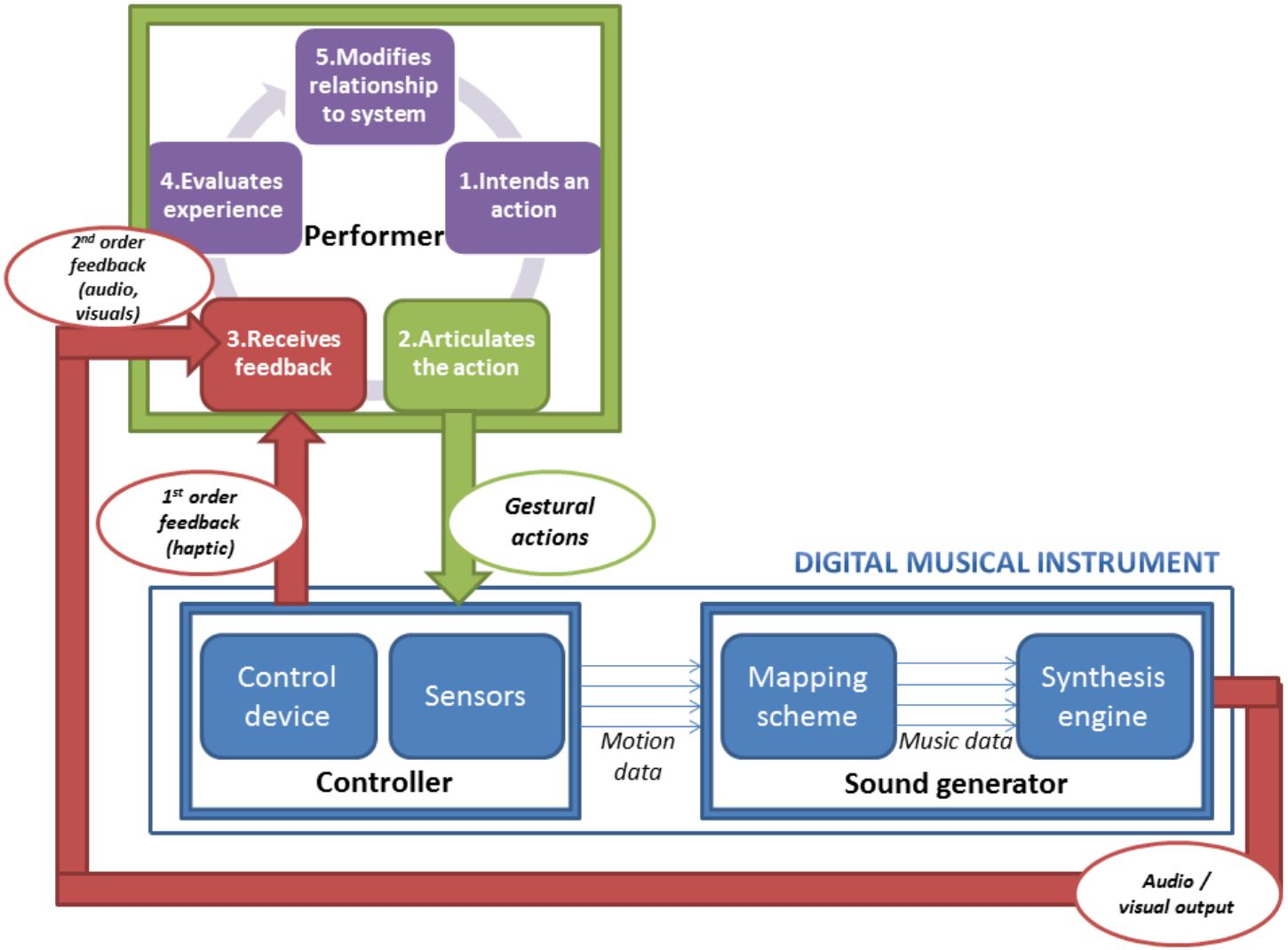

Figure 3.2: Adding 'the performer' to the digital musical instrument model

The relationship between our perception of acoustic instruments and digital instruments is something that is worth examining for the prospective designer; an awareness of the relative strengths and weaknesses of both fields can often act as a catalyst for both stimulating discussion and design innovation. The comments gathered by Magnusson and Mendieta in their 2006 survey (reproduced here as Table 3.1) provide a succinct introduction to some of the most pertinent issues that arise when comparing both types of instrument:

Acoustic – Positive	Acoustic - Negative
Tactile feedback	Lacking in range
Limitations inspiring	No editing out of mistakes
Traditions and legacy	No memory or intelligence
Musician reaches depth	Prone to cliché playing
Instrument becomes 2nd nature	Too much tradition/history
Each instrument is unique	No experimentation in design
No latency	Inflexible – no dialog
Easier to express mood	Less microtonality or tunings
Extrovert state when playing	No inharmonic spectra
Digital – Positive	**Digital - Negative**
Free from musical traditions	Lacking in substance
Experimental – explorative	No legacy or continuation
Any sound and any interface	No haptic feedback
Designed for specific needs	Lacking social conventions
Freedom in mapping	Latency frequently a problem
Automation, intelligence	Disembodied experience
Good for composing with	Slave of the historical/acoustic
Easier to get into	Imitation of the acoustic
Not as limited to tonal music	Introvert state when playing

Table 3.1: Frequent comments on the positive and negative aspects of acoustic instruments and digital instruments [106]

3.2 Classification of digital musical instruments

There is a very real danger, when looking at the vast selection of musical devices at our disposal today, of becoming distracted by the sheer variety of approaches to electronic music creation [107]. Systems of classifying digital musical instruments vary in their approach due to the somewhat abstract nature of their composition, as seen in the previous section. Regardless of their different priorities, these systems provide useful conceptual tools for discussing and developing our understanding of interactive musical systems.

3.2.1 The Hornbostel-Sachs system

The science of classifying musical instruments is known as organology. The most widely-adopted system was proposed by Erich von Hornbostel and Curt Sachs in 1914. This macrotaxonomy – known as the Hornbostel-Sachs system – groups instruments in a hierarchical structure with a numerical referencing system based-upon the Dewey Decimal System. Traditionally the Hornbostel-Sachs system had four main categories, each of which are divided into a multitude of sub-classifications:

1. *Idiophones*: "The substance of the instrument itself, owing to its solidity and elasticity, yields the sounds, without requiring stretched membranes or string"
2. *Membranophones*: "The sound is excited by tightly-stretched membranes"
3. *Chordophones*: "One or more strings are stretched between fixed points"
4. *Aerophones*: "The air itself is the vibrator in the primary sense" [64]

Two different approaches to accommodating electronic instruments within the H-S system are introduced in [64]. The first of these emphasises the importance of the

presence of an oscillator in an authentic member of the electrophone family [10]. The second adopts a more modular view – seeing the electronic instrument as an assemblage of distinct elements and using a mathematical system to give a more-complex and accurate description of its constituent parts.

These discussions, while lively and thought-provoking, serve to illustrate the futility of classifying devices of such intrinsic malleability as electronic musical instruments within a scheme that was not originally designed to accommodate them. The rapid growth of new musical ideas, technology and ways to combine the two has established electronic instruments as the perfect reinforcement to Hornbostel and Sach's opening caveat:

> Treatises on systems of classification are by and large of uncertain value. The material to be classified, whatever it may be, came into existence without any such system, and grows and changes without reference to any conceptual scheme. The objects to be classified are alive and dynamic, indifferent to sharp demarcation and set form, while systems are static and depend upon sharply-drawn demarcations and categories. [64]

3.2.2 Atau Tanaka – physical/mechanical

Atau Tanaka was the first musician to work with BioMuse – a unique performance system that generated control data for music and visuals by measuring bioelectric signals produced by the human body. Tanaka used the system to perform with a trio called Sensorband (alongside Zbigniew Karkowski and Edwin van der Heide) from1993-2003 [163]. His classification for gestural controllers is concise and focuses upon the relationship between the performer's body and the system [179]. Controllers are simply grouped according to two different categories – namely, their

mode of interaction (physical or nonphysical) and the kind of manipulation which takes place (mechanical or nonmechanical).

Although it may appear basic at first glance, this approach serves to illustrate how a simple descriptive model can enable a designer to clearly differentiate actions such as turning a potentiometer (physical and mechanical) from performing gestures in front of a camera (nonphysical and nonmechanical) or using biosignals such as electroencephalogram/EEG (physical and nonmechanical). It is primarily a system which focuses upon the nature of the sensors themselves and the means through which the performer engages with them.

3.2.3 Miranda and Wanderley – acoustic similarity

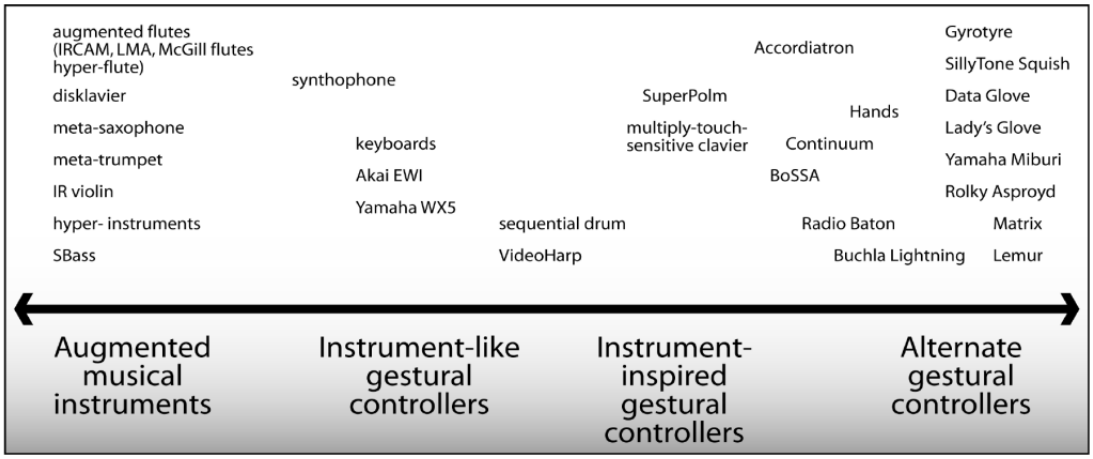

Figure 3.3: Comparing controllers with respect to their resemblance to existing acoustic instruments [120]

Miranda and Wanderley propose a didactical method of comparison based upon the similarities between gestural controllers and existing acoustic instruments. This is best seen as a continuum, rather than a series of rigidly-defined categories – a fact that is reflective of the organic evolution of digital musical instruments in recent decades. The four categories are shown in Figure 3.3 (along with some examples of how controllers can be placed within this model) and are described as follows:

1. *Augmented musical instruments:* Also known as hybrid instruments, hyperinstruments or extended instruments – these are acoustic (sometimes electric) instruments that have been extended by the addition of extra sensors. In general, the instrument functions exactly as it would have prior to modification – the technological components serve to increase the range of expression through added extra features or parameters to alter the sound.
2. *Instrument-like gestural controllers*: These instruments feature a control surface that is modelled after an existing acoustic or electric instrument with the goal of emulating the original. These are often used by musicians who wish to expand-upon the sonic capabilities of their existing instrumental technique.
3. *Instrument-inspired gestural controllers:* These instruments feature control surfaces that are directly-derived from those of an existing instrument, yet they do not aim to reproduce exactly the functionality of their acoustic counterparts. Sometimes they can be designed with a view to overcoming some of the intrinsic physical limitations of the original and provide alternative ways to employ existing instrumental skills (e.g. by providing an alternate fingering system for a flute-like controller).
4. *Alternate gestural controllers:* Instruments that belong in this category do not bear any striking resemblance, in appearance or means of operation, to any existing instruments. Given the rather broad spectrum of devices that might be placed in this category, a number of subdivisions have been proposed. For example, in [127], Mulder suggests a further breaking-down into one of the following three groups:

a. *Touch controllers* that require the performer to make physical contact with the control surface and provide a haptic representation.
b. *Expanded-range controllers* that may require a limited form of physical contact or do not require any physical contact but have a limited 'range' – that is to say, the performer is free to make certain movements that do not have musical consequences.
c. *Immersive controllers* have few or no restrictions on performer movements. Consequentially, the performer is within the sensing field of the controller at all times. A further three subdivisions are suggested according, not to the actual physical form of the controller or sensors, but to the visualisation of the surface and the accompanying mapping strategies used:
    i. *Internal controllers:* The control surface visualisation is the physical shape of the human body itself.
    ii. *External controllers:* The control surface is visualised as separate from the performer's body. It may even be impossible to visualise it as a physical shape.
    iii. *Symbolic controllers:* The control surface is not visible; it requires some sort of formalised gesture set (sign language, conducting) to be operated.

A catalogue of interesting examples for each of the above categories can be found in [120], pp.21-101.

3.2.4 Controllerism – ITCH system

An interesting alternative approach is to look at technique and practice, as opposed to physical characteristics or acoustic similarities, when classifying devices. An

example of this kind of system is proposed by a contributor to Controllerism.com (an online community of electronic musicians that use generic controllers in their live performances) in an article entitled ‘Types of Controllerism’ [28]. The ITCH system is an abbreviation of the four categories it comprises (Instrument, Transfiguration, Conjuration and Hybrid) and groups interfaces and/or musicians into one of these categories based entirely upon their personal approach to audio creation during performance. This of course means that two physically-identical interfaces, when used by two musicians with contrasting styles of performance, will be placed in two totally-different categories. With regard to versatile control surfaces, which can be employed in a theoretically-limitless variety of ways, this seems to make a lot of sense for the practicing musician. The categories are defined as follows:

1. *Instrument:* Musicians in this group use an external audio source (e.g. an electric guitar, a didgeridoo, a hardware synth, an MPC, beatboxing, etc.) in conjunction with audio created by the software. This category can be combined with any of the following approaches.

2. *Transfiguration:* The focus of the interaction is changing elements into something else. For example, using pre-recorded loops and patterns that are altered using effects and combined together in various ways to create a track or live set.

3. *Conjuration:* Creating something out of nothing. The focus here is using virtual instruments or a collection of samples to create a track or live set through overdubbing, live looping or playing everything manually from start to finish.

4. *Hybrid:* Performance setups in this category rely heavily upon techniques from both the Transfiguration and Conjuration groups. While 'C' performers might use some occasional loops and backing, and 'T' performers might occasionally play one-shot samples/rhythms/melodies manually, 'H' performers use both to the extent that the omission of either would result in an incomplete or impossible setup.

This approach is proposed with a very definite audience and type of performer in mind but it has an extremely useful core concept: if we do insist upon classifying interfaces for this purpose, perhaps we should focus upon the individual configuration, intent and style of a given performer. This certainly seems like a line of thought that begs further investigation.

3.2.5 Human-machine interaction approach

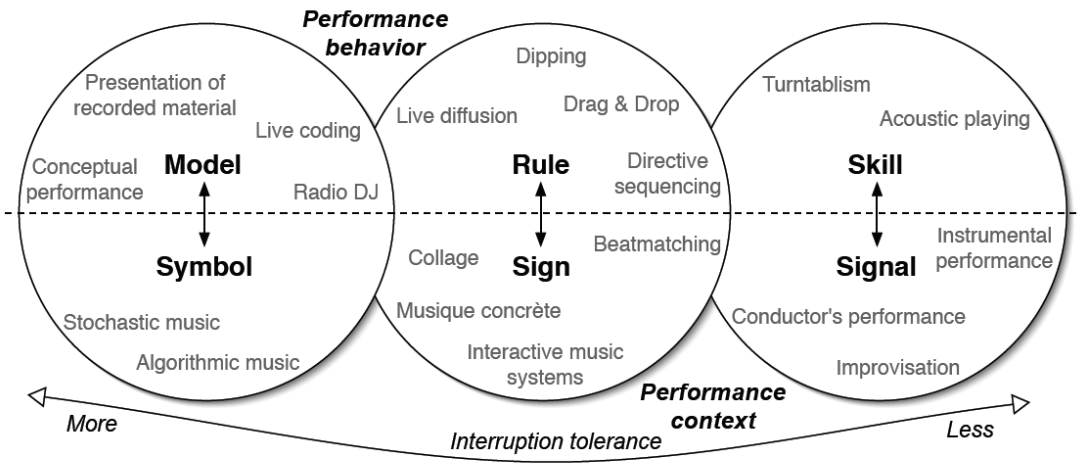

Figure 3.4: A visualisation of interaction and musical context based on Jens Rasmussen's model of human information processing [232]

This model focuses upon the context of a performance and views the computer as "a semiotic, connotative machine that hypothesizes design criteria" [107]. It is based upon Jens Rasmussen's model of human information processing and defines three distinct types of performance behaviour: *skill-*, *rule-* and *knowledge-based.* The latter is renamed *model-based* interaction behaviour in accordance with a suggestion of Rasmussen himself and also to avoid the conflicting connotations that might arise with the concept of musical knowledge. The categories of behaviour are shown in Figure 3.4 and are described as follows:

1. *Skill-based:* Typified by physical gestures made in response to a continuous signal, this behaviour is closest to the typical understanding of instrumental performance (in the traditional, acoustic sense). It has been observed that very few activities are restricted entirely to the skill-based category – a musician usually depends on the experience of previous attempts in conjunction with the real-time signal input that characterises this section.
2. *Rule-based:* This category sees the performer's focus shift away from controlling a signal towards controlling higher-level processes, such as selecting and sequencing previously-arranged material. As in the skill domain, interactions and interfaces within this category can be further differentiated according to the rate at which the performer can effect change.
3. *Model-based:* A musician operating at this level can only exert a low level of control over the outcome at a low rate. Interactions in this domain are goal-oriented and goal-controlled – the performer is typically involved with the rational formulation of a plan to reach a particular goal.

The lower portion of the accompanying diagram also shows how these three types of behaviours can be coupled with Rasmussen's categories of human information processing in accordance with the performance context and environmental conditions therein: *signals, signs,* and *symbols* are representative of the kind of information that is being processed in a given domain (i.e. skill-based=signals, rule-based=signs, model-based=symbols).

Despite coming from two very different perspectives, we can see a clear correlation between this model and the ITCH system:

- *Conjuration-type* performances/interfaces operate primarily on the level of *signals* and thus can be placed comfortably within the *skill-based* domain.
- *Tranfiguration-type* setups place more of an emphasis upon the manipulation of *signs* and occupy the *rule-based* domain.
- As the human-machine interaction model is represented as a continuum, rather than discrete categories, *Hybrid-types* can be accommodated by an appropriate location between the *skill-based* and *rule-based* domains according to their primary reliance upon either *signs* or *signals*.

The *model-based* domain has no close counterpart in the ITCH model but this is understandable given the live-performance focus of the Controllerism.com community.

3.2.6 Timeline-oriented versus procedural performance

This framework is concerned more with the design and use of software interfaces onscreen in laptop-based performances, rather than hardware devices, but it contributes further to the discussion on approaches to practice that has been established by 3.2.4 and 3.2.5.

This model, proposed by Zadel and Scavone in [223], differentiates music performance software based upon the way that it handles sequencing tasks and control data. Two categories emerge:

- *Timeline-oriented performance control:* This solution focuses upon linear pieces of audio and control data that are positioned in time, processed and overlaid to create full pieces of music. Analogous to offline sequencing except that certain aspects of the piece are left to be triggered and/or manipulated in real-time during a performance. Both Ableton Live and Reason are cited as examples of this kind of system.
- *Procedural performance control:* These interfaces focus-upon allowing the user to define and modify processes in real-time to shape the musical output during a performance. For visual dataflow languages, such as Pure Data and Max/MSP [147], this typically involves the manipulation of a patch comprising signal generators and modifiers that has been prepared prior to the performance. In the case of more text-centric languages, such as Csound [33], SuperCollider [116] and ChucK [204], the practice of *live coding* is more common. The authors point out that, in this latter case, the creation of the procedure itself *is* the performance, or at least part of it.

3.2.7 Taxonomy of sequencer user-interfaces

A music sequencer is, in some respects, analogous to a written score in traditional music composition. It comprises a piece of hardware or software that stores data related to a piece of music (e.g. note values, melodies, timing) and sends this data to a sound generation module. This model aims to provide an analytical framework for the categorisation of sequencer-based user-interfaces or indeed any interface that

features a sequencing component for the linear arrangement of musical material in time [42]. There are five axes defined within this system:

1. *Medium:* This can be more *textual* or *graphical*. The extent to which the interface relies upon either visual or text-based abstractions – the immediacy and learnability of the former is in marked contrast to the flexibility and customisation-potential of the latter.
2. *Abstraction level:* This can be more *predetermined* or *custom*. Abstractions reduce cognitive load during performance by highlighting relationships between similar objects and hiding/reusing details. Common predetermined abstractions include MIDI and audio 'clips'. Custom abstractions require more of an initial investment on behalf of the user, to understand and create their own hierarchies of objects and their associated behaviours, but they can offer more flexibility and control in a well-designed system.
3. *Linearisation stage:* This can be more *delayed* or *eager*. The linear ordering of musical material can occur at different stages throughout the composition/preparation process of a performance. At its most extreme, or delayed, the ordering is not determined until the actual moment of performance itself. More eager systems demand a predetermined ordering of material which can, in turn, lead to more simple interfaces and allow the musician to concentrate on other aspects of their performance.
4. *Event-ordering:* This can be more *data* or *control-flow based*. Control-flow systems allow the user to specify the final order of sequencing in terms of events. This may include programming techniques such as conditional tests, loops and suchlike. The data-flow paradigm, on the other hand, is found in systems where the user must determine the final sequence of data flowing

through a computational system. This is commonly-used for effects-control and automation in digital audio workstation software, for example.

5. *Applicability:* This can be more *special* or *general-purpose*. Special-purpose sequencers are defined as demonstrating a preference towards a particular style or styles of musical sequencing. This can allow certain aesthetic considerations to be taken into account when designing the interface and therefore increase the simplicity and efficiency of the interface. General-purpose applicability refers to systems which are equally-useful when performing a variety of sequencing styles.

Combining these characteristics in all of their various permutations gives a total of 28 distinct types of sequencer. The authors also apply the taxonomy to a number of common performance applications in order to demonstrate its use.

3.2.8 Thoughts on classification

The selection of taxonomies outlined in this section represent a broad spectrum of approaches – from the simple and universal down to the most complex and specific. It is most striking to observe how difficult it can be to develop a single all-encompassing model for digital musical instrument classification in the same vein of the H-S system – the sheer expanse of creative ground covered by even the most basic of computer music tools makes it hard to conceive of such a system. Perhaps this is the wrong goal to be aiming towards. While no one system that we have outlined above can claim to include all the factors as broad as sensor-type, interface-type, performance-style, musical-context, etc., each one manages to shed a little more light on a different aspect of arguably the most rapidly-evolving approach to musical expression in history. As musicians, designers and scholars we should welcome any opportunity to view our discipline in a new way – every new taxonomy

that might be proposed should be considered as offering a unique new perspective on the field, rather than a prescriptive labelling system.

3.3 The instrumental paradigm

Issues of terminology often arise when discussing digital musical instruments. A musical instrument consists of an excitation source which the performer causes to oscillate using their own physical energy. The sound may be modified by the performer using the available control mechanisms of the instrument before, during or after it reaches a resonating system that conveys the resulting vibrations to the air. On the other hand, the only part of a digital musical instrument that the performer comes into contact with – the controller – merely sends data to a sound generating system. It does not allow the performer to directly excite or modify the sound from a physical standpoint in the same way that an acoustic instrument does – so where does the *instrument* live in the digital musical instrument?

The acoustic piano provides an interesting discussion point – the strings (excitation source) are excited by the hammer mechanism and subsequently send vibrations throughout the frame of the piano (resonating system). The performer, however, does not have any direct physical contact with this system – it is all enclosed within the body of the instrument. By applying our basic mapping model to the acoustic piano, it could be said that the user (pianist) interacts with the gestural controller (piano keys) which sends instructions to the sound generator (hammer and strings) and hence produces music. Of course the piano still relies upon the laws of physics to excite, modify and sustain sound, but from the performer's perspective these aspects are obscured by the elaborate and sophisticated interface that we call the piano keyboard. It is interesting to consider, physically speaking, that the only control afforded to the performer is the ability to determine the speed and depth with

which the key is struck and subsequently released [128] . The organ takes another step towards the digital, conceptually-speaking, as the energy used to excite the vibrations in its pipes comes from a mechanical or electric bellows, not from the performer at all, and the modern organ keyboard is effectively a set of binary switches.

This might seem like an exercise in polemics – as far as the pianist is concerned, this distinction between his/her musical gestures and the actual means of sound production is understandably arbitrary. However, the purpose of this comparison is not to critique the piano but rather to situate it as a kind of stepping-stone between acoustic/mechanical instruments and electronic/digital systems. The same feeling is shared by the digital musician: "during the process of production or performance, [the music and the means of its generation] are inseparable" [88]. A well-designed digital musical instrument is identical to a well-designed acoustic instrument in the sense that they are both "vessels for expression of human thought". The physical object that we refer to as the instrument is really just "an energy conversion device" that is employed in a musical context [78].

Perhaps it is better to think of the instrument as a concept, rather than a physical object? For example, an oil drum is clearly not designed with musical expression in mind – it is not an instrument, but can become one when it is added to a percussion ensemble. When does this change occur? Is it when it is played upon like an instrument, or placed alongside the other 'intentional' instruments, or even when the musician first considers the possibility of playing it? The perceptual structure of tasks is key to understanding this transformation in that it determines how we chose to use a particular input device [73]. We can use the concept of *affordances* here to describe the perceived properties of an object that determine how

we approach interacting with it. Research by the Ubiquitous Music Group [190] has pointed-out that the musical affordances of a system "are not properties of the environment or properties of the actors. They are relational properties that arise while activities are been carried out" [85].

In this line of thought, the instrument becomes an abstract concept: the point of intersection between tools, concepts and activities [85]. This perspective liberates the digital musician from steering their interaction metaphors towards the restrictive ideal of Western classical virtuosity – an ideal that has led to the predominant view that tightly-coupled interactive systems, analogous to acoustic instruments, provide the best support for creative musical expression. The Ubiquitous Music group has also suggested that this *instrumental paradigm* [86] might not be the best approach. Systems where agent and object are tightly-coupled can prevent the natural emergence of affordances and the implicit interdependence of modules in tightly-coupled systems also serves to reduce code reusability – making testing, maintaining and modifying the individual components difficult and time-consuming. Loose-coupling has been proposed as a more-suitable solution – by sharing the performance demands between agent and object, we allow users to "explore the epistemic space of relationships among material objects and creative outcomes…[i.e.] loose coupling fosters natural affordance formation".

Armed with this new perspective on what exactly we are doing when creating a digital musical instrument, we can make further enquiries into the most effective design strategies than can be adopted.

3.4 Mapping

In a digital musical instrument, mapping describes the manner in which data gathered by the input device(s) is related to the musical parameters of a system. The

importance of selecting or devising an appropriate mapping scheme cannot be understated – effective and elegant systems can lead to "a more holistic performance exploration of the parameter space" [69].

This is not to say that a performance system should necessarily be overly simplistic or immediately accessible. In the study of human-computer interaction (hereafter referred to as HCI), it has been suggested that the distinct practices of (a) designing for efficiency, and (b) focusing upon aesthetic considerations and the user experience cannot successfully proceed in isolation [39]. In a musical context, an expressive interface design must accommodate the capacity to practise, learn, make mistakes, and develop skill:

> Mapping is at least as important to musicians as the physical interface, and even more so over the long term. Using a different mapping strategy results in a new control paradigm to explore [25].

Literature devoted specifically to the definition of effective mapping schemes is scarce – the theoretically limitless combinations of devices and musical goals that a musician might seek to accommodate render the discussion of general mapping principles quite difficult. However, there has been growing interest in the development of more detailed conceptual frameworks for mapping – examples include strategies specific to sound synthesis [203], digital audio effects [194], Max/MSP [15], PD [174] and algorithmic composition [40].

Musical mapping schemes are generally classified according to the number of parameters over which the user can exert control at once - the most straightforward of these being *one-to-one* mapping, where a single control device influences a single parameter. This kind of precision is exactly what is needed in, for example, a mixing console. However it has been suggested that human operators naturally expect more

complex schemes and ultimately find these interactions more rewarding and intuitive than simple *one-to-one* mappings [69]. More complex setups can be said to employ *convergent* and *divergent mapping*. Convergent mapping employs a number of devices to control a single parameter (*many-to-one*) whereas devices which use divergent mapping operate several parameters at once (*one-to-many*). Most acoustic musical instruments can be thought of as combining elements of both of these schemes.

Outside of a musical context, mapping schemes for human-technology interaction are more efficiency-focused and hence easier to discuss. In *The Design of Future Things,* Donald A. Norman encourages designers to utilize what he refers to as *natural mappings* wherever possible (citing the oft-inconsistent positioning of hobs and their controls on a cooker as an example). In this context, it is preferable that controls should be laid out "in a manner spatially analogous to the layout of the devices they control" and that the principle can be extended to "numerous other domains" including sound [132]. With this consideration in mind, it is surprising how many supposedly-intuitive musical performance systems opt for the most convenient or visually-appealing layout for their controls, rather than considering the perception of the user.

In the same volume, Norman provides a summary of the essential design considerations discussed throughout his work. His rules of interaction state that interactive technology should:

1. Provide rich, complex, and natural signals
2. Be predictable
3. Provide a good conceptual model
4. Make the output understandable

5. Provide continual awareness, without annoyance
6. Exploit natural mappings to make interaction understandable and effective

It should be stressed that these considerations are clearly intended for functional applications which can be effectively used almost instantly - a description which cannot reasonably accommodate the level of skilled practice and gradual progress that we associate with learning a musical instrument. However, they do provide a model of simplicity and efficiency which can be useful to bear in mind while working on more complex multimedia environments.

Another interesting set of general design principles, in this case specific to digital musical instrument design, have been defined by Perry Cook in [29] (the explanations following each principle have been added by the author of this thesis) :

Human/Artistic Principles:

1. *Programmability is a curse*

   This refers to the danger of versatile and customisable systems – namely, their ability to facilitate endless experimentation, modification, paper writing and time wasting without ever producing any artistic or musical product.

2. *Smart instruments are often not smart*

   Instruments that are capable of learning and modifying their behaviour in response to user input are hazardous in that they react differently to conventional physical interactions and potentially lead to frustration and confusion. Instruments that constantly change prevent the user from developing and refining their own interactions accurately.

3. *Copying an instrument is dumb, leveraging expert technique is smart*

Attempting to emulate an existing instrument, while an interesting technical exercise, seldom leads to exciting or practical designs. However, devising new ways to utilise the fine motor skills of expert musicians can be a fertile starting-point for musical innovation.

4. *Some players have spare bandwidth, some do not*

   Taking a combination of individual musical ability and the physical demands of certain instruments into account, some cases are better suited to the addition of extra control devices and sensors than others.

5. *Make a piece, not an instrument or controller*

   Setting out to design a 'super instrument', with endless expressive potential, generally yields plenty of interesting research questions but not so much interesting music.

6. *Instant music, subtlety later*

   This observation refers to the (almost) universal ability of acoustic instruments to react and produce sound instantly in response to even the most basic of beginner interactions. Electronic instruments that follow this pattern, as opposed to front-loading their more complex features, are often more likely to encourage and stimulate the user.

Technological Principles:

7. *MIDI = Miracle, Industry Designed, (In)adequate*

   A simple cautionary point – while MIDI is often a quick and easy solution to get new systems communicating, it was designed with very particular commercial concerns in mind and far from a perfect protocol.

8. *Batteries, Die (a command, not an observation)*

Another simple warning against the use of unpredictable power sources unless they are absolutely necessary.

9. *Wires are not that bad (compared to wireless)*

   Designers are often eager to dispense with wires for ergonomic and aesthetic reasons. However, the added complexity, expense and potential inaccuracy of wireless systems can sometimes lead to problems. In situations where performers are mostly stationary or seated, wires are still a reasonable option.

Some Other Principles:

10. *New algorithms suggest new controllers*

    The development of new synthesis or signal processing techniques can often prompt investigation into new methods of control.

11. *New controllers suggest new algorithms*

    In a similar fashion, novel controller designs can often be the catalyst for the development of unusual or radical musical processes.

12. *Existing instruments suggest new controllers*

    A reference to the wide variety of ergonomic, musical and technical insights that can be derived from studying established musical instruments.

13. *Everyday objects suggest amusing controllers*

    We possess a remarkably broad vocabulary of techniques for interaction with objects in our daily lives. Electronic instruments that utilise or repurpose our non-musical interactions can be entertaining and stimulating.

This kind of reflective practice with regard to mapping is indicative of a community seeking to deepen the collective understanding of a neglected area, and it is maturing rapidly. The importance of developing an objective approach, one that avoids

didacticism and device-specific discussion, is outlined clearly by Hunt, Wanderley and Paradis:

> Since there will not always be ready models for inspiration when designing mapping strategies for new digital musical instruments, the task then becomes one of proposing guidelines for mapping and also, if possible, devising models that can facilitate the implementation of mapping strategies other than simple one-to-one relationships. [70]

3.5 The design cycle

Cooper and Reimann give a succinct summary of the process of interaction design in [31]:

1. Researching the domain
2. Understanding the users and their requirements
3. Defining the framework of a solution
4. Filling in the design details
5. Testing the validity of the solution with users

Note the emphasis on users and solutions, as opposed to technology and features – this approach to interaction design encourages a behaviour-oriented design approach that is strongly influenced by cognitive principles and user perception. In other words, *goal-directed design*. The process of identifying, empathizing with and facilitating user goals is the most important part of this approach and is referred-to as the "bedrock upon which interaction design is practiced."

This philosophy, and the comprehensive documentation that accompanies it, places a high priority on achieving elegant communication between the user and the system. According to the arguments presented in this chapter so far, the designer of

digital musical instruments should make every effort to develop their understanding of this process which is at the heart of every successful interactive experience.

Having established this priority, we can explore approaches to design that have been developed specifically with musical expression in mind. Miranda and Wanderley propose a 5-step design process for the creation of digital musical instruments:

1. Decide on the gestures that will be used to control the system
2. Define gesture capture strategies that will best translate these movements into electrical signals. This is typically done using a variety of sensors to measure hand, arm, lip, or other body movement, velocity of movement, pressure, or any other variables of interest.
3. Define sound synthesis algorithms that will create the sounds to be played; or, define the music software to be used for control of prerecorded musical processes.
4. Map the sensor outputs to the synthesis and music-control inputs. This mapping can be arbitrary, so any unusual combinations would be as feasible to instantiate as any coupling of gesture to sound known in acoustic instruments.
5. Decide on the feedback modalities available (apart from the sound generated by the system): visual, tactile and/or kinaesthetic. [120]

Depending on the circumstances, the available technology or musical goal at the heart of a particular project might very well provide the answer to several of these questions before the design process even begins. Therefore it is highly unlikely that these steps will proceed in a strict order, both for this reason and the fact that

adjustments will often need to be made before the desired functionality is attained [120].

The effective design process, therefore, should be conceived of as a cyclical rather than a linear process. In order to facilitate a smooth and efficient transition between designing and refining, some process of evaluation is necessary.

3.6 Evaluation

There has been relatively little research dedicated solely towards the evaluation of digital musical instruments (see Figure 1 in [13]). Traditional methods of evaluating user interfaces from the field of HCI focus upon efficiency and clarity. *Fitt's law* is a prime example of this emphasis – measuring the difficulty of movement-related tasks and the human rate of information-processing as these tasks are realised [102]. However, the evaluation of digital musical instruments must take into account concepts as diverse and far-reaching as efficiency, potential for extension, difficulty, learning curve, and so on [76] – prompting the suggestion of categories such as "reproducibility, reliability and expressive potential" [51]. Furthermore, there are a variety of perspectives, each demanding different techniques, from which we can evaluate digital performance tools [140].

To illustrate the difference between evaluating traditional interactive systems and digital musical instruments, consider the presence of the spectator or audience implicit in the musical context of the latter. It has been perspicaciously observed that we cannot simply transplant our understanding of spectatorship from the domain of acoustic musicianship to that of digitally-mediated performance [57]. Accordingly the creation of meaningful and perceivable connections between human action and sound has been identified as a key point for making a performance convincing for the audience [140]. The ability to evaluate the extent to which an

audience can understand these connections would prove a valuable asset to digital musical instrument designers.

According to Davis [38], a performance ecosystem comprises four parts: the instrument – an artefact that is manipulated to produce music; the performer – an agent who directly interacts with the instrument; the listener (referred-to here as 'the audience') – who watches the interaction and has an indirect relationship with the instrument; and the environment – the place where the performance takes place.

In traditional HCI design, there is no equivalent to the audience as defined above. Its models focus almost exclusively upon the direct user of the system. In digital musical instrument research, this has led to a predominance of performer-centred design (assisted by the instrumental paradigm, as previously discussed in 3.3) and an insufficient treatment of the audience. A synthesis of techniques is proposed in [13] to address this deficit – this is indicative of a new interest in evaluation methodologies tailored-specifically to the needs of the digital performer.

Further discussion on the issue of evaluation can be found in [120], pp95 and in [16].

3.7 Conclusion

This chapter has given a comprehensive overview of the concepts at the heart of digital musical instrument design. The basic model is explained and expanded-upon with regard to the concept of control dislocation and its effect upon the user experience. Several different approaches to classifying digital musical instruments are summarised to illustrate the broad selection of taxonomic approaches that may be taken to aid design, practice and pedagogy.

Having established a clear picture of the role played by the digital musical instrument and the different ways that researchers have tried to classify it, the

chapter progresses onto a more conceptual treatment of the subject, albeit one that has considerable practical application. The idea of the *instrument* itself is challenged in order to highlight the shortcomings of sticking too closely to established conventions of performer-instrument relationships when using computers.

With the emphasis now firmly upon the emergent perceptions of the performer, we discuss the importance of mapping and justify its position of determining the essence of the interactive experience. Finally, the chapter outlines several pertinent strategies for approaching the design process itself and introduces the promising new developments taking place concerning the evaluation of digital musical instruments.

Chapter 4. Interaction design for the digital musician

"The interface defines a sort of landscape, creating valleys into which users tend to gather, like rainwater falling on a watershed. Other areas are separated by forbidding mountain ranges, and are much less travelled. A good interface designer optimizes the operations that will be most often used."

-David Rockeby, The Construction of Experience: Interface as Content [157]

This chapter discusses an alternative approach to digital musical instrument design. By treating an input device as a selection of independent data-generating sensors, we can define a system of modular interaction components. These, in turn, can be combined in various ways in order to create effective interactive systems for musical performance.

The chapter begins with a summary of the reasons behind this approach and the intended goals of developing it (4.1). Sections 4.2 and 4.3 establish a vocabulary for discussing design models and different categories of input sensor, respectively. Section 4.4 proposes a modular approach to interaction design and summarises the core concepts in a series of tables. Section 4.5 lays the foundation for this approach by outlining a selection of simple one-to-one interaction strategies for dealing with live sensor input.

While separating these elements is useful for illustrative purposes, in reality they are seldom used in isolation. Therefore, this rest of this chapter discusses strategies for combining inputs and distinguishing various layers of functionality from one another in a digital musical instrument. It also introduces more abstract

concepts which can be used to augment the functionality of a system's actual physical controls.

Section 4.6 discusses multiple controllers which operate independently of one another – both in order to perform different musical tasks or in order to control multiple instances of similar tasks (polyphony). Next we classify different ways in which controllers can inform the behaviour of other controllers in a system – symbol, executive and modifier keys (Buxton's key-action-model), interdependent controls that only operate in combination (selection + excitation model, such as the guitar, Theremin etc.) and controllers that occasionally interact when a given condition is met (4.7). This is followed by a discussion on various ways to combine controllers that ideally may lead to synergistic roles and complementary modalities (4.8).

This section is followed by a description of 'virtual' controllers – abstract variables that can be used to alter the behaviour of a system (4.9). These are further divided into two categories – statistical variables (including those influenced by time, averages, etc.) and variables related to modal behaviour. The section on modes identifies several different types of modes and suggests strategies for accessing them fluidly within the context of a live performance (toggling modes, quasi-modal systems, advancing systems, etc.). It also discusses the importance of clearly delimiting different functionalities and the construction of unambiguous state transitions (Buxton's 3-state model of graphical input is used as an example).

The chapter concludes with a case study that discusses an interface designed for a series of performances with the Trondheim Electroacoustic Music Performance group (4.10). Particular attention is given to those aspects of the instrument that embody the ideas described so far – goals are identified and refined, the design process is discussed in detail and personal reflections upon the success of the project

are outlined. This section provides a conclusion of sorts to the first half of the thesis – subsequent chapters focus upon the development of a complex software controller for a specific type of technology (multi-touch) that is also intended to facilitate the design approach embodied by the opening chapters.

4.1 The importance of a conceptual foundation

The field of live electronic music has always been markedly innovative - for many practitioners, the design of a personalised interactive system is considered a significant component of their artistic statement which is just as important as the live performance itself. Andrew Hugill describes this important relationship:

> The types of interface to be used…how those map onto the sounds that will be produced…these are all performative decisions, equivalent in significance and musical qualities to the traditional 'tone' that an instrumentalist might produce from their instrument. In other words, the technological set-up is not just a way of making an 'instrument' upon which to perform, it is integral to the nature of the sound that is produced, to the distinctive sound that makes the digital musician into the performer, to the musicianship itself. [67]

This tendency towards idiosyncratic technique, combined with the considerable variations in digital musicians' available resources, has made generalising about design processes difficult unless specific hardware or software is involved. While studies on the use of specific devices can be useful in assessing the suitability of a particular controller to a particular musical task, it is unreasonable to expect a general model of human performance to emerge from such studies [21].

This system has been developed with a strong awareness of these factors. The goal is not to devise a linear, instructive or didactic system for digital musical

instrument design - instead it is proposed that a conceptual toolkit, independent of any particular musical style, hardware requirements, or programming languages, will prove a useful addition to the pre-existing theory on interaction design for the digital musician. The toolkit will ideally provide:

- A simple, incremental and easily-taught system of musical interaction design which is not hardware or software-dependant
- A selection of independent input paradigms which can be combined into more complex input metaphors
- Guidelines to help construct complex and flexible interfaces using simple hardware
- A way to assess the suitability of a piece of hardware for a given musical task
- A starting point in the design process which encourages incremental and methodical design
- A means of looking beyond the common usage trends and design clichés of a given piece of hardware

A concise and efficient vocabulary for interaction design, written especially with the digital musician in mind, has the potential to both expedite and enhance the development process. With a clear idea of the logical tools at his disposal, the digital luthier is well-equipped to articulate his musical ideas by breaking-down a complex interaction into its constituent processes. This, in turn, facilitates the tight matching of the device's control structure with the perceptual structure of the task as perceived by the user – arguably the best way to improve the responsiveness of an interface [107].

Note that the strategies being discussed in this thesis are primarily in terms of imperative programming – other programming paradigms may not make use of these tools in the same way. A familiarity with basic programming tools or building blocks is essential for constructing even the most basic of interactive behaviours. A selection of pedagogical references for beginning programmers can be found in [50, 142, 166, 209].

At its most basic operational level, the digital musical instrument consists of an array of interdependent interactive processes. While the overall design might behave in a very complex and nuanced fashion, the individual processes can often be quite simple in programming terms. The following section serves as a bridge between the language of the digital musician and the language of computer logic. While the approach being proposed might sound overly straightforward, dry or methodical in the context of musical projects, one must remember that "these are precisely the kinds of physical computing projects that need this kind of planning the most" [142].

4.2 Models

Models, in the design context, are simplifications of real-world scenarios. They can be especially useful for the digital musician - permitting an exploration of the validity of an instrument design concept prior to embarking upon the often costly, and time-consuming, process of implementation. In [103], models are described as existing on a continuum - with *predictive* and *descriptive* models occupying the extrema.

4.2.1 Predictive models

Predictive models represent a hypothetical analysis of how users will perform using a proposed interactive system [103]. These predictions are generated a priori and therefore circumvent the time and effort that might be required to both implement a system and perform observational testing with real users. Predictive models are commonplace in HCI where the measurement of efficiency and motor skills are concerned.

4.2.2 Descriptive models

Descriptive models are not designed to generate empirical or quantitative analyses of user performance in the same way as predictive models. Instead, they aim to equip the designer with a new conceptual framework or perspective on the user experience of a proposed interactive system [103]. This framework can take the form of a graphical representation, verbal description, or re-structuring of the system using categories, comparisons or metaphors.

Generating descriptive models for existing digital musical instruments can be a powerful tool for highlighting important issues that might otherwise be obscured by the creative and/or technical aspects at play. The role of a descriptive model in this case is to present a useful way of thinking or categorising the behaviour of an interactive system. Their simplicity, ease of use and potential for problem-solving makes them a valuable asset to the digital musician. The modular system of interaction design that follows is an example of applying this type of planning with regard to musical use. Some non-musical examples that illustrate the role of descriptive modelling in a concise manner are the Key-Action Model/KAM [103] and Buxton's 3 State Model [21].

4.3 Describing sensors

The following section establishes some terminology needed to discuss digital musical instrument design.

4.3.1 Degrees of freedom vs. dimensions

Two terms that are often used interchangeably are dimensions (D) and degrees of freedom (DOF), referring to the number of parameters in a system that are free to change independently of one another. Here we use degrees of freedom to describe the number of data streams that the user can manipulate independently using a given device.

A mouse is often described as a 2-D input device, with respect to the X and Y planes of the graphical environment that it typically navigates. This might also be hastily-described as having 2-DOF. However, as pointed out in [103], a true 2-D device actually has 3-DOF - translation across the X-axis, Y-axis and rotational motion around the Z-axis. It is true to say that a standard mouse does not permit movement in this way, unlike a 2-ball mouse or a device such as the Reactable [77] which allows rotational movement of its control cubes.

For the purposes of digital musical instrument design, it is advisable that the term 'dimensions' is avoided, due to the spatial/graphical connotations illustrated by the previous example. Instead, when referring to the number of data streams that a user can manipulate independently via a given device, it is preferable that the term *degrees of freedom* or DOF is used. A standard mouse device in this context possesses 4-DOF - that is to say, independent freedom of movement on the X and Y-axes combined with the left and right mouse buttons. If a scroll-wheel is present, the

device can be said to have 5-DOF, if the scroll-wheel happens to be clickable, 6-DOF, etc.

There are complications with this model. Firstly, ergonomic and physiological concerns must be accounted for (see, for example, the introduction to kinesiology in Chapter 2 of [159]). While a simultaneous rotating and clicking of the scroll-wheel is certainly technologically possible, there is no doubt that one must influence the other in terms of the comfort and accuracy with which such a combination can be performed by the user. Any statement of a device's DOF, therefore, should be accompanied with a caveat as to which channels of control might reasonably interfere with one another in practice. It should also be noted that sensor combinations that are difficult to operate simultaneously can be advantageous - by using data from these sensors to control aspects of the musical output that should not function together, user error and accidental triggering of certain functions can be reduced significantly.

A further complication is introduced by our means of interpreting the input data. It is perfectly possible, for example, to take a simple X/Y-axis reading from a mouse and derive a further speed value using simple mathematics. This speed value can be further broken-down into horizontal and vertical speed, and so on. These kinds of augmentations to a device are very useful indeed and are covered in detail below. However, it cannot strictly be said that they are integral to the sensor itself. Furthermore, depending on how the various channels of control are utilised, these values may not be entirely independent from others - making it difficult to apply our definition as outlined at the start of this section.

It must be concluded that, while the term degrees of freedom is indeed a useful tool when designing interaction strategies, it is not a fixed value when

anything other than a simple one-to-one mapping is concerned. One must be careful to distinguish between the DOF property of a device in a strictly hardware-related sense (out-of-the-box) and the DOF property that is arrived-at when mapping strategies and musical output have been established. A variety of such strategies for obtaining useful performance information are outlined below (section 4.5). It is important, however, to remember that an interface is not-necessarily improved by the addition of more DOF – rather it is how closely the control structure of the interface matches the perceptual structure of the user approaching the task [73].

4.3.2 Resolution

A further property that will be utilised in discussing design strategies for digital musical instruments is resolution. It is used here to indicate the smallest change that can be detected in the input of a given sensor and, as such, can be used as an indicator of the potential accuracy of the sensor. For example, a 2-button mouse being used to navigate a graphical environment has 4 sensors - two of which have a high-resolution (the X and Y axes) and two of which have a low-resolution (the buttons, which can be said to have a binary resolution, on/off or simply 0-1).

Examples of typical musical devices would be a 4x4 button matrix (16 DOF / 0-1 RES) and simple MIDI mixer with 4 faders and 4 dials (8 DOF / 0-127 RES). These properties become useful when selecting what interaction strategies to use as well as deciding upon appropriate mappings to musical parameters. The exact resolution of a sensor is somewhat trivial for the purposes of this discussion - what is important is the ability to distinguish between those sensors which simply behave as switches and those which allow for a greater degree of expression.

4.4 A modular approach

In the following sections we will look at micro-strategies for interpreting the data generated by individual controller devices. While designing highly-minimalistic interfaces is a useful exercise for digital musicians of all levels of experience, in practice we will generally be dealing with hardware that comprises a wide variety of input devices in various combinations. The major advantage of a modular design approach is the ability to quickly experiment with different ways of using the devices available in a given piece of hardware. This in turn facilities rapid prototyping, compared to more tightly-coupled systems, which allows for more time testing and refining the design with the performer's experience in mind. This reflective space is a vital commodity in digital musical instrument design; our ability to add new features "is constrained by the musician's physical and psychological capacities of accomplishing multiple and simultaneous tasks" [96]. Rapid prototyping affords the designer more opportunities to assess the performer's capacity for expression using the current interface.

The following sections discuss the modular building blocks of musical interface design. Perry Cook's caveat that digital musical instrument design "proceeds as more art than science" [29] is particularly pertinent in this chapter – these strategies and rules are useful when followed but even better, in some cases, when they are broken correctly.

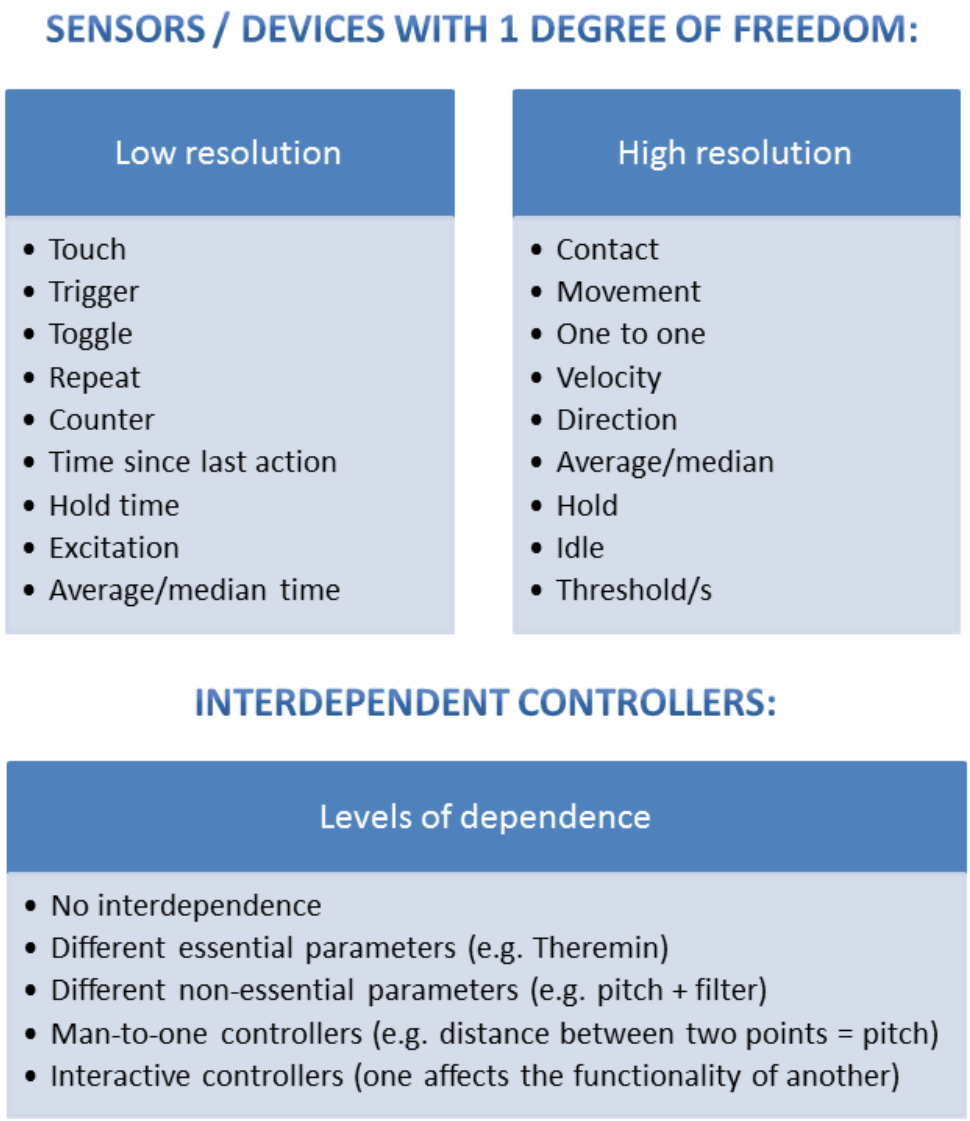

Figure 4.1: Summary of strategies for single devices and combined controllers

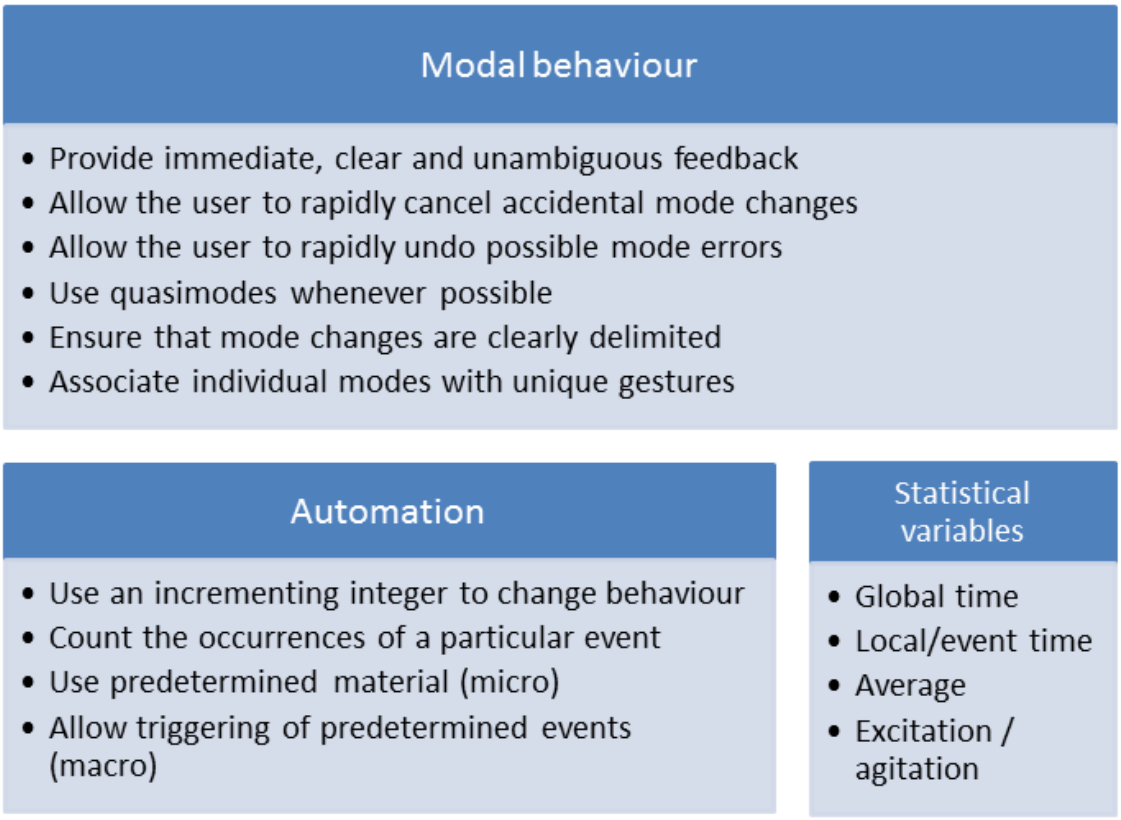

Figure 4.2: Summary of abstract control strategies

4.5 Interaction strategies

This section defines a series of original strategies for interpreting the input of various sensors. These relatively simple strategies are grouped according to the criteria established above and form the foundations of a novel and concise framework of interaction design for the digital musician. The concepts and logic behind the strategies are generic, enabling them to be applied to a variety of different sensor types.

Several of the strategies may seem redundant, obvious or a waste of potential data from a given sensor (the Contact strategy from section 4.5.2, for example). In these cases, it is important to consider that the strength of this approach can often lie in the ability to combine the strategies with one another to generate complex results.

The explicit definition and demonstration of even simple strategies makes for a comprehensive inventory of design components with which to construct elegant interfaces.

4.5.1 One DOF sensors with low resolution (on/off)

Examples include QWERTY keyboard keys, non-pressure sensitive buttons and pads, piano-style keyboards without touch sensitivity, mouse buttons, many videogame buttons, switches, etc. Note that sensors with a higher-resolution can be used in conjunction with these strategies by employing a threshold, or similar technique (see 4.5.2.9). The first three strategies that follow can be thought of as basic one-to-one mappings.

4.5.1.1 Touch

The sensor causes something to happen while it is depressed, but ceases to have an effect once contact is broken. Typical uses include sample-playback, toggling temporary effects, non-progressive sustain pedal on keyboard, non-velocity sensitive synth playing, etc.

4.5.1.2 Trigger

Description: The sensor causes something to happen when it is pressed, typically an event that resolves of its own accord, breaking contact makes no difference. Typical uses include playing drum-like or percussive samples.

4.5.1.3 Toggle

Pressing the sensor once causes a change in how the system works, pressing it a second time returns the system to its initial state. Typical uses include switching on/off effects.

4.5.1.4 Repeat

Holding down the sensor causes an event to repeatedly occur at a certain interval until contact is broken. 'Stuttering' effects of sample playback are often achieved in this way.

4.5.1.5 Counter

Pressing the sensor generates a different result each time, incrementally cycling through a preset array of results.

4.5.1.6 Time since last action

Pressing the sensor generates a different result, depending on the time that has elapsed since its last action.

4.5.1.7 Hold time

The amount of time the sensor is activated is recorded and subsequently used to trigger different behaviour.

4.5.1.8 Excitation

While the sensor is activated, a value increases at a preset rate. While idle, the value decreases at a preset rate.

4.5.1.9 Average/median time

May apply to any of the time-based strategies. An array of recent values from the sensor is maintained, with either the average or median value of the array being used to influence behaviour.

4.5.2 One DOF sensors with high resolution

Examples include dials/potentiometers, faders, touch sensitive piano keys, pads or buttons, ribbon controllers, modulation wheels, etc. While these sensors all fall under this category, it is important to take the physical means of operation into account as there are some notable differences.

For some of these sensors, hereafter referred to as *fixed-state* sensors, the value being output remains the same after the user has manipulated it - for example, dials/potentiometers, faders, certain modulation wheels (that don't spring back into place). These sensors also serve to provide feedback on their current state, which can be noted at a glance or, in some cases, kinaesthetically without actually disturbing the sensor.

In other cases, the extra resolution is due to a velocity or pressure component, which may only be delivered upon impact (pressure/velocity sensitive keyboard keys and pads/buttons). These could be said to be analogous to the 'ballistic' playing style of acoustic percussion or piano in the sense that, once the initial impact has taken place, the velocity component has been determined and cannot be changed. The exceptions are cases wherein it is possible to continually-measure the pressure being applied to a sensor, for example, certain ribbon controllers, pads, spring-loaded modulation wheels, etc. Another important characteristic of these sensors to bear in mind is the fact that the output will always start and end at zero, unless some kind of alternative or auxiliary control is implemented.

A final idiosyncrasy to consider is the ability of some of these sensors to allow discontinuous output, or teleporting of values. This is best explained by comparing a fader and a ribbon sensor, both being used to control the volume of a track. In order to bring the volume from the minimum possible level up to the

maximum using the fader it is necessarily to progress (however rapidly) through the full range of values in between. The ribbon, conversely, allows the user to make jumps in the signal by simply breaking contact with the sensor and depressing their finger elsewhere. Whether or not this kind of behaviour is a help or a hindrance, or even acknowledged, depends on the application, but it is certainly important to be aware of.

4.5.2.1 Contact

A behaviour is triggered when the user makes contact with the sensor (not possible with fixed-state sensors, for which the next strategy is a close alternative)

4.5.2.2 Movement

A behaviour is triggered when the user changes the value of the sensor

4.5.2.3 One-to-one

The value of the sensor is tied to the value of a musical parameter

4.5.2.4 Velocity

The rate of change in the sensor is tied to a musical parameter

4.5.2.5 Direction

The direction of movement (incrementing/decrementing) is used to influence a parameter

4.5.2.6 Average/median

An array of recent values from the sensor is maintained, with either the average or median value of the array being used to influence behaviour. The size of the sampling window must be adjusted, according to the speed of changes in the value,

in order to provide the most accurate reading. The average/median velocity or direction can also be calculated.

4.5.2.7 Hold

A behaviour is triggered when the sensor remains at a specified value for a predetermined period of time

4.5.2.8 Idle

A behaviour is triggered when the sensor remains untouched for a predetermined period of time

4.5.2.9 Threshold

A value, or number of values, is designated as a crossing-point. When the sensor passes a point, a behaviour is triggered. Alternatively, thresholds may be used to assign different functions to several areas over the total range of the sensor. One or more of these areas may be 'dead', where nothing happens or a previous effect is negated. Analogous to splitting up a visual control surface into 'zones'. Many of the strategies outlined above can be applied once areas are split up in this fashion.

4.6 Independent controllers

In this context, the term ‘independent controllers’ refers to the use of more than one input device simultaneously in an interface but without the data interacting in any significant way. The devices remain separate both physically, in terms of the hardware itself, and computationally.

The most common approach in this category can be referred to as polyphony – where a selection of similar input devices allow the user to control multiple instances of similar events. Dissecting the ubiquitous digital keyboard

provides us with a clear example of this model: each key provides access to a single musical note and allows the performer to independently actuate, sustain and terminate individual instances of notes across the range of the keyboard. Difficult combinations, temporally and spatially-speaking, can be accommodated through practice and appropriate fingering technique.

We can extend this understanding of the digital keyboard without any modification to the performance sampler or drum machine. Any piece of hardware that is designed primarily with live triggering of samples in mind will feature an array of buttons or pads that each provide the user with access to a particular sound. In both cases, external modifiers are available: additional velocity sensors for each key/button/pad are a standard feature in middle to high-range equipment, the piano keyboard is typically augmented by pedals and the sampler will generally provide a means of switching between 'banks' of different preprogrammed sounds. These features will be discussed in 4.7 and 4.8.

The concept of integrality and separability as two classes of perceptual structure are useful in this context. Primary input devices found on a piece of hardware are seldom 1DOF, as in the example of velocity above. When separate attributes of a single device are used to control more than one parameter, we can characterise the device as multidimensional. When a number of attributes combine perceptually, they become *integral*; attributes that remain distinct are *separable* [73]. In the example of the velocity-sensitive piano keyboard, the act of individual note-selection and volume/timbre-selection can be classified as an *integral* action as the movement "is in Euclidian space and cuts across all dimensions of control" [73].

It is important to consider this perceptual structure of individual input devices when approaching the design of an interactive system. Some devices with

more than 1DOF are better characterised as integral as opposed to separable. These are better-suited to controlling aspects of the musical output that are perceptually similar or closely related in terms of their effect on the sound. One device which is typically used to control two variables independently with a single gesture is the XY pad. Some observations upon typical mapping schemes are provided in 5.1.2.

4.7 Interdependent controllers

Devices that are used together, without influencing one another, are covered by the previous definition of independent controllers. Interdependent controllers are different in that they can be said to inform one another's decisions upon how to classify a given input action by the user. There are varying degrees to which one controller can influence the behaviour of another but the defining characteristic here is the necessity for the devices in question to be operated together in order to achieve their full functionality within the system.

The prevalence of software user-interface design conventions within the field of music technology tends to discourage complex interdependent and multi-functional interfaces (see Chapter 5 for a specific commentary upon this). However, it has been proposed that these are precisely the kinds of interfaces that generate interesting and rewarding interactive experiences [69, 39]. When combining controllers we are aiming, ideally, to define synergistic roles and discern complementary modalities [60]. Some of our most powerful tools when working-towards this ideal are descriptive models.

The key-action model (KAM) proposed by MacKenzie in [103] is a descriptive model that illustrates an everyday example of interdependent controllers. KAM sorts the keys on a standard QWERTY keyboard into three distinct categories: symbol keys (deliver graphic symbols such as alphanumeric characters, punctuation

marks etc. to the system), executive keys (perform meta or system-level tasks such as the function keys, ENTER or ESC) and modifier keys (SHIFT, ALT, CTRL, etc.). Modifier keys establish a condition that alters the effect of a subsequent key press but do not immediately or directly invoke behaviours or deliver symbols in the same way as the other two categories. This can be categorised as the most separate method of using interdependent controllers – the modifier keys change the functionality of the symbol keys entirely (e.g. holding SHIFT capitalises simultaneously character entry) but the symbol keys can be operated without using the modifier keys. The modifying controllers augment the functionality of the basic controllers, but they are not required for simple tasks.

We can identify a further variation on this idea without leaving the QWERTY keyboard – keys such as CAPS LOCK and INSERT can be described as a hybrid between the modifier and executive key categories. Both affect the behaviour of subsequent key-presses while also toggling an application-level change of functionality. This can also be categorised as modal behaviour (covered in detail in 4.9.2).

Further along the continuum of interdependency we encounter controllers that only function correctly when used simultaneously. Acoustic instruments that can be classified in this way are typically designed for bimanual operation where each hand performs a different task (separate selection and articulation of notes). Most chordophones are designed with this kind of interaction in mind. The same could be said of many wind instruments, where note selection (keys/holes) and articulation (mouthpiece) are two interaction modalities that are, in normal circumstances, entirely dependant on one another. In both cases there are certain musical results that omit one of the channels (e.g. fretboard-tapping and open notes on guitar, key-noises

and open notes in wind instruments) but these are the results of physical, rather than musical, design conventions. It is interesting to note that the Theremin closely adheres-to this paradigm of dual-channel control – with each hand allocated separate control of the instruments pitch and volume – while remaining a difficult instrument that requires great physical discipline to master [26].

Somewhere between these two extremes we encounter occasionally-interdependent controllers – control devices or techniques that become co-dependant when a certain condition is met. Examples of this kind of behaviour can be found in Akustich [5] (when the user's hands cross over to trigger a distortion effect) and Subcycle Labs [178] (where touch points moving above/below one another switch the kind of effect being applied). When applied intelligently, with clear delimiting considerations and feedback to indicate the newly-activated interdependence of the controllers, this can be an elegant technique to nest a variety of behaviours within a system without introducing new hardware or confusing layers of functionality.

4.8 Strategies for combining controllers

Having clearly distinguished between independence and interdependence between devices in digital musical instruments, we can identify a number of distinct approaches towards combining their functionality. Device/hardware-specificity and ergonomic considerations weigh heavily on these decisions and vary massively from case to case – therefore the following categories assume that the designer is proposing a combination that is physically possibly, both from the perspective of the technology being used and the reasonable ability of the intended performer.

These definitions are not to suggest a strict categorisation of approaches towards combining controllers – a difficult and redundant task, given the infinite variety of devices and applications. However they do allow us to look a little more

closely at the relationships that exist between different parts of a digital musical instrument's input components and assess their role in creating a satisfying and robust channel of communication between man and machine.

4.8.1 No interdependence

Both controllers affect different, unrelated parameters of the instrument.

Included for completion – no interdependence is implied in this case.

4.8.2 Different essential parameters

Both controllers affect different, but related, essential parameters of the instrument.

As seen in the Theremin example – the parameters of pitch and volume are related to the same sonic event and are perceptually integral. Both parameters are also necessary for the basic operation of the instrument.

4.8.3 Different non-essential parameters

Both controllers affect different, but related, non-essential parameters of the instrument.

In this case, the parameters might be pitch and some kind of timbre-shaping property – both parameters are perceptually-integral but one or more can be deemed optional/non-essential in terms of the system's priorities.

4.8.4 Many-to-one controllers

Both controllers affect the same parameter in the instrument.

Data that describes the relationship between the behaviour of both controllers can be used – for example, the cumulative velocity of a pair of trackballs or the distance between two touch points. Further levels of abstraction can also be introduced to

invoke different types of behaviour – for example, which of the two controllers was activated first, which is moving faster, positioned higher/lower etc.

4.8.5 Interactive controllers

One controller alters the functionality of another.

This is closest to the acoustic model, typical of chordophones, that is described in section 4.7. The piano sustain pedal can also be placed in this category. The behaviour of one device acts as a modifier – for example, a fader selects a position within a stored audio loop and a button triggers playback from that position. Depending on the context of use, and the level of influence being exerted upon the system as a whole, it may be helpful to categorise this strategy as modal or quasimodal behaviour (see section 4.9.2).

4.9 Abstract controllers

Alongside the control opportunities that are afforded by any physical hardware, we also have access to a variety of abstract controllers. These are programming techniques that are distinct from those summarised in 4.5 in that they are not designed explicitly to interpret or modify data that is generated by the user interacting with the hardware, although they may often be employed in that way. These virtual controllers provide the designer with additional tools to contextualise and delimit the behaviour of a digital musical instrument without requiring additional hardware or sensors. It must be pointed out, however, that the use of any abstract controllers should be clearly signposted to the user – either via some kind of feedback mechanism or through prior explanation – in order to avoid confusion.

4.9.1 Statistical variables

Statistical variables are global values that exist separately to individual sensor readings, although they may be derived from or influenced by them, and usually have a temporal component. They can be used to imbue a digital musical instrument with a sense of movement and activity by fluctuating parameters in response to, or independent from, user input. Some examples include:

1. *Global time*

   Values related to the time since the system or performance began can be useful in cases where the duration of the performance, and certain changes associated with its progress, are known. A timer can be used to ascertain when a new section of the performance should begin and used to automate some of the processes required (see also 4.9.3)

2. *Local/event time*

   Smaller timers that are started in response to individual events or actions can be extremely useful for delimiting certain behaviours. The most commonly-used example is a tap-and-hold style gesture on a touch screen which is often used to invoke alternative behaviour. Figure 3 in [60] shows a wide variety of touch gestures that are differentiated from regular interactions using a simple time-based hold cue. This kind of cue (holding a posture/button/etc. for a predetermined period) is very difficult to perform by accident and represents a powerful way to move between states. However, it should be avoided in the case of rhythmic or time-critical events that might be rendered inaccurate or flimsy as a result of the implied delay. Aside from providing a convenient means to construct delimiting functions, the values from local timers can also be useful when

used directly for synthesis and signal processing. For example, a sound that is initiated by a button press can be made to increase in volume using the value of a timer, with the button release signifying a note release at the final volume that is reached. This kind of non-obvious interpretation of user input can be used to give a sense of dynamics to even the most basic of hardware inputs.

3. *Average*

   Aside from its use as a tool for smoothing noisy input data, averaging can be used to generate interesting values for synthesiser control. A fader which uses an average value, rather than its current value, to control the pitch of an oscillator, for example, will 'drift' smoothly from the previously-held value to its destination. This can be used to implement a portamento-style effect and also, at slower speeds, to free up the performer to concentrate on other tasks – the delayed reaction that this technique produces can be used like an instant form of automation programming, where the performer selects a value that the controller will move gradually towards and proceeds to concentrate on other aspects of the performance.

4. *Excitation/agitation*

   This is a metaphorical implementation of unsettling a physical system. A threshold is set for a particular sensor input – for example, the velocity of a mouse being moved – along with a simple conditional loop. When the value being output from the device exceeds the threshold, an additional 'excitation/agitation' value is incremented. Conversely, when the sensor input drops below the threshold the value decreases. This value is used

elsewhere to control volume, distortion ratios, effects sends, etc. Experimenting with different threshold positions and the rate of addition/subtraction to/from the excitation variable can lend a sense of life to properties of a synthesis or signal processing algorithm that might otherwise be static or linear-sounding. A simple example is described in 5.3 of [117].

4.9.2 Modal behaviour

We have already encountered the concept of modes when discussing the keyboard action model in 4.6 – the modifier keys (SHIFT, CAPS LOCK, INSERT, etc.) all alter the interaction scheme and allow the same interface, the QWERTY keyboard buttons, to be used for entirely-different purposes. The concept of a modal interface is described as follows in *The Humane Interface*:

> A human-machine interface is modal with respect to a given gesture when (1) the current state of the interface is not the user's locus of attention and (2) the interface will execute one among several different responses to the gesture depending on the system's current state [149].

Thus we can describe a digital musical instrument as modal if it comprises multiple states or modes that each exhibit a distinct set of behaviours and rules for the interpretation of user input.

There are mixed opinions as to the inclusion of modes in physical computer interfaces due to their capacity to cause confusion and 'mode errors' (i.e. any kind of unpredictable or unwanted response to user input as a result of a system being in a different mode than the user assumes). Many designers advocate the avoidance of modal systems altogether [149] and cite a preference for mapping each control "to a unique and consistent response" [142]. However, modes are very useful

for the digital musical instrument designer: a well-designed modal interface allows the performer to employ the same physical gestures and devices for multiple purposes both quickly and unambiguously. Modes can be used to reduce the number of gestures that a user needs to learn and also dispense with the need for extra devices or sensors.

The primary concern in such a system is the avoidance of mode errors – there are several pertinent strategies that can be employed:

1. *Provide immediate, clear and unambiguous feedback*

   Signifying that an alternative mode has been triggered is the most direct and simple way to avoid mode errors. The system can alert the user immediately once a new mode is engaged, provide some kind of consistent ambient indication while it is engaged, or some combination of both. Visual indicators (e.g. lights, screens, colour-changes, etc.) are usually convenient, provided that they do not disrupt the performance, as they can be ignored once the user becomes proficient. Sonic cues may also prove useful in certain cases where the change of mode has a drastic effect on the sound anyway (e.g. entering a mode that applies a granular distortion effect on the sound) but can be disruptive or fatiguing in many scenarios. If it is available, haptic feedback can provide subtle reinforcement cues in response to user input without alerting the audience.

2. *Allow the user to rapidly cancel accidental mode changes*

   Providing some kind of dedicated ‘escape’ button can assist the user in rapidly correcting a false switching of modes and returning the system to its previous, or default, state. One example in a popular application is the

use of the ESC key to exit note editing mode in the Sibelius family of digital notation software [167]. Both this technique and the one that follows differ slightly from the rest of the strategies in this section in that they aim to retrospectively amend errors related to modal behaviour. Aside from the practical benefits of rapid error-correction, this kind of feature can help users learn to navigate through complex performance systems by reducing the damage, and hence frustration, caused by mistakes.

3. *Allow the user to rapidly undo possible mode errors*

   In cases where mode errors can potentially have a devastating result on the performance (e.g. deleting an entire sequence of notes or altering a live-looping setup) it can be necessary to provide an emergency button analogous to the undo function featured in most desktop applications. This should be immediately accessible and difficult to trigger by accident. For more advanced applications, the ability to save and recall various states can be a practical extension of this idea (4.9.4).

4. *Use quasimodes whenever possible*

   Quasimodes, also known as "spring-loaded modes" [149], invoke changes of behaviour in a system in exactly the same sense as a mode but they require a conscious and sustained input cue from the user in order to remain active. Once more, we can refer to the key-action model and the concept of modifier keys (4.6) for an everyday example: the altered functionality modes of the SHIFT, CTRL and ALT keys are seldom activated by mistake because they require a constant physical effort on behalf of the user and cease to have an effect once the key is released.

The sustain pedal on a piano provides us with another good example – despite completely altering the behaviour of the instrument, it is seldom triggered in error due to the decisive physical effort required. The kinaesthetic aspect of maintaining a quasimode serves as a form of natural feedback, which further reduces the capacity for errors, but also necessitates a comfortable and ergonomic design in order to avoid strain or injury.

5. *Ensure that mode changes are clearly delimited*

   The user actions that result in a transition between states or modes should be clearly defined and distinct (both from one another and any other kinds of action that use the same input channel or device). While quasimodes, as described above, provide a relatively safe way to accomplish this, they are impractical for invoking modes that are sustained for long periods of time due to cognitive load on the user, the compromised physical faculties of the user and the potential for fatigue. Careful attention should be given to this issue with regard to the choice of hardware, number of different modes and the frequency/speed with which they need to be changed. A robust example of a comparison between several input devices and their states can be found in Buxton [21].

6. *Associate individual modes with unique gestures*

   Reserving specific gestures (e.g. unique button combinations) for toggling modal behaviours is a worthwhile option to consider for digital musical instruments that require a variety of operating states. This presents a more abstract or symbolic approach that requires the user to commit a set of executive gestures and the modes associated with them to

memory. While this approach implies an investment of time and an adjustment period on the users' behalf, there are considerable benefits in terms of user familiarity and the potential for layering many different levels of functionality without confusing. The ability to rapidly and unambiguously switch between a variety of layered performance modes in, for example, hardware samplers and drum machines has enabled dedicated users to develop an extraordinary level of precision and efficiency while generating and editing patterns and samples in real-time. The primary difficulty with interfaces designed in this style is that many of the features are obscured from the user due to the level of abstraction that is involved. Care must be taken, therefore, to ensure that the user has quick and easy access to the most salient features of the system when it comes to live performance. A contemporary commercial example is Yamaha's Tenori-On [183] which features ten hardware buttons on either side of the main interface that provide access to a variety of functions – changing tempo, altering note lengths and octaves, transposing, etc. Holding the R1 button and selecting a horizontal row from the main grid interface changes between one of sixteen different 'layers' – each of which are pre-assigned to use one of the Tenori-On's six distinct 'performance modes', which range from a step sequencer (Score Mode) to more generative behaviours (Random Mode, Bounce Mode). While initially quite complex and overwhelming, each layer (and hence behaviour) can be associated with a simple two-button combination that allows habitual users to navigate between them with speed and accuracy.

4.9.3 Automation

The concept of using time as a variable was introduced in 4.9.1 – the extent to which the use of system time as a cue can be classified as automation depends largely on the level of complexity that is involved. Without necessarily adopting this approach, and hence moving towards the design of a predetermined interactive score, several more subtle forms of automation can be used to reduce cognitive load and therefore permit the user to focus upon more critical aspects of the performance:

1. *Use an incrementing integer to change behaviour*

   This is an extremely simple yet powerful technique. One or more global variables store an integer that the user can increment/decrement at will. These variables are used elsewhere in the code to alter aspects of the system – examples could be to transpose a section, alter the scale that is being used, change the sample bank assigned to a certain device, move on to the next part of a looped sequence, etc. The most powerful aspect of this approach is the ability to invoke a large number of changes in response to a simple user action. It can also be used to simultaneously change operating mode, as discussed above, and musical material in a system where the general progression of events is known in advance of a live performance.

2. *Count the occurrences of a particular event*

   This strategy is a variation of the previous concept that uses the same technique of incrementing abstract counters. The difference is that these counters are tied to a particular event, such as a note/sample trigger, rather than being manipulated by the user directly. Aspects of a system can, therefore, be set to evolve in direct response to the performer's

actions without requiring specific attention, manipulation or devices to control. These evolutions can take the form of anything from subtle drifting of sample-playback positions and reverb parameters, to more drastic effects such as changing the note value or volume of a particular key every time it is struck.

3. *Use predetermined material (micro)*

   The use of pre-prepared samples, sequences, loops and patterns in live performance is taken for granted in most forms of electronic music. However it is worth declaring this strategy explicitly in order to highlight that the 'predetermined material' in question need not be audio or note event data. All sample-based performance systems use predetermined amplitude envelopes on a micro-scale to ensure smooth playback of samples, but the ability to define and trigger more macro-level parameter control envelopes is featured less frequently. Commonly-used musical techniques such as fade-ins/outs, crossfades, sustaining of notes, scrubbing through samples, and suchlike can also be automated and set to begin in response to a specific input device or gesture. Once more, it must be stated that this kind of functionality is not intended to make performing easier or less-human, but rather to free the faculties of the user in order to concentrate more fully on other aspects of the music.

4. *Allow the user to trigger predetermined events (macro)*

   This technique does not refer to the use of smaller, composite parts such as those described in the previous strategy. The approach in question refers to the preparation of key transitions, musical gestures and transformations that are likely to form part of the overall performance at

some stage – a loose analogy can be made with motivic jazz improvisation [87]. Unlike the use of a timer or linear score, the user is provided with the means to trigger certain automated processes at will throughout the performance. This method allows complex hooks and progressions to be preserved and produced at will during the performance without restricting the user to a preset timing, duration or score. Obviously the boundary between this strategy and the previous is loosely-defined and depends largely on the structure of the music and the role of the performer.

4.9.4 Saving and recalling settings

The ability to save and recall preset sounds, arrangements and parameter settings is typically reserved for the preparatory stages of developing a live performance – patches and presets are often loaded up during a show, but seldom edited and saved again. Interfaces for memory access on both hardware and software instruments generally reflect this trend, with detailed multi-level menus and file system navigation being the norm.

The only scenarios where performers typically generate, store and recall material onstage tend to be where live looping/sampling or sequencing is taking place. In such cases it is common for dedicated hardware/software to provide a set of quick-access banks, patches or presets that can be altered, saved and recalled rapidly during performance. With respect to dynamic control of a system, there is ample reason to explore the ability to save/recall the current state of an instrument, its parameters, and other abstract variables such as those discussed in this section. Analogous to a ‘screengrab’, ‘snapshot’, ‘bookmark’ or ‘quick-save’ in gaming and other media, digital musical instruments that allow the user to dynamically store and

retrieve information during a performance can facilitate a sense of freedom and complexity with regard to developing musical material live.

4.10 Case study: LoopBlender

Figure 4.3: T-EMP ensemble performance at Rockheim, Trondheim

This section describes the creation of a digital musical instrument using the strategies defined in the preceding chapters. *LoopBlender* was used in a series of improvised performances with the Trondheim Electroacoustic Music Performance ensemble (T-EMP) in August 2012.

4.10.1 Background

The T-EMP ensemble (shown in Figure 4.3) explores some of the peculiarities of digital musicianship through live performance and group improvisation [152]. The author was invited to play as a guest musician for a small tour in August 2012 with two days of rehearsal and two consecutive concerts – each consisting of a 50-minute performance comprising three sets. The intention was to bring two separate sources of sound – acoustic and electronic – in order to accommodate a broad range of

improvisatory material. The acoustic source was an unmodified shakuhachi flute, but it was decided that an entirely new interface should be designed for the manipulation of electronic material.

For both of the concerts in question, the ensemble consisted of: Øyvind Brandtsegg (Hadron partikkel synthesizer), Trond Engum (guitar & electronics), Bernt Isak Wærstad (guitar & electronics), Tone Åse (voice & electronics), Ingrid Lode (voice & electronics), Carl Haakon Waadeland (drums), Bryan Quigley (acoustic bass) and Patrick McGlynn (shakuhachi & electronics). The large size of the ensemble, improvised concert format, predominance of electronic instruments and unfamiliarity of the author with the group's performance style meant that the development of a robust, versatile and adaptable digital musical instrument was key to a successful integration with the group.

4.10.2 Design brief

Preparation for the sessions was guided entirely by three points of interest that had been conveyed by the T-EMP ensemble:

1. *Emphasis upon a non-visual performance style*

   Visual communication between members of the ensemble was not encouraged. While visual cues (both predetermined and spontaneous) are frequently used to communicate in improvised settings [136], the intention was to build a group rapport solely based-upon audio stimulus.

2. *No monitors for individual performers*

   In a live concert setting, particularly where amplified and electronic instruments are used, it is common practice to provide a monitoring system to allow the performers to clearly hear their own contribution alongside the rest of the parts. In order to maintain a sense of focus upon

the overall textures and gestures taking place within the group, there were to be no individual monitors or mixes for the musicians. The only form of feedback would be a stereo mix delivered to the stage which would be identical to the front-of-house mix heard by the audience. Musicians were thus expected to be self-regulating with regard to their overall dynamic placement in the mix.

3. *Performers need to be able to respond quickly*

This was the most striking of the guidelines provided – the ability to rapidly exchange more percussive or rhythmic gestures with the rest of the group was made a priority when designing the performance interface.

4.10.3 Hardware selection

The ideal controller for this scenario would be lightweight, compact, and feature an equal mix of continuous and discrete control devices. Physical interfaces and traditional mechanical input devices such as potentiometers and buttons were given preference over digital systems due to the necessity to accommodate rapid and dynamic responses to fellow musicians. The Korg NanoKontrol2 was identified as the optimum available controller – a slimline USB mixing desk [94].

4.10.4 Interface components

The interface was designed to perform sample-based synthesis using a set of preloaded loops. The user can individually control the volume of both a dry and reverb-effected signal from four separate sample banks – each of which contains six loops that can be muted/played independently. The loop start point and loop length of each bank can also be altered dynamically by the user.

In addition to this main sound-generating architecture, a selection of articulatory tools are implemented as toggle-able master effects – including a killswitch, selection of filters (LP, HP, BP), bit depth/resolution reducer and overdrive effect. Finally, the unexpected nature of both the individual performances and the dynamic of the ensemble itself made the integration of a vast and varied sample library a necessary addition – a separate and independent 'layer' of banks was added, with an alternate selection of samples loaded into each slot, in order to increase the sound material available to the performer.

4.10.4.1 Bimanual interface for selection and articulation

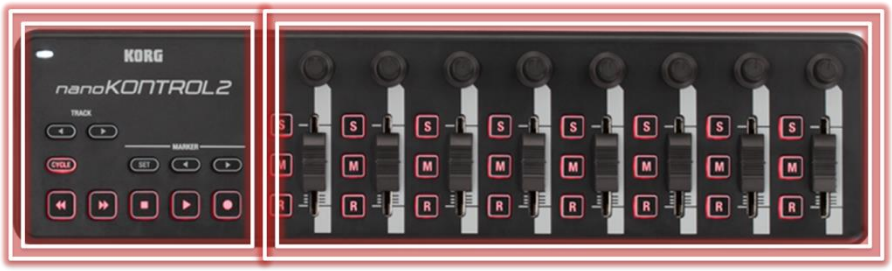

Figure 4.4: Bimanual division of Korg NanoKontrol2

Partly inspired by the layout of the control surface itself, and partly due to the intrinsic seperability of the tasks taking place (sample selection/manipulation and master effects triggering), a bimanual model of interaction was adopted. This involved an abstract division of the physical control surface into two halves: selection and articulation. Selection tasks (sample triggering, looping and scrubbing) are performed using the faders, potentiometers and buttons of the right side of the interface, and articulation tasks (master effects and killswitch) are performed on the left side.

This model of allocating distinct yet complementary tasks to the left and right hands was inspired by the research of Bill Buxton into framing and articulatory [98] roles in two-handed input and also Guiard's kinematic chain model [213]. With regard to Guiard's model, this instrument design represents a hybrid between the orthogonal and serial categories of bimanual interaction – both hands perform different tasks (orthogonal approach) but the output of the right-hand sample banks also provides the input for the left-hand effects processors (serial approach).

4.10.4.2 Modal sample toggles with LED flags

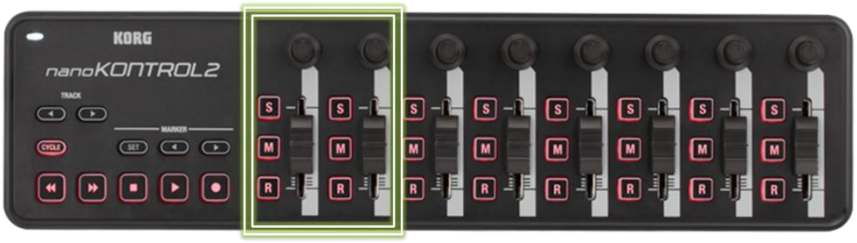

Figure 4.5: Control section for sample group A

A wide variety of sounds from the author's personal collection were auditioned, edited and categorised in preparation for the performances – including found sounds, field recordings, synthesized material and excerpts from compositional work-in-progress. To facilitate rapid memorising of sample locations and enable fluid access onstage, four abstract categories were defined based upon the sonic qualities of the samples:

- Group A: *Low-frequency / low-energy sounds*

  (e.g. rumbling, dense, slow-moving textures)

- Group B: *Low-frequency / high-energy sounds*

(e.g. active, dark, rhythmic, alive textures)

- Group C: *High-frequency / low-energy sounds*

  (e.g. subtle, high-pitched, gradual, airy textures)

- Group D: *High-frequency / high-energy sounds*

  (e.g. busy, dynamic, shrill, cutting textures)

Each separate category or bank is allocated a section with two faders, two potentiometers and six buttons. In addition, toggling the alternate bank mode gives access to a second layer of samples, all of which are organised using the same system. Each group therefore contains 12 samples giving a total of 48 unique samples to draw-upon. Each button on the NanoKontrol2 features an LED light that is used here as a flag to indicate that a given sample is active. The samples range from short (<5 second) clips to longer (c. 2 minutes) montages that can be scrubbed-through and looped in different ways to generate new rhythms and textures live during performance.

4.10.4.3 Touch versus toggle behaviour

Figure 4.6: Sample buttons for group A

At any given time, each bank features 6 samples that are assigned to individual buttons. The buttons behave as switches and use the toggle behaviour described in

4.5.1.3. Once activated, each button lights-up and loops a sample continually until pressed again. An initial prototype design used the touch strategy (4.5.1.1) but this was deemed impractical due to the difficult and uncomfortable hand positions that certain combinations of sample required. Access to more sudden, rhythmic and percussive gestures was delegated to the articulation section (4.10.4.6 below).

4.10.4.4 Integrality and multiple outputs of banks

Figure 4.7: Master volume and reverb send for group A

The pair of faders in each group are dedicated towards controlling a dry and reverb-effected mix of the currently looping samples within that group. In both cases, a simple one-to-one mapping scheme is used. There are no individual volume controls for the samples themselves – each group is used as a sound collage generator. This approach was chosen in order to emphasise the integrality (see 4.7) of each bank in terms of the timbral similarity of the material (4.10.4.2). The result is a system which emphasises the cumulative product of each bank and thus mirrors the concept of the self-regulating performer defined in the brief (point 2 in 4.10.2).

4.10.4.5 Combined looping controllers

Figure 4.8: Loop start and length controls for group A

There are 2 variables related to looping behaviour that are unique to each group – loop start point and loop length. These variables are controlled by pairs of potentiometers that are linked using the second strategy described in section 4.8: *Both controllers affect different, but related, essential parameters of the instrument.* Each value is controlled via one of the potentiometers and is dynamically-scaled in order to prevent read-errors (i.e. loop start time and length are both expressed as a percentage of each individual loops size). Integrating the controls for multiple loops within the same group in this way prevents micro-management of sample playback and encourages the performer to explore the sample library in search of interesting emergent patterns.

4.10.4.6 Touch strategy for articulation

Figure 4.9: Articulation controls

As discussed in 4.10.4.1 above, the articulation section was designed explicitly to assign the left-hand to a more percussive and rhythmic role. The right-hand selection section tends to provide an analytical and measured way to blend the various samples together, so the left-hand section needed to inject a sense of immediacy and spontaneity into the instrument.

The killswitch (labelled [A] in Figure 4.9) was included as a direct result of the toggle behaviour used for the sample buttons (4.10.4.3). The need for two consecutive button presses in order to quickly start and stop a sample led to a poor response time when rapid bursts of silence and sound were needed. Also, the extra effort required on behalf of the performer was fatiguing and inelegant. Therefore a dedicated percussive button was introduced – the killswitch, when depressed, mutes all outgoing audio from the instrument. When released, playback immediately continues (the button uses a touch strategy, as described in 4.5.1.1). This muted state can be described as a quasimode, as it requires the user to physically maintain contact with the button, and is impossible to invoke by accident. During performance, the killswitch is typically operated using the thumb while the right-hand alters the configuration of the material being looped.

The layer switch (labelled [B] in Figure 4.9) uses a simple toggle strategy to invoke the only constant mode change within the instrument design – switching between the two banks of samples. Any sample-slots that are currently playing are replaced with their counterparts. While this prevents material from the first and second layers being used simultaneously, it is preferable to having loops running in the background that are not represented by a lit button on the device itself. Every loop being played is represented by a lit button.

Finally, the effects section (labelled [C] in Figure 4.9) comprises 5 larger buttons in a row, each of which activates a different effect on the master output channel. From left to right these effects are: low-pass filter, high-pass filter, band-pass filter (fixed to a mid-range), bit depth/resolution resampler and overdrive. Like the killswitch, these effects are all quasimodes that use the touch strategy – the effect remains active while the button is depressed. The effects are not mutually-exclusive and can be triggered in various combinations for interesting and distinctive results (e.g. LP filter + resampling effect generally generates brittle, low textures). An important role of the effects is to allow the performer to instantly modify sampled material to fit loosely within a spectral or timbral space that is being established by the other performers during an improvisation. The spatial configuration of these buttons lends itself well to this role – the fingers of the left-hand can rest comfortably upon the buttons and operate them fluidly, after some practice, without the need for the performer to glance at the interface.

4.10.5 Discussion

Analysis of this case study can be undertaken with regard to two separate issues: (a) the effectiveness of applying the interaction design strategies developed throughout this thesis, and (b) the success of the *LoopBlender* interface itself in practice.

Every sensor on the NanoKontrol2 was reappropriated in some way to suit the requirements of the performance. Once the intended functionality of the interface was outlined, the interaction strategies allowed for rapid prototyping using the available potentiometers, faders and buttons. For example, individual samples were initially activated using the touch strategy (4.5.1.1) to facilitate rapid, percussive play, but this was quickly deemed impractical due to both the physical layout of the buttons and the tendency for the user's hands to obscure other vital controls (faders and potentiometers) while maintaining contact. A quick survey of the available strategies revealed toggle (4.5.1.3) to be a viable alternative. The touch behaviour was subsequently assigned to the left-hand, or articulation, section in order to provide a comparable amount of percussive or rhythmic control.

An awareness of modes and strategies for combining controllers (4.9.2 and 4.8, respectively) led to the simultaneous and independent activation of the various master effects. Although the effects were perceptually very different, their similar method of activation and close physical location coupled them together into a single control modality that encouraged a particular style of play – the juxtaposition of various combinations of effects became associated with particular fingering patterns.

While the strategies proved useful and easy to apply, the instrument design itself has a number of shortcomings when it comes to performance. Navigating the sample library is heavily-reliant upon the user's memory. This had the dual disadvantage of lowering reaction time and discouraging intrepid explorations through the samples due to the significant risk of triggering unwanted sounds. Also, the effects section was allocated half the physical space of the sample section and none of the continuous controllers (faders and potentiometers), yet it swiftly became apparent that its features were far more practical in an improvisatory context. A

direct reversal of priorities, providing more emphasis on the articulatory controls and effects, would possibly lead to a more versatile instrument.

However, there were also many interesting benefits to the design. Most performance-oriented samplers do not provide the user with a quick, non-destructive means to navigate-through and modify loop points – *LoopBlender* facilitates rapid and precise modification of samples during playback without permitting the user to enter a distracted or analytical state. The experience of performing with this instrument in an improvisatory context, once the sample locations themselves have been memorised sufficiently, is similar to playing with a collection of found objects and physical sound sources. *LoopBlender* afforded the author the ability to partake in a series of long-form improvised performances, using exclusively pre-recorded material, without becoming repetitious or requiring a visual display.

It is interesting to revisit the design brief subsequent to implementing and performing with *LoopBlender* and note how the apparent-limitations of the goals in fact led to some liberating performance concepts:

1. *Emphasis upon a non-visual performance style*

   The lack of visual communication between the improvising musicians meant that the instrument itself could feature a visually-complex interface without compromising the performer.

2. *No monitors for individual performers*

   The ability of the performer to identify and modify their material unambiguously during performance was vital. Therefore, the clear organisation of samples and the ability to rapidly respond to changing dynamics (using both the killswitch and the faders) became vital features.

3. *Performers need to be able to respond quickly*

The sudden changes and reactions that the musical style demanded led to a system where play modes could be quickly switched in an intuitive and error-free way. Once memorised, the simple toggle controls in the articulation section become an unambiguous and versatile expressive tool.

4.10.6 Future work

There is scope for both development and improvement with this concept in the future. Performance oriented samplers generally place a high priority, in terms of the layout of the control surface, upon turning on and off individual samples – features dedicated towards the editing of samples are usually restricted to a set of hierarchical menus. The close physical relationship between continuous controllers (potentiometers and faders) and discrete controllers (buttons) upon the NanoKontrol2 makes the hardware well-suited to a performance approach that places equal importance upon triggering and scrubbing through looped material. However, the controllers in question are amongst the most common found on music hardware – distributing these roles across a number of dedicated fader, potentiometer and button control surfaces might lead to a more evenly-distributed instrument design in terms of its functionality.

Therefore, expanding the hardware setup is a planned development for the future. The controller itself is lightweight, compact and ideal for mounting onto additional pieces of equipment. An additional button array or touchscreen device would expand *LoopBlender*'s capacity for supporting complex sample triggering behaviour or real-time manipulation of effects, respectively. Integration with the

Oscar system, as described in Chapter 6, would also greatly enhance the number of control modalities open to the performer.

In response to these observations, a number of guideline questions can be generated in order to help musicians approach a similar interface design project in the future:

1. *Does the performer need to simultaneously operate a large number of samples?*

   If so, an interface that accommodates a large number of 1D controllers (i.e. buttons) should be used, ideally without allocating multiple samples to the same controllers. Consider an alternative, ergonomic layout that might allow smooth playing (see 2.2 for some suggestions).

2. *Does the performer need to pay close attention to visual stimulus?*

   If so, the interface should avoid using screens or other forms of input that rely heavily upon visual communication. Visual feedback should be simple, unambiguous and available at a glance (e.g. LEDs, fader positions, etc.).

3. *To what extent can the performer edit the loop contents live?*

   If the performer just needs to trigger the loops at a set volume, a simple button will suffice. Additional mixer-style controls allow for some variation (e.g. of volume, panning, etc.) but more precise real-time editing, such as loop boundaries and playback speed, demand a further set of responsive, dedicated controllers.

4. *Is it important to be able to manipulate effects parameters?*

   For more dynamic live sample processing, consider adding an additional control surface dedicated towards effects and parameter control.

Frequently used effects, such as the killswitch in *loopblender*, should be assigned to a comfortable and accurate input device.

4.11 Conclusion

This chapter has proposed a modular approach for the construction of interactive strategies in digital musical instrument design. Having established the key benefits of a modular system and highlighted the goals this approach aims towards, a selection of essential programming concepts are discussed in relation to musical application development. A concise comparison of predictive models and descriptive models reaffirms the direction of this approach. The next section takes a critical look at some easily-misused terminology – degrees of freedom, dimensions and resolution – and clarifies the distinction between them for the purposes of discussing hardware devices when used for musical interaction.

Following this groundwork, a list of interactive strategies is proposed for both high and low-resolution controllers with one degree of freedom. These strategies, while quite simple, represent a new way of seeing a piece of hardware: as a selection of flexible devices that are open to interpretation individually and as a group. This perspective highlights the expressive potential of even the most basic devices and actively discourages the kind of simple one-to-one mapping techniques that were criticised in Chapter 3.These strategies form the fundamental building-blocks of a new conceptual toolkit which is expanded-upon in the following sections.

Having outlined the benefits of a modular approach to digital musical instrument design, we looked at the distinction between controllers operating independently and interdependently, from the performer's perspective, and thus identified a number of different strategies for combining controllers.

Another important category of tools – abstract controllers – was then introduced to complete our model. A summary of interaction techniques that employ statistics, modal behaviour, automation and the saving/recalling of settings were discussed in terms of their ability to augment the hardware components of digital musical instruments. The complete model is illustrated, in brief, by the tables provided in Figures 4.1 and 4.2. Finally, we took a detailed look at the design of *LoopBlender* – a sample-based performance instrument designed for use in an improvisatory context. The utility of the terminology we have established throughout this chapter is demonstrated via this case study, which employs many of the interaction strategies that have already been discussed.

Chapter 5. Recontextualising the multi-touch surface

"Through eons of human evolution, we have developed sophisticated skills for sensing and manipulating our physical environment. However, most of them are not used when interacting with the digital world where interaction is largely confined to graphical user interfaces."

-Hiroshi Ishii, The tangible user interface and its evolution [72]

This chapter discusses design issues for digital musical instruments which utilize multi-touch technology. The focus is firmly upon experimental and/or innovative instrument designs which engage with the users' sense of tacit knowledge [132] and facilitate spontaneity and improvisation. There are four main sections:

- Surface-based Interfaces (5.1) describes in detail the data generated by two popular types of controller – the XY pad and button array – and how it influences their use in digital musical instrument design. The multi-touch interface is then discussed in the same context and a summary of notable uses is provided.
- Designing Multi-touch Interfaces (5.2) discusses the often-restrictive use of graphic user interfaces (GUIs) in multi-touch systems and suggests an alternative approach with an emphasis on gestural, as opposed to visual, interaction.
- SurfacePlayer (5.3) describes the development of a multi-touch interface paradigm designed with non-graphical performance techniques in mind. This tool moves beyond the simple use of coordinate data to the development of

multi-touch interaction algorithms using a standard tangible interface protocol. This work became the foundation of a new interface design, featured in Chapter 6.

- The conclusion (5.4) summarises the main points made within the chapter and describes the link between these findings and the proof-of-concept described in the next chapter.

5.1 Surface-based interfaces

This section consists of a review of various surface-based interfaces when used as musical controllers. The surfaces in question are simple XY pads, button arrays (also known as 'grids') and multi-touch surfaces. The grouping of these devices under the heading 'surface-based interfaces' is not to suggest some kind of abstract category, but rather to emphasize their shared physical characteristics – all are basically flat sensor devices which respond to human finger-touches, albeit in different ways.

5.1.1 Historical roots

There is a rich history of analog synthesizers designed to respond to touch – the *Ondes Martenot* (see 2.3.1), *Trautonium* (2.3.3), and *Theremin Cello* (2.3.4) all used precise finger movements as their primary means of control and laid the foundation for more contemporary devices such as the ribbon controller – a popular addition to performance setups since Robert Moog's modular synthesizers (2.1.8) [26]. Pen-based interfaces such as *UPIC* (conceived by Xenakis and implemented by Centre d'Etudes de Mathématique et Automatique Musicales (CEMAMu) in Paris) also inspired computer musicians to begin working with tablets. The "quantitative merits" of the tablet as a musical controller have been well-established, practically as well as

theoretically, by research carried out at the Centre for New Music and Audio Technologies at University of California, Berkeley (CNMAT) [224, 225].

5.1.2 XY pads

The XY pad is a control surface which offers 2 degrees of freedom via its horizontal and vertical axes (note that the strategies outlined in Chapter 4 can augment this number – in this case, we are just considering the basic physical properties of the interface). Resolutions vary, but are typically high enough to accommodate continuous parameter control. The XY pad can be seen as combining the functionality of two faders into a single interface, as it offers simultaneous and independent control of two streams of data (although this comparison highlights some interesting differences, as discussed below).

The *Korg Kaoss Pad* (1999) [91] range brought mainstream attention to the use of XY pads for a variety of musical tasks with a selection of high-profile users from genres as diverse as experimental rock (Radiohead's Johnny Greenwood [221]), dance (Jon Hopkins [214]), ambient electronic (Brian Eno [215]), beatboxing (Beardyman [216]), and alternative rock (Muse's Matt Bellamy [220]). The *KP* [93] range use the surface to control various live signal processing patches while the spin-off *Kaossilator* (2007) series are designed for pattern recording/playback using a selection of onboard synthesis patches. The manual for the *Kaoscillator Pro* (2010) [92] gives a comprehensive list of the mapping schemes employed and is indicative of the typical function of these devices within a performance setup.

The continuous nature of the output means that this kind of device lends itself well to glissandi and sweeping effects. Typical mapping schemes establish a one-to-one connection between each axes and a pair of parameters – cutoff/resonance of a filter, for example, or pitch/ loudness of a synthesizer. Some interesting observations

upon the combinations of parameters are discussed in [168]. It has been suggested that any two parameters mapped in this way (i.e. controlled by a single point of contact from the user) have a high degree of integration [73] and should ideally influence closely-related elements of the sound. Of the examples given above, the cutoff/resonance combination is preferable as it deals exclusively with the behaviour of the filter and allows users to associate a particular space on the surface with a certain type of sound or effect. Pitch/loudness are not so closely-coupled, as they deal with perceptually-separate aspects of the sound, and it has been observed that users may find this kind of mapping less intuitive [168].

As mentioned above, it is worth noting a number of differences not made explicit in the pair of faders analogy. While the potential for simultaneous and independent manipulation of a pair of data streams is theoretically identical in both cases, there are three major differences between an XY pad and two faders:

1. The ability to jump from one value to another while skipping the intermittent values ('teleportation') – it is possible for a user of an XY pad to break contact with the surface and reconnect at a higher/lower position.
2. Lack of feedback – an XY pad does not provide any feedback (unless it is combined with a visual display) – a fader provides both tactile and visual feedback indicative of its current state.
3. One-touch input - an XY pad can be manipulated with a single fingertip, whereas certain manipulations with the faders are difficult without the use of multiple fingers or hands.

XY pads are typically allocated an ancillary role in a performance system – playing a similar role to pitch-bend/modulation wheels or controlling effects – while primary tasks such as note selection or event triggering are left to devices such as keyboards

or samplers. While they are often used to control the continuous parameters of various effects, it should be noted that XY pads do have a certain resolution. This may not be audible and depends mostly on the hardware and communications protocol being used.

5.1.3 Grid-based interfaces

A style of interface that has seen comparatively more musical experimentation is the grid-based layout popularized by devices such as the *Tenori-On* [182], *Monome* [124] and *Novation Launchpad* [2]. While generally represented by an array of separate buttons, the device is essentially a discretized version of the XY pad – replacing a high-resolution 2 degrees-of-freedom controller with a matrix of low-resolution (binary) 1-degree-of-freedom controllers. The grid-interface can therefore be described as an array of switches.

Given the relative lack of precision that this description seems to imply, one could be forgiven for assuming that the usage scenarios are comparatively less-musical and flexible compared to those of the continuous XY pad. However, the opposite is true – grid-based interfaces have been employed in a vast array of musical tasks including sample-triggering [219], multi-effects processing [217], FM synthesis [125], step sequencer-control [177], visualization [196] and animation [197].

There are a number of reasons why this is the case. Firstly, the physical nature of an array of buttons provides a kind of tactile feedback which an XY pad cannot replicate. The importance of a tactile relationship between performer and instrument is well-acknowledged [139]. With an array of buttons, it is possible to discern the location of your fingers without relying upon visual feedback or actually triggering a reaction from the device. Secondly, many button-array controllers

(including those listed above) light up individual buttons in order to indicate their individual status or to form a collective abstract shape. This capacity for unambiguous, immediate visual feedback is significant, as it allows the user to maintain a relationship with any number of abstract variables or multiple layers of functionality once the corresponding symbolism has been established and committed to memory. Accordingly, this added channel of communication with the user encourages more complex multimodal systems. Finally, it should be mentioned that the visual appeal of the lights themselves can be a motivation for employing these devices in a live context, even as works of art in themselves [198].

Together these factors give an impression of the increased potential of the button array as part of a robust live performance system. What appear to be trivial additions (buttons and lights) are actually partly-responsible for the variety of creative digital musical instrument designs that employ button arrays.

5.1.4 Multi-touch surfaces

This section discusses approaches to musical performance using multi-touch surfaces within three categories – covering hardware, academic and mobile application development, respectively.

5.1.4.1 Commercial hardware

Commercial hardware for multi-touch music performance began with the *JazzMutant Lemur* [74] - a high-resolution touchscreen with a flexible and powerful interface editor. The Lemur arguably set the standard for multi-touch music control – the direct influence of its approach, from the futuristic visual style to its use of Open Sound Control (or OSC, see 2.9.3 or [141]), can be seen across a broad range of projects today.

While the Lemur was a generalized controller, recent trends in multi-touch music interfaces tend to be designed with more specific tasks in mind such as mixing (*Line 6 Stagescape* [100], *KS-1974* [169], *Mackie DL1608* [104]), synthesizer performance (*Haaken Continuum* [58], *Soundplane* [105] and *Misa Kitara Era* [121]) and portable composition (*KDJ-One* [82]). One notable exception is *QuNeo* from Keith McMillen Instruments [83] - a multi-touch pad controller that first appeared on the crowd funding site Kickstarter [84].

5.1.4.2 Academic research

Academic research into multi-touch music performance is widespread and diverse. Projects such as the *Reactable* [77], *Linnstrument* [156] and David Wessel's *SLABS* [207] provide interesting and progressive examples of contemporary work. One particularly useful online presence is maintained by the Natural User Interface Group – both their forum [134] and free book *Multi-Touch Technologies* are invaluable sources of up-to-date information and advice [181].

5.1.4.3 Mobile applications

Mobile applications are understandably a popular way to package and distribute multi-touch music software. There is a vast selection of musical 'toys' available on both the iOS App Store and the Google Play store which demonstrate an extremely-limited range of possibilities and are accordingly of little interest to musicians. There have been a number of attempts at 'serious' instruments – most of which are designed to resemble an existing piece of hardware (*Yamaha TNR-i* [182], *Korg iElectribe* [90]), though exceptions do exist (*TC-11* [180], *Mugician* [155]).

Some of the more flexible musical tools available on mobile devices are dedicated 'controller' applications. These perform tasks only at the input stage of the

digital musical instrument architecture and produce no sound. Instead, the users' interactions with onscreen widgets prompt the device to send data wirelessly to a computer via protocols such as MIDI (2.9.1), OSC (2.9.3) and TUIO (2.6.9) [79]. The host computer can then use this data to control synthesis or signal processing.

While a number of applications are specifically designed to complement existing hardware or software (*DL1608 Master Fader* [104], *V-Control Pro* [193], *Omni TR* [138]) the majority of controller applications allow the user to customize the layout of the screen in some respect – for example, to accommodate alternative keyboard layouts (*Musix* [114], *ExpressionPad* [48]). Most applications consist of a widget-based GUI - in this case the screen forms a canvas which can be populated by a selection of pre-designed faders, buttons, dials and touchpads (*Control* [27], *mrmr* [126], *TouchOSC* [186], *Lemur* [99]). This approach to musical performance using multi-touch technology is by far the most popular due to its relative ease-of-use and familiar visual associations.

5.2 Designing multi-touch interfaces

5.2.1 Rethinking the GUI

As outlined above, the most popular way to design multi-touch user-interfaces is via a toolkit of widgets that provide typical GUI-like elements such as windows and menus. For musical interfaces, these toolkits usually contain a selection of hardware-inspired widgets such as faders, dials, drum pads, etc. While multi-touch interfaces often resemble typical GUIs, there are vastly different design issues that need to be considered. These issues are well-established and have been under investigation for many years (see [159] for a comprehensive introduction and the work of Bill Buxton

[23] for more detailed analysis). We must be cautious not to blindly apply design strategies that are ill-suited to the medium of multi-touch itself.

The explanation can be illustrated with a comparison to music controllers in general. A well-established criticism of MIDI interfaces has been their over-reliance upon the piano-keyboard metaphor, which by its nature cannot accommodate many of the features unique to synthetic sound (freedom from discrete pitch-structures, continuous control over timbre, etc.). There are many practical reasons, however, why the keyboard interface dominates – the most prevalent being that it allows pianists to leverage their existing musical skills and explore new sounds by interacting with a wide range of hardware/software [120]. It is for this reason also that basing a new controller or synthesizer around the keyboard interface represents less of a financial risk to manufacturers, causing some speculation as to the developmental distortions that can arise when commercial interests influence the evolution of musical interfaces [120].

For the same reason, it makes perfect sense for designers of new digital musical instruments to adhere to familiar GUI/WIMP (Windows, Icons, Menus, Pointers) paradigms. These design clichés allow us to exploit several decades-worth of embedded cultural and technological knowledge in our interfaces and there are abundant resources which enable us to do so. However, in much the same way as the piano keyboard was not designed to accommodate continuous pitch changes or gradual manipulation of timbre, the GUI was not designed with multi-touch input or live music performance in mind.

The GUI paradigm has been optimized for use with a keyboard and mouse combination – it is therefore misguided to adopt this style of interaction on multi-touch surfaces without any modification [69]. There are arguably some benefits to

using a multi-touch GUI in performance – the inability of a mouse to manipulate more than one onscreen object simultaneously is a limitation that the multi-touch surface does indeed surmount. However, there is a vast array of negative repercussions – for example, over-reliance upon visual feedback, tendency for users' hands to obscure the screen (and hence, the only source of feedback) and the rigorous precision demanded by most multi-touch GUIs make them a less-than-ideal solution for live musical performance.

Widget-based GUIs by their very nature encourage one-to-one mapping and tight-coupling at the procedural stage of digital musical instrument design – both restrictive approaches that lead to systems bound by 'the instrumental paradigm' [69, 86]. This kind of design approach imposes a cognitive load on the user which can impair their level of engagement with the performance, especially when other musicians are involved. It has been acknowledged that the emergence of social affordances during music-making can be seriously compromised by tightly-coupled digital musical instruments [85].

This is not to suggest that robust and innovative GUI-based digital musical instruments cannot be designed for multi-touch surfaces. Rather it is being proposed that we should investigate, with equal vigour, the possibility of creating new interaction paradigms that best exploit the unique properties of the multi-touch surface as a performance interface.

5.2.2 Beyond the GUI

Interactions with multi-touch surfaces generate extremely rich data. A cursory glance at the capabilities of any multi-touch device which uses, for example, the TUIO protocol allows us to infer the following:

- The location of individual fingers at any given point in time
- Whether or not the surface is being touched
- The total number of fingers in contact with the surface
- The distance and angle between any of these points
- The location, area, perimeter and shape of a space defined by these points
- Whether or not a point is static or moving
- The speed at which a point is moving
- The direction in which a point is moving
- The length of time a point has been present on the surface
- The previous movements and average position of a given point…etc.

This list serves to illustrate the problem with widget-based music software on a multi-touch platform. Such environments solely employ the first point above, the location of individual fingers at any given point in time, to interact with various onscreen widgets such as buttons, faders, etc. The other types of data outlined above, while they might appear abstract or trivial, can in fact be combined in a wide variety of ways to create rich metaphors and gestural cues. It is plain to see how, in terms of designing software for a role as potentially nuanced as musical performance, the dominant GUI-based approach fails to utilize the available data in an intelligent manner.

There are many resources which can help digital musical instrument designers to access this data – Reactivision [151], TUIO [187], CCV [133] and the NUI Group all provide a variety of tools for accessing raw touch data and generating higher-level information such as speed of travel, point history, etc. A number of interesting projects have sought to utilize this data for musical performance and fittingly treat the multi-touch surface as a complex and sensitive tool rather than just

a novelty controller. Kevin Schlei's *MDrumSynth* and *MStretchSynth* both rely heavily upon relationship-based analysis for multiple parameter control [161] and his iPad app *TC-11* presents a customisable synthesiser engine that is especially designed to respond to multi-point performance [180]. Balz Rittmeyer's *Akustisch* recognizes and responds to a selection of expressive gestures using an elegant interpreter [5]. Christian Bannister's *Subcycle Labs* cleverly analyses the number of touches present on the surface to toggle various DSP effects [178]. However, the vast majority of applications fail to make use of this data in any meaningful way.

One possible reason is the volatility of geometrically-derived data. Some of the examples mentioned use algorithms that calculate, for example, the angle to the previous point or the distance to the first touch. There is a danger in mapping this kind of data to any kind of prominent synthesis parameter as it is highly-dependent upon the order of touch initialization upon the surface – two perceptually-identical gestures can quite easily result in the establishment of totally different point-relationships.

Another reason is the difficulty of implementing high-level 'gestural' response systems. Anyone intending to design a gesture-based multi-touch digital musical instrument must have, at the very least, a competent grasp of the hardware and protocol being used, coordinate geometry and intermediate programming concepts such as event handling, control flow and multi-threading. This overhead is a significant deterrent to any musician, composer or performer who wants to explore multi-touch interaction. There are many solutions which offer high-level gesture support, but none specifically-designed for musicians.

Figure 5.1 is a purely illustrative graph which places some popular approaches to multi-touch music control on a two-dimensional continuum. The

different systems are situated according to the programming expertise required (vertical axis) and how closed-off they are (horizontal). Naturally, these systems all function very well in certain contexts – the purpose of this diagram is to suggest how these approaches relate to one another and also to establish a point at which there may be a deficit of resources.

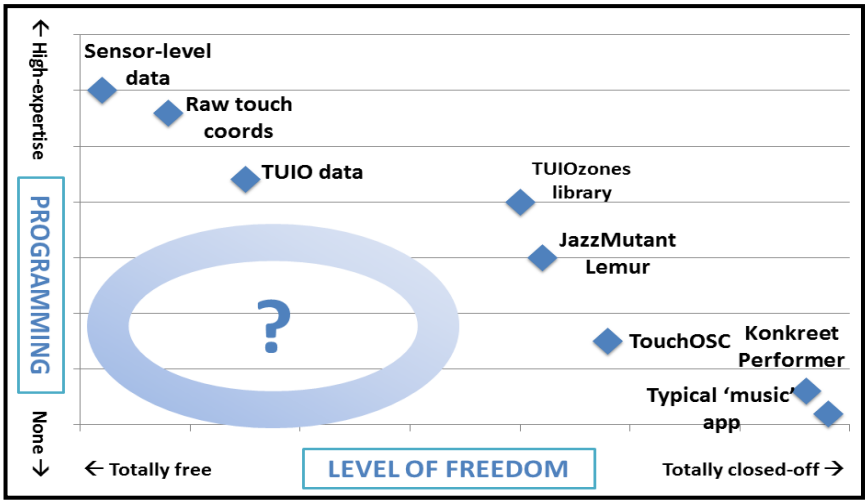

Figure 5.1: Comparison of development options for multi-touch musical apps

We can hypothesise that the area in the lower left of the diagram is an ideal space to aim for when developing tools for digital musical instrument design. An approach that could be placed within this area would allow more freedom to experiment, with less specialist requirements and prescriptive boundaries influencing the design process.

The ability to engage in reflective practice is indispensable to the digital musician [67] – therefore, a fluid transition from evaluation to implementation (and indeed all stages of the digital musical instrument design cycle) is vital [13, 140]. Tools which allow rapid and transparent development ensure that the designer can concentrate upon the critical aspects of mapping and user experience.

5.3 SurfacePlayer

This section describes SurfacePlayer – a project which was developed in order to explore the space identified above. It is designed with non-graphical interaction techniques in mind – treating the multi-touch surface as a sensitive data-gathering device rather than a canvas for widget-based interactions. This was the first step towards developing research tools which will enable future studies into multi-touch interface design for music performance and subsequently inspired the creation of a standalone app, *Oscar*, which is described in-depth in Chapter 6.

5.3.1 Aims and objectives

One of the main reasons for the relative scarcity of experimental interfaces, such as those mentioned above, is the amount of work required to analyse the data generated by the multi-touch surface. The requisite knowledge of basic networking, control flow, geometry and human-computer interaction serves to form a significant barrier for even the most experienced users. While there are plenty of libraries and applications available to obtain raw touch data, there is a lack of support for high-level data which may prove to be more perceptually-relevant in a live performance context.

The objective of SurfacePlayer was to develop a modular set of tools to facilitate the construction of expressive touch-based performance interfaces. A set of high-level interpretive tools, devised specifically with musical interaction in mind, could allow designers to concentrate their attention on more musically-critical aspects of the interface, such as mapping, and encourage more experimentation with multi-touch music performance.

5.3.2 Dependencies

The algorithms for SurfacePlayer were developed within *Processing* – an open-source creative coding platform launched by Casey Reas and Benjamin Fry in 2001 [146]. The language is based upon Java but features a simplified syntax and emphasis upon graphics to help non-programmers learn to code. Processing is especially popular amongst graphic designers, musicians and visual artists.

The Tangible User Interface Objects protocol, or TUIO, was initially developed as part of the *Reactable* project at Universitat Pompeu Fabra [77]. The `TuioObject` class handles data for tangible interface objects (such as the coloured blocks of the *Reactable*) and the `TuioCursor` class is used to represent user touches directly upon the surface itself. The SurfacePlayer algorithms were designed to derive high-level gestural cues from `TuioCursor` data sent from an external interface (in this case, a tablet device).

5.3.3 Implementation

The project comprises a selection of algorithms which generate high-level information in response to multi-touch data. This information can be quickly accessed via concise function calls, thus allowing the user to circumvent a considerable amount of programming.

Prior to this work, designers using Processing were restricted to the use of raw data which describes the coordinates, speed and path history of a point, for example. Hard-coding even simple gestures using this raw data can be a time-consuming and tedious process. The SurfacePlayer algorithms assist in this process via a selection of functions that represent common multi-touch gestures – such as taps, flicks, etc. These are set to receive TUIO data and check for certain conditions.

When these conditions are met, a gesture is recognized and relevant data related to that gesture can be used within the performance patch.

For example, in order to infer the direction of movement for a given touch, it has previously been necessary to undertake a cumbersome analysis of the path history and the average angle between points (or, alternatively, devise an algorithm which infers the direction based upon the relative speeds of X and Y-axis movement). Similarly, an action as ubiquitous as a 'multiple-tap' (where taps made using more than one finger are differentiated) requires an analysis of touch coordinates, birth/death time and the use of multithreading in order to be of any practical use. The complexity of these processes is likely to discourage the widespread use of the often useful information which they can generate.

In response to this issue, the SurfacePlayer functions allow access to this kind of information using succinct and easily-readable commands such as `movementDirection()` and `multiTap()`. This made it possible to experiment with different combinations and sequences of cues which were previously difficult and time-consuming to implement.

The functions are all defined separately in the code, allowing for the possibility of user-defined algorithms, and are compatible with existing TUIO implementations for Processing.

5.3.4 Example of use

This section describes how SurfacePlayer was integrated into the architecture of a prototype multi-touch music performance system.

A typical use of SurfacePlayer may be broken into three distinct components – the input layer, interpretation layer, and output layer. These layers are illustrated in Figure 5.2 and described in the following sections.

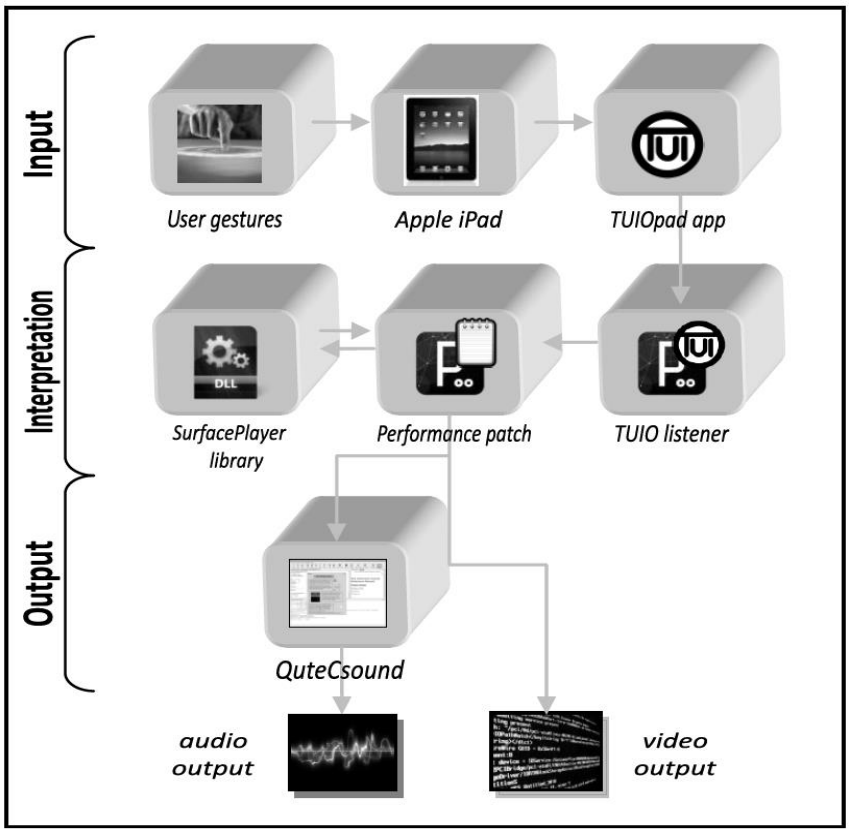

Figure 5.2: SurfacePlayer in use

5.3.4.1 Input layer

This layer consists of any device, or number of devices, capable of generating TUIO data in response to user gestures. In the example above, an iPad running the open-source application TuioPad [188] sends multi-touch data to a computer via a wireless network. TuioDroid[189], available on Android devices, is also open-source and free to download.

The TUIO protocol was chosen due to its flexibility and active user community. It also renders the system hardware-independent – allowing the algorithms implemented within SurfacePlayer to be used with any device capable of outputting TUIO-formatted cursor data.

5.3.4.2 Interpretation layer

The composite elements of this layer are implemented within the Processing development environment. The Processing TUIO Client API [80] listens for incoming TUIO events and generates data related to touch positions, such as time tags and coordinate paths. This data is subsequently interpreted by the SurfacePlayer functions which are called from within the user-created performance patch.

5.3.4.3 Output layer

According to the needs of the user, the gestures described by SurfacePlayer's functions can be used to send OSC or MIDI data to other applications. Generating simple visual feedback in response to these gestures is easy to implement using Processing itself; projected or displayed on a convenient screen during performance, this feedback can eliminate the need for a performer to look down at the surface itself constantly while playing.

5.3.5 Results

The SurfacePlayer algorithms provided easy access to some of the most commonly-used multi-touch cues – such as tap and double-tap recognition, multiple-taps supporting up to ten fingers, and directional swipes of varying speeds. They could also be used to determine the surface area, diameter, centroid and perimeter of shapes formed by surface touches. These cues were combined in complementary ways, using the strategies described in previous chapters, to investigate the feasibility of creating novel and expressive musical interfaces based mostly around multi-touch gestures.

While the architecture described above was useful as a prototyping platform, it became apparent that the highly specific components and investment of time

required to design a new interface using the software might be a significant deterrent for potential users. A more tightly integrated system, with specific musical functionality, would represent a more efficient and elegant way to utilise the algorithms for musical performance. This new system, Oscar, represents the final embodiment of the research described so far in this thesis and is discussed in detail in Chapter 6.

5.4 Conclusion

This chapter has discussed in detail many of the design issues particular to digital musical instruments that employ multi-touch surfaces. Through a comparison with two other touch devices that are used in a similar context – the XY pad and button array – we have looked at how the implicit physical characteristics of a device exert a strong influence upon their optimum role within a live performance context. A more detailed look at specific musical applications of multi-touch technology in recent years allows us to paint a picture of accepted design conventions.

These conventions are challenged on the grounds that they are not ideally-suited to the means of interaction provided by multi-touch technology and tend to overlook some of the more unique properties of this kind of device – in particular, the rich gestural cues that can be inferred from point data. Several unusual music interfaces are cited as examples that demonstrate successful alternative approaches.

An explanation for the markedly-conservative design conventions is offered by identifying a gap in the selection of development tools that are open to musicians using multi-touch. SurfacePlayer, a set of algorithms implemented in the Processing development environment, is introduced as a first step into exploring this promising space. The work described in Chapter 6 carries on directly from these findings and

attempts to establish a stronger grasp upon the concepts of non-visual-centred interfaces that have been established over the course of this chapter.

Chapter 6. Designing a new multi-touch instrument

"Often overlooked is the need to work on an instrument that responds sufficiently to the nuances of touch."

-Boris Berman, Notes from the pianist's bench [14]

This chapter describes the development of a new multi-touch interface called *Oscar* that is designed especially to facilitate the creation of multi-layered performance tools using the model established in Chapter 4. *Oscar* is a generic controller and stand-alone synthesizer for tablet devices that uses a novel non-visual interaction model inspired by the research described in this thesis. We discuss the motivation for designing *Oscar*, establish an explicit list of design goals, and describe in detail the various interface features at the heart of the software. This chapter concludes with a description of *DroneTilt* – an example of an alternative performance instrument designed with *Oscar* – and a description of how the interaction strategies described in Chapter 4 can be implemented using *Oscar*'s gestural interface.

6.1 Introduction to Oscar

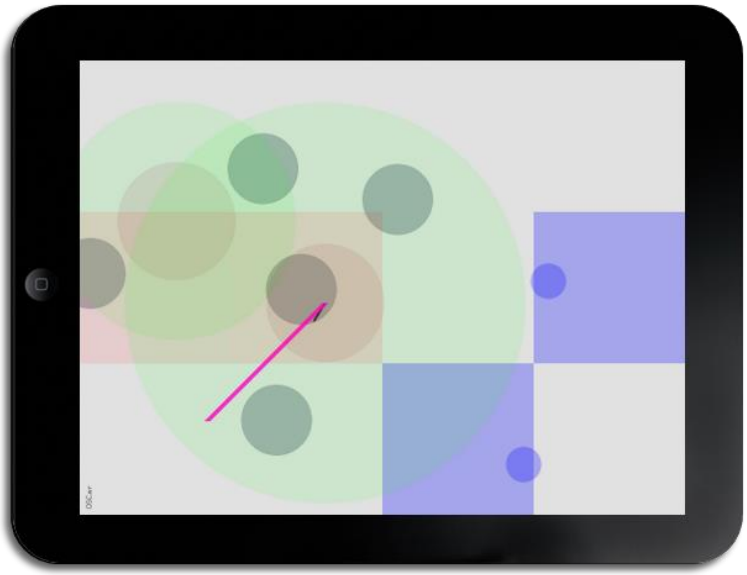

Figure 6.1: Oscar running on a 2nd generation iPad – the graphical feedback represents user touches, groups of touches and their centre-points, discreet zones and the direction of movement

Oscar is a music synthesizer and OSC controller that runs on iPad and Android. It utilises a unique interface paradigm that relies exclusively upon multi-touch gestures - there are no widgets or GUI controls employed during play. *Oscar* is powered by the audio programming language Csound [33] which can be used to generate and process sound in response to user input. Dropbox [41] is also integrated into the app to enable users to easily import their own Csound code and audio files. All of the data generated in response to user input can be sent to external hardware/software via OSC messages over a wireless connection - allowing remote control of other music software, synthesizers, graphics, etc.

Oscar was designed to test the viability of the descriptive model that we have already established (Chapter 4) and serves to demonstrate how this approach can maximise the musical potential of the output from a given piece of hardware. Note that the same ideas could be applied to any type of interface and not just multi-touch

devices. Also, while *Oscar* might have the potential to facilitate more ergonomic, minimalistic interaction styles (due to the simplicity of its gestures and the rich data they produce) it is, in essence, a development environment. It is intended that *Oscar*'s flexible interface will provide ample room for experimentation with new approaches to music control using tablets.

6.2 Design objective

The overall goal of *Oscar* is to provide electronic musicians - composers and performers - with an elegant, portable and highly-customisable tool for live performance using multi-touch surfaces. Existing solutions were either too complex (programming a gesturally-controlled music app from scratch) or too simplistic (commercial music apps with a particular performance or musical style in mind) to accommodate the digital musician who wants to experiment with the multi touch surface as a unique interface in its own right. Achieving this balance between complexity and accessibility is essential for a new musical interface – under the right circumstances, the user will gradually adapt the controller to suit their own musical needs and therefore prolong its lifespan [25].

Through experimentation with various techniques of multi-touch music control, and a comprehensive study of existing research, a number of explicit design goals were identified and implemented. It was decided that, in order to offer a genuinely useful platform for musical interface development, the app must provide a number of key features. To summarise, *Oscar* must:

- Incorporate an entirely gesturally-controlled performance mode that does not rely upon platform-specific widgets or GUI elements
- Ensure that performance mode cannot be interrupted by the accidental opening of menus, options, etc.

- Provide graphical feedback relating to the processing of user input and allow users to change the visual layout for aesthetic and/or feedback purposes
- Allow users to easily import their own Csound programs and audio material for rapid prototyping [13, 54]
- Allow users to quickly switch programs ('hot-swapping') during performance without needing to negotiate through the menu
- Be accompanied by a clearly-commented and easily-customisable template
- Allow users to employ the iPad's built-in sensors in their program designs (i.e. easy access to accelerometer and gyroscope data)
- Send gesture data via a wireless connection for control of external audio and visual software

The following sections describe, in order, how each of these features were implemented.

6.3 Gestural interface

This section contains an explicit step-by-step explanation of how *Oscar* processes user input. The interface was designed and modified over a lengthy period to accommodate the largest number of unambiguous, data-rich and complementary gestures possible.

The system organises individual persistent touches (i.e. fingers) into groups called ‘clusters’ which represent the users’ hands. The process by which this is achieved is laid-out below in section 6.3.1. This concept leverages the users’ own intuitive knowledge about the movement of their hands and gathers data about the kind of gesture being performed. This data is made accessible to the user via

variables defined within the Csound template (see 6.8) or output over a wireless network (6.10).

Unlike many other multi-touch gestural systems there are no separate events that represent, for example, pan/drag or pinch/zoom gestures. The information required to invoke musical behaviour in response to these gestures is indeed present (see the definitions of `kdirection`, `kvelocity` and `kisZooming` in 6.3.1. below) but the way that it is utilised is very much left up to the designer of the Csound code. There are several reasons for this – chiefly, the goal of *Oscar* to circumvent interface paradigms such as these that were created solely to interact with graphical systems and also the desire to accommodate users who wish to experiment with new approaches to multi-touch musical control.

Positional information, where applicable, is given by a point in two-dimensional space. The x and y values are translated from the native coordinate system of the iPad's own sensors to a range between 0.0 and 1.0. All positional data is recorded in pairs – i.e. the current position and the last position. This is an alternative to keeping a complete point history, most of which will never be used, which still allows the accurate detection of all the gestures that *Oscar* accommodates. The origin of the native coordinate system of *Oscar* is located at the top-left corner of the iPad itself in a landscape orientation. This is fixed and unaffected by device movement (analogous to the 'screen-lock' option present in many apps) in order to accommodate the fullest possible use of the other motion sensors in the iPad (see section 6.9).

The *Oscar* interface responds to four different gestures: clusters, touches, taps and flicks. These are perhaps best described as four varieties of event, each with

a selection of unique properties or attributes (although some properties, such as x and y location, are common to all four). Each gesture is described in detail below.

6.3.1 Clusters

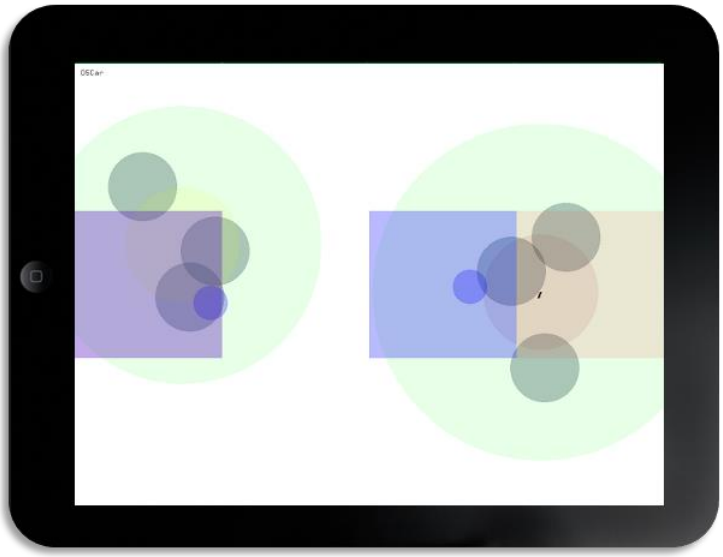

Figure 6.2: Two separate clusters, represented by large green circles that encompass the user's individual finger touches (shown as smaller grey circles)

The cluster abstraction is a key component of the *Oscar* interface. A cluster is a group of individual touches, where a touch is a persistently-tracked point with a unique ID that represents a finger making contact with the surface. The purpose of the cluster is to act as an abstraction of the user's physical hands. This allows a variety of high-level data relating to the group of touches to be calculated and subsequently interpreted by the synthesis engine. Each cluster is a continuous entity - its properties are updated constantly throughout its life-cycle – that is represented by its own instrument in the Csound environment. Therefore, events like cluster creation and destruction can be used to trigger various behaviours without the need for any separate 'cluster-is-created/destroyed'-style events.

A maximum of two clusters can exist at any given time. Each cluster can accept up to five separate touches, which are added according to their distance to the

cluster centroid. There is a very brief intentional delay between touches arriving on the surface and cluster creation/modification – this is to facilitate the independent articulation of discrete gestures, such as taps and flicks, without invoking musical behaviour associated with clusters (see 6.3.3. and 6.3.4.). A cluster is destroyed if all of its touches leave the surface. New clusters are created when a touch is added to the surface and one of the following conditions is met:

- There is no other cluster present
- In the case of there being one other cluster present, the touch is too far away to join it
- The nearest cluster contains five touches and cannot accept any more

Each cluster has the following properties (note the use of Csound variable name formats, where an *i* signifies an initialisation-time variable that does not change during play and a *k* signifies a control-rate variable that is updated dynamically) – `iclusterID, ix, iy, izone, izonex,izoney, inumTouches,kx, ky, kzone, kzoneX, kzoneY, knumTouches, ksize, kdir, kvel, kisHeld, kisZooming`. Each property is described in detail as follows:

- `iclusterID` (int) – the unique identifier of the cluster.
- `ix, iy` (int) – the centroid of the cluster at the time of its creation. This is calculated by averaging the position of each touch contained in the cluster. The centroid is therefore given by:

$$(x, y) \rightarrow \left(\frac{\sum_{i=1}^{n} x_i}{n}, \frac{\sum_{i=1}^{n} y_i}{n} \right)$$

  Where n is the number of touches in the cluster. Note that n cannot equal zero as a cluster must contain at least one touch.

- `izone, izoneX` and `izoneY` (int) – integers that represent a discretised location upon the surface where the cluster centroid was located at the time of its creation. The default settings divide the surface into 12 distinct zones in a 3x4 matrix (assuming portrait orientation). The `izoneX` and `izoneY` values provide convenient access to the column and row values respectively.
- `inumTouches` (int) – the number of touches contained within the cluster at the time of creation.
- `kx, ky` (int) – the current centroid of the cluster. See notes for `ix, iy` above.
- `kzone, kzoneX` and `kzoneY` (int) – the current discrete location of the cluster centroid. See notes for `izone, izoneX` and `izoneY` above.
- `knumTouches` (int) – the number of touches currently contained within the cluster.
- `ksize` (float) – a numerical value that represents the radius of a circle whose centre point is the cluster centroid and which contains the position of each touch in the cluster. If the number of touches, n, equals one then size = 0.1. Otherwise size is given by the distance from the centroid to the touch position which is located farthest away from it. This value is multiplied by a scaling factor in order to scale the largest comfortable hand span to equal 1.0. This scaled value cannot exceed 1.0.
- `kdir` (int) – the current direction the cluster is travelling in. The direction is represented discretely by one of several integer values representing the cardinal (N, S, E, W) and intercardinal/ordinal (NE, SE, SW, NW) directions. While the cluster is in motion, this value ranges from 1-8 in a clockwise fashion where 1=N, 2=NE, 3=E, etc. When the cluster is static `kdir`=0.

- `kvel` (float) – a numerical value denoting the speed at which the centroid of the cluster is moving. This can be used, in combination with `kdir`, to delimit behaviour that might be expressed via a panning/dragging movement. The speed is expressed in positional units per second. This value is normalised, for convenience of mapping to musical parameters, and cannot exceed 1.0.
- `kisHeld` (boolean) – a true/false value which specifies if the cluster has remained stationary since its creation. This is determined by checking how far the centroid has moved since its initial creation values. There is a margin of error to accommodate natural slight movements (<0.05 positional units) and a time delay (2 seconds) between the cluster creation time and the setting of this variable state. When the `kisHeld` check is successful, the value is set permanently (i.e. until the cluster is destroyed) and the graphical feedback that represents the cluster changes colour to reflect the fact. The user is then free to move the cluster without cancelling the `kisHeld` state and any audio processes that might be associated with it.
- `kisZooming` (boolean) – a true/false value that indicates whether or not the size of a cluster is increasing or decreasing. There is a small margin of error to accommodate natural fluctuations on behalf of the user or sensors. This property exists as an alternative to explicit 'pinch/zoom' gestures common to multi-touch systems – users can, if needs be, make use of this value to delimit certain behaviours without cancelling or overriding other gestures associated with the cluster.

6.3.2 Touches

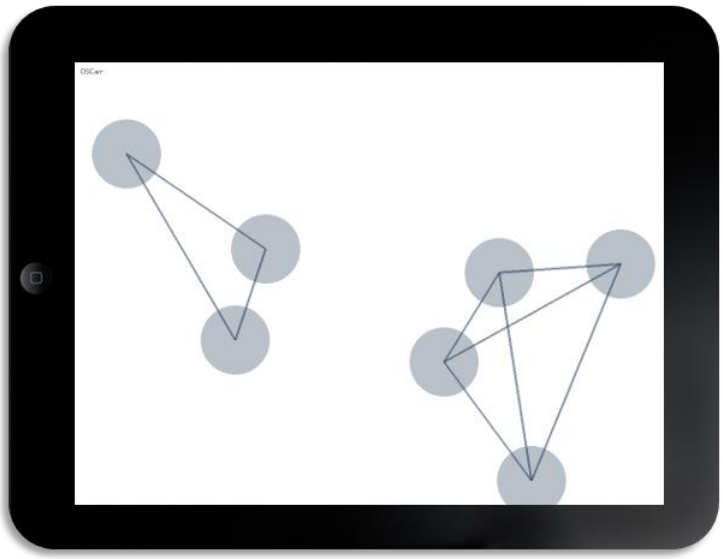

Figure 6.3: Individual finger touches represented by grey circles

Touches represent individual fingertips in contact with the multi-touch surface. Unlike many other gesture-recognition systems, touches in *Oscar* are not 'swallowed-up' or consumed when they join a cluster or become part of a gesture. This gives the user a great deal of freedom when designing a Csound program – if the interface in question requires individual touch data, but no clusters, the user simply ignores the cluster data in the code (and vice versa). As stated in the cluster definition above, this gives a great deal of flexibility and contributes to the non-prescriptive flavour of *Oscar*.

Another important benefit of giving the user access to touches that are independent of taps, flicks and clusters is the immediacy they provide. As described in the following section (6.3.3.), tap recognition involves a certain latency that may be unsuitable for rhythmic or time-critical event-triggering. It is recommended that the touch gesture is used for any events that require precision timing.

Touches are also represented by individual and unique instrument-instances within Csound with the following properties - `itouchID, ix, iy, izone, izonex,izoney,kx, ky, kzone, kzoneX, kzoneY`:

- `itouchID` (int) – the unique identifier of the touch.
- `ix, iy` (int) – the location of the touch at the time of its creation.
- `izone, izoneX` and `izoneY` (int) – the discrete location of the touch at the time of its creation. See equivalent description in 6.3.1.
- `kx, ky` (int) – the current location of the touch.
- `kzone, kzoneX` and `kzoneY` (int) – the current discrete location of the touch centroid. See equivalent description in 6.3.1.

6.3.3 Taps

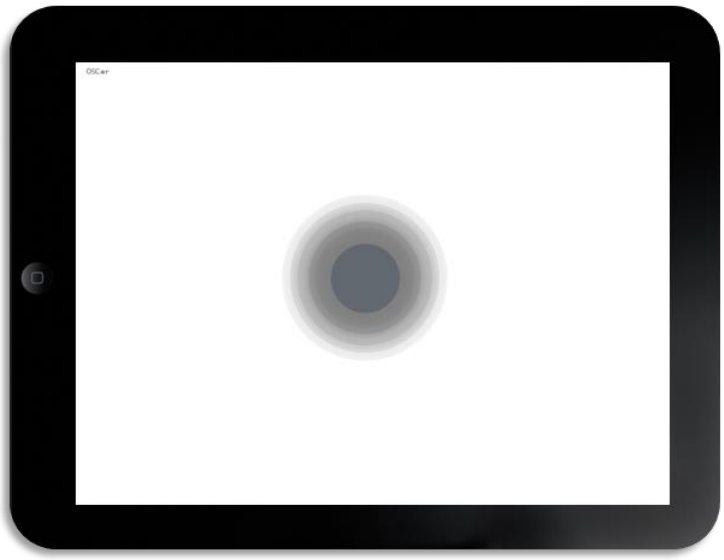

Figure 6.4: A tap event being recognised

The ubiquitous tap gesture common to most touch-screen devices. Taps in *Oscar* are discrete, one-off events that launch a brief (0.05 seconds) instance of a dedicated Csound instrument. This gesture is perfectly-suited to the triggering of samples/notes/events, changing modes, toggling various effects, etc.

There are two important points to note regarding the way *Oscar* processes tap events. Firstly, there is a small intentional delay between a touch arriving upon the surface and cluster-related behaviour (i.e. the touch being added to an existing cluster or forming a new cluster of its own). This allows the user to perform swift tap events without triggering any behaviour related to clusters – the graphical feedback clearly shows the brief delay between touch addition and cluster activity. Secondly, there is a further latency between tap performance and recognition that is necessary to allow the *Oscar* to recognise multi-finger taps. While the ability to accurately differentiate between taps of up to 5 fingers opens up many options for the designer, the inevitable latency may prove troublesome when it comes to time-critical event-triggering (e.g. MPC-style sample-triggering, playing notes like a piano, etc.). For this reason it is strongly-suggested that rhythmic activity is triggered using touch events (as described in 6.3.2.) and that taps are reserved for making more global decisions or triggering quantised samples, for example.

Tap events have the following properties - `ix iy izone izoneX izoneY inumTouches`:

- `ix, iy` (int) – the location where the tap was performed. This is obtained by calculating the average position of the taps constituent touches using the cluster centroid formula as described in 6.3.1.
- `izone, izoneX` and `izoneY` (int) – the discrete location of the tap. See equivalent description in 6.3.1.
- `inumTouches` (int) – the number of fingers used to perform the tap.

6.3.4 Flicks

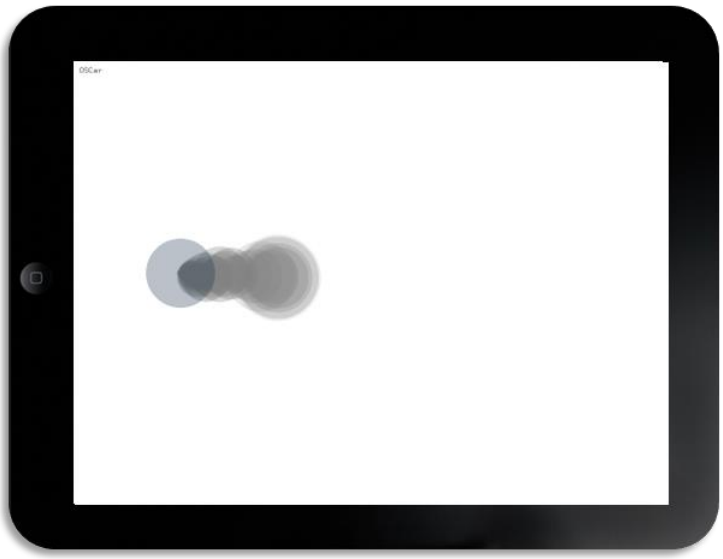

Figure 6.5: A flick event being recognised

The flick gesture is similar to tap in that it is a discrete, once-off event with a short duration (0.05). The only difference from an articulatory standpoint is that the user's touches deviate significantly from the initial point of contact prior to leaving the surface – hence the additional direction property. All considerations related to timing and latency using the tap gesture are equally-applicable here.

Flick events have the following properties - `ix iy izone izoneX izoneY inumTouches, idir`:

- `ix, iy` (int) – the location where the flick was performed.
- `izone, izoneX` and `izoneY` (int) – the discrete location of the flick.
- `inumTouches` (int) – the number of fingers used to perform the flick.
- `idir` (int) – the direction of the flick. See the description of `kdir` in 6.3.1

6.4 Hidden menus

Figure 6.6: Accessing the hidden menu

Oscar was designed specifically to facilitate experimentation and non-standard performance techniques - therefore, it was vital to ensure that a performance could not be disrupted by a user accidentally switching out of performance mode and into the menu system. Other full-screen apps place menus on the edge of the screen (Mugician [154]), provide a small icon that requires a double-tap (TC-11 [180]), or use a shaking gesture (TUIOpad [188]) to access options while still retaining most of the screen real-estate for actual gesture performance. The first two options were considered inappropriate, given the emphasis on non-visual interaction at the heart of the gesture engine itself, and the shake-to-exit approach would make it impossible for musicians to fully-utilise the gyroscope data in their performance programs.

After considerable experimentation with a number of gestures, a hybrid approach was chosen to allow access to the menu without compromising the main performance area or risking accidental activation. Rather than reserve a specific gesture, and therefore prohibit its use for actual performance, a combination of location, movement and time data is used. The user must place a single touch in the extreme top-left hand corner of the screen (location) and keep it reasonably still

(movement) for a count of two seconds (time). The touch point also appears red to signify that the user is about to open the menu.

In practice, this has been a successful solution - the menu is easy to open once the user has learned how to do so and almost impossible to trigger by accident. The highly-specific nature of the gesture itself makes it unlikely to compromise any design that a programmer might have.

6.5 Customisable graphical feedback

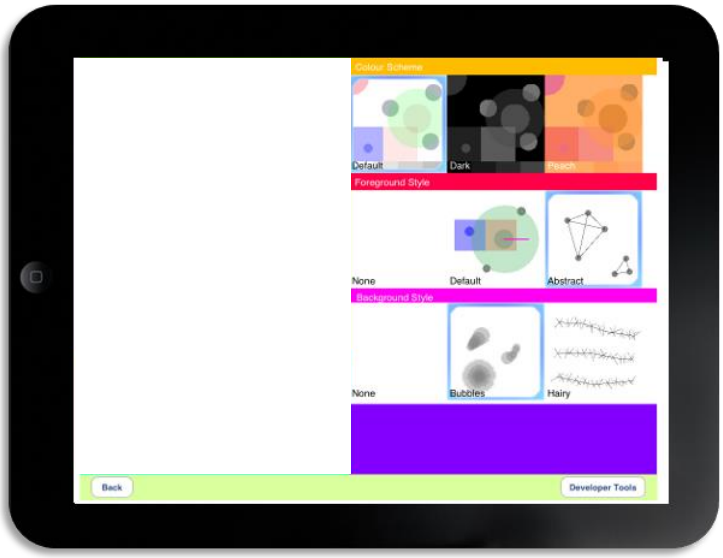

Figure 6.7: Graphics selection menu

Early iterations of the *Oscar* concept [118] did not feature any graphics - with all of the options hidden in the iPad's Preferences menu. This proved unsatisfactory for many reasons but mostly made it difficult for a user to comprehend how the system perceived his/her actions onscreen.

The initial priority during design was to visualise as many aspects of the gesture processing activity as possible - for debugging and fine-tuning the engine. It quickly became apparent that too much information might also be a problem - for example, seeing the individual touches represented graphically in a performance

patch that only uses clusters might obscure the process and confuse performer and audience alike.

Therefore a customisable graphics feature was added in order to allow users to select from a range of colour schemes and data visualisations. This feature also supports the idea that *Oscar* is not limited to a particular genre of music or style of performance - combinations can range from the subtle, to the informative, and to the futuristic and garish, as the performance context demands.

6.6 Import user programs and audio

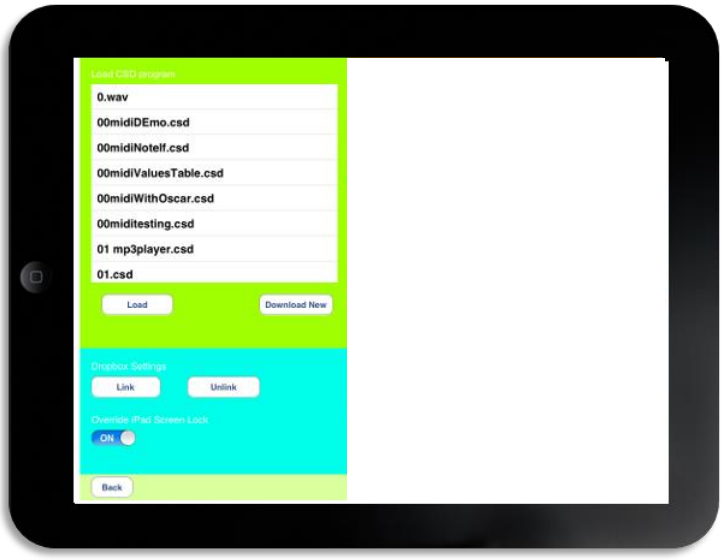

Figure 6.8: Dropbox menu

Oscar's behaviour is determined by customisable code written in the Csound Unified File Format (often abbreviated to, and hereafter referred to, as a CSD file [35]). These files can be easily created and modified using the template designed especially for *Oscar* (see 6.8 below).

Programming even a simple CSD for *Oscar* typically involves an iterative trial-and-error approach, as different gesture types are combined with different audio outputs and bugs/typos are hunted down. It was vital to ensure that the process of

downloading and testing a new version of a CSD is as simple, easy and fast as possible.

By syncing up the app with their Dropbox account, users can store their CSD files in a Dropbox folder, open and edit them using another app or computer, and simply press 'update' in *Oscar*'s menu when they want to download and test their code. Audio files are stored and retrieved in the same way which allows users to play and process pre-arranged material in their *Oscar* patches.

6.7 Hot-swapping of programs

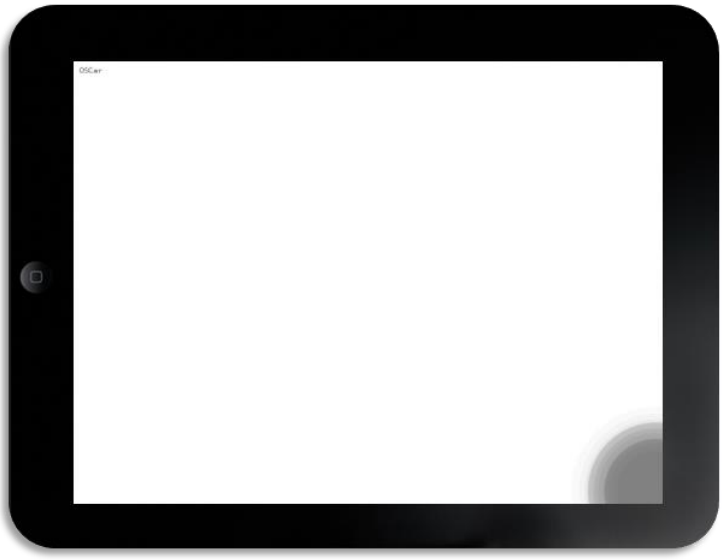

Figure 6.9: Selecting a hot-swappable program

The ability to switch patches or presets fluidly during performance is a key feature of many hardware synthesisers. Different patches are typically accessed via entering an ID number via a numeric keypad, but some manufacturers provide a jogwheel or customisable banks to ensure quick and error-free switching mid-performance. It was important to facilitate this kind of play to encourage the creation of small, modular programs for *Oscar* or even re-using of the same program with different content (e.g. two versions of the same sampler program that load different banks of

samples into memory). It was also vital to ensure that this behaviour could not be triggered by mistake or confused with the other gestures that *Oscar* provides.

A similar approach to the hidden menu system described in 6.4 above was implemented - the user must place a single touch in one of the remaining corners of the screen (top-right, bottom-left and bottom-right) and hold it in place for a count of two seconds. These three gestures instruct *Oscar* to change immediately to one of three pre-selected CSD files labelled A, B and C. These labels can be assigned via the file browser in the main menu system. While there is a slight compromise on time, due to the mandatory two second delay, it was deemed more important to ensure that accidental-triggering was made impossible than to allow split-second switching of patches during play. The intended use of this feature is to change the functionality of *Oscar* in-between sections of a piece, separate songs or sets - there are ample gestures described in 6.3 that can potentially be used to alter program functionality during play if necessary (e.g. tapping in a particular zone to select a particular musical scale).

6.8 Csound template

A comprehensive and clearly-commented Csound CSD template has been developed and maintained throughout the design process. The template features default audio settings for iPad, user-defined-opcodes for Cluster and Touch events, blank instruments that are called in response to Cluster, Touch and accelerometer/gyroscope updates and Tap/Flick events, global reverb and master channels. Each section is explained clearly via comments and variables related to *Oscar* are pre-cast and ready for use in instrument definitions.

6.8.1 CsOptions and global variables

Default audio settings for *Oscar* are provided in CsOptions. In order to allow the user to access their own audio within the program, *Oscar* needs to know the directory where files imported from Dropbox are stored. As the iOS file structure is not made explicit to the user and is difficult to read, the **chnexport** opcode is used to receive a directory path from the Csound API. This is stored as a string named ***gSresourcePath*** which can be prepended to any references to filenames within the code (e.g. for reading an audio file into a table). Finally, global variables to keep track of accelerometer values (*gkaccX*, *gkaccY*, *gkaccZ*) are initialised alongside left/right audio-rate channels for reverb and master output (***gareverbL/R*** and ***gamasterL/R*** respectively).

```
<CsoundSynthesizer>

/*
Oscar program template
1st of April 2014
*/
<CsOptions>
-odac -dm0 -+rtmidi=null -+rtaudio=null -+msg_color=0 -
M0
</CsOptions>

<CsInstruments>

sr        = 44100
ksmps     = 32
nchnls    = 2
0dbfs     = 1

/* GLOBAL SETUP */

; Resource path
gSresourcePath chnexport "resourcePath", 1

; Accelerometer variables
```

```
gkaccX    init 0
gkaccY    init 0
gkaccZ    init 0

; Global reverb channel
gareverbL init 0
gareverbR init 0

; Master output channel
gamasterL init 0
gamasterR init 0
```

Figure 6.10: csOptions and global variables

6.8.2 UDOs for touch and cluster events

Two user-defined opcodes (or UDOs) are used as a bridge between the data arriving from the *Oscar* interface itself and the local variables in Csound. The main purpose of these opcodes is to streamline the process of mapping the gestures to audio output by distancing this process from the instrument definitions as much as possible. The user does not ever need to alter the contents of this section and it is intended that this code will be hidden from the end user in the final release version of *Oscar* (within a text document that is accessed in the header via an `#include` command).

The **`sprintf`** opcode is used to generate strings referencing the appropriate variables being passed from *Oscar* intro the API. These strings are subsequently used to assign the values to local Csound variables using **`chnget`**.

```
; UDO for Touch events
opcode Touch, iiiiiiikkkk, iiiiii p4, p5, p6, p7, p8, p9
xin

itouchID = p4

; Dynamically-generated channel names
S_x       sprintf "touch.%d.x", itouchID
S_y       sprintf "touch.%d.y", itouchID
S_zone    sprintf "touch.%d.zone", itouchID
S_zoneX   sprintf "touch.%d.zoneX", itouchID
```

```
S_zoneY   sprintf "touch.%d.zoneY", itouchID

; K-rate variables for touch
kx        chnget S_x
ky        chnget S_y
kzone     chnget S_zone
kzoneX    chnget S_zoneX
kzoneY    chnget S_zoneY
xout p4, p5, p6, p7, p8, p9, kx, ky, kzone, kzoneX,
kzoneY
endop
```

Figure 6.11: Touch event UDO

```
; UDO for Cluster events
opcode Cluster, iiiiiiiikkkkkkkkkk, iiiiiii p4, p5, p6,
p7, p8, p9, p10 xin

iclusterID = p4

; Dynamically-generated channel names
S_x          sprintf "cluster.%d.x", iclusterID
S_y          sprintf "cluster.%d.y", iclusterID
S_zone       sprintf "cluster.%d.zone", iclusterID
S_zoneX      sprintf "cluster.%d.zoneX", iclusterID
S_zoneY      sprintf "cluster.%d.zoneY", iclusterID
S_numTouches sprintf "cluster.%d.numTouches", iclusterID
S_size       sprintf "cluster.%d.size", iclusterID
S_direction  sprintf "cluster.%d.direction", iclusterID
S_velocity   sprintf "cluster.%d.velocity", iclusterID
S_isHeld     sprintf "cluster.%d.isHeld", iclusterID
S_isZooming  sprintf "cluster.%d.isZooming", iclusterID
; K-rate variables for cluster
Kx             chnget S_x
ky             chnget S_y
kzone          chnget S_zone
kzoneX         chnget S_zoneX
kzoneY         chnget S_zoneY
knumTouches    chnget S_numTouches
ksize          chnget S_size
kdir           chnget S_direction
kvel           chnget S_velocity
kisHeld        chnget S_isHeld
kisZooming     chnget S_isZooming
```

```
xout p4, p5, p6, p7, p8, p9, p10, kx, ky, kzone, kzoneX,
kzoneY, knumTouches, ksize, kdir, kvel, kisHeld,
kisZooming
endop
```

Figure 6.12: Cluster event UDO

6.8.4 Instrument definitions for touch and cluster events

Instruments 1 and 2 are reserved for receiving data from touch and cluster events (this is an important factor to remember when using external MIDI with *Oscar*, given the default mapping of MIDI channels to instruments in Csound). This is necessary to facilitate dynamic instrument number allocation - each new touch and/or cluster that is detected creates a new instance of its corresponding instrument with an incremental decimal point naming system.

These instrument definitions are kept relatively free from clutter through the use of the UDOs – a single line of code pulls-in all of the gesture event data and assigns it to a selection of local variables (as described in 6.3). All the user needs to do is add synthesis and/or processing code that makes use of these variables.

```
instr 1
/* ---TOUCH---

Score format: i1.N 0 -1 N x y zone zoneX zoneY
Each individual touch generates a new instance of this
instrument, which is killed upon touch removal. */

; Touch properties
itouchID, ix, iy, izone, izonex,izoney,kx, ky, kzone,
kzoneX, kzoneY Touch p4, p5, p6, p7, p8, p9

;-----Add synths here-----;

; Master output
; gamasterL = gamasterL +
; gamasterR = gamasterR +
; Reverb send
; gareverbL = gareverbL +
```

```
; gareverbR = gareverbR +
endin
```

Figure 6.13: Instrument 1 – touch event

```
instr 2
/* ---CLUSTER---

Score format:i2.N 0 -1 N x y zone zoneX zoneY numTouches
Touches arriving within a certain distance of one
another are grouped into a cluster. Each cluster has a
set of shared parameters (number of touches, size, etc.)
There can only be a maximum of 2 clusters present,
intended to be used for left and right-hand. Clusters
die when all of their touches are removed. */

; Cluster properties
iclusterID, ix, iy, izone, izonex,izoney,
inumTouches,kx, ky, kzone, kzoneX, kzoneY, knumTouches,
ksize, kdir, kvel, kisHeld, kisZooming Cluster p4, p5,
p6, p7, p8, p9, p10

;-----Add synths here-----;

; Master output
; gamasterL = gamasterL +
; gamasterR = gamasterR +
; Reverb send
; gareverbL = gareverbL +
; gareverbR = gareverbR +
endin
```

Figure 6.14: Instrument 2 – cluster event

6.8.5 Instrument definitions for tap and flick events

Tap and flick events are both sent to *Oscar* in the form of score statements - this negates the need to create UDOs (there is no messy patching to do) and also allows the use of non-numeric instrument names.

```
instr tap
/* ---TAP---
Score format: i "tap" 0 0.05 x y zone zoneX zoneY
numTouches
```

```
When a group of touches hits and leaves the surface
quickly, without moving far, a tap event is triggered.
*/

; Tap properties
ix = p4
iy = p5
izone = p6
izoneX = p7
izoneY = p8
inumTouches = p9

;-----Add synths here-----;

; Master output
; gamasterL = gamasterL +
; gamasterR = gamasterR +
; Reverb send
; gareverbL = gareverbL +
; gareverbR = gareverbR +
endin

instr flick
/* ---FLICK---
Score format: i "flick" 0 0.05 x y zone zoneX zoneY
numTouches dir
Identical to a Tap event, except the touches have moved
prior to leaving the surface. Gives direction value. */

; Flick properties
ix = p4
iy = p5
izone = p6
izoneX= p7
izoneY = p8
inumTouches = p9
idir= p10

;-----Add synths here-----;

; Master output
; gamasterL = gamasterL +
; gamasterR = gamasterR +
; Reverb send
```

```
; gareverbL = gareverbL +
; gareverbR = gareverbR +
endin
```

Figure 6.15: Tap and flick instrument definitions

6.8.6 Reverb, master and accelerometer instruments

A simple reverb channel is included to demonstrate how to use auxiliaries and a master channel is also provided to ease workflow during patch design. The latter is equipped with a **clip** opcode to prevent new users from damaging their speakers and/or hearing while getting to grips with the Csound language. A final helper instrument reads the accelerometer values and feeds their values into the global variables described in 6.8.1.

```
instr reverb
/* ---REVERB---
Score format: i "reverb" 0 3600
A basic global reverb instrument. */

aL, aR    reverbsc gamasterL*0.05, gamasterR*0.05, 0.9,
10000
outs aL, aR
clear gareverbL, gareverbR
endin

instr master
/* ---MASTER---
Score format: i "master" 0 3600
Master output bus */

aoutL clip gamasterL
aoutR clip gamasterR
      outs aoutL, aoutR
      clear gamasterL, gamasterR
endin

instr accel
; Accelerometer update instrument
gkaccX chnget "accelX"
gkaccY chnget "accelY"
```

```
gkaccZ chnget "accelZ"
; printks "X = %f, Y = %f, Z = %f\\n", 0.25, gkaccX,
gkaccY,gkaccZ
endin
```

Figure 6.16: Reverb, master and accelerometer instrument definitions

6.8.7 CsScore

The score section is typical of live Csound programs and simply contains commands to run Csound and the three helper instruments from section 6.8.7 indefinitely.

```
<CsScore>

; Run Csound indefinitely
f 0 6600

; Run reverb instrument
i "reverb" 0 6600

; Run master instrument
i "master" 0 6600

; Run accelerometer instrument
i "accel" 0 6600

e
</CsScore>
```

Figure 6.17: CsScore

A copy of the template in its entirety is provided as Appendix A.

6.9 iPad sensors

While the original intention of *Oscar* was to provide a purely multi-touch driven interface, it made little sense to omit the other iPad sensors from the selection of potential controls available to the user. We have focused upon the accelerometer and gyroscope sensors - which can detect acceleration and rotation, respectively, along the x, y and z-axes. Simply reading the gyroscope and accelerometer data within

Oscar opens up a number of exciting possibilities by offloading aspects of "what would otherwise be purely touch-based visual interactions onto the motion channel" [60] and allows the integration of *ancillary* gestures into performance setups [24]. As seen in the examples below, this data can be used to facilitate *background* interaction that complements the *foreground* interaction offered by the multi-touch gestures themselves [59]. Some of the possibilities are described as follows.

6.9.1 Hard-linking motion data to global variables

Simply using the variables associated with the device orientation (pitch/yaw/roll) within the Csound file can give interesting and dynamic results. Examples include scaling the volume of various channels according to the orientation, changing effects sends, modulation of synthesised sounds, etc. This technique is used as the main form of interaction for the ambient drone performance instrument described below in 6.12. It can also be used to enhance the articulatory options available to a performer for a sound event that is being triggered by a multi-touch gesture. Depending on the respective influence that the touch and motion data exert over the parameters of a musical event, this kind of approach might be more suited to the next category. Even if they are not actually part of the same gesture, with regard to how the program itself processes user input, from a phenomenological perspective they appear to be linked during performance and can hence be considered a cross-modal gesture – combining aspects of both motion and touch [62].

6.9.2 Combined touch and motion gestures

There are many different ways that sensor data can be used to augment the user's touch input, and vice versa. Gestures that are comprised of both touch and motion data exhibit a number of interesting properties that make them a valuable addition to

the vocabulary of the mobile application designer [46]. These can be further divided into *Touch-Enhanced Motion Techniques* and *Motion-Enhanced Touch Techniques.*

Touch-Enhanced Motion Techniques can be used to infer the context of use or add detail to the expression of a touch gesture using the incidental vibrations induced by finger contact [46]. In the case of *Oscar*, for example, a held touch could generate a tone using a VCO (voltage controlled oscillator) and subsequent device motion could be used to control the depth of an LFO (low frequency oscillator). The essential factor to consider in this kind of interaction is how to differentiate intentional motion gestures from incidental device-handling. Hinckley and Song advocate the use of a "comfortable and imprecise target to delimit motion…[e.g.] let the user gently rest a finger anywhere on the screen while moving the device. Such motions demand less attention, do not impose a particular hand-grip, and may be more comfortable to articulate" [46].

Motion-Enhanced Touch Techniques use the accelerometer/gyroscope data to infer characteristics of a touch event that cannot be detected solely by the touch screen. This can be used as a proxy for pressure sensitivity - the gap between the successive accelerometer peaks generated by a tap can be measured and used to differentiate between different intensities or strengths of touch impact. This can be used to assign varying functionality to hard and soft taps, due to the "clear signatures" of the impacts themselves, but more complex emulation of acoustic-style pressure sensitivity with this method has proven unreliable due to the sheer range of variability that the signal can exhibit [46].

6.9.3 Changing behaviour based on device orientation

The process of changing behaviour based upon device orientation is a familiar paradigm in mobile/tablet applications that is used to facilitate context-sensitive

interaction [22]. The most commonly-used application of this technique is to rotate the view of a web-browser, document reader or photo viewer according to the orientation of the device - this allows the user to seamlessly switch between landscape and portrait-style views without the need for additional onscreen controls.

In a musical context, depending on the intended use, this kind of threshold based mode switching can be used to add several different layers of control to a program. Assuming that the intended use scenario facilitates the changing of device orientation, this means of mode-switching has many advantages - it is unambiguous, requires physical effort to change and maintain, is usually impossible to trigger by accident and the physical state of the hardware itself provides naturally-occurring feedback on the state of the program – a contextual cue that is sensed in the background without disrupting other performance gestures [61]. A mode of operation that is selected in this way is intrinsically delimited by the physical tension required to hold the device in a specific way [61]

It should be noted that, in order to implement this type of behaviour effectively, it is important to take into account the default or 'resting' position of the device. This can be hard-coded into the Csound program in cases where the orientation changes can be clearly defined and differentiated. Another option is to provide a calibration function within the code which reads the current sensor data and sets this as the point (0, 0, 0). In this case the device can be held in a variety of ways and the displacement can still be calculated and subsequently used to invoke behavioural changes. This can be useful, for example, for performance setups that are designed to be played while suspended from the musician like a guitar. This can be used to detect and facilitate both left and right-handed guitar playing styles.

In any case it is advisable to provide a means to re-calculate the resting position dynamically during performance. This allows the performer to drift considerably from their initial physical posture, a reasonable phenomenon, without jeopardising the accuracy of the system's response to their playing. A button upon an external input device or an obscure multi-touch gesture can be reserved for this 'orientation-reset' function for performance setups that are likely to involve performer movement.

6.9.4 Purely motion-based gesture recognition

It is also possible to infer certain gestures solely based upon a statistical analysis of the accelerometer/gyroscope readings. There are currently no plans to implement features within *Oscar* to assist in this process - the usefulness of any motion-based gesture depends entirely on how the device is intended to be held and touched in a given performance setup. The strength of *Oscar* lies in its flexibility and any explicit indicators of how it is supposed to be held or operated run contrary to the design goals.

There have been, however, numerous musical projects in recent years that make extensive use of this kind of gesture. Literature describing digital musical instruments that use the Nintendo Wii Remote as a primary input device can give a clear sense of how to analyse 3D motion data from a held device [211, 52, 162] and software such as Wiigee [208] and GlovePIE [55] can assist users in defining and recognising such exclusively motion-based gestures. Certain non-obvious gestures that can be detected using this data have also been identified - for example, detecting taps upon different corners of the device body without using multi-touch information [62].

These categories provide some indication of the vast potential opened-up by simply reading the accelerometer and gyroscope sensors into *Oscar*. The key to creating successful interaction paradigms in this way is congruency - designers who seek to explore the "untapped possibilities" of contextual sensing must carefully consider the aesthetic and ergonomic experience of the intended performer [59]. Generating a logical and separated list of recognised multi-modal gestures and their associated behaviour (see Figure 3 of [60]) can be a useful way to identify potential problems or heavily-weighted gestures.

6.10 Wireless control

Oscar can be used as a wireless controller for external audio, graphics, or gaming software. All of the gestural data that is generated internally and used by the Csound engine can be made available to other devices on the same wireless network as the iPad itself. This data can be sent alone, in order to use *Oscar*'s gestures as a remote control, or also in combination with the on-board synthesis engine for the control of additional, external graphics or sound alongside those generated within the app.

Gesture data is sent wirelessly via Open Sound Control (OSC) messages. The messaging format is defined as follows:

```
/oscar/touchadded -
 float x, float y, int zone, int zoneX, int zoneY

/oscar/tap -
 float x, float y, int zone, int zoneX, int zoneY, int n
umTouches

/oscar/flick -
 float x, float y, int zone, int zoneX, int zoneY, int n
umTouches, int dir

…
```

```
/oscar/zoom -
 int clusterId, float x, float y, int zone, int zoneX, i
nt zoneY, int numTouches, float size, int type, float ve
locity

/oscar/pan -
 int clusterId, float x, float y, int zone, int zoneX, i
nt zoneY, int numTouches, int dir, float velocity

/oscar/held -
 int clusterId, float x, float y, int zone, int zoneX, i
nt zoneY, int numTouches
```

Figure 6.18: OSC message format

6.11 Typical workflow

The ideal work scenario for designing *Oscar* programs involves a computer and an iPad running *Oscar* – both of which should be online. The user opens the *Oscar* template in a code editor on the computer (e.g. CsoundQT [148]) and adds some content. The modified template is then saved/uploaded to a Dropbox folder using the same account that is synced with *Oscar*. Once the file is uploaded successfully, the user accesses the Dropbox menu in *Oscar* and searches for new content by clicking 'refresh'. *Oscar* will then download and overwrite any existing programs, where applicable, with the newly-edited files. The user tests the file running on *Oscar*, notes any changes that need to be made, and returns to QuteCsound to refine the design.

Needless to say it is possible to use any iPad word-processing app that can access Dropbox to edit *Oscar* templates – all that is necessary is an internet connection in order to sync up content. A future design goal of *Oscar* is to integrate a code editor and debugger into the app itself – this would serve the dual purpose of completely integrating the design process and also dispensing with the need for an internet connection when developing new programs. Another future development

that will increase the flexibility of *Oscar* programs is the addition of individual in-app settings for each template that allow the user to change parameters like zone size and configuration, tap/flick recognition speed, tempo/BPM and musical scale.

6.12 Case study: DroneTilt

DroneTilt is a performance instrument that uses *Oscar*, with no peripheral equipment, to generate and modify a sample-based dronescape using a combination of cluster objects and the accelerometer data. It is an example of what Miranda and Wanderley describe as an 'alternate gestural controller' and demonstrates *Oscar*'s ability to facilitate diverse multimodal interface design (see [120] and 4.11.2). A video example of *DroneTilt* being played is included on the CD that accompanies this thesis.

6.12.1 Concept

The central goal of this instrument design was to allow the performer to make subtle and nuanced changes to the texture of a drone-based soundscape in a physical, non-analytical sense. In other words, the typical approach of slowly-modifying a large selection of parameters via faders or the equivalent was to be avoided in favour of a more complex, one-to-many mapping scheme that would allow the coupling of gestures to particular 'flavours' or 'colours' of sonic texture. One of the core strengths of *Oscar* as a controller is its wide array of continuous controls and the gradually-shifting evolution of drone music was identified as an interesting means to explore the potential of these controls.

While drones are found in many different musical genres and ethnic traditions around the world, the aesthetic in this case belongs to a more modern school of composition that traces its roots back to the compositions of Terry Riley

and La Monte Young. The key aspect of a drone-based composition is the sustained tone that usually persists, albeit in different forms, throughout the development of the piece:

> …sustained intonation that establishes a harmonic center for its accompanying elements…the drone might utilize a single note repeated indefinitely or, at the opposite extreme, all of the scale's notes spread across numerous octaves. Other key aspects include extended duration, modular repetition, and a focus on overtones…the trance-inducing drone with its extended tones and layered pitches does change but glacially…[184]

Drone-based music is strongly centred upon the listening experience and artists working in the genre often aim to induce a kind of altered-state of consciousness in their audience: the listener often discovers "what seemed to be a single drone sound shifts and changes as the listener scans and focuses on different parts of it, opening up into a universe of overtones, microtones and combination tones" [17]. The primary goal in mind when designing this particular instrument was to ensure that the performers could access the same holistic listening space as the audience and feel like they were navigating through the soundscape rather than just shaping it through the manipulation of abstract parameters. A secondary goal was to utilise the built-in accelerometers of the iPad as the main channel of expression in order to investigate their usefulness as the main performance sensor, rather than the ancillary role they are typically allocated in mobile performance apps.

The performer cradles the iPad in one hand and uses the other to touch the surface itself. This divides up the roles between both hands – the touching hand performs selection gestures (altering individual loop parameters and triggering

effects) while the cradling hand is responsible for articulation gestures (using the tilt sensors to control the complexity of the overall drone texture).

6.12.2 Loop parameters

The main texture is comprised of three samples that loop continuously throughout performance: a high-pitched tonal texture, a low-pitched tonal texture, and an abstract percussive sample. The loops used are provided in the appendix. Each loop has three parameters – start time, length/end time and volume – that can be changed by touching the surface.

The screen is divided lengthwise into three sections, each representing one of the loops and responding to one and two-finger clusters. A single touch along any of the loop sections sets the volume of the loop and a two-finger cluster sets both the start position (the lengthwise location of the cluster) and the length (size of cluster) of the portion of the sample being repeated.

6.12.3 Low pass resonant filter

A low pass filter can be applied to the whole texture by performing a held two-finger cluster gesture. The x and y locations of the cluster control the cutoff and Q properties, respectively, and the size of the cluster alters the pre-gain of a slight distortion that is incorporated into the effect.

6.12.4 Accelerometer

The primary means of altering the texture of a DroneTilt performance is by changing the orientation of the device from its default, face-up position. Each axes has a different effect on the overall timbre of the drone and uses a different physical movement, assuming the suggested way of holding the device.

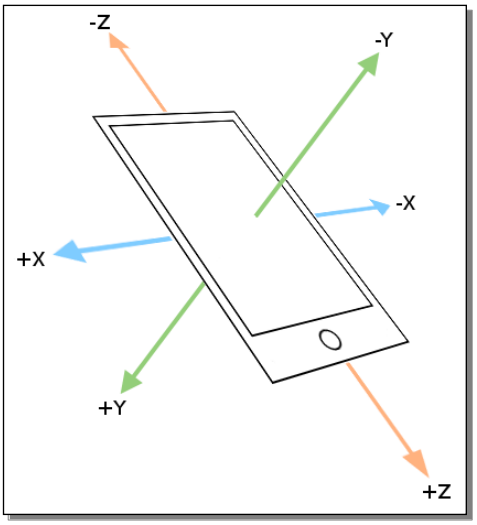

Figure 6.19: Visualisation of accelerometer axes

The x-axis value is changed using wrist-rotation – tilting controls the volume of two copies of the drone signal, hard-panned left and right, that have been pitch-shifted slightly up and down. Negative x-values (tilting towards the performer's body) have different detuning ratios than positive x-values (tilting away) to provide a clearly-delineated choice of timbre using a similar gesture. There is also a small threshold above the resting position (±0.25) that must be exceeded – this serves to allow for slight deviations and a comfortable holding position for the hand supporting the device.

Y-axis tilting is performed by relaxing and tensing the bicep and elbow joint. This gesture works in a similar way to the wrist-tilt and allows the user to fade between the original drone texture and a copy that is being fed through a multi-tap delay line and pitch shifted using a phase vocoder. Positive and negative y-axis tilts produce the exact same behaviour, unlike the x-axis, due to the ergonomic difficulty of suspending the iPad and raising it to face the body.

Z-axis movement is a special gesture that is difficult to perform and reserved for the closing of a performance. The z-value ranges from -1 (device is face-up) to 0 (device is on its side) to 1 (device is face-down). Once a threshold is exceeded

(>0.25) a fourth sample is faded-in to replace the main soundscape, which is faded-out at the same rate. This extra sample, a noisy granular rumbling, overwhelms the entire piece and acts as a final punctuation to the performance.

6.12.5 Code excerpts

```
instr 10
kSpeed init 1; playback speed
iSkip init 0; inskip into file (in seconds)
iLoop init 1; looping switch (0=off 1=on)
ifn = p4
ichns = ftchnls(ifn)
isamps = ftlen(ifn)
ilength = (isamps/sr)/ichns
if (ifn == 1) then
kamp = gkloop1vol
kloopstart = gkloop1start
klooplength = gkloop1length
elseif (ifn == 2) then
kamp = gkloop2vol
kloopstart = gkloop2start
klooplength = gkloop2length
else
kamp = gkloop3vol
kloopstart = gkloop3start
klooplength = gkloop3length
endif
kpitch = 1
kloopend = (klooplength*8)+0.05
kcrossfade = 0.05
asig flooper2 kamp, kpitch, kloopstart, kloopend,
kcrossfade, ifn

if (abs(gkaccX)>=0.25) then
```

```
kshift1 = (abs(gkaccX)-0.25)*(1/(1-0.25))
printk 0.25, gkaccX
if(gkaccX>0.1) then
asigShift1 flooper2 kamp*(kshift1*0.5), 0.74, kloopstart,
kloopend, kcrossfade, ifn, 0, 2
asigShift2 flooper2 kamp*(kshift1*0.5), 1.22, kloopstart,
kloopend, kcrossfade, ifn, 0, 2
else
asigShift2 flooper2 kamp*(kshift1*0.5), 0.33, kloopstart,
kloopend, kcrossfade, ifn, 0, 2
asigShift1 flooper2 kamp*(kshift1*0.5), 1.62, kloopstart,
kloopend, kcrossfade, ifn, 0, 2
endif
gamasterL = gamasterL + asigShift1
gamasterR = gamasterR + asigShift2
endif

multitap asig, 0.1, 0.4, 0.5, 0.3, 0.7, 0.2, 0.9,
fsig1 pvsanal adelayL, 1024, 256, 2048, 1
fsig2 pvscale fsig1, 4*((gkwob*0.5)+1), 0, 1.5
abackL pvsynth fsig2
adelayR multitap abackL, 0.2, 0.4, 0.6, 0.3, 1, 0.2,
1.3, 0.1
gamasterL = gamasterL + asig + abackL*(gkaccY*0.4)
gamasterR = gamasterR + asig + adelayR*(gkaccY*0.4)
endin
```

Figure 6.20: Looping instrument with X/Y auxiliaries. Each flooper2 instance holds a pitch-shifted variant on the loop, while the pvsanal/pvscale/pvsynth chain uses a phase vocoder to perform pitch shifts.

```
kflag release
if(kflag==1) then
gkeffect=0
endif
```

```
if(kisHeld==1)then
if(knumTouches==2)then
gkeffect=2
gkpregain = ksize*5
gkcf = kx*9000
gkq = ky*15
gkwob=1
endif
else
if(knumTouches==1) then
if(izoney==0) then
gkloop1vol = kx
elseif(izoney==1)
then gkloop2vol = kx
elseif(izoney==2)
then gkloop3vol = kx
endif
elseif(knumTouches==2) then
if(izoney==0) then
gkloop1start = kx
if(ksize!=0) then
gkloop1length = ksize
endif
elseif(izoney==1) then
gkloop2start = kx
if(ksize!=0) then
gkloop2length = ksize
endif
elseif(izoney==2) then
gkloop3start = kx
if(ksize!=0) then
gkloop3length = ksize
endif
```

```
endif
endif
endif
endin
```

Figure 6.21: Except from cluster instrument controlling loop parameters and filter. The variable kisHeld activates a bandpass filter controlled by a cluster of two fingers. Otherwise, single touches scale the volume of each loop while two-finger clusters control the loop start point and length.

6.12.6 Discussion

DroneTilt was used to perform a live improvisation as part of a small concert entitled 'Interfaces & Psychoacoustics' [71] that was organised and co-hosted by fellow PhD candidate Brian Connolly. The performer found the design both expressive and easy-to-operate and the performance proceeded without any problems.

The most surprising aspect of playing with *DroneTilt* was the expressive power of the accelerometer data. In most interactive music applications this information is normally employed in a very simple way – either to detect discrete 'shaking' type movements or to change menu orientation. Allocating a primary performance role to the motion sensors is impossible in most cases due to the predominance of the graphical interface and/or the need for the user employ both of their hands during performance – leaving no way to move the device itself. The lack of visual emphasis required to operate this particular design allowed for a great deal of freedom to change the position of the device while carefully monitoring the resulting change in drone texture.

Brief timbral fluctuations caused by sudden movements also provided an interesting, if unpredictable, contribution to the performance – a quick jerking of the device in space results in a momentary teleportation of values and a jarring glimpse

into the timbre present at another orientation. It was observed that jerking the device in different directions produced markedly different results – especially in relation to up/down movements which provided a way to foreshadow the arrival of the final sample.

The posture used throughout the performance, with one-hand suspending the device like a clipboard, was comfortable and not as limiting as expected. It also prompted an investigation into ways to suspend the iPad during performance so as to enable the use of the accelerometer data without disabling one or more of the performer's hands, which will be explored in future performances.

6.13 Linking Oscar to the descriptive model

Oscar is a configurable platform with a gestural interface that is particularly well-suited to designing performance tools using the descriptive model established in Chapter 4. Many of *Oscar*'s core components can be used to construct instances of the interaction strategies outlined in sections 4.4 – 4.9, as demonstrated in the following examples:

- The *Touch* strategy is easily used – every touch and/or cluster creates a unique instance of itself (as a Csound instrument) which can be populated with sound generating code. *Repeat* and *contact* behaviours are easily performed in the same way.
- *Trigger* behaviour can be easily invoked by sending a statement to the Csound score using the **`event`** opcode.
- *Toggles* can be implemented as above, with the addition of a boolean flag variable, or in response to *Oscar*'s tap or flick gestures.

- *Counter* can be implemented with the use of a global variable holding an integer.
- *Time since last action, hold time, idle* and *excitation* strategies can all be performed by creating a counting behaviour either within touch/cluster instruments or by defining a separate, dedicated counter instrument that updates a global variable. Other instruments can access this number to affect their parameters and the *average/median time* strategy can also be derived from it.
- *One-to-one* mapping is easily to perform by plugging the properties of *Oscar*'s gestures into sound generating parameters. *Movement* can be detected by keeping a record of the last known value and performing comparisons with the current value or (as *velocity* values are provided directly in the case of clusters and touches) simply setting a *threshold* of speed above which action is taken.
- *Thresholds* are also built-in to *Oscar*'s interface in the form of the zones. Conditional zone checks (as shown in Figure 6.21) can be used quickly and effectively to reserve sections of *Oscar*'s surface for the control of specific tasks.
- *Modal* behaviours are strongly supported by the variety of independent variables available to the user. The number of clusters, number of touches within a cluster, starting location, current zone, 'is held' property and the device orientation are all well-suited to delimiting various modes of operation depending on the needs of the user.
- *Automation* strategies of varying complexity are easily created and accessed via Csound's vast library of opcodes for storing information in tables. Pre-

arranged scores or event lists can be copied directly into the *Oscar* template and read without any additional configuration.

- *Strategies for combining controllers* are well-supported between the various gestures within *Oscar* itself and also the ability to use external MIDI devices via USB. MIDI signals from a connected device can be read directly into an *Oscar* template and combined with the built-in gesture data to control performance.

6.14 Assessment of Oscar

While formal assessment of *Oscar*'s suitability as a multi-touch music performance system is beyond the scope of this thesis, anecdotal evidence suggests that it has the potential to be a useful development platform:

- Beta-testing of the app has gathered very positive feedback from subjects with a self-described knowledge of computer music tools ranging from 'expert' to 'none'.
- An undocumented performance program was demonstrated successfully by Dr. Victor Lazzarini at the 2013 UbiMus workshop. Participants were easily able to figure out how the gestures corresponded to the sonic results.
- An *Oscar* workshop was given to students of the MA in Computer Music at NUI Maynooth – over the course of two hours, students with some Csound training and no prior experience of *Oscar* were able to create fully-functional programs to control synthesized sounds and mix together pre-recorded samples. Several students remarked that the session helped their understanding of signal flow within Csound and suggested that having access

to a tool like *Oscar* would be very helpful when learning synthesis and music programming languages.

Oscar is still being developed and refined. In the near future, a team of sound designers and programmers will be contracted to develop interesting sound programs for *Oscar* and explore its abilities for various genres of music. Further academic research is also planned with the Ubiquitous Music Group – performing experiments on usability and expression with *Oscar* as the test platform.

6.15 Conclusion

An overview of *Oscar* has been given in this chapter, followed by a detailed description of the design goals and the steps taken in order to achieve them. This is complemented by a breakdown of the template that *Oscar* uses to communicate with the Csound API and an example of a performance tool developed using the system.

The concepts embodied in *Oscar* represent an alternative way of designing a digital musical instrument, inspired by the interaction strategies established in earlier chapters. *Oscar* demonstrates the viability of dissecting a gestural controller into its composite parts and reassembling them using the descriptive model outlined in this thesis. This model is not limited to touch screen devices – the versatility and modular nature of this approach makes it easy to apply to any kind of human-computer interface device. The strategies integral to *Oscar* could, with little modification, be ported to devices that detect hand and finger movement in a different way, such as the contactless tracking offered by the Leap [97]. The cluster abstraction could be repurposed in this case, to accommodate the Leap's system of tracking finger/hand relationships, possibly causing a number of subtle changes in how the model works.

As a standalone tool, *Oscar* represents a novel and versatile means to develop interactive audio software on a tablet device. It occupies a unique middle-ground

between controller and synthesizer, and also allows users to develop their own standalone programs without the need for a Mac, a developer account, or knowledge of any programming language beyond Csound. *Oscar* has become an invaluable part of the author's own creative process – it is hoped that its commercial release will inspire a wide variety of musicians and artists to express their musical ideas in a new way.

Chapter 7. Conclusion

"The goal we seek is nothing less than the free expression of our imaginations. No one else should decide for us the best way to get there. All hardware and software intended for musical use should be designed with that in mind."

-Simon Emmerson, The Language of Electroacoustic Music [45]

This thesis set out to enhance the existing vocabulary of the digital musician by establishing new conceptual tools for musical interaction design. An account of influential developments in the field of live electronic music performance, followed by a comprehensive survey of existing theory and practice, served to contextualise the work and introduce the core concepts of digital musical instrument design. The thesis then presented a system of modular interaction strategies and defined a variety of complementary techniques for augmenting their functionality. This section of the work concluded with a case study that used the strategies to approach a typical performance technique: live sample playback and manipulation.

Having established a theoretical framework, the thesis focused in upon a particular kind of device – the multi-touch surface – in order to investigate its musical potential from a new perspective. A survey of multi-touch musical applications, followed by some preliminary experiments with alternative interaction techniques, led to the development of *Oscar* – a novel system for musical performance using multi-touch. The discussion following this section consisted of a detailed description of the design philosophy, gestural interface, feature

implementation and user template of *Oscar* and concluded with a final case study that demonstrates the system in action.

In terms of original contributions to the field of study, three main aspects of this thesis should be considered: (i) a critical assessment of recent trends in digital musical instrument design, (ii) a descriptive model for digital musical instrument design using modular interaction strategies, and (iii) a novel, customisable, integrated platform for the development of multi-touch music performance systems.

The critical assessment of digital musical instrument design has its basis in a number of design aphorisms inspired by the design trends of the last century (discussed in section 2.11). While it is difficult to predict the direction that future developments will take, an understanding of the dynamic developmental history of controllers for electronic music is vital in order to appreciate, and contribute to, the current state of the art. Chapter 5 builds upon this understanding to present a critical comparison of the design techniques common to popular contemporary hardware controllers (i.e. button grids, XY pads and multi-touch screens).

The descriptive model was developed in Chapter 4, which summarised basic programming techniques, and discussed various data-handling strategies, predictive and descriptive models, describing sensors in terms of degrees of freedom (DOF), dimensions and resolution, and a series of fundamental interaction strategies for 1DOF sensors with high and low resolutions.

We then discussed the application of the modular approach and provided strategies for using controllers independently and interdependently, with a summary of techniques to combine the data from several different input sensors. Chapter 4 also proposed a complementary toolkit of abstract controllers that employ statistics, multi-modal behaviour, automation and saving/recalling of settings to augment the

functionality of the physical sensors present in the system. These findings are supported by a thorough literature review that documents the evolution of controllers for the performance of live electronic music (Chapter 2) and provides a comprehensive summary of conceptual tools that have been developed to assist in the design and classification of digital musical instruments (Chapter 3). The practical application of the descriptive model was illustrated via a detailed account of the design of a live performance sampler interface – *LoopBlender*.

The development of a novel multi-touch performance platform was informed by a detailed study of surface-based interfaces, encompassing contemporary musical applications of XY pads, grid-based interfaces and multi-touch surfaces (Chapter 5). This survey examined popular approaches to musical interface design using multi-touch devices and proposed an alternative control strategy that aims to leverage the intrinsic strengths of multi-touch technology for expressive and nuanced musical control. This strategy was investigated by devising and refining a series of gesture recognition algorithms – described herein as *SurfacePlayer.*

Both the descriptive model and these algorithms were used to create a standalone platform for the design of expressive multi-touch performance systems – *Oscar.* In Chapter 6, we discussed the goals that motivated the creation of *Oscar* and described their realisation in terms of the application's features and how they were implemented. In particular, we examined closely the core components of the gestural interface (clusters, touches, taps and flicks), the template for designing new performance programs, and the various strategies that can be employed to combine the motion data generated by the iPad's built-in sensors with multi-touch gestures performed by the user. The chapter concluded with a second design case study – *DroneTilt* – which combines the interaction strategies from our descriptive model

with some of *Oscar*'s unique features to create an expressive live performance instrument.

The work presented in this thesis has been motivated by a dedication to the development of practical and universally-applicable tools that encourage more expressive, radical and intuitive digital musical instrument designs. The descriptive model is both practical and generic – the interaction strategies and concepts at its core can be used with any kind of sensor or interface that generates digital data. It is also not a closed system – there is ample room for designers to discover and contribute new strategies within the prescribed framework.

Oscar is a unique addition to the selection of multi-touch software designed for serious musicians. It is a fully-customisable, integrated platform that anyone can develop programs for, with a basic knowledge of the Csound language. The software occupies a unique space with regard to the level of flexibility it provides and the accessibility of writing new code for it – the end user can design new programs for *Oscar,* and modify existing programs, using nothing but a basic text editor and Dropbox account. Aside from this specific implementation, the gestural interface at the core of *Oscar* can be easily adapted to other devices that use manual, open-handed control, such as the Leap Motion [97].

The electronic music community's growing interest in ergonomic, intuitive and flexible control devices is evidence of an evolution in our collective approach to technology in performance. Our emphasis is moving away from the powerful equipment at our disposal towards the development of powerful musical interactions using that equipment. It is my hope that this thesis will make a significant contribution to our understanding of interactive systems, multi-touch controllers, and the unique art that is the design of digital musical instruments.

Bibliography

[1] https://www.ableton.com/, accessed April 2014

[2] http://www.ableton.com/launchpad, accessed April 2014

[3] http://acousmata.com/post/27443169341/jorg-mager, accessed April 2014

[4] http://www.airpiano.de/concept, accessed April 2014

[5] http://akustisch.digitalaspekte.ch/, accessed April 2014

[6] http://www.altkeyboards.com/instruments/jammer, accessed April 2014

[7] http://www.altkeyboards.com/instruments/sonomes, accessed April 2014

[8] http://www.apple.com/ipad/, accessed April 2014

[9] https://developer.apple.com/technologies/ios/cocoa-touch.html, accessed April 2014

[10] Bakan, M., Bryant, W., Li, G., Martinelli, D. and Vaughnm, K. Demystifying and Classifying Electronic Music Instruments, Selected Reports in Ethnomusicology 8: 37–63, 1990

[11] http://baranoff-rossine.com/optophonic-piano/, accessed April 2014

[12] Barbosa, J., Calegario, F., Magalhães, F., Teichrieb, V., Ramalho, G. and Cabral, G. Towards an evaluation methodology for digital music instruments considering performer's view: a case study, SBCM, 2011

[13] Barbosa, J., Calegario, F., McGlynn, P., Teichrieb, V., Ramalho, G. Considering Audience's View Towards an Evaluation Methodology for Digital Musical Instruments, NIME, 2012

[14] Berman, B. Notes from the Pianist's Bench, Yale. University Press, New Haven and London, 2000

[15] Bevilacqua, F., Muller, R. and Schnell, N. MnM: aMax/MSP mapping toolbox, NIME, 2005

[16] Birnbaum, D., Fiebrink, R., Malloch, J. and Wanderley, M. M. Towards a dimension space for musical devices, NIME, 2005

[17] Boon, M. The Drone. In Young, R. (Ed.), Undercurrents: The Hidden Wiring of Modern Music, Continuum Books, 2003

[18] http://www.buchla.com/historical/, accessed April 2014

[19] http://www.buchla.com/historical/thunder/index.html, accessed April 2014

[20] http://www.buchla.com/lightning3.html, accessed April 2014

[21] Buxton, W. Three-state model of graphical input, INTERACT, 1990

[22] Buxton, W. Integrating the Periphery and Context: A New Model of Telematics**,** Proceedings of Graphics Interface'95, 239-246, 1995

[23] http://www.billbuxton.com/papers.html#anchor1442822, accessed April 2014

[24] Cadoz, C. and Wanderley, M. M. Gesture-Music. From Trends in Gestural Control of Music. Paris: IRCAM (PDF version), 2000

[25] Casciato, C. and Wanderley, M. M. Lessons from Long Term Gestural Controller Users, ENACTIVE, 2007

[26] Chadabe, J. Electric Sound: The Past and Promise of Electronic Music, Prentice Hall, New Jersey, 1997

[27] http://charlie-roberts.com/Control/, accessed April 2014

[28] http://www.controllerism.com/types-of-controllerism, accessed April 2014

[29] Cook, P. Principles for designing computer music controllers, NIME, 2001

[30] Collins, N., Schedel, M. and Wilson, S. Cambridge Introductions to Music: Electronic Music, Cambridge University Press, 2013

[31] Cooper, A. and Reimann, R. About Face 2.0: The essentials of interaction design, Wiley, 2003

[32] Crab, S. 120 Years of Electronic Music, http://www.mathieubosi.com/zikprojects/120YearsOfElectronicMusic.pdf, 2004

[33] http://www.csounds.com/, accessed April 2014

[34] http://csound.sourceforge.net/doc/html/index.html, accessed April 2014

[35] http://www.csounds.com/manual/html/CommandUnifile.html, accessed April 2014

[36] http://www.c-thru-music.com/cgi/?page=prod_axis-64, accessed April 2014

[37] http://www.davidrokeby.com/vns.html, accessed April 2014

[38] Davis, T. Towards a Relational Understanding of the Performance Ecosystem, Organised Sound, 16, 2011

[39] Djajadiningrat, J.P., Matthews, B. and Stienstra, M. Easy doesn't do it: skill and expression in tangible dynamics, Personal and Ubiquitous Computing, vol. 11, no. 8, pp. 657-676, 2007

[40] Doornbusch, P. Composers' views on mapping in algorithmic composition, Organised Sound, 7(2), 145 – 156, 2002

[41] https://www.dropbox.com/, accessed April 2014

[42] Duignam, M. A Taxonomy of Sequencer User-Interfaces, ICMC, 2005

[43] Dunn, D. A History of Electronic Music Pioneers, Eigenwelt der Apparate-Welt.(Katalog), Linz, 1992

[44] http://www.eigenlabs.com/, accessed April 2014

[45] Emmerson, S. Living Electronic Music, Ashgate Publishing Ltd., 2007

[46] Essl, G., Rohs, M. and Kratz, S. Use the Force (or something) - Pressure and Pressure-Like Input for Mobile Music Performance, NIME, 2010

[47] https://emhistory.wikispaces.com/1876+Musical+Telegraph, accessed April 2014

[48] http://expressionpad.com/, accessed April 2014

[49] http://faderfox.de/home.html, accessed April 2014

[50] Farrell, J. Just Enough Programming Logic and Design. Course Technology, 2010

[51] Ferguson, S. and Wanderley, M. M. The McGill Digital Orchestra: Interdisciplinarity in Digital musical Instrument Design, CIM, 2009

[52] Fitz-Walter, Z., Jones, S., and Tjondronegoro, D. Detecting gesture force peaks for intuitive interaction, IE, 2008

[53] Flanagan, J. L. Speech Analysis, Synthesis and Perception, Springer-Verlag, 1965

[54] Frisson, C., Macq, B., Dupont, S., Siebert, X., Tardieu, D. and Dutoit, T. DeviceCycle: rapid and reusable prototyping of gestural interfaces, applied to audio browsing by similarity, NIME, 2010

[55] https://sites.google.com/site/carlkenner/glovepie, accessed April 2014

[56] Gruenbaum, L. The Samchillian Tip Tip Tip Cheeepeeeee: A Relativistic Keyboard Instrument, NIME, 2007

[57] Gurevich, M. and Fyans, A. Digital Musical Interactions: Performer–system relationships and their perception by spectators. Organised Sound, 16(02), pp.166-175, 2011

[58] http://www.hakenaudio.com/Continuum/, accessed April 2014

[59] Hinckley, K., Pierce, J., Sinclair, M., and Horvitz, E. Sensing techniques for mobile interaction, UIST, 2000

[60] Hinckley, K., and Song, H., Sensor Synaesthesia: Touch in Motion, and Motion in Touch, CHI, 2011

[61] Hinckley, K. Manual deskterity: Combining pen and touch, http://kenhinckley.wordpress.com/2011/08/02/classic-manual-deskterit/, 2011, accessed April 2014

[62] Hinckley, K. The hidden dimension of touch, http://kenhinckley.wordpress.com/2011/09/07/classic-post-the-hidden-dimension-of-touch/, 2011, accessed April 2014

[63] Holms, T. Electronic and Experimental Music: Technology, Music and Culture, Routledge, 2008

[64] von Hornbostel, E. M. and Sachs, C. Classification of Musical Instruments: Translated from the Original German by Anthony Baines and Klaus P. Wachsmann, The Galpin Society Journal, Vol. 14, pp. 3-29, 1961

[65] Hsu, W. and Sosnick, M. Evaluating interactive music systems: An HCI approach, NIME, 2009

[66] http://www.hughlecaine.com/en/sackbut.html, accessed April 2014

[67] Hugill, A. The Digital Musician, Routledge, 2008

[68] Hummels, C., Smets, G. and Overbeeke, K. An intuitive two handed gestural interface, Journal of KISS, 1997

[69] Hunt, A. Radical User Interfaces for Real-time Musical Control, PhD thesis, University of York, 1999

[70] Hunt, A., Wanderley, M. M. and Paradis, M. The Importance of Parameter Mapping in Electronic Instrument Design, NIME, 2002

[71] http://direct.journalofmusic.com/listing/04-04-13/interfaces-and-psychoacoustics, accessed April 2014

[72] Ishii, H. The tangible user interface and its evolution, Communications of the ACM: Organic user interfaces, Volume 51(6):32-36, 2008

[73] Jacob, R., Sibert, L., McFarlane, D. and Mullen Jr., M. P. Integrality and separability of input devices, ACM Trans. Human Computer Interaction, vol. 1, no. 1, pp.3–26, 1994

[74] http://www.jazzmutant.com/lemur_overview.php, accessed April 2014

[75] Jenkins, M. Analog Synthesizers: Understanding, performing, buying, Focal Press, 2007

[76] Jorda, S. Digital instruments and players: Part I - Efficiency and Apprenticeship, NIME, 2004

[77] Jorda, S., Kaltenbrunner, M., Geiger, G. and Bencina, R. The Reactable*, ICMC, 2005

[78] Jorda, S. Digital Lutherie, PhD thesis, Universitat Pompeu Fabra, 2005

[79] Kaltenbrunner, M., Bovermann, T., Bencina, R. and Costanza, E., TUIO - A Protocol for Table-Top Tangible User Interfaces, GW, 2005

[80] Kaltenbrunner, M. Processing TUIO, http://www.tuio.org/?processing, accessed April 2014

[81] Kartomi, M. The Classification of Musical Instruments: Changing Trends in Research from the Late Nineteenth Century, with Special Reference to the 1990s, Ethnomusicology, Vol. 45, No. 2, pp. 283-314, 2001

[82] http://www.kdj-one.com/, accessed April 2014

[83] http://www.keithmcmillen.com/QuNeo/tech-specs/, accessed April 2014

[84] http://kck.st/taqBsn, accessed April 2014

[85] Keller, D., Barreiro, D. L., Queiroz, M. and Pimenta, M. S. Anchoring in Ubiquitous Musical Activities, ICMC, 2010

[86] Keller, D., Flores, L. V., Pimenta, M. S., Capasso, A. and Tinajero, P. Convergent Trends Toward Ubiquitous Music, Journal of New Music Research 40 (3): 265-276, 2011

[87] Kernfeld, B. What to Listen For in Jazz, Yale University Press, 1997

[88] Kirn, P. Why DIY Music? Reflections from STEIM's Patterns & Pleasure Fest, Handmade Music Amsterdam, http://createdigitalmusic.com/2011/09/why-diy-music-reflections-from-steims-patterns-and-pleasure-fest-handmade-music-amsterdam/, 2011, accessed April 2014

[89] Kirn, P. Reinventing the Whee: Engineering arc2 – Digital Instrument from Monome Creator, http://createdigitalmusic.com/2012/10/reinventing-the-wheel-engineering-arc2-digital-instrument-from-monome-creator-gallery-interview/, 2012, accessed April 2014

[90] http://www.korg.com/ielectribe, accessed April 2014

[91] http://korg.com/products.aspx?ct=4, accessed April 2014

[92] http://www.korg.com/uploads/Support/KAOSSILATOR_PRO_OM_EFG1_634067737513300000.pdf, accessed April 2014

[93] http://www.korg.com/uploads/Support/KP3_OM_EFG1_633659261667720000.pdf, accessed April 2014

[94] http://www.korg.com/nanoseries2, accessed April 2014

[95] Kurtenbach, G. and Hulteen, E. Gestures in human-computer communication. The Art of Human-Computer Interface Design, Addison-Wesley Publishing Co., 1990

[96] Lähdeoja, O., Wanderley, M. M. and Malloch, J. Instrument Augmentation using Ancillary Gestures for Subtle Sonic Effects, SMC, 2009

[97] https://www.leapmotion.com/, accessed April 2014

[98] Leganchuk, A., Zhai, S. and Buxton, W. Manual and Cognitive Benefits of Two-Handed Input: An Experimental Study. Transactions on Human-Computer Interaction, 5(4), 326-359, 1998.

[99] http://liine.net/en/products/lemur/, accessed April 2014

[100] http://line6.com/stagescape-m20d/, accessed April 2014

[101] Lockwood, D. Inspired Instruments – You Rock Guitar, Sound on Sound, August 2011

[102] MacKenzie, I. S. Fitts' law as a performance model in human-computer interaction. Unpublished Doctoral Dissertation, University of Toronto, 1991

[103] MacKenzie, I. S. Motor behaviour models for human-computer interaction. In J. M. Carroll (Ed.) HCI models, theories, and frameworks: Toward a multidisciplinary science, pp. 27-54. San Francisco: Morgan Kaufmann, 2003

[104] http://www.mackie.com/products/dl1608/, accessed April 2014

[105] http://madronalabs.com/hardware, accessed April 2014

[106] Magnusson, T. The Acoustic, the Digital and the Body: A Survey on Musical Instruments, NIME, 2007

[107] Malloch, J., Birnbaum, D., Sinyor, E. and Wanderley, M. M. Towards a new conceptual framework for digital musical instruments, DAFx, 2006

[108] Malloch, J. and Wanderley, M. M. The T-Stick: From Musical Interface to Musical Instrument, NIME 2007

[109] Manning, P. Electronic and Computer Music, Oxford University Press, 2013

[110] Marshall, M. and Wanderley, M. M. Evaluation of Sensors as Input Devices for Computer Music Interfaces, CMMR, 2006

[111] McMillan, K. ZIPI: Origins and Motivations, Computer Music Journal, vol. 18, no. 4, pp. 48–96, 1994

[112] Mathieu, W.A. This Musical Life: Reflections on what it is and how to live it, Shambhala, 1994

[113] Matthews, M. V. Radio-Baton Instruction Manual. MARMAX, 2000

[114] Maupin, S., Gerhard, D. and Park, B. Isomorphic Tessellations for Musical keyboards. SMC, 2011.

[115] http://www.maxforlive.com/library/device.php?id=534, accessed April 2014

[116] McCartney, J. Continued evolution of the SuperCollider real time synthesis environment, ICMC, 1998

[117] McGlynn, P. Towards more effective mapping strategies for digital musical instruments, LAC, 2011

[118] McGlynn, P., Lazzarini, V., Delap, G., and Chen, X., Recontexualizing the Multi-touch Surface, NIME, 2012

[119] http://www.midi.org/techspecs/, accessed April 2014

[120] Miranda, E.R. and Wanderley, M. M. New Digital Musical Instruments: Control and Interaction Beyond the Keyboard, A-R Editions Inc., 2006

[121] http://www.misadigital.com/, accessed April 2014

[122] Miyama, C. Peacock: A Non-Haptic 3D Performance Interface, NIME 2010

[123] http://www.moldover.com/press/Moldover_Remix_Oct-2007_w.jpg, accessed April 2014

[124] http://monome.org/, accessed April 2014

[125] http://docs.monome.org/doku.php?id=app:straw, accessed April 2014

[126] http://mrmr.noisepages.com/, accessed April 2014

[127] Mulder, A. Towards a choice of gestural constraints for instrumental performers, From Trends in Gestural Control of Music. Paris: IRCAM (PDF version), 2000

[128] Nadolski, K. Listening to Our Bodies: How Pianists Can Create Unnecessary Difficulties Through Excessively Rule-Bound Approaches to Legato and Fingering, PhD Thesis, Texas Tech University, 2012

[129] Nagle, P. Dewanatron Swarmatron, Sound on Sound, June 2011

[130] http://www.native-instruments.com/en/products/maschine/, accessed April 2014

[131] http://www.nintendo.com/wii/console/controllers, accessed April 2014

[132] Norman, D. A. The Design of Future Things. Basic Books, New York, 2007

[133] http://ccv.nuigroup.com/, accessed April 2014

[134] http://nuigroup.com/forums/, accessed April 2014

[135] http://www.numark.com/product/orbit, accessed April 2014

[136] Nunn, T. The Wisdom of the Impulse: On the nature of musical free improvisation, http://www20.brinkster.com/improarchive/tn_wisdom_part1.pdf, 2008, accessed April 2014

[137] Oliver, J. and Jenkins, M. The Silent Drum Controller: A New Percussive Gestural Interface, ICMC 2008

[138] http://www.spectrasonics.net/products/omni_tr.php, accessed April 2014

[139] O'Modhrain, S. Playing by Feel: Incorporating Haptic Feedback into Computer-Based musical Instruments, Ph.D. Thesis, Stanford University, 2000

[140] O'Modhrain, S. A framework for the evaluation of digital musical instruments, Computer Music Journal, vol. 35, no. 1, pp. 28–42, 2011

[141] http://opensoundcontrol.org/, accessed April 2014

[142] O'Sullivan, D. and Igoe, T. Physical Computing: Sensing and controlling the physical world with computers, Thomson Couse Technology, 2004

[143] http://www.patchmanmusic.com/manuals/DH100Manual.pdf, accessed April 2014

[144] Pimenta, M.S., Flores, L., V., Capasso, A., Tinajero, P. and Keller, D. Ubiquitous Music: Concepts and Metaphors, SBCM 2009

[145] Prendergast, M. J. The Ambient Century: From Mahler to Trance – The Evolution of Sound in the Electronic Age, Bloomsbury Publishing PLC, 2000

[146] http://processing.org/, accessed April 2014

[147] Puckette, M. Max at seventeen, Computer Music Journal 26(4), 31–43, 2002

[148] http://qutecsound.sourceforge.net/index.html, accessed April 2014

[149] Raskin, J. The humane interface: New directions for designing interactive systems, Addison Wesley, 2000.

[150] http://www.reactable.com/, accessed April 2014

[151] http://reactivision.sourceforge.net/, accessed April 2014

[152] http://www.researchcatalogue.net/view/48123/53020, accessed April 2014

[153] Roads, C. The Computer Music Tutorial, MIT Press, 1996

[154] https://github.com/rfielding/Mugician, accessed April 2014

[155] http://rrr00bb.blogspot.com/2010/08/mugician-heiroglyphics.html, accessed April 2014

[156] http://www.rogerlinndesign.com/preview-linnstrument.html, accessed April 2014

[157] Rokeby, D. The Construction of Experience : Interface as Content, in Dodsworth Jr., C. (Ed.), Digital Illusion: Entertaining the Future with High Technology, ACM Press, 1998

[158] Russ, M. Sound Synthesis and Sampling, Focal Press, 2009

[159] Saffer, D. Designing Gestural Interfaces: Touchscreens and Interactive Devices. O'Reilly Media, Sebastopol, CA, 2009

[160] Samagaio, F. The Mellotron Book, artistpro.com LLC, 2002

[161] Schlei, K. Relationship-Based Instrument Mapping of Multi-Point Data Streams Using a Trackpad Interface, NIME, 2010

[162] Schlömer, T., Poppinga, B., Henze, N. and Boll, S. Gesture recognition with a Wii controller, TEI, 2008

[163] http://www.sensorband.com/, accessed April 2014

[164] Shapiro, P. Turn the Beat Around: The Secret History of Disco, Faber & Faber, 2006

[165] Shepard, B. K. Refining Sound: A Practical Guide to Synthesis and Synthesizers, OUP USA, 2013

[166] Shiffman, D. Learning Processing: A beginner's guide to programming images, animation, and iteration, Morgan Kaufman Publishers, 2008

[167] http://www.sibelius.com/, accessed April 2014

[168] Sinclair, S. A guitar-inspired touch pad controller, http://www.music.mcgill.ca/~sinclair/touchpad-guitar_sinclair.pdf, accessed April 2014

[169] http://www.smithsonmartin.com/kontrol-surface-ks-1974/, accessed April 2014

[170] http://www.sonami.net/works/ladys-glove/, accessed April 2014

[171] http://www.sourceaudio.net/products/hothand/, accessed April 2014

[172] Spiegel, L. Music Mouse manual, http://retiary.org/ls/progs/mm_manual/mouse_manual.html, 1986, accessed April 2014

[173] http://www.starrlabs.com/, accessed April 2014

[174] Steiner, J. C. [hid] toolkit: a unified framework for instrument design, NIME, 2005

[175] http://www.stephenhobley.com/blog/laser-harp-2009/the-laser-harp-pages/, accessed April 2014

[176] Stretta, Making music is process, http://stretta.blogspot.com/2011/07/making-music-is-process.html#disqus_thread, 2011, accessed April 2014

[177] Stretta, Plane – m | vi | cv, http://stretta.blogspot.com/2011/05/plane-m-vi-cv.html, 2011, accessed April 2014

[178] http://www.subcycle.org/, accessed April 2014

[179] Tanaka, A. Musical performance practice on/with sensor based instruments. From Trends in Gestural Control of Music. Paris: IRCAM (PDF version), 2000

[180] http://www.bitshapesoftware.com/instruments/tc-11/ , accessed April 2014

[181] Teiche, A., Rai, A., Yanc, C., Moore, C., Solms, D., Cetin, G., Riggio, J., Ramseyer, N., D'Intino, P., Muller, L., et al. Multi-touch technologies. NUI Group, 2009

[182] http://www.global.yamaha.com/tenori-on/, accessed April 2014

[183] http://download.yamaha.com/api/asset/file/?language=en&site=au.yamaha.com&asset_id=13041, accessed April 2014

[184] Textura. A History of Dronology, http://www.textura.org/archives/articles/dronesarticle.htm, accessed April 2014

[185] Toop, D. Haunted Weather: Music, Silence and Memory, Serpent's Tale, 2005.

[186] http://hexler.net/software/touchosc, accessed April 2014

[187] http://www.tuio.org/, accessed April 2014

[188] https://code.google.com/p/tuiopad/, accessed April 2014

[189] https://code.google.com/p/tuiodroid/, accessed April 2014

[190] http://compmus.ime.usp.br/ubimus/en, accessed April 2014

[191] http://www.uusikaupunki.fi/~patalus/new_stuff/Dynacord%20Rhythm%20Stick/Dynacord%20Rhythm%20Stick%20manual.pdf, accessed April 2014

[192] Vail, M. Vintage Synthesizers: Groundbreaking Instruments and Pioneering Designers of Electronic Music Synthesizers, Backbeat Books, 2000

[193] http://www.neyrinck.com/en/products/v-control-pro, accessed April 2014

[194] Verfaille, V., Wanderley, M. M. and Depalle, P. Mapping strategies for gestural and adaptive control of digital audio effects, Journal of New Music Research 35(1):71–93, 2006

[195] Vigliensoni, G. and Wanderley, M.M. Soundcatcher: Explorations in Audio-Looping and Time-Freezing using an Open-Air Gestural Controller, ICMC 2010

[196] http://vimeo.com/29517018, accessed April 2014

[197] http://vimeo.com/30976072, accessed April 2014

[198] http://vimeo.com/1338613, accessed April 2014

[199] http://www.vintagesynth.com/casio/vl1.php, accessed April 2014

[200] Waisvisz, M. The Hands: A set of remote MIDI controllers, ICMC, 1985

[201] Wait, B. Guitar Synth and MIDI, H. Leonard Books, 1988

[202] Wanderley, M. M. and Orio, N. Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI. Computer Music Journal, 26(3):62-76, 2002

[203] Wanderley, M.M. and Depalle, P. Gestural control of sound synthesis. Proceedings of the IEEE, Special Issue on Engineering and Music—Supervisory Control and Auditory Communication, 92(4), 2004

[204] Wang, G. and Cook, P. On-the-fly programming: Using code as an expressive musical instrument, NIME, 2004

[205] Weisser, S. and Quanten, M. Rethinking Musical Instrument Classification: Towards a Modular Approach to the Hornbostel-Sachs System**,** Yearbook for traditional music , Vol. 43, pp. 122-146, 2011

[206] Wessel, D. An enactive approach to computer music performance. In Orlarey, Y. (ed.) Le Feedback dans la Creation Musical, pp. 93–98, Lyon, France: Studio Gramme, 2006

[207] http://cnmat.berkeley.edu/user/david_wessel/blog, accessed April 2014

[208] http://www.wiigee.org/index.html, accessed April 2014

[209] Willoughby, G. Purebasic: A beginner's guide to computer programming. http://www.purearea.net/pb/download/PureBasicBook.pdf, 2006, accessed April 2014

[210] Wobbrock, J.O., Wilson, A.D. and Li, Y. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. UIST, 2007

[211] Wong, E., Yuen, W. and Choy, C. Designing Wii Controller - A Powerful Musical Instrument In An Interactive Music Performance System, MoMM, 2008

[212] http://www.xbox.com/kinect/, accessed April 2014

[213] Xia, X., Irani, P. and Wang, J. Evaluation of Guiard's Theory of Bimanual Control for Navigation and Selection, EHAWC, 2007

[214] http://www.youtube.com/watch?v=1t3fk4QPlD4, accessed April 2014

[215] http://youtu.be/yw3Gs3tx8xo, accessed April 2014

[216] http://www.youtube.com/watch?v=apqF40rf5FY, accessed April 2014

[217] http://youtu.be/umsO-KLjRX8, accessed April 2014

[218] http://youtu.be/WyNDWZhYZ3U, accessed April 2014

[219] http://youtu.be/CYv5jqHMe5c, accessed April 2014

[220] http://www.youtube.com/watch?v=aulbMkOLqKg, accessed April 2014

[221] http://www.youtube.com/watch?v=PwdWfPxpq8g, accessed April 2014

[222] http://www.youtube.com/watch?v=WVDw7uPnfa8, accessed April 2014

[223] Zadel, M. and Scavone, G. Laptop Performance: Techniques, Tools, and a New Interface Design, ICMC, 2006

[224] Zbyszynski, M., Wright, M., Momeni, A. and Cullen, D. Ten years of tablet musical interfaces at CNMAT, NIME, 2007

[225] Zbyszynski, M. An Elementary Method for Tablet, NIME, 2008

[226] http://www.zendrum.com/, accessed April 2014

[227] http://cycling74.com/products/max/, accessed April 2014

[228] http://www.nime.org/, accessed April 2014

[229] http://puredata.info/, accessed April 2014

[230] http://scratchpad.wikia.com/wiki/P5_Glove:Musical, accessed April 2014

[231] http://supercollider.sourceforge.net/, accessed April 2014

[232] Rasmussen, J. Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering, Elsevier Science Inc., New York, NY, USA, 1986.

[233] Wanderley, M. Performer-Instrument Interaction: Applications to Gestural Control of Sound Synthesis, Ph.D. Thesis, Université Pierre et Marie Curie, Paris VI, 2001.

Appendix A: Oscar program template

```
<CsoundSynthesizer>

/*
Oscar program template
1st of April 2014
*/
<CsOptions>
-odac -dm0 -+rtmidi=null -+rtaudio=null -+msg_color=0 -
M0
</CsOptions>

<CsInstruments>

sr        = 44100
ksmps     = 32
nchnls    = 2
0dbfs     = 1

/* GLOBAL SETUP */

; Resource path
gSresourcePath chnexport "resourcePath", 1

; Accelerometer variables
gkaccX    init 0
gkaccY    init 0
gkaccZ    init 0

; Global reverb channel
gareverbL init 0
gareverbR init 0

; Master output channel
gamasterL init 0
gamasterR init 0

; UDO for Touch events
opcode Touch, iiiiiikkkkk, iiiiii p4, p5, p6, p7, p8, p9
xin
```

```
itouchID = p4

; Dynamically-generated channel names
S_x       sprintf "touch.%d.x", itouchID
S_y       sprintf "touch.%d.y", itouchID
S_zone    sprintf "touch.%d.zone", itouchID
S_zoneX   sprintf "touch.%d.zoneX", itouchID
S_zoneY   sprintf "touch.%d.zoneY", itouchID

; K-rate variables for touch
kx        chnget S_x
ky        chnget S_y
kzone     chnget S_zone
kzoneX    chnget S_zoneX
kzoneY    chnget S_zoneY
xout p4, p5, p6, p7, p8, p9, kx, ky, kzone, kzoneX,
kzoneY
endop

; UDO for Cluster events
opcode Cluster, iiiiiiiikkkkkkkkkk, iiiiiii p4, p5, p6,
p7, p8, p9, p10 xin

iclusterID = p4

; Dynamically-generated channel names
S_x          sprintf "cluster.%d.x", iclusterID
S_y          sprintf "cluster.%d.y", iclusterID
S_zone       sprintf "cluster.%d.zone", iclusterID
S_zoneX      sprintf "cluster.%d.zoneX", iclusterID
S_zoneY      sprintf "cluster.%d.zoneY", iclusterID
S_numTouches sprintf "cluster.%d.numTouches", iclusterID
S_size       sprintf "cluster.%d.size", iclusterID
S_direction  sprintf "cluster.%d.direction", iclusterID
S_velocity   sprintf "cluster.%d.velocity", iclusterID
S_isHeld     sprintf "cluster.%d.isHeld", iclusterID
S_isZooming  sprintf "cluster.%d.isZooming", iclusterID
; K-rate variables for cluster
Kx             chnget S_x
ky             chnget S_y
kzone          chnget S_zone
kzoneX         chnget S_zoneX
kzoneY         chnget S_zoneY
knumTouches    chnget S_numTouches
```

```
ksize           chnget S_size
kdir            chnget S_direction
kvel            chnget S_velocity
kisHeld         chnget S_isHeld
kisZooming      chnget S_isZooming
xout p4, p5, p6, p7, p8, p9, p10, kx, ky, kzone, kzoneX,
kzoneY, knumTouches, ksize, kdir, kvel, kisHeld,
kisZooming
endop

instr 1
/* ---TOUCH---

Score format: i1.N 0 -1 N x y zone zoneX zoneY
Each individual touch generates a new instance of this
instrument, which is killed upon touch removal. */

; Touch properties
itouchID, ix, iy, izone, izonex,izoney,kx, ky, kzone,
kzoneX, kzoneY Touch p4, p5, p6, p7, p8, p9

;-----Add synths here-----;

; Master output
; gamasterL = gamasterL +
; gamasterR = gamasterR +
; Reverb send
; gareverbL = gareverbL +
; gareverbR = gareverbR +
endin

instr 2
/* ---CLUSTER---

Score format:i2.N 0 -1 N x y zone zoneX zoneY numTouches
Touches arriving within a certain distance of one
another are grouped into a cluster. Each cluster has a
set of shared parameters (number of touches, size, etc.)
There can only be a maximum of 2 clusters present,
intended to be used for left and right-hand. Clusters
die when all of their touches are removed. */

; Cluster properties
```

```
iclusterID, ix, iy, izone, izonex,izoney,
inumTouches,kx, ky, kzone, kzoneX, kzoneY, knumTouches,
ksize, kdir, kvel, kisHeld, kisZooming Cluster p4, p5,
p6, p7, p8, p9, p10

;-----Add synths here-----;

; Master output
; gamasterL = gamasterL +
; gamasterR = gamasterR +
; Reverb send
; gareverbL = gareverbL +
; gareverbR = gareverbR +
endin

instr tap
/* ---TAP---
Score format: i "tap" 0 0.05 x y zone zoneX zoneY
numTouches
When a group of touches hits and leaves the surface
quickly, without moving far, a tap event is triggered.
*/

; Tap properties
ix = p4
iy = p5
izone = p6
izoneX = p7
izoneY = p8
inumTouches = p9

;-----Add synths here-----;

; Master output
; gamasterL = gamasterL +
; gamasterR = gamasterR +
; Reverb send
; gareverbL = gareverbL +
; gareverbR = gareverbR +
endin

instr flick
/* ---FLICK---
```

```
Score format: i "flick" 0 0.05 x y zone zoneX zoneY
numTouches dir
Identical to a Tap event, except the touches have moved
prior to leaving the surface. Gives direction value. */

; Flick properties
ix = p4
iy = p5
izone = p6
izoneX= p7
izoneY = p8
inumTouches = p9
idir= p10

;-----Add synths here-----;

; Master output
; gamasterL = gamasterL +
endin

instr reverb
/* ---REVERB---
Score format: i "reverb" 0 3600
A basic global reverb instrument. */

aL, aR    reverbsc gamasterL*0.05, gamasterR*0.05, 0.9,
10000
outs aL, aR
clear gareverbL, gareverbR
endin

instr master
/* ---MASTER---
Score format: i "master" 0 3600
Master output bus */

aoutL clip gamasterL
aoutR clip gamasterR
      outs aoutL, aoutR
      clear gamasterL, gamasterR
endin

instr accel
; Accelerometer update instrument
```

```
gkaccX chnget "accelX"
gkaccY chnget "accelY"
gkaccZ chnget "accelZ"
; printks "X = %f, Y = %f, Z = %f\\n", 0.25, gkaccX,
gkaccY,gkaccZ
endin

<CsScore>

; Run Csound indefinitely
f 0 6600

; Run reverb instrument
i "reverb" 0 6600

; Run master instrument
i "master" 0 6600

; Run accelerometer instrument
i "accel" 0 6600

e
</CsScore>
</CsoundSynthesizer>
```

Appendix B: CD contents

- **T-EMP 29-08-12.mp4**

  Video recording of T-EMP ensemble performance (Rockheim, 29th August 2012) featuring the *loopblender* performance interface discussed in Chapter 4

- **DroneTilt demo.mp4**

  Video recording of the author demonstrating the *DroneTilt* instrument design from Chapter 6 with *Oscar* running on a 2nd generation iPad

- **OSCAR project folder**

  XCode project for *Oscar* application*

- **McGlynn, P. Interaction Design for Digital Musical Instruments.pdf**

  Digital copy of thesis

Appendix C: Papers and publications

26/04/2013	*Carte blanche: Designing for live performance with a novel interface* Music Department Postgraduate Conference, NUI Maynooth.
02/08/2012	*Spatial Tagging: A Preliminary Study* (with Victor Lazzarini, Damián Keller, Marcelo Soares Pimenta & Marcelo Queiroz) 2nd Irish Sound, Science & Technology Convocation, Cork School of Music, Cork.
07/07/2012	*Multi-touch gestures for Musical Performance – live demo, singing bowls* Sonic Arts Forum, School of Music, University of Leeds, UK
22/05/2012	*Recontextualizing the Multi-touch Surface* (with Victor Lazzarini, Gordon Delap & Xiaoyu Chen, NUIM) 12th International Conference on New Interfaces for Musical Expression, University of Michigan, Ann Arbor, USA.
22/05/2012	*Considering Audience's View Towards an Evaluation Methodology for Digital Musical Instruments* (with Jerônimo Barbosa, Filipe Calegario, Veronica Teichrieb & Geber Ramalho) 12th International Conference on New Interfaces for Musical Expression, University of Michigan, Ann Arbor, USA.

04/05/2012	*OSCar: A non-visual multi-touch controller* 3rd Ubiquitous Music Workshop, Sao Paulo, ES. Brazil.
27/04/2012	*Live demonstration/performance of work* International Festival for Innovation in Music Production & Composition, Leeds College of Music, UK.
20/01/2012	*Non-Visual Interfaces for Musical Performance using Multi-touch* (with Edward Costello, NUIM) Society for Musicology in Ireland Postgraduate Conference, DIT Conservatory of Music & Drama, Dublin.
25/11/2011	*Expression through Design: Unifying the Creative Tools of the Electronic Performer* 'Echoes and reflections' Inaugural Interdisciplinary Seminar, An Foras Feasa, NUI Maynooth.
02/9/2011	*Developing a Method for Multi-touch* 2nd Ubuiquitous Music Workshop, Vitória, ES. Brazil.
01/09/2011	*Analysing Multi-touch Data for Expressive Musical Control* 13th Brazilian Symposium on Computer Music, Vitória, ES. Brazil.
09/07/2011	*Improving the Efficiency of Open Sound Control with Compressed Address Strings* (with Jari Kleimola, Aalto University, Finland) 8th Sound & Music Computing Conference, Padova, Italy.
25/06/2011	*Evaluating the Expressive Potential of New Gestural Interfaces through Experimental Musical Application*

	Development 9th Annual Society for Musicology in Ireland Conference, RIAM, Dublin.
07/05/2011	*Towards More Effective Mapping Strategies for Digital Musical Instruments* 9th Annual Linux Audio Conference, NUI Maynooth.
13/04/2011	*Sound Augmented Vision* Irish National Finals 'Imagine Cup', Microsoft, Dublin.
27/01/2011	*Motion & Metaphor* Society for Musicology in Ireland Postgraduate Conference, Queen's University, Belfast.
14/01/2011	*Intelligent Mapping in Digital Musical Instrument Design* Irish Workshop on Music and Audio Signal Processing, Trinity College, Dublin.
12/07/2010	*Sound Sculptures: Exploring Music through Motion* 3 Minute Gong Competition, NUI Maynooth.
05/03/2010	*Acoustic Intimacy and Electronic Possibility: Exploring the Expressive Potential of Gesture in Performer-Instrument Interfaces* Music Department Postgraduate Conference, NUI Maynooth.