

Investigation of the use of Multi-Touch Gestures in Music Interaction

Nicholas Arner

MSc (by research)

University of York

Department of Electronics

November, 2013

Abstract

The thesis explores the use of multi-touch gestures in the use of interactive music apps run on mobile devices. With the growing popularity of mobile, multi-touch devices such as the iPad, more and more people have the chance to interact with music in a creative setting. However, many of these apps are based on traditional analogue music equipment, employing metaphors to physical interaction elements such as rotary knobs and faders.

User preference for musical interaction is researched. Three apps were developed for this investigation: two apps employ skeuomorphic UI elements (rotary knobs and faders), and the third employs multi-touch gestures. All three apps allow the user to interact with a granular synthesizer. User tests show that if users want an app that will allow them to explore music in a manner they would describe as “intuitive, interactive, creative, and playful”, then a multi-touch gestural app is the preferred option. However, if users want to be able to intricately modify and edit music, then an app implementing a skeuomorphic design paradigm is the preferred option.

Table of Contents

Abstract	i
List of Figures	viii
List of Tables	xi
List of Accompanying Material	xii
Acknowledgments	xiii
Declaration	xiv
1. Introduction and Hypothesis	1
1.1 Thesis Overview.....	1
1.2 Thesis Structure.....	2
2. Literature Review	4
2.1 Music Therapy	4
2.2 Community Music	5
2.3 The Role of Technology in Music Therapy and Community Music	5
2.4 Problems and Solutions	12
2.5 Accessibility	13
2.6 Human-Computer Interaction	15
2.7 Interactive Composition	19
2.7.1 Sound Composition	23
2.7.2 Granular Synthesis and Sound Composition	24
2.8 Designing an Instrument	26
2.8.1 Control Surfaces	26
2.8.2 Mapping	28
2.8.3 Gestures	32
2.8.4 Multi-touch Interaction	34
2.8.5 The iPad and Mobile Music	35
2.9 iOS App Examples	37

2.9.1	iMaschine	39
2.9.2	Figure	40
2.9.3	The Akai SynthStation	41
2.9.4	iKassoliator	44
2.9.5	The Animoog	45
2.9.6	The Filtatron	47
2.9.7	GrainProc	48
2.9.8	Reactable Mobile	50
2.9.9	NodeBeat	52
2.9.10	csGrain	53
2.9.11	Portable Dandy	56
2.9.11	iPulsaret	57
2.9.12	Grain Science	59
2.9.13	MegaCurtis (Lite)	60
2.9.14	CP1919	62
2.9.15	TC-11	63
2.9.16	Borderlands	66
2.10	App Conclusions	69
2.11	Interactive Composition Systems or, How Do We Make Music Now?	69
2.12	Problems with Digital Musical Instruments	71
2.13	What can Music Technology teach HCI?	73
3.	Preliminary Study on Gestural Intuitiveness	76
3.1	Purpose	76
3.2	Test Overview.....	77
3.3	Test Subjects	77
3.4	Test Procedure.....	80
3.5	Technical Details	80
3.6	Analysis	81

3.6.1	Sound Generation and Modification	85
3.7	Conclusions	87
4.	Technical Details of User Tests	89
4.1	iOS	89
4.1.1	Multi-touch on iOS	89
4.1.2	Gesture Recognizers	93
4.1.3	Types of Gestures in iOS	94
4.1.4	iOS Development in Xcode	96
4.2	Audio Programming for iOS	97
4.2.1	Core Audio	97
4.2.2	libpd	98
3.2.2.1	Pure Data	98
4.2.3	The Amazing Audio Engine	99
4.2.4	Mobile Csound Platform	99
4.2.4.1	Csound.....	101
4.3	History and Overview of Granular Synthesis	102
4.4	Granular Synthesis in Csound	106
4.4.1	The <i>grain</i> Opcode	106
4.5	Design of Test Apps	112
4.5.1	Rotary Knob Test App	113
4.5.2	Faders Test App	116
4.5.3	Multi-Touch Test App	118
4.5.3.1	Multi-Touch Implementation Details	121
5.	App User Tests Design	127
5.1	Test Purpose	127
5.2	Test Subjects	128
5.3	Test Procedure	130
5.4	Technical Setup	131

5.5	Test 1 Results.....	131
5.6	Test 2 Results	133
5.7	User Comments.....	134
5.8	Hypothesis Discussion	136
5.9	Potential Test Improvements	138
6.	Conclusions and Future Work	141
6.1	Conclusions.....	141
6.2	Further Work.....	141
6.3	Significance of Research.....	142
	Appendix A - Interactive Composition App Sketches	146
	Appendix B -Tutorial	157
	Appendix C - Csound for iOS Tutorial Code.....	190
	Appendix D - CDM blog post, emails from developers/Boulanger	191
	Appendix E - Gesture Intuitiveness Study Participant Handout	193
	Appendix F- Gesture Intuitiveness Study Questionnaires	196
	Appendix G - Gesture Intuitiveness Study Spreadsheets	217
	Appendix H - Csound Code/audio files for Gesture Intuitiveness Test	218
	Appendix I - Video Clips from Gesture Intuitiveness Test	219
	Appendix J - App User Test Participant Handout	220
	Appendix K - App User Test Questionnaires	225
	Appendix L - App User Test Spreadsheets	256
	Appendix M – Video Clips from App User Tests.....	257
	Appendix N – Subject Audio Clips.....	258
	Appendix O -App User Test Xcode Projects	259
	Appendix P – App User Test .CSD File	275
	Appendix Q - Literature Repository	276

Appendix R - Fraunhofer MSc Presentation277
References278

List of Figures

Figure 2.1	MidiGrid	7
Figure 2.2	The MIDICreator	8
Figure 2.3	MidiGesture Sensor	9
Figure 2.4	The “Shell Instrument”	10
Figure 2.5	The Squeezable Cluster	11
Figure 2.6	Illustration of HCI	17
Figure 2.7	An Interactive Composition System	22
Figure 2.8	Input to Output Mapping	29
Figure 2.9	Multi-Parametric Mapping	31
Figure 2.10	iMaschine Drum Pad View	39
Figure 2.11	iMaschine Keyboard view	39
Figure 2.12	Reason Figure	41
Figure 2.13	Akai SynthStation Performance View	42
Figure 2.14	Akai SynthStation Sequence Editing View	43
Figure 2.15	Akai SynthStation Drum Kit View	43
Figure 2.16	iKaossilator	45
Figure 2.17	Animoog iPhone App	46
Figure 2.18	Animoog iPad App	46
Figure 2.19	Filtatron Main View	47
Figure 2.20	Filtatron Sample Editor View	48
Figure 2.21	Filtatron Pad View	48
Figure 2.22	GrainProc	49
Figure 2.23	The Reactable	51
Figure 2.24	Reactable Mobile App	51
Figure 2.25	NodeBeat	53

Figure 2.26	csGrain	54
Figure 2.27	Dandy	57
Figure 2.28	iPulsaret	58
Figure 2.29	Grain Science	60
Figure 2.30	MegaCurtisLite Performance View	61
Figure 2.31	MegaCurtisLite Editing View	61
Figure 2.32	CP1919	63
Figure 2.33	TC-11 Patching View	64
Figure 2.34	TC-11 Interaction View	65
Figure 2.35	Borderlands, showing audio grains	67
Figure 2.36	Borderlands, showing grain parameters/editing toolbar	68
Figure 3.1	iPad's multi-touch sensitivity area	84
Figure 3.2	Human Energy Input and Control	86
Figure 4.1	iPhone Touch Detection	92
Figure 4.2	Discrete and Continuous Gestures	94
Figure 4.3	Screenshot of Screenshot of Xcode IDE	97
Figure 4.4	A Pure Data Patch	99
Figure 4.5	Csound for iOS API Relationships	100
Figure 4.6	Screenshot of a Block of Csound Code	102
Figure 4.7	View of a Grain in the Time Domain.....	103
Figure 4.8	Grain Amplitude Envelopes	104
Figure 4.9	A Hanning Window	108
Figure 4.10	Code for generating a square wave and applying a Hanning Window.....	109
Figure 4.11	Block Diagram of <i>grain</i> opcode	110
Figure 4.12	.CSD Instrument Flowchart	111
Figure 4.13	Code for rendering control	113

Figure 4.14	Rotary Knob Test App	114
Figure 4.15	Setting the value parameters of Knob1	115
Figure 4.16	Sending knob values to Csound variables	115
Figure 4.17	Csound code for reading in variable control information ...	116
Figure 4.18	Code for sending UISlider values to Csound.....	117
Figure 4.19	Faders Test App.....	118
Figure 4.20	Screenshot of Touch Control Zones.....	119
Figure 4.21	Touches App.....	120
Figure 4.22	Creating Two Touch Areas.....	121
Figure 4.23	<i>touchesBegan</i> Method.....	122
Figure 4.24	Parameter declaration in <i>touchesMoved</i>	122
Figure 4.25	X/Y position tracking.....	123
Figure 4.26	Delegate Code for sending X/Y Values	123
Figure 4.27	Calling <i>OneTouch</i> Delegates	124
Figure 4.28	<i>muteCsound</i> Method	124
Figure 4.29	Delegate Method Implementation	126

List of Tables

Table 3.1	Subjects' Musical Backgrounds	79
Table 3.2	Number of Hands Used	82
Table 3.3	Orientation Positions	83
Table 3.4	Did Gestures Stay in multi-touch Area?	83
Table 3.5	Generation and Modification Events	87
Table 4.1	Apple iOS Gestures	95
Table 4.2	<i>grain</i> Initialisation Parameters	106
Table 4.3	<i>grain</i> Performance Parameters	107
Table 5.1	Subjects' Music/Audio Background	129
Table 5.2	Test1- Subjects' Most and Least Preferred Apps	132
Table 5.3	Test2- Subjects' Most and Least Preferred Apps	133
Table 5.4	User Change in Preference	134

List of Accompanying Material

The accompanying DVD contains a PDF file of this document, as well as the supporting materials for various appendices. These are listed below:

Appendix C – Csound Tutorial Xcode project

Appendix F – Gesture Test Questionnaires

Appendix H – Gesture Test .csd file and audio clips

Appendix K – App user test questionnaires

Appendix L – App user test Excel spreadsheets

Appendix N – App user test audio commentary

Appendix O – App user test Xcode projects

Appendix P – App user test .csd files

Appendix Q – Literature Repository

Appendix R – Fraunhofer IIS MSc Research Presentation

Acknowledgments

To my parents and brothers, thank you all for your love and support.

Papaw, thanks for teaching me to take things apart and figure out how they work.

Grandpa, thanks for giving me the travel bug. I wish I could show you where I've gone.

Andy, I owe you tremendous thanks for all your help and guidance during the past year. I couldn't have done this without you.

Helena and Jude, thank you both for your advice during the past year; especially your grammar lessons. I probably wouldn't have survived my year in the UK without them.

Thanks to Mark "Doc" Lochstampfor for having faith in me. I would not have gotten this far were it not for you pushing me to succeed.

To Steve, Andrew, and Sarah, thanks for the good friendship, and for the many curry nights at the Deramore this past year. And to Andrew...that was the best brownie I've ever had. Ever.

Thanks to Jelle and Dimitri for helping me become a better programmer.

Danielle and Jarrod, thank you for your witty banter and conversation, as well as for being good listeners and good friends.

To my dear friend Sarah; thank you for your conversation and encouragement over the past year.

To my best friend Jeremiah, thank you a friendship that has remained constant throughout our life despite the many miles between us.

Declaration

The work presented in this thesis is entirely my own. None of the content as part of this thesis has been previously published by the author in any form.

Chapter 1

Introduction

This chapter describes the motivation behind the research, as well as the preliminary hypothesis. Subsequent chapters are also described in Section 1.2.

1.1 Thesis Overview

In spite of the advances that have occurred in music interaction research, the majority of music-centric computer programs and mobile apps implement user interaction based around mouse and keyboard input (in the case of desktop computer programs), or around skeuomorphic user interface elements (in the case of mobile apps). These skeuomorphs include virtual sliders, rotary knobs, and even piano keyboards that the user “plays” with a single finger. This approach is persistent in music-centric mobile apps, despite the fact that the multi-touch platforms they run on are capable of intricate and often intuitive multi-touch gestures for interaction.

The hypothesis of this project is that users prefer using multi-touch gestures to interact with music as opposed to traditional skeuomorphs. If this is indeed the case, it will most likely be because users find multi-touch gestures more intuitive for controlling musical parameters than traditional user interface skeuomorphs. This hypothesis will be tested through user tests conducted with three separate iPad apps, developed specifically for this purpose, and described in Chapter 5. The technology used in the creation of these apps is discussed in Chapter 4.

The research took several changes in direction, scope and outcome goals throughout the project period. Originally, the goal was to develop an app that would assist individuals with disabilities in composing original music. Upon further consideration, this was deemed impractical due to the wide range of disability types; it is impossible to develop a single system that would meet the needs of such a diverse set of users.

The goal then became to develop an interactive composition system that would be useful to those with no musical background or training. Instead of focusing on making the app helpful to one specific segment of the population, the goal was to make the app as *accessible* to as many people as possible.

As the research developed the aforementioned hypothesis emerged to investigate users' preferences when using multi-touch devices for musical purposes. The hypothesis' evolution is discussed in detail in Chapter 5.

1.2 Thesis Structure

Chapter 2 provides an overview of the literature relevant to this thesis, specifically the role of technology in music therapy, interactive composition, Human-Computer Interaction, Music Interaction, the iPad and mobile music, and it concludes with an overview of existing music-centric iOS apps. Chapter 3 is an overview of the preliminary study on gestural intuitiveness, which investigates how users relate multi-touch gestures to various musical parameters. Chapter 4 provides a technical overview of the hardware and programming technologies used in the creation of the apps built for the testing of the hypothesis, as well as specific design details of each test app. Chapter

5 details the user tests carried out to test the hypothesis, as well as presenting a modified version of the hypothesis. Finally, Chapter 6 concludes the thesis with a discussion of the relationship between Music Interaction and Human-Computer Interaction research, and how Music Interaction can benefit the HCI discipline as a whole.

Chapter 2

Literature Review

A literature review was undertaken to investigate the theoretical principles that inform the goals of the research, including the role of technology in music therapy, Human-Computer Interaction, Music Interaction/Interactive Composition, the iPad and mobile music. It concludes with a review of currently available iOS interactive music applications.

At the start of the project, the goal was to design an app that would help music therapy clients with the act of music composition. Though that goal evolved, the research conducted on the role of technology in music therapy and community music is included here for completeness and to provide an appropriate context to the goals of the project.

2.1 Music Therapy

The British Association for Music Therapy defines music therapy as “...*a psychological therapy which uses the unique qualities of music as a means of interaction between therapist and client*” (BAMTa, 2012, online). Ability and musical performance are not necessarily the outcome goals of music therapy, but rather the focus is on helping people communicate in their *own musical language*. Clients of music therapy include children and young people, individuals with learning disabilities or autistic spectrum conditions, individuals in need of mental health care, the elderly, and those with neuro-disabilities (BAMTb, 2012, online).

2.2 Community Music

Olseng (1990) states (as cited in McKay and Higham, 2003, p.5), community music is “...*characterized by the following principles: decentralisation, accessibility, equal opportunity, and active participation in music making*”. Distinct from music therapy, or even community music therapy, community music signals an effort to “...*move outside a clinical or restricted practice to a wider, more socially engaged one*” (McKay and Higham, 2003, p.7). Essentially, community music is not clinically oriented, but rather focused on helping a wide variety of individuals to express themselves musically, who may or may not have a prior musical background or training.

2.3 The Role of Technology in Music Therapy and Community Music

Music technology has a vital role to play in music therapy. Music therapists have traditionally used a variety of acoustic instruments in their practice, such as guitar, piano, drum-sets, percussion, etc. (Cole, 1996). However, according to Magee and Burland, “...*music therapists turn to technology to enable a client to participate actively or to widen the client’s musical expression. Technology offers improved access for people with complex physical needs to engagement in active methods of music therapy*” (Magee and Burland, 2008, p.3). Magee and Burland also state that electronic music technology may be specifically helpful for people with limited movement, as well as for children and adolescents suffering from emotional disorders; a group that has been traditionally difficult to engage in music therapy (Magee and Burland, 2008).

Though immensely expressive, acoustic instruments may not be able to be used by those with certain physical and/or mental disabilities. Often, the therapist will assist the individual with physically manipulating the instrument. This can be a problem, as

the patient will not feel that they are in complete control of the musical situation, leading to feelings of frustration and/or dependence. In a 2012 paper, Burland and Magee state that an electronic music technology system can help disabled users to “...*explore sound and offers a way to communicate with others*” (Burland and Magee, 2012, p.7), thus providing clients with more control over the music creation process.

A variety of music technology systems utilized in music therapy have been developed at the University of York. One of these is the *MidiGrid* system, developed by Andy Hunt and Ross Kirk. The program allows the user to perform chord sequences, arpeggios, and scales in a variety of timbres by moving a mouse over specific grids of musical material (see Figure 2.1). In addition to the mouse control, it is also possible for users to connect external MIDI devices that allow gestural actions to control pitch-bending, pan positions, and volume modulation (Hunt and Kirk, 2003). According to the authors, *MidiGrid* has been utilized in music therapy to allow the user to improvise on a wide palette of timbres. Such timbres may not have been accessible to the users without the use of music technology.

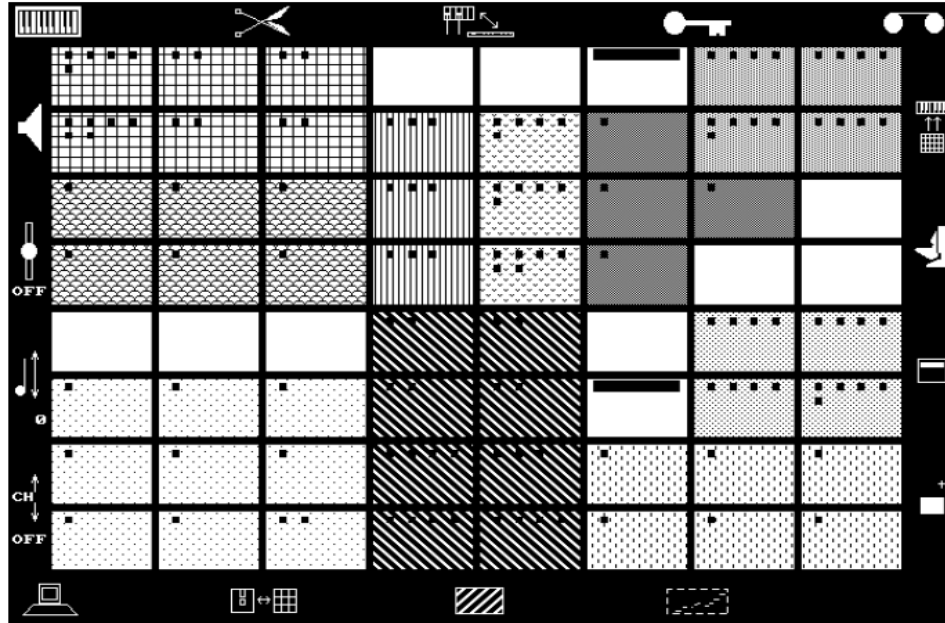


Fig. 2.1
MidiGrid
(Hunt and Kirk, 2003, p.136)

University of York student Phil Bates designed a system known as *MIDIcreator*, which can be used as an external controller to *MidiGrid*. *MIDIcreator* converts voltages from a variety of electronic sensors and converts them into MIDI messages, which control a variety of parameters on electronic musical instruments. The *MIDIcreator* is shown below in Figure 2.2.



Fig. 2.2
The MIDIcreator
(MIDIcreator User Manual)

One of the sensors that was specifically developed for use with the *MIDIcreator* is the *MIDIGesture*, a small portable sensor capable of detecting objects from one to three meters away (Cole, 1996). This allows individuals to interact with programs such as MidiGrid through the use of free-form gestures, rather than with a mouse. Other sensors developed for the MIDIcreator include a squeezable cushion, piano keyboard, light, and air pressure sensors (MIDIcreator User Manual). The *MIDIGesture* is shown below in Figure 2.3.



Fig. 2.3
MidiGesture sensor
(MIDIcreator-Resources)

More information regarding the types of sensing technologies that are able to be used in musical applications, as well as specific examples each sensing technology, is available on the SensorWiki website (Wanderely, 2004).

University of York researchers also explored the concept of using tactile physical gestures to modify sound. A “shell instrument”, shown in Figure 2.4, consists of a “...*fibreglass mould set in transparent resin, in which piezoelectric sensors are embedded*” (Hunt, Kirk, and Neighbour, 2004, p.52). The sensors respond to the user’s touch, which is then mapped to control and synthesis parameters on an external sound generator.



Fig 2.4
The “Shell Instrument”
(Hunt 2004, p.52)

Similarly, Weinberg explored the concept of squeezing or sculpting sound in his Squeezable Cluster while at MIT (Weinberg, 1998). The Squeezable Cluster is a controller that allows for musical exploration and is designed for use by children.

The user squeezes foam balls embedded with sensors mapped to synthesis parameters. Though not explicitly designed for use in music therapy or community music, the mode of interaction used by the Squeezable Cluster, shown in Figure 2.5, potentially lends the device to use in such a context.

Although the test apps documented in this thesis (discussed in more detail in later chapters) utilize a touch-screen for interaction rather than tactile, physical objects, the concept of sound sculpting plays a significant role in the design of the interface and synthesis mapping to gestures.



Fig 2.5
The Squeezable Cluster
(Weinberg, 1998, p.43)

According to Nagler, hand-held devices (e.g., mobile phones) are of use in music therapy due to the fact that there are “...fewer physical limitations, abundantly rich sonic possibilities and robust algorithms that negate the need for prerequisite music making skills or task readiness” (Nagler, 2011, p.198). Nagler presents these characteristics as advantages that mobile devices have over traditional acoustic instruments. Many music therapy patients may have difficulty in interacting with acoustic instruments due to possible physical limitations. As a result of this, mobile technology offers greater potential for clients with physical handicaps to interact with music creation in a music therapy context. The fact that mobile devices also offer the

potential for the creation of new sonic possibilities due to the digital signal processing (DSP) algorithms they are able to run further enhances the possibility of use in a music therapy context. Since clients are able to create sounds that have never been heard before, they are able to freely explore and create; making music that is truly unique to their identity.

Additionally, users are able to share their musical creations with others through email or uploading to social media sites. This will help the user feel a sense of accomplishment and self-actualization. Through the incorporation of Application Programming Interfaces (APIs) provided by websites such as SoundCloud, Twitter, and Facebook, users of a mobile music app will be able to share their creations with their friends and loved ones.

In addition to the interfaces discussed so far, a variety of other computer programs, applications, and interfaces have been developed specifically for use in a music therapy/community music context (Challis and Smith, 2012; Gorman et al., 2007; Corrêa et al., 2009; Hözl et al., 2009; Boulanger, 2004).

2.4 Problems and Solutions

Despite technological advances, music therapists are often reluctant to incorporate electronic music technologies in their practice. Magee and Burland state that one of the reasons for this is that many therapists do not have a technological background, and thus do not feel confident in the use of modern musical technology (Magee and Burland, 2008). An application that does not require *any* technical training would be helpful in solving this problem.

Additionally, as many music therapists are employed part-time across multiple locations or perform various outreach workshops, any technology needs to be as portable and mobile as possible. The fact that many music technology systems require time for setting up, means that they may be inappropriate for many music therapists' needs (Magee, 2006). This problem can be eliminated by the creation of interactive musical applications that do not require any additional setup by the user. One of the justifications for using the iPad as the platform for the original application goal of this project was the factor of portability and ease of setup by users. More discussion on the iPad follows in Chapter 4.

Due to their multisensory characteristics, acoustic instruments are often perceived as more aesthetically appealing than electronic instruments (Magee and Burland, 2008). With the advances in Digital Signal Processing (DSP) technology and synthesis algorithms in the past decades, this is becoming less and less of a problem. These technologies are now available on mobile devices via Core Audio, Pure Data, and Csound, etc; as will be shown in Section 4.2. The problem of the lack of visual appeal in electronic systems is also solved with visually captivating interface design on devices such as the iPad.

2.5 Accessibility

During the course of the research period, it was determined that the task of developing an iOS application for the field of music therapy and/or community music was too specific a task for the scope of this project. Instead, the research focus shifted to the investigation of how to provide an intuitive, interactive compositional environment

on a multi-touch device, aimed at individuals either without prior knowledge or access to traditional musical instruments.

Castro, writing for The Information Technology and Innovation Foundation, defines accessible technology as “...*technology that has many broad applications but helps remove barriers and make the world more accessible for people with disabilities, giving them more access to information, communication, and independence*” (Castro, 2008, p.52). Such technology may not be designed *specifically* for users with disabilities, but such users are better able to engage with the technology. Similarly Bergman and Johnson state that providing users with accessibility in a computer system means “...*removing barriers that prevent people with disabilities from participating in substantial life activities, including the use of services, products, and information*” (Bergman and Johnson, 1997, p.2).

The notion of designing computer systems specifically for users *with* and *without* disabilities is challenged by the claim that *all* potential users of a computer system have a wide and diverse skill set depending on the user’s life stage, task and environment (Beaudouin-Lafon, 2004). The notion of categorizing users into those with or without disabilities is summarily invalid.

Bergman and Johnson also argue that not only does awareness of accessibility benefit users who may have physical disabilities; it also provides a higher quality of user experience for those who do not have a disability, stating that usability testing with disabled users “...*can uncover usability defects that are important in the larger population*” (Bergman and Johnson, 1997, p.9). Such usability defects can include font

and colour conflicts, problems with layout and context, poor interface flow, tab order and tasks that require an excessive number of steps or a wide range of movement (Bergman and Johnson, 1997).

Instead of categorizing users into disabled and non-disabled, designers and programmers should focus instead on creating interactive computer systems that address a more expansive range of interaction modes and experiences, therefore broadening the base of potential users.

Designers and programmers must also focus on the interactions that the user will be performing with the system. The capabilities of the computer system, for example processing power or memory size, are meaningless unless the user is able to interact with the system in an intuitive and meaningful way. Beaudouin-Lafon explores this problem by stating “...our goal is to control the quality of the interaction between user and computer: user interfaces are the means, not the end” (Beaudouin-Lafon, 2004, p.16). Prior to finalizing the interface design for an interactive music system, a decision must be reached regarding the way in which the user will interact with an audio synthesis engine for the purpose of composing music.

2.6 Human-Computer Interaction

By the very nature of their design goals, interactive computer music systems are based around a human user interacting with a computing platform; in the case of this project, a mobile computing platform, specifically the iPad. As such, an overview of Human-Computer Interaction follows.

Human-Computer Interaction is defined by the ACM Special Interest Group for Computer-Human Interaction as “...a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them” (Hewett et al., 1992). An inherently broad field of research, HCI incorporates the diverse disciplines (Rogers, Sharp, and Preece, 2011, p.10) of:

- Computer Science
- Artificial Intelligence
- Linguistics
- Philosophy
- Sociology
- Anthropology
- Design
- Engineering
- Ergonomics and human factors
- Social and organisational psychology
- Cognitive psychology

Figure 2.6 below illustrates the general process of human-computer interaction.

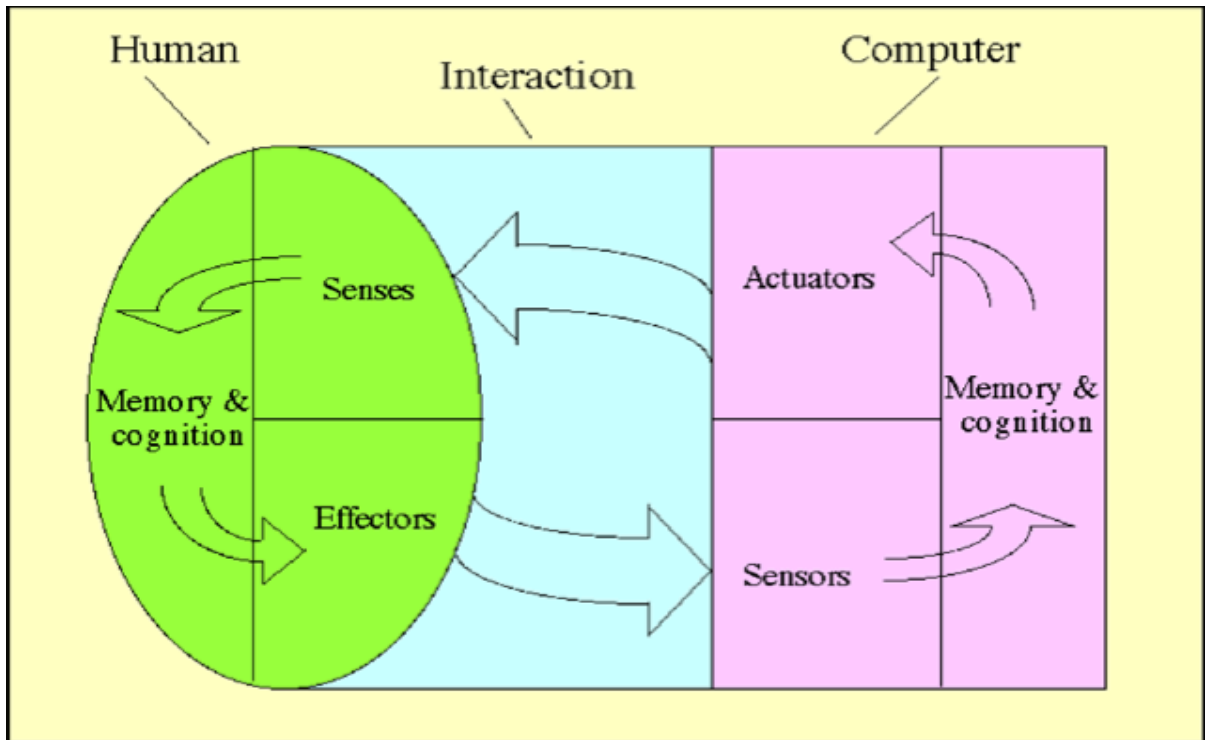


Fig. 2.6
Illustration of HCI
(Bongers, 2000, p.44)

When a user interacts with a computer, they approach the system with their memory and cognition, the sum total of all their life experiences and knowledge. They interact with the computer system through *Effectors*. In the case of a multi-touch system, effectors are hands and fingers. The computer system receives this information via various *Sensors*, and processes it through its own memory and cognition (programmed *Algorithms*), and outputs data through *Actuators*, such as a visual screen or a speaker. The user then perceives this data through their visual, auditory, or tactile senses. They process this information, and keep interacting with the computer in a continuous control-feedback loop (Bongers, 2000).

When the user of a software system is able to *subconsciously* apply their prior knowledge and experiences to interacting with it, the system is considered to be intuitive, or intuitively usable. The system approaches intuitiveness the more that prior knowledge can be classified as “...*innate, sensory-motor, embodied, cultural, or expert*” (Wilkie et al., 2010, p.37). Users of such systems can be described as experiencing Csikszentmihalyi’s concept of “flow”, which Leman describes as “*an experience in which the subject’s skills are fully preoccupied with a task*” (Leman, 2010, p.139). Users experiencing flow are fully involved in accomplishing their intended task with the tool they are using, not with *consciously thinking* how they can use the tool to accomplish their tasks. In order for users of computer music software to enter a state of creative flow, the interfaces of such programs must... “*be built around tacit knowledge, and also afford the opportunity for users to discover and form their own perspectives*” (Nash, 2011, p.58).

In order to help developers create intuitive iOS apps, Apple provides a list of guidelines for the design for User Interfaces in its guide on iOS Human Interface Principles :

Aesthetic Integrity – How well the appearance of the app integrates with its function.

Consistency – The ability of users to transfer their knowledge and skills from one app to another.

Direct Manipulation – Using gestures gives people a greater affinity for, and sense of control over, the objects they see onscreen, because they're able to touch them without using an intermediary, such as a mouse.

Feedback – Apps should acknowledge user actions and reassure them that processing is occurring.

Metaphors – Virtual objects and actions in the app are metaphors for objects and actions in the real world; users quickly grasp how to use the app.

User Control – People, not apps, should initiate and control actions.

(Apple 2012e)

Prior to a discussion of specific HCI topics (mapping, gestures, and multi-touch) in the context of an interactive music composition system, the notion of what interactive composition is, and its history in relation to computing technology will be discussed.

2.7 Interactive Composition

The Harvard Dictionary of Music defines composition as “The activity of creating a musical work [...]” (Harvard Dictionary of Music, 4th ed., 2003, p.194). While an overview of the theory of musical composition is beyond the scope of this work, a brief discussion of the intersection between composition and technology follows to provide a background on the motivation of the second goal of the thesis: that of making composition more accessible to users.

Composers have long made use of technology in general, and computers in particular, as a way to compose music that would be impossible to compose and perform with traditional acoustic instruments (Holmes, 2008; Brunner, 2009). In 1957, Max Mathews became the first person to program a computer to synthesize music. Using a program he created, *MUSIC I*, Mathews composed a short (seventeen second) monophonic tune (Holmes, 2008). *MUSIC I* was later developed into a variety of music synthesis languages, including the synthesis environment that this project uses, Csound. Csound will be discussed in more detail in Chapter 4.

At first, the performance power of computers limited their use in a musical context, both compositional and performance-based. Composers, initially, were restricted to composing on punched cards, or using a pseudo-random number generator to determine performance rules for a composition with acoustic instruments (Holmes, 2008).

The development of microprocessors led to the creation of simultaneously smaller and more powerful computers, and musicians started to leverage the available computing power to compose and perform more musically intricate pieces. A variety of music-focused software environments began to develop: software synthesizers, virtual analogue instruments, audio development environments, software samplers, percussion synthesizers, and digital audio workstations (Holmes, 2008).

In addition to being able to compose and perform in real-time, musicians began to use the malleability and flexibility of the computer to explore a new domain of musical activity: interactive composition.

In a *Computer Music Journal* article published in 1984, Joel Chadabe describes the use of an interactive composition system as being a two-stage process:

1. Creating an interactive composing system.
2. Simultaneously composing and performing by interacting with that system as it functions.

(Chadabe, 1984)

A composer engages in a high-level musical dialogue with an interactive system after it has been created. An interactive composition system can be created with a variety of hardware and/or software components.

While the system is creating music based on the rules of its algorithms set by the composer, the composer is reacting to this music in real-time via some kind of external interface that is capable of adding to the musical output of the system. Interaction between the composer and the system means that “...*the computer’s internal state depends on the performer’s action, and that the latter may itself be influenced by the computer output*” (Di Scipio, 2003, p.270).

The composer of such a system is doing more than creating a particular musical structure; they are composing “...*a mode of functioning for computer system and performer that, in operation, generates a new particular structure in every performance*” (Chadabe, 1984, p.26). The composer is not necessarily performing music in response to the system; often they are performing control information that *alters* the performance output of the system (Chadabe, 1977). Essentially, the composer determines the overall high-level structure of the composition while the computer performs signal processing to produce the sonic output desired by the composer.

Figure 2.7 depicts the interaction of a musician and a computer music system. Here, “performer” is taken to mean any human user, and “instrument” is taken to mean any tool for the facilitation of interactive electronic composition.

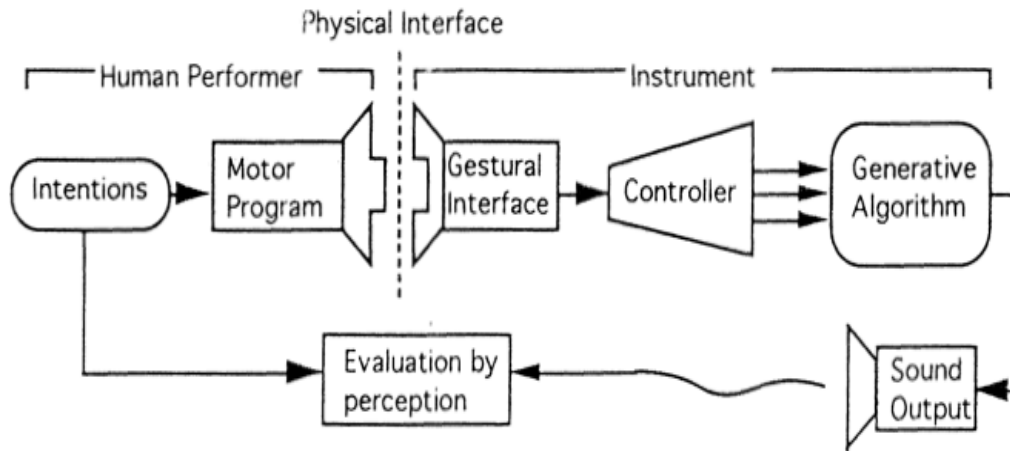


Fig. 2.7
An Interactive Composition System
(Lee and Wessel, 1992, p.278)

A user approaches the compositional system with the desire to create a new piece of music. They engage with the system by using their hands and fingers in contact with some kind of gestural interface. These gestures are mapped to the synthesis algorithms used by the system, which generates audible output. When the user hears the output, they determine how they wish the music to evolve, and continue interacting with the system accordingly.

A number of interactive compositional systems have been developed with the goal of making composition more accessible and easier to engage with, including QSketcher (Abrams et al., 2001), Smart Harmony (Abrams et al., 1999), DSP (Rudi, 2007), and Hyperscore (Farbood, 1997). However, these systems are restricted to use on desktop computing platforms that require the use of a mouse and keyboard.

In creating an engaging and intuitive interactive composition system, the designer has to consider the underlying synthesis methods and algorithms in order to form the desired aesthetic characteristics of the system.

2.7.1 Sound Composition

One consideration of designers of interactive music systems is whether they wish to compose music ‘of sounds’ (often referred to as sound, or timbre composition), or if they wish to compose music ‘of notes’ (“traditional” composition). Sound composition was not widespread until the advent of powerful computers, and is an example of what Treadaway describes as digital technology’s ability to “...*support the artist’s creative practice by providing access to tools and processes that enable work to be generated that could be made no other way*” (Treadaway, 2009, p.185).

Many computer-based composition systems adopt metaphors that are based on music theory concepts (Wright et al., 1997). As such, only those individuals with a previous background of music theory concepts are able to approach such a system and interact with it in a creative manner. Viewed from an accessibility-design point of view, such a system inherently limits the number of users who are able to interact with the system to only those who have music theory knowledge, as opposed to those who may not have such knowledge but are gifted with an inherent sense of musicality.

The second design goal of this thesis was to make music composition more accessible to a wider variety of users. A system that requires users to have previous knowledge of music theory and standard notation for interaction is therefore unsatisfactory to the design goals of this thesis. One way to make composition more accessible to a wider variety of users is through the aforementioned use of *sound*

composition, rather than note-based composition. Before the advent of computers, such musical works were realised through the use of analogue processing equipment, mixing consoles, and tape recorders.

According to Challis and Smith, composition can include “...*the selection of sounds to form a palette of sounds which can be shaped, manipulated, varied, and combined to create a piece or musical performance*” (Challis and Smith, 2012, p.65). In such a system, the user will still be able to explore the use of sound and timbre modification without worrying about whether or not the compositional sequence is correct within the framework of traditional harmony, as they would in a note-based environment.

The idea of composing with sounds is not something that evolved along with the digital age. Pierre Schaeffer, a radio engineer, and Pierre Henry, a composer, pioneered the idea of *musique concrète* in France in the late 1940s. *Musique concrète* is defined as “...*the construction of music using sound recording tools, natural sounds, electronic signals, and instrumental sounds*” (Holmes, 2008, p.49).

One synthesis method that allows for composing with sounds is granular synthesis. Granular synthesis allows for sampled sounds (either stored files or live audio input) to be manipulated in real time, creating a composition of sound rather than of musical notes.

2.7.2 Granular Synthesis and Sound Composition

Curtis Roads describes granular synthesis as “...*generating thousands of very short sonic grains to form larger acoustic events*” (Roads, 1988, p.11). Grains can then

be organized into higher-level compositional events. Roads describes an event as consisting of the following parameters:

- Beginning time
- Duration
- Initial waveform
- Waveform slope (the transition rate from a sine to a band limited pulse wave)
- Initial centre frequency
- Frequency slope
- Bandwidth
- Bandwidth slope
- Initial grain density
- Grain density slope
- Initial amplitude
- Amplitude Slope

(Roads, 1988, p.12)

Due to the high number of grains that can be organised at once, the result is perceived by the listener to be a fused sonic texture (Bencina, 2006). Once the grains, (which can number in the thousands) are organized into events, the user can alter the above parameters in real time. The user can use these grain events to create sound clouds of evolving spectra (Roads, 1988), enabling the user to compose directly with sounds without the intermediate interface of musical notation.

It was decided that an interactive system utilizing granular synthesis would be the best means of achieving the second project goal of developing an accessible interactive composition system. This is due to the fact that this approach allows for exploration of sound, not exploration of notes. As such, granular synthesis meets the original project goal of creating an interactive composition system that users with varied musical backgrounds would be able to use. It was decided to use granular synthesis in the Test Apps described in Chapter 4 for both consistency of development, as well as

allowing the user to interact with a complex synthesis system while testing their preferences for interaction paradigms on a mobile device, in order to test the main hypothesis.

2.8 Designing an Instrument

Computers have played a large part in the history of the development of new forms of music. Though primitive and cumbersome at first, computing technology has accelerated so that the average person has access to powerful mobile computing platforms, with a rich variety of sensing capabilities, in the palm of their hands.

In a paper presented at the 2008 Mobile Music Workshop, Essl, Wang, and Rohs presented on the on-going effort to turn mobile devices into *generic* devices for musical expression. The authors define generic as “...*a platform that is not designed with a specific performance in mind [...] alternately, a design that is open to flexible, varied use without trying to prefigure artistic intent*” (Essl et al., 2008, p.1). The authors describe the actions one needs to take in designing a mobile instrument:

- Decide what input modalities to use
- Manipulate them for synthesis control
- Choose appropriate synthesis algorithms

The answers to these questions will dictate the interaction paradigm, sound output, and overall style character of a mobile, interactive sound app.

2.8.1 Control Surfaces

In 1977, Pierre Schaeffer stated that “*Musical ideas are prisoners, more than one might believe, of musical devices*” (as cited in Roads 2001, p.44). An instrument’s

physics of sound generation, as well as of interfacing with the methods of sound generation, determine the ways in which a musician will think about performing and composing music.

The use of keyboard-based controllers in computer music systems is quite common due to persistent industry preference (Wessel and Wright, 2002). This is primarily due to the fact that the MIDI 1.0 specification was designed around the paradigm of keyboard performance (Roads, 1996).

However, a keyboard is not always appropriate for control of an electronic music system, particularly those that utilize multi-touch screens. There are two reasons for this. The first is that a mechanical keyboard is sensitive to the velocity, after-touch, pressure, and action of a user (Roads, 1996). These physical parameters are then translated into auditory feedback to the user in terms of musical nuance. However, they are not easily replicated on a multi-touch system.

Secondly, multi-touch based platforms, such as the iPad, have the capability of recognizing a variety of user gestures. These gestures can be mapped to whatever the designer of the application desires. As such, synthesis parameters can be controlled by other methods than just keyboard interfaces. If developers of mobile-based music apps want to set new ground and fully utilize the capabilities of multi-touch devices such as the iPad, they will need to develop other ways of interacting with musical structure besides keyboard interfaces.

2.8.2 Mapping

The method of connecting the control device to the parameters of sound generation is known as *mapping* (Hunt and Wanderley, 2002). When a user is performing with an acoustic instrument, the player is directly manipulating a physical object. As the interface and the method of sound generation are intrinsically connected, the mappings between interface and sound source are “...*complex, subtle, and determined by physical laws*” (Hunt, Wanderley, and Paradis, 2002, p.1). Jordà describes the intrinsic mapping between input and output in acoustic instruments by stating “*acoustic instruments impose their own playability rules, which allow listeners to infer the type and form of the gesture from the sound being generated*” (Jordà, 2005, p.6). Merrill and Raffle further describe the differences between acoustic and electronic instruments by stating that “*electronic instruments lack the subtle affordances and potential for acoustic spontaneity featured by acoustic instruments*” (Merrill and Raffle, 2013, p.213).

Unlike acoustic instruments, the method of control and the parameters that are controlled in a computer music system are separate, as there is no implicit mapping of one to the other (Winkler, 1995). As such, it can be difficult to simulate basic interaction characteristics that are an inherent part of acoustic instruments, such as tactile and/or force feedback (Wanderley and Depalle, 2004; Giordano and Wanderley, 2013). It is up to the composer/programmer to determine what performance actions are mapped to various synthesis parameters. The result of these interactions is the output sound. Figure 2.8, taken from Fels, Gadd, and Mulder (2002) below depicts the mapping process.

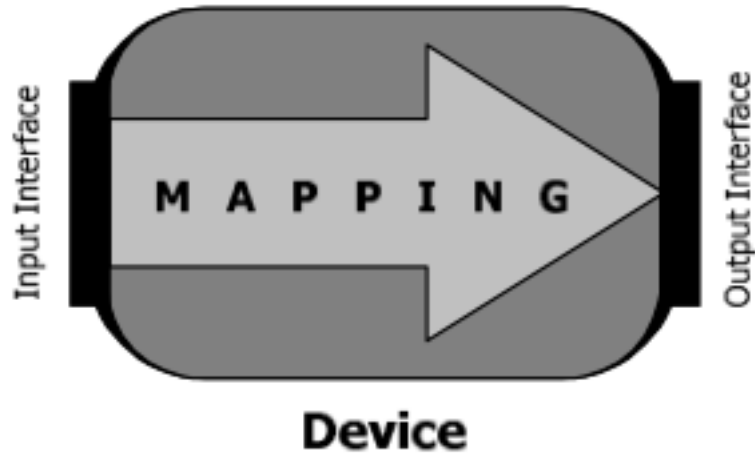


Fig. 2.8
Input to Output Mapping
(Fels et al., 2002, p.6)

Mapping determines the nature of how users will interact with a computer music system and, as such, is an extremely important consideration in the design phase of project development. Wessel et al. emphasise this point by noting that *“The success or failure of a live computer music instrument is determined by the way it maps performers’ control gestures to sound”* (Wessel et al., 2002, p.2). In comparison with music written for acoustic instruments, computer-based music has a shorter historical context from which to draw upon and, as such, can often be more difficult to program and compose. Consequently, the creation of meaningful mappings for computer-based music systems may be problematic for programmers and/or composers. While potentially challenging to design, the quality of the input-to-sound mappings help to determine the success or failure of the interactive music system, as *“Mapping is at least as important to musicians as the physical interface, and even more so over the long term”* (Casciato and Wanderley, 2007, p.4).

While the timbral possibilities of computer music systems are endless, unless the mappings are created to be intuitive, meaningful, and to allow the performer to advance in skill; the instrument will likely be used very rarely. In fact, many computer music systems do not have a shelf life longer than the single performance of a composition - a sharp contrast to a masterfully designed and crafted acoustic instrument such as a Stradivarius violin.

In order for a music-centric multi-touch app to be capable of eliciting intuitive interaction between the interface and the user, the mapping between the multi-touch gestures and the parameters of the synthesis engine must be semantically meaningful and easy for beginners to perceive. The goal for an interactive music system is “...*the emancipation of expressivity in computer music through the incorporation of multiple levels of human inflection*” (Overholt, 2009, p.219).

When designing the mappings for such a system, the designer should keep in mind that it is not necessary (or desirable) for the user to have control access to every parameter of the synthesis engine. In many analogue and software-based synthesizers, the user does indeed have manual access to every possible synthesis parameter, but this does not always equate with clarity or intuition. In such situations, “...*the number of elements assigned to different variables makes it difficult to understand how a sound is programmed with a glance at the control surface*” (Gómez et al., 2007, p.327).

In a study conducted at the University of York in 2000, Hunt and Kirk showed that multi-parametric musical user interfaces are more engaging for users than one to one mappings. They allowed the users to “...*think gesturally, or to mentally rehearse*

sounds as shapes” (p.255) when using multi-parametric interfaces. Figure 2.9 below depicts an illustration of multi-parametric mapping.

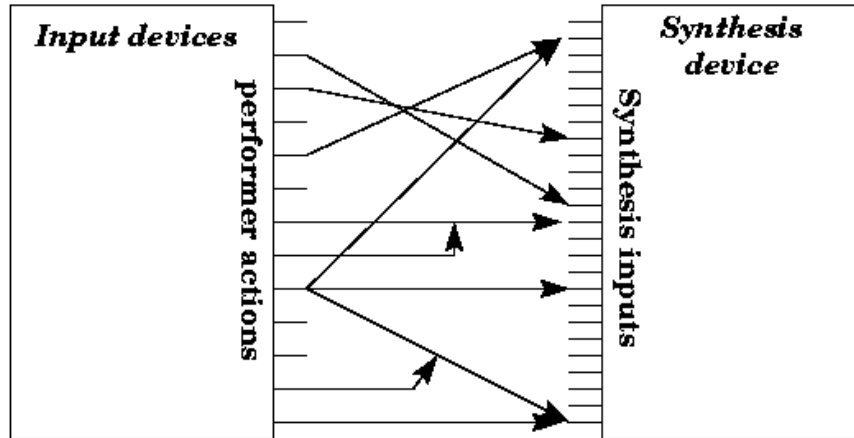


Fig. 2.9
Multi-Parametric Mapping
(Hunt, Wanderley, Kirk 2002, p.1)

Hunt and Kirk term this interaction ‘Performance Mode’; an explorative operation where the user “...discovers how to control a device by exploring different input control positions and combinations” (Hunt and Kirk, 2000, p.233), by which the user gains immediate feedback response from the system.

They list the characteristics of Performance Mode as:

- Continuous control of many parameters in real time,
- More than one conscious body control (or limb) is used,
- Parameters are coupled together,
- User’s energy is required as a system input.

During performance mode, the user discovers hidden relationships in the system that are not perceivable without repeated interaction. In such a system, a performer is able to emotionally express themselves “...by making the most of the available sensitivity and dynamic range of a given physical interface, and using different gestural interactions in each performance” (Overholt, 2009, p.219). Instead of gestures being mapped to individual synthesis parameters, a more appropriate solution would be to have a *conceptual* mapping layer that lets users control higher-level parameters such as brightness, sharpness, or other evocative timbral descriptors (Hunt et al., 2002).

As the sound production of acoustic instruments is determined by their mechanical/physical nature, a musician’s performance with them involves a close connection to the musician’s body (Magnusson and Mendieta, 2007). A computer interface, however, does not necessarily depend on the physical characteristics of the underlying synthesis algorithms that are utilised. Rather, the interface design is completely up to the programmer.

In a multi-touch environment, the process of mapping is influenced by the available gestures that are possible on the specific multi-touch platform. A general overview of the concept of gestures follows in the next section, including a description of available gestures for use on the iPad.

2.8.3 Gestures

There are many definitions of the word ‘gesture’ in both HCI and music related literature. From the HCI perspective comes the definition of a gesture as “*a set of measured points P in space and a corresponding set of time intervals T between measurements*” (Cleveringa et al., 2009, p.2). A less mathematical but equally valid

definition given by Leman and Godøy is that a gesture is “*a movement of part of the body, for example a hand or the head, to express an idea or meaning*” (Leman and Godøy, 2010, p.5). From the point of view of music (specifically performance), Miranda and Wanderley broadly describe musical gestures as “*...any human action used to generate sounds*” (Miranda and Wanderley, 2006, p.5).

All three definitions must be kept in mind during the design and implementation of a mobile music-based app. The gestures that are used must be capable of effectively expressing musical ideas, while at the same time being intuitive in their execution by the user.

The challenge for designing an interactive music technology system is that there is no standardized set of gestures for the performance of computer music. As such, users of a system may have different ideas than the designer of the meaning of gestures in that system. This is in contrast to acoustic instruments, whose physical characteristics lead to the use of gestures by musicians that are constrained to the design of the instrument.

Another design challenge is that there is a lack of gesture standardization across the possible hardware platforms available for development. Yet a further challenge is that each individual user, compared to other users, has differing finger and hand sizes with which to perform the gesture (Bachl et al., 2010). This makes it possible for users to experience different outcomes when using the same gestures.

The use of a single hardware device such as Apple’s iPad partially solves the problem of gesture standardization. Apple provides a standardized set of gestures for the iOS operating system that all developers are able to implement for various mobile apps.

According to the Apple Human Interface Guidelines, the use of gestures in the iOS environment gives users “...a greater affinity for, and sense of control over, the objects they see onscreen, because they’re able to touch them without using an intermediary, such as a mouse” (Apple, 2012e). However, just because an app may use multi-touch gestures does not inherently make it intuitive. If the user interaction design is unfamiliar to a user based on their primary experiences, they will find the gestures unintuitive. Additionally, as users generally prefer simple gestures, if the user is required to use excessive physical effort to complete a gesture, then the interaction experience will be considered less meaningful and rewarding (Ingram et al., 2012).

The use of standard Apple gestures is helpful for mobile applications. Atkins-Wakefield (2012) showed that Apple gestures are in fact inherently intuitive, and therefore applications developed on Apple’s iOS platform, which uses these gestures, have a high likelihood of being operated intuitively by a user. An overview of available Apple gestures is given in Section 4.1.2. For a thorough discussion on the use of gestures in interactive music, the reader is encouraged to read “Gesture-Music” (Cadoz and Wanderley, 2000).

2.8.4 Multi-touch Interaction

Treadaway states that “*The hands are our primary interface with the world, and provide the brain with rich sensory information which is instrumental in building imagination and novel ideas*” (Treadaway, 2009, p.185). As such, multi-touch user interaction is one possible method for an interactive music compositional system, and offers a possibility for novel musical interactions apart from traditional *Windows Icons Menu Pointer* interfaces operated by mouse and keyboard control. Multi-touch

interaction is capable of allowing the user to utilize natural, intuitive gestures to control a variety of possible musical parameters (Wöldecke et al., 2012). Jordà describes the limitations of WIMP interfaces by *stating* “...there are limits to what can be efficiently achieved in real-time by means of a mouse and a computer keyboard” (Jordà 2003, p.4). Alternatively, multi-touch devices allow more intimate control over complex structures that can exist in music by virtue of employing the use of hand gestures for control information input (Brunner, 2009). As many musicians tend to instinctively use gestures when either recalling or producing sounds (Hauelsen and Knösche, 2001), multi-touch gestural input seems a natural choice for the control of musical parameters.

Many mobile devices, including Apple’s series of iPads, have multi-touch capability. Given these devices’ ubiquity, portability, and computing power; they are well suited for music software development.

The iPad is also a widely available computing device. As of October 23, 2013, Apple has sold 170 Million iPads (Hughes, 2013). Additionally, Apple states that there have been over 50 billion downloads from the Apple App Store as of May 16th, 2013 (Apple 2013a). Part of the popularity of Apple’s mobile computing platforms is that the devices utilize simple multi-touch gestures that represent physical metaphors (Selker, 2008).

2.8.5 The iPad and Mobile Music

In addition to providing multi-touch capabilities, the iPad has a variety of other of sensors, data input, and interaction methods (Apple, 2013c), including:

- Three-axis gyroscope
- Accelerometer
- Ambient light sensor
- Wi-Fi
- Digital Compass
- GPS (on cellular model)
- Photo/video recording
- Lightning Connector
- 3.5mm headphone mini-jack
- Built-in speaker
- Microphone

As such, the iPad offers numerous possible input modalities, potentially providing a rich set of mapping possibilities for controlling musical parameters.

According to Xambó, et al., many multi-touch music apps, some of which are surveyed at the end of this chapter, exist for a variety of reasons (Xambó et al., 2011):

- Popularity and ubiquity of personal and shared multi-touch devices
- Ease of development for the available devices
- Consumer interest in the creative products.

Geiger (2006) describes the requirements that a mobile interactive music app should have:

1. It should remain as one piece (the mobile device); not a collection of controllers and synthesis engines
2. It should stay a portable instrument
3. It should have an interface that maximizes control and gives immediate feedback
4. It should be a “...*learn-able and master-able instrument*”

(Geiger, 2006).

For widespread use amongst people who may or may not have a technical background or similar skills, the app must be ready to use as soon as the user launches it. It should not require any external hardware or outside software. As the iPad is a

single hardware device that does not rely on any external peripherals and is exceptionally portable, points 1 and 2 are immediately satisfied.

According to Geiger, one characteristic of standard multi-touch screens is that they do not provide kinaesthetic feedback, unlike traditional acoustic instruments. Since feedback is an important part of an interactive system, not only should the user have auditory feedback, but visual feedback through the use of computer graphics as well, substituting for the lack of kinaesthetic feedback.

In the following section, a variety of interactive music applications for the iOS platform are reviewed. In addition, the reader is encouraged to read “A Quantitative Review of Mappings in Musical iOS Applications” (Kell and Wanderley, 2013), in which the authors examined the mappings and metaphors of 337 music creation apps.

2.9 iOS App Examples

When designing an interactive computer music-system, it is important that the system effectively makes use of the specific hardware and software capabilities that the computer offers (Magnusson and Medietta, 2007). In the case of mobile touch-screen devices such as the iPad, it is important that an interactive-music app takes advantage of the large amount of multi-touch screen real-estate. Carlson and Wang further comment on the ubiquity of traditional musical interfaces in software synthesis applications: “...*there are few that enable interactions that go beyond the standard set of knobs, sliders, XY control surfaces, and single waveform displays*” (Carlson and Wang, 2011, p.2). Such interfaces originated in the analogue realm of music technology, and were

further used in desktop and laptop computers utilizing WIMP interfaces. This is an example of *skeuomorphism* (Gross, 2012), and is discussed further in Section 2.9.10.

With the emergence of multi-touch gestural devices, it is now more feasible to develop and design new paradigms for musical interaction. Oh et al. (2009) state that the ubiquity, mobility, and accessibility of mobile phone devices have begun to break down the barriers of traditional musical experiences. The authors note that there is a “...blurring of once distinctive roles of a composer, performer, and audience, as one can now more easily partake in the integrated music making experience” (p.86). In some instances, users may be performing and composing at the same time, or sequencing and playing a game, etc...

Given this, many of the apps reviewed below do not necessarily fall into a clearly defined category. In fact, it has been a considerable challenge to categorize them. While a complete review of all available music-related iOS apps and the unique interaction paradigms they embody is beyond the scope of this thesis, the following approximate categories have been defined:

- Synthesizers
- Generative/Immersive
- Production
- Sequencers
- Effects

As will be seen, many of the reviewed apps fall into several of the above categories, further evidence that computers, and mobile platforms in particular, are drastically changing the way that music is composed, performed, and accessed. (Note: all images are screenshots captured from an iOS device unless otherwise stated).

2.9.1 iMaschine

- Production
- Sequencer

According to its page on the App Store, *iMaschine* is an “*intuitive beat sketchpad perfect for developing song ideas, anytime, anywhere*” (Apple, 2012g, online).

The app has two interaction modes: a drum-pad view (Figure 2.10), and two separated keyboard views (Figure 2.11). The app is also able to record; letting the user record and edit their own samples to use, in addition to the samples included with the app. Screenshots of the various modes are shown below.



Fig. 2.10
iMaschine Drum Pad View



Fig. 2.11
iMaschine Keyboard View

In addition to the multiple modes for interaction, the app has an onboard mixer, with the ability to send audio to two different effects processors. Although the user interface does not allow for intricate compositional development, it is possible to create a musical sketch for song material that can be further developed later.

2.9.2 Figure

- Production
- Sequencer/Tracker

According to the developers' (Reason) description on the App Store *Figure* is the “...*fun music-making app for instant inspiration*” (Apple, 2012b, online). The description goes on to say that “*Figure will have you making music within seconds yet is deep enough for endless play on a transatlantic flight*”. From these descriptions, it is clear that the app is also more of a musical sketchpad, similar to *iMaschine*, as opposed to a complete composition environment. Both applications allow the user to develop harmonic and rhythmic ideas, but do not provide a mechanism for meaningful control of timbre characteristics or DSP processes. The interface to *Figure* is shown below in Figure 2.12.



Fig. 2.12
Reason Figure

2.9.3 The Akai SynthStation

- Synthesizer
- Production
- Sequencer/Tracker

According to its page on the App Store, the *Akai SynthStation* “...transforms your iPhone, iPod Touch, or iPad into a mobile music production studio for mobile music creation” (Apple, 2012k, online). *Akai* proudly claims that the *SynthStation* is loaded with features. While true, the menu layout format does not lend itself well to composing on a structural level. It appears to be more suited to jotting down quick musical ideas that can be recreated in a more complete composing environment later.

Similar to the *Figure* app, the *Akai SynthStation* does not take full advantage of the iOS multi-touch capabilities.

The app does give control over multiple synthesis parameters, but again; this is based on a menu system, as well as sliders and knobs. As such, users are not able to truly take advantage of iOS's multi-touch architecture. The three views of the application are shown in Figures 2.13-2.15.



Fig. 2.13
Akai SynthStation Performance View

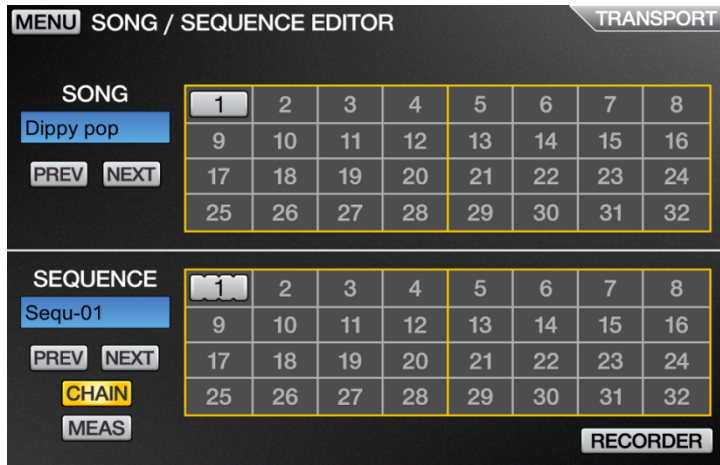


Fig. 2.14
Akai SynthStation Sequence Editing View

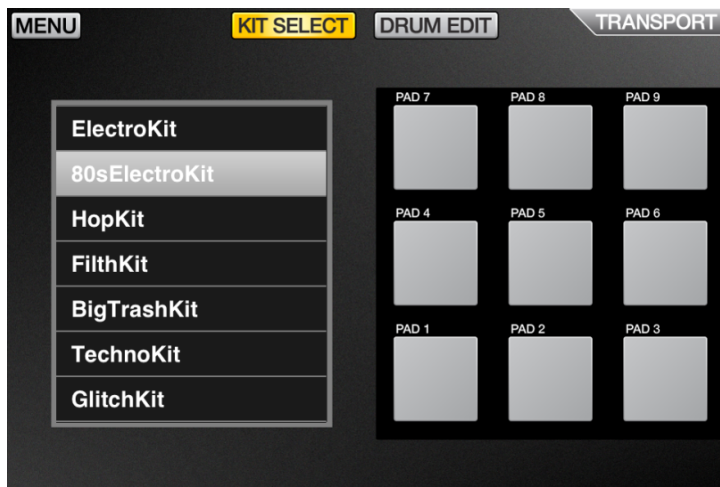


Fig. 2.15
Akai SynthStation Drum Kit View

2.9.4 iKassoliator

- Synthesizer

A highly commercially successful iOS app is the *iKaossilator*, whose design is based on a hardware device of the same name. According to its App Store Page (Apple, 2012d), it was at one time the internationally number one selling music app on the App Store. The app reached a number 1 sales mark in 8 countries (Korg, 2013).

iKaossilator allows the user to use multi-touch gestures to play a wide palette of instruments either independently, or along with a variety of included loops. The user is then able to record their actions, and to continue to record loops on top of each other. In addition to controlling instrument timbres, the user is also able to change the scale, tempo, and note length they are performing.

According to Korg's description, the app is capable of being used by both musicians and non-musicians. The question is whether the app is for composing or performing? According to Korg's website, the "*...loop sequencer and Mix Play capability give you total freedom for creating tracks and performing live*" (Korg, 2013). Note the allusion to both a compositional and performance paradigm. However, even if the user is able to compose a sequence using the app, they will most likely export the file to a conventional computer-based Digital Audio Workstation (DAW), such as Ableton Live or Pro Tools for further editing, sequencing, and production. The app could best be described as both an instrument for live performance and a simple loop generator/sequencer, but is in itself not a compositional app. The interface for the iKaossilator is shown below in Figure 2.16.

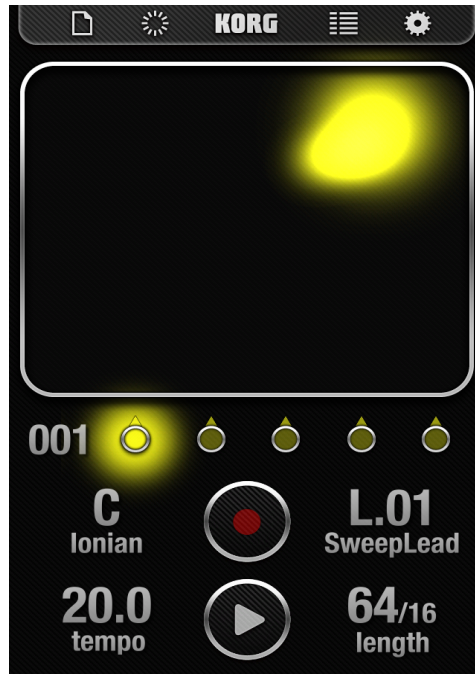


Fig. 2.16
iKaossilator

2.9.5 The Animoog

- Synthesizer

The *Animoog* was the first music app to hit the number one selling spot in the App Store on the day of its release. It was also named by Apple as one of the best apps of 2012 on the App Store page (Apple, 2012a).

While difficult to use on the limited screen-space of an iPhone, the Animoog for iPad app is much more user-friendly in terms of user control of the various synthesis parameters. It would seem that the iPad would be a more appropriate platform for this application (the app is available on both the iPhone and the iPad). The interface for Animoog is shown below in Figures 2.17 and 2.18.



Fig. 2.17
Animoog iPhone App



Fig. 2.18
Animoog iPad App
 (Apple 2012a, online)

2.9.6 The Filtatron

- Synthesizer

The *Moog Filtatron* is described on its App Store page (Apple, 2011b, online) as a “...real-time audio effects suite and powerful studio tool for your iPhone or iPod Touch”. The user is able to alter, in real time, sounds from the iDevice’s line/microphone input, the app’s sampler, or the app’s built-in oscillator. Though the user is able to alter the sound in real time via the multi-touch interface of the iDevice, the application is not intended as a live performance tool; it is intended as an augmentation to the user’s studio/audio-production workflow. While such an app may assist in the making of a composition (the app includes a generous selection of quality presets), it is not an app in which the sole focus of the user is the *construction* of a composition. The three main views of Filtatron are shown below in Figures 2.19-2.21.



Fig. 2.19
Filtatron Main View



Fig. 2.20
Filtatron Sample
Editor View

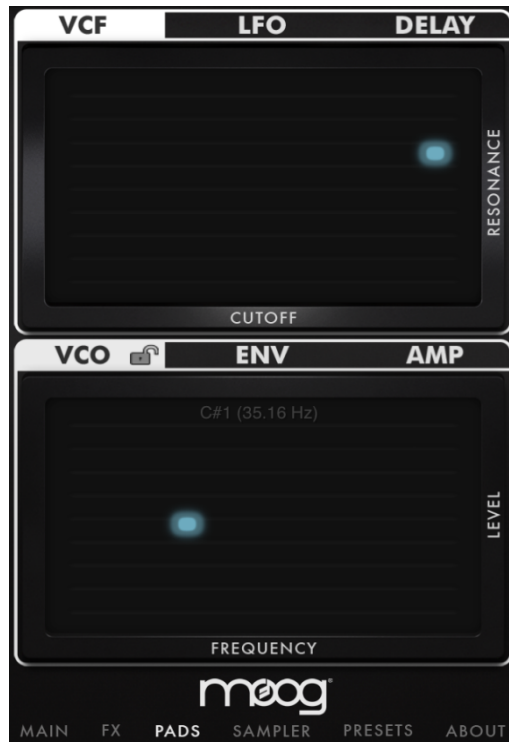


Fig. 2.21
Filtatron Pad View

2.9.7 GrainProc

- Synthesizer
- Generative/Immersive

GrainProc is described on its App Store page (Apple, 2012c, online) as “...providing an expressive control surface of granular manipulation of real-time audio input, well suited for sonic sculpting and self-accompaniment”. The app is designed for quick and intuitive control by a user’s fingers. The user is able to control four granular

synthesis parameters via level controls that resemble faders, but stripped of their skeumorphic design elements. A waveform view is shown in the bottom left of the screen, as well as a “Freeze” button that stops audio.

The waveforms display provides visual feedback to the user as they are altering these parameters (see screenshot below). In a video on the developers’ website, control with toes is also shown to be possible. This is an especially interesting capability of the app, as those without hand movement are still able to access the capabilities of the touch-screen interface. The app is available both in iPad and iPhone/iPod Touch versions, and is shown below in Figure 2.22.



Fig 2.22
GrainProc

2.9.8 Reactable Mobile

- Generative/Immersive
- Production
- Sequencer

The *Reactable Mobile* is based on the hardware version of *Reactable*, a round table-based Tangible User Interface that lets multiple users share control of the music by “...caressing, rotating, and moving physical artefacts” (Jordà et al., 2007 p.142). The *Reactable* is shown below in Figure 2.23.

The *Reactable Mobile*'s App Store Page says that the user is able to “*Create and improvise music in an intuitive and visual way*” (Apple, 2012j). The app is rated an average of four stars, based on input from 205 users.

The *Reactable Mobile* app lets users control both high-level interactions between various musical objects (such as sequencers, loops, synthesizers, and filters) and the low-level parameters of each component object. In both aspects, the application makes effective use of the iPad's multi-touch capabilities. The *Reactable Mobile* App is shown in Figure 2.24.



Fig. 2.23
The Reactable
(Music Technology Group, online)

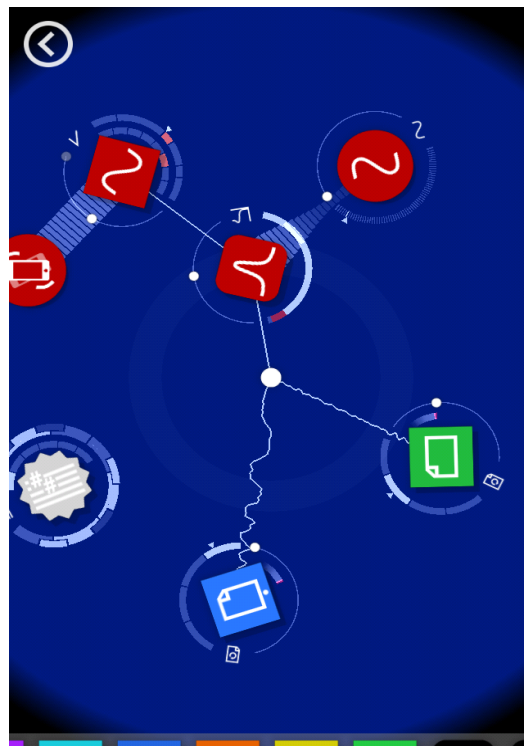


Fig. 2.24
Reactable Mobile App

2.9.9 NodeBeat

- Synthesizer
- Generative/Immersive
- Production
- Sequencer

NodeBeat is advertised as an “...intuitive and fun visual music app for all ages” (Apple, 2012i, online). It is also available on the BlackBerry, Amazon and Android platforms and as a desktop version. The user is able to generate their own original music, or listen to generative music scenes included in the app.

A screenshot of the app is shown below in Figure 2.25. It makes use of the iOS multi-touch architecture effectively, resulting in a highly intuitive application interface. The user controls the output of the app with a variety of coloured and connected nodes. These nodes are *Generators* and *Notes*. The developers state that “*Generators pulse and play notes within proximity. A Note is played in sequence, based on the distance it is from its connected Generator. Pause Notes to create your own beats or let them roam free to have them generate their own*” (AffinityBlue, 2013, online). Additionally, users are able to modify audio parameters such as echo, attack, decay, and release. Panning and tempo may also be adjusted. (AffinityBlue, 2013).



Fig. 2.25
NodeBeat

2.9.10 csGrain

- Synthesizer

csGrain is the first iOS app that makes use of the Csound for iOS SDK (discussed further in Chapter 4), and its maker, Boulanger Labs, hopes to release more in the near future. A granular synthesis based app, it uses a variety of Csound opcodes to granularize loaded audio files, as well as live input from the iPad microphone. The app also includes effects such as a pitch-shifter, ring modulator, chorus, flanger, tap delay, reverse, high-pass filter, low-pass filter, stereo waveguide reverb, and output mix/submix (Boulanger Labs, 2012). A screenshot of the primary app view is shown below in Figure 2.26.

While versatile in terms of the synthesis performance, the app does not make full use of the iPad’s multi-touch capabilities which is discussed further below.

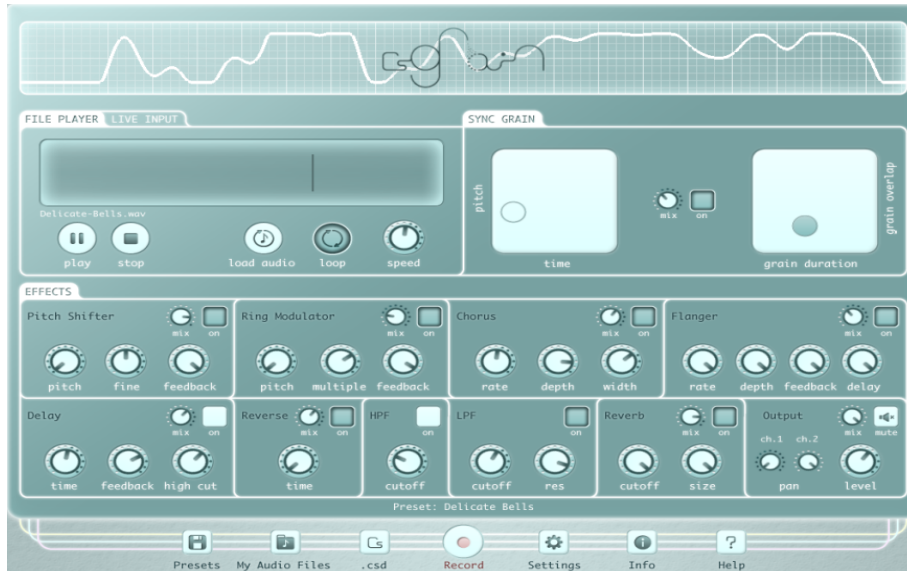


Fig. 2.26
csGrain

As can be seen, the synthesis parameters are controlled via touch-screen representations of On/Off switches and rotary knobs. Thus the app uses skeuomorphs, “...holdovers from previous material construction requirements of an artifact” (Gross, 2012, p.1). In analogue synthesizers, parameter changes necessitated the use of rotary knobs, switches, and faders for manual control. The knobs, switches, and faders, being physical objects, were intrinsically mapped to the parameters they controlled via electrical circuitry. Thus, the user had access to every possible synthesis parameter via a physical control.

Given that multi-touch based computer systems allow for new metaphors for user interaction, the use of rotary knobs or faders for synthesis control is no longer needed, as they are not a *necessary* component for parameter control. Additionally, touch-screen based pictorial representations of these physical control elements do not provide the user with the tactile feedback that they would on an analogue system. When a user twists a knob, changes a switch, or moves a fader in a certain direction, the parameter being controlled responds accordingly due to the physical correlation between the knob or slider and the controlled parameter. As such, knobs and faders provide the user with *affordances*, fundamental properties that determine how an object should be used (Norman, 2001). A knob affords the action of turning; a fader affords the action of a sliding motion.

In a multi-touch computer system, this is not the case: as parameter mappings are completely up to the designer of the interactive system, he or she is able to create mappings and metaphors for user interaction of parameters that are based on the computing device on which the interactive system runs. Therefore it is unlikely that the best representation would use methods of interaction based on systems that work on analogue electro-mechanical principles.

Due to the small size of the On/Off switches and rotary knobs compared to the rest of the app, they are difficult for the user to manipulate quickly, thus inhibiting the ease with which the user may interact with the app. While the app does give the user explicit control over all possible synthesis parameters, it does so in a way that is ineffective for multi-touch user interaction, as the UI is using design metaphors that are

inherited from analogue synthesis systems; rather than utilizing the rich multi-touch gestures afforded by the iPad's touch-screen.

2.9.11 Portable Dandy

- Sequencer

Another app that makes use of the Mobile Sound API is *Portable Dandy*. *Portable Dandy* is a simple sequencing app that was inspired by Dandy Desmond's 1939 composition "Magnetic Loops for 15 Tape Decks" (barefoot-coders, 2012). Users click on buttons that signify audio loops. These audio loops can be modified by filters, a ring-modulator, and a pitch-jog wheel, which are controlled via faders. Users are able to use the pre-loaded samples, or add their own samples using iTunes. The interface for *Portable Dandy* is shown in Figure 2.27.

While the app does allow users to interactively sequence sound files and modify them in real-time, it does not allow for the composition of music, as there is no record and playback functionality.

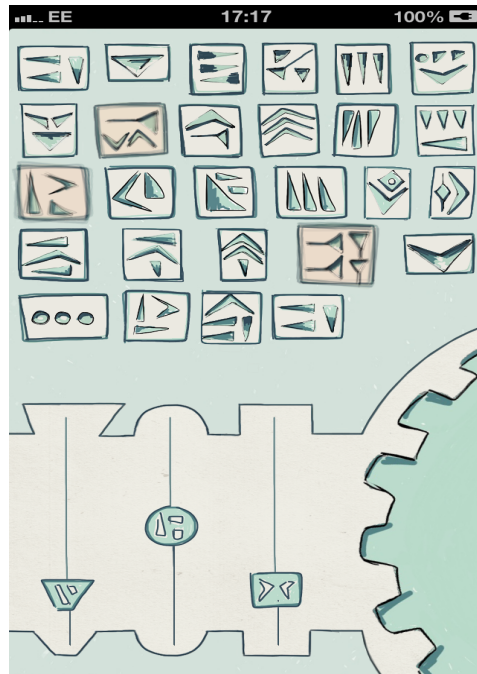


Figure 2.27
Dandy

2.9.11 iPulsaret

- Synthesizer

iPulsaret is another app that uses the Mobile Csound API. The user is able to modify a variety of granular synthesis parameters in real time:

- Grain Amplitude
- Grain Density
- Grain Length
- Grain Amplitude Masking
- Grain Frequency Modulation
- Stereo Width
- Random Density
- Random Length
- Random Frequency (Semitones)
- Grain Index Frequency Modulation

Like many apps, however, these parameters are controlled via virtual sliders, soft buttons, and knobs. Additionally, the app sticks to a traditional method of performance interaction via a virtual keyboard. Kell and Wanderley speculate that the reason many iOS music creation apps make use of keyboards is that the keyboard is such a well-known metaphor for musical interaction (Kell and Wanderley, 2013). Although the app makes use of a powerful audio engine capable of producing many interesting sounds, the choice of interaction methods means that users are confined to traditional ways of interacting with musical material.

While *iPulsaret* contains a powerful audio engine, the app does not take full advantage of the available multi-touch screen real-estate that the iPad offers. The interface for *iPulsaret* is shown in Figure 2.28.



2.28
iPulsaret

2.9.12 Grain Science

- Synthesizer

Grain Science is a granular synthesizer for both the iPhone and the iPad. The developers describe it as being able to create “...everything from crunchy basslines to spooky soundscapes” (Woojjjuice, 2012, online). The user is able modify an FX chain for sound modification, generate arpeggios, and create custom loop patterns using a 32-step sequencer. Of most interest is that the app allows for arbitrary parameter mapping between the synthesis parameters and XY pads, pitch/mod wheels, or an external MIDI controller.

The only obvious limiting factor of the app is the keyboard layout at the bottom half of the screen. By including a traditional music interface (the piano keyboard), the app is not taking full advantage of the available touch screen capabilities when interacting with the harmonic and rhythmic structure of a musical piece. The interface for *Grain Science* is shown in Figure 2.29.

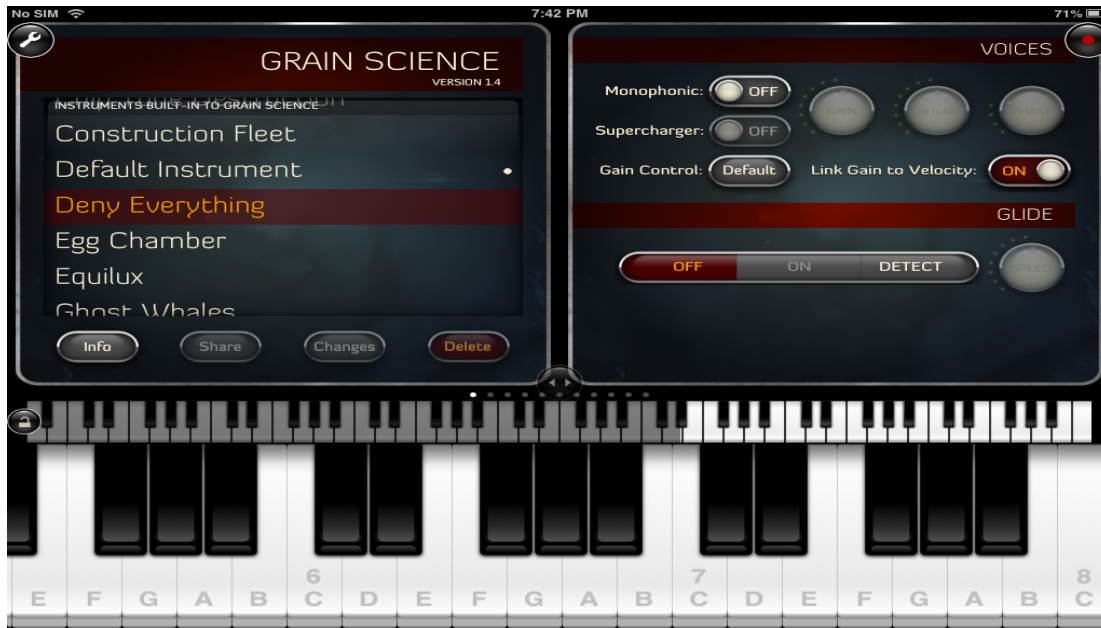


Fig. 2.29
Grain Science

2.9.13 MegaCurtis (Lite)

- Synthesizer

MegaCurtis (Lite) is described by its developers on its App Store page as being able to “*Turn any recording into a unique synthesizer!*” (Apple, 2012h, online). The app uses a combination of granular and wavetable synthesis to alter either a live microphone input or recorded samples. The sample view is shown in Figure 2.30 below. The user is also able to alter the amplitude, oscillator, envelope, and key of the sampled material, shown in Figure 2.31 below.

While the app is capable of generating rather interesting sounds, it also relies on a keyboard interface. The user is once again restricted to the use of a traditional acoustic instrument paradigm for interacting with synthesis algorithms that are capable of

generating new and unexpected sounds, but are limited by the fact that the user is forced to approach them with the interaction mind-set of a pianist.

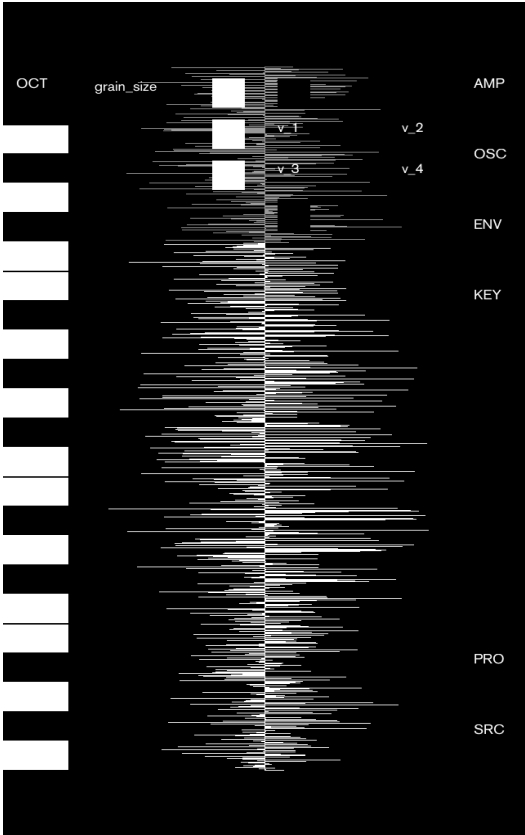


Fig. 2.30
MegaCurtisLite
Performance View

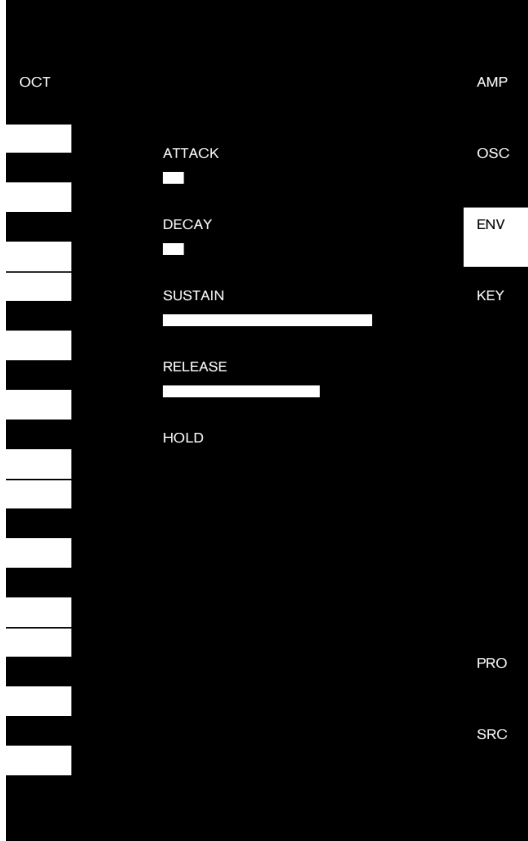


Fig. 2.31
MegaCurtisLite
Editing View

2.9.14 CP1919

- Synthesizer

CP1919 is an app created by The Strange Agency, and is described on its App Store page as a physics-based fluid simulation that “...drives an additive synthesizer [...]. The multi-touch interface lets you control the fluid with your fingers, exciting the oscillators like strings on a liquid harp” (Apple, 2011a, online).

The app seems to succeed in finding a balance between traditional modes of musical performance (keyboard at bottom of screen) and parameter control (virtual sliders at top of the screen), and the use of multi-touch gestures. Additionally, there are two track-pad controls in the bottom left of the screen for controlling LFO parameters. The most interesting interface aspect, however, is the fluid mesh for interacting with the additive synthesizer bank. This appears to be a refreshing metaphor of interaction using a combination of multi-touch gestures and synthesis parameters. The interface is shown in Figure 2.32 below.

The app is capable of recording, but the user would need to import the material into a DAW for further editing in order to create a finished composition. Therefore, it is not suitable for use as a standalone composition app.

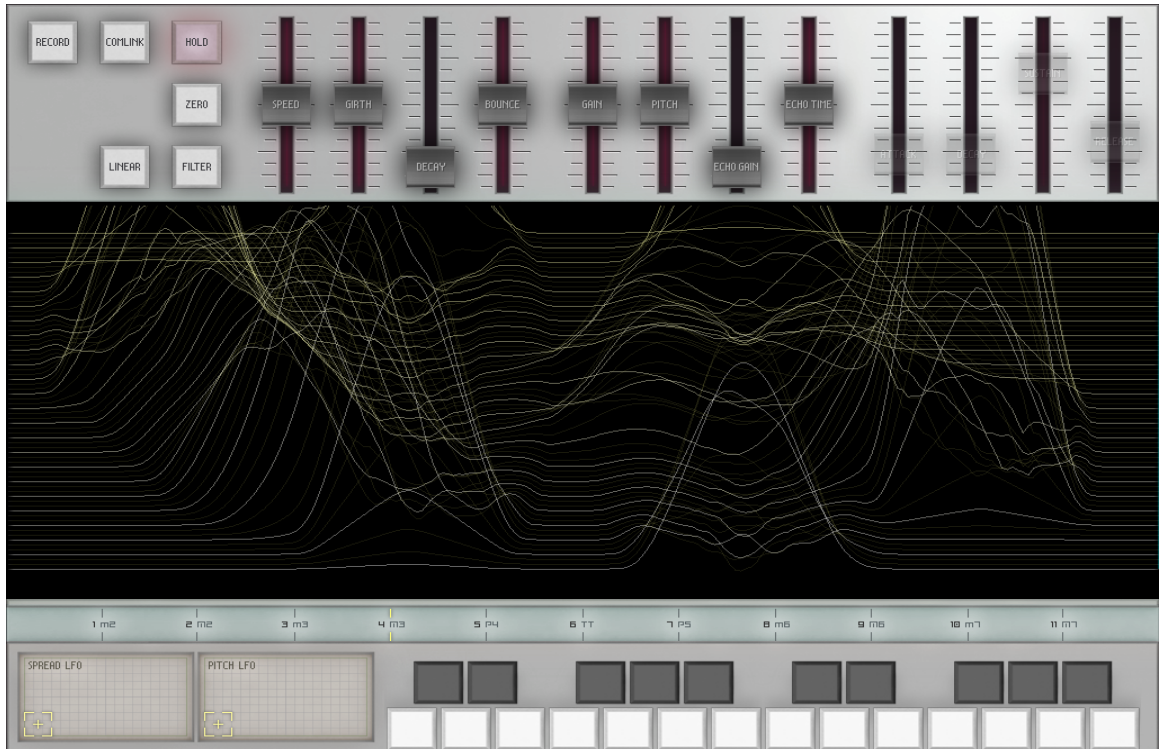


Fig. 2.32
CP1919

2.9.15 TC-11

- Synthesizer
- Sequencer

TC-11 is an iPad app created by Kevin Schlei that is “...*built around the idea that any synthesis parameter should be controllable by any multi-point controller*” (Schlei, 2012, p.1).

The app comes with several pre-programmed synthesis templates, consisting of a variety of DSP modules such as envelope generators, step sequencers, and low-frequency oscillators. The user is able to modify the routing of these modules in each

template, and is allowed to save them as a new custom patch for later use. The Pure Data for the iOS library, *libpd*, is used as the synthesis engine. The patching view is shown below in Figure 2.33.



Fig. 2.33
TC-11 Patching View

The synthesizer is controlled via multi-point controllers that analyse the raw multi-touch information generated by the user via the iPad touch-screen. There are two types of these controllers: single and group touch; based on the number of touches detected by the iPad's screen. As users move their hands across the iPad's screen, they "...see graphic representations of the multi-point controllers in use, such as connecting lines, circles, and angle vertices" (Schlei, 2012, p.3). Additionally, the app makes use of the iPad's accelerometer, gyroscope, and compass as controllers for the synthesis engine.

Users are able to record music with the app's on-board transport. The interaction view is shown in Figure 2.34 below.

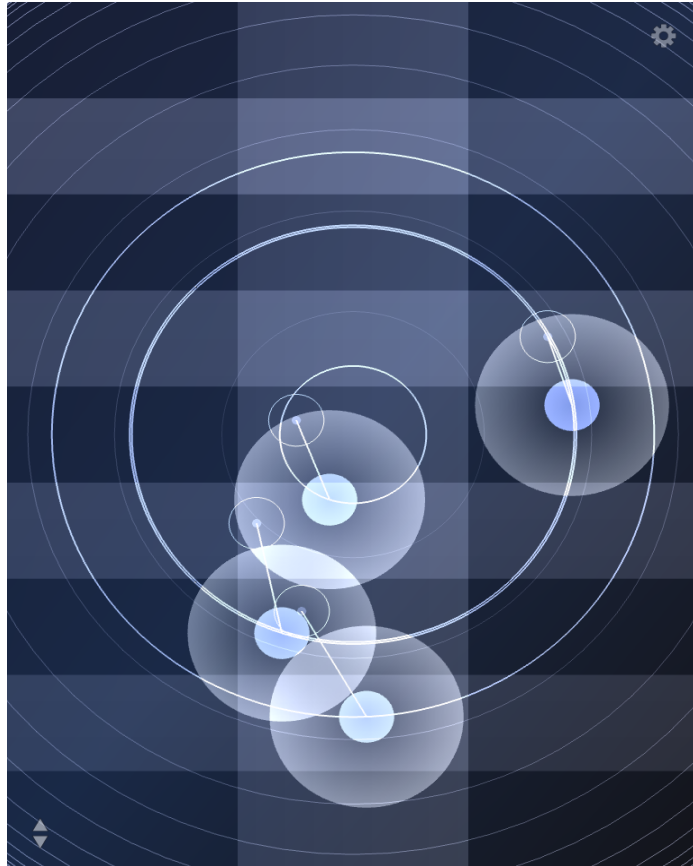


Fig. 2.34
TC-11 Interaction View

Given that *TC-11* comes with a 68-page user manual, it is not well suited to the musically inexperienced user. Additionally, some knowledge of synthesis is necessary to make full use of the app. That being said, *TC-11* is a powerful app that a dedicated user will be able to make interesting compositions with, thanks to its well-implemented use of the available screen-space and multi-touch gestures of the iPad.

2.9.16 Borderlands

- Synthesizer

Borderlands is an iPad app created by CCRMA student Chris Carlson in (Carlson and Wang, 2011). The app has a single window that displays the waveforms of audio samples, both included audio files and user-supplied files. The user is able to use multi-touch gestures to zoom in and out of desired audio files. When the user double taps on an audio waveform, a circle enclosing an animated waveform and red and white dots appears. This object granulates whatever part of the audio file it is on top of. When the user double-taps on the object, a series of circles appear that allow the user to modify the volume, LFO frequency, pitch, grain overlap, duration in milliseconds of the grain, and the number of voices of the grain. The user is also able to change the length and width of the screen that the grain object will sample from. The grain objects are shown in Figure 2.35 below.

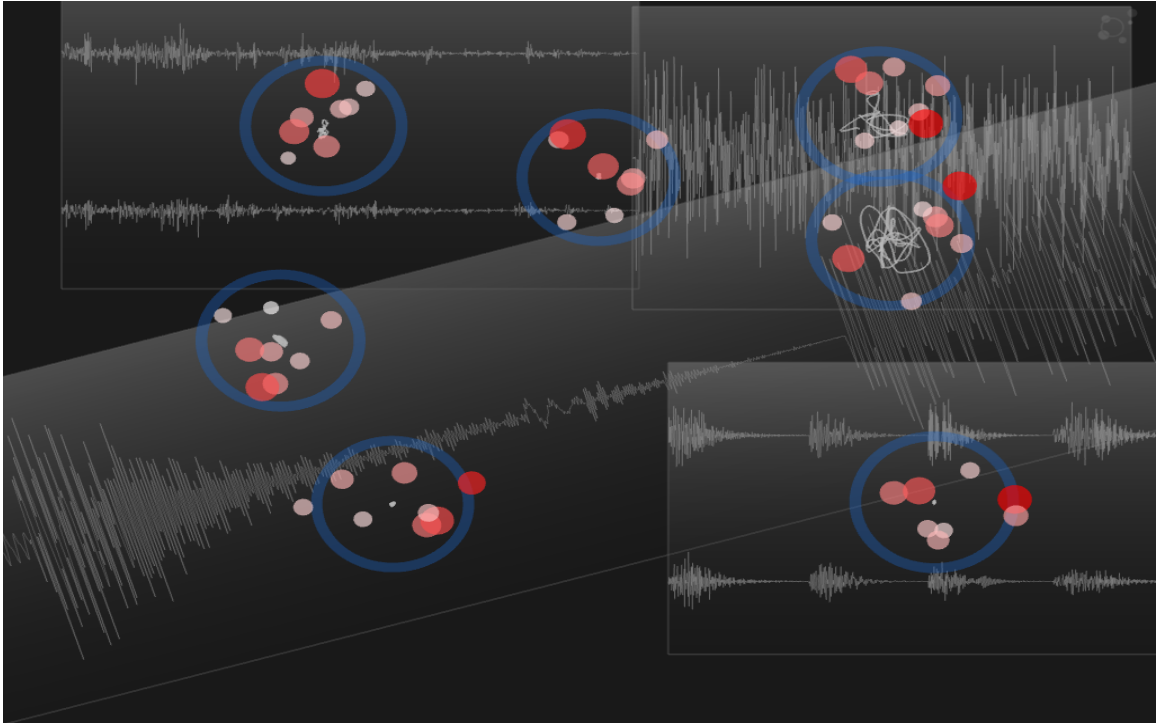


Fig. 2.35
Borderlands,
showing audio grains

Another event that occurs when the user double taps the grain object is the appearance of a small toolbar at the top of the screen that allows for control of meta-parameters. This is shown in Figure 2.36 below.

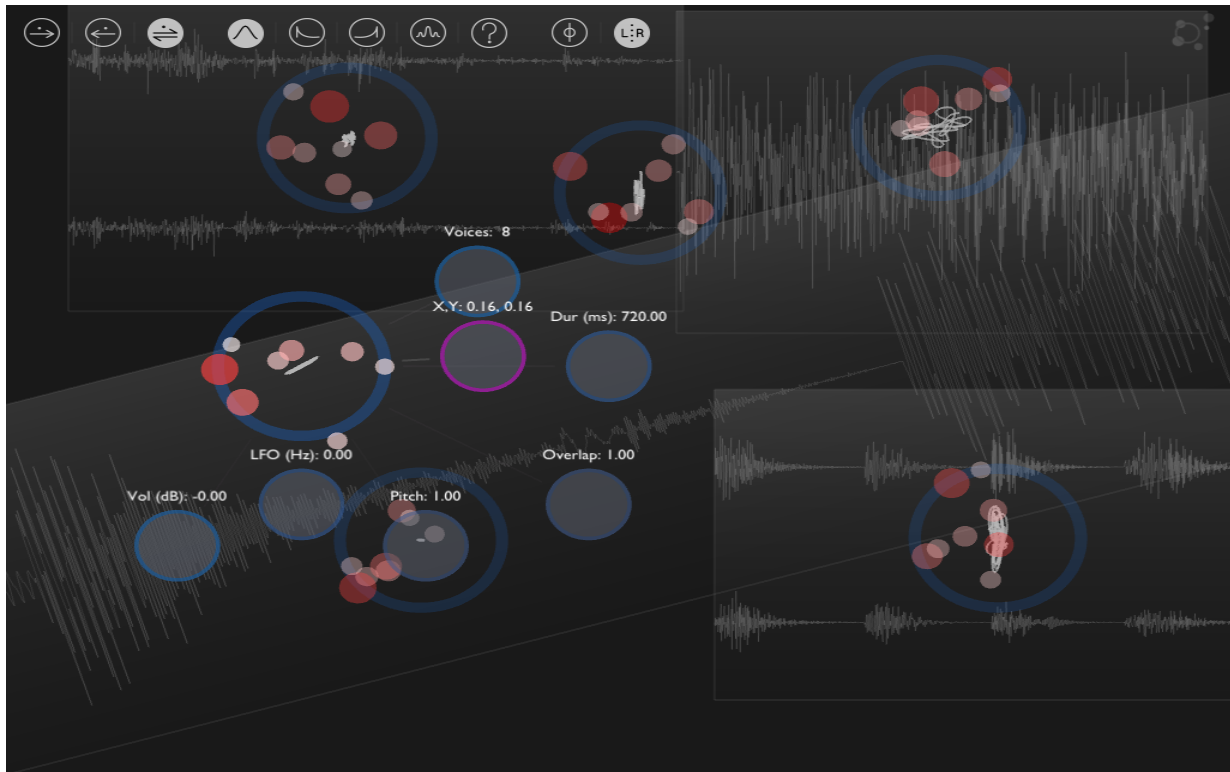


Fig. 2.36
Borderlands,
showing grain parameters and editing toolbar

As the app uses granular synthesis as a method of interacting and composing with sound, users are able to focus on creating large-scale sound compositions rather than note-level ones, providing a more accessible route to music composition.

Due to the absence of traditional instrumental metaphors the Borderlands app appears makes the most effective use of the iPad's multi-touch capabilities in the realm of musical synthesis control out of all the reviewed apps.

2.10 App Conclusions

The majority of current music applications for iOS are not taking enough advantage of the platform, both in terms of utilizing multi-touch; and as a consequence, not doing enough to serve a compositional need. Most apps that are composition oriented do not allow for much more than the ability to sketch out simple rhythmic, harmonic, and melodic ideas that can be modified at a later time. The majority of apps do not allow for intricate and intimate control of timbral and DSP parameters. As such, they are not particularly well suited to perform as high-level, accessible compositional tools.

2.11 Interactive Composition Systems or, now how do we Make Music?

As discussed in detail in Section 2.7, Composer Joel Chadabe defines interactive composition as a “*method for using performable, real-time computer music systems in composing and performing music*” (Chadabe, 1984, p.22). The definition itself gives some idea as to the difficulty in defining what interactive composition is. In the conventional tradition of Western music composition, a composer uses notation to individually write out each part of a composition. The composer may go through many revisions until settling on the final musical work. The composer may then rehearse the piece with a group of musicians, who later perform the work for a live audience. In other genres, such as rock, compositions may be written by a member of the band and then rehearsed with the rest of the group, or the entire group will write the composition in a group setting. Regardless of genre, the same principle applies: composers write the music, a group of musicians play it, and an audience will listen.

A new dynamic was afforded with the advent of computers. New methods of interaction meant that the lines between the definition of composer, performer, and audience member were increasingly being blurred. In an overview on the topic, Paul Lansky states that computers have created an additional two spheres in the realm of musical interaction: those of the *sound-giver* and the *instrument-builder* (Lansky, 1990).

Lansky states that a *sound giver* can be on a spectrum consisting of wanting to share musical experiences that one appreciates at one end, and sharing one's newest composition at the other. Mobile apps, such as the ones described in Section 2.9, often accomplish these goals. Some apps allow users to document sound and music from their everyday experiences and share them with friends via social media networks, such as Facebook and Twitter. Many music-centric apps also allow users to upload their creations to these social networks.

An *instrument builder* is an individual or group of individuals who designs and builds sound generating hardware and software. Composers can use these tools in the writing and performing of their musical works. A digital instrument, in some cases, such as one containing pre-sequenced material, may be considered in itself to be a composition or part of a composition. This occurs most often if the hardware and/or software created by the instrument builder are to be used for a single performance of a composition. Lansky describes this by saying "*Playing someone else's instruments becomes a form of playing someone else's composition*" (Lansky, 1990, p.4). When describing their own approach to electronic composition, Settel and Lippe state that "*the instrument is present in the composition process at its very inception*" (Settel and Lippe, 2003, p.4).

However, an instrument builder may also create hardware and/or software that can be used by a multitude of other composers that would let them write additional works. If this is the case, the composers' music is under the influence of what the instrument designer determines to be important musical considerations, which influence the design and construction of the instrument.

This can be seen throughout history. For example, the piano was invented out of a desire to improve the harpsichord by introducing note-independent dynamic range to keyboard playing. The music of composers such as Chopin or Debussy was dependent on and influenced by this development. Additionally, the music of Charlie Parker or Jimmy Hendrix was inspired by and made possible through the respective inventions of the saxophone and electric guitar (Settel and Lippe, 2003).

2.12 Problems with Digital Musical Instruments

As has been shown throughout this chapter, technology has had a profound impact on the way in which music is composed, performed, and accessed. With today's computer technology, it is possible for a composer to write a symphony from their bedroom while immediately hearing what it will sound like, for an artist to perform new genres of electronic music live, and for fans to listen to any style of music they choose at any time through their mobile devices.

Cook offers some suggestions as to what influences the design and construction of digital music interfaces (Cook, 2011):

- Music the designer likes
- Music the designer wants to make
- Instruments the designer already knows how to play
- The artists the designer wants to work with
- Available technologies

Digital musical interfaces are designed and created based on the preferences and needs of the designer. A wide variety of outcomes can be achieved due to the flexibility of modern computing technology. This is in contrast to “traditional” musical instruments that have evolved over millennia based on the physical, acoustic properties of materials found in nature.

As flexible as they can be, digital instruments are not without their problems. One issue is reproducibility. Given the fast paced evolution of computer hardware and software, an instrument that is created with today’s technology may not be able to be recreated in a decade’s time, as the technology used to create the instrument may become outdated and obsolete. In fact, if someone other than the original designer wanted to create a specific digital musical instrument, they may find it difficult, as they may not have access to the schematics, hardware, algorithms, or source code used to create the original digital instrument. Tod Machover describes this by saying “*While we have been successful in designing controllers and interactions capable of virtuosity and subtlety, the best of these [...] have been customized for particular compositions, performances, or performers, and have not been standardized in a way that I associate with ‘instrumentality’*” (Machover, 2002, p.1). In other words, these digital musical interfaces do not have the same level of longevity as traditional instruments.

As discussed in Section 2.8.3, a solution to the second problem is through the use of a standard hardware platform that a digital musical instrument can run on. Consumer devices such as the iPad allow users to download multiple instances of a digital instrument across devices, thus encouraging longevity.

2.13 What can Music Technology teach HCI?

The field of Music Technology, in particular areas concerning music interaction design, has many opportunities to benefit the field of Human-Computer Interaction as a whole. Holland et al. (2013, p.2) illustrates this by saying “*As music is an evolutionary, deep-rooted, complex social activity, Music Interaction makes unusual demands beyond everyday verbal and mathematical matters, which can lead to inspirational or novel solutions of wider relevance to mainstream HCI*”. Additionally, Khooshabeh et al. state that musical interaction deals with research areas such as “*multi-modal input, analysis, and mapping of a complex array of human communication signals*”, all of which have applications in the larger field of HCI (Khoosabeh et al., 2005, p.2).

When someone is using computer technology to interact with music, they are interacting with a highly complex system. Users bring to the interaction all past experiences, emotions, tastes, and preferences in an attempt to create an artistic product. They want the experience to be intuitive and straightforward, without having to spend time learning every aspect of an interface.

According to Hurtienne and Blessing (2007, p.2), “*a technical system is intuitively usable if the users’ subconscious application of prior knowledge leads to*

effective interaction". The study of music interaction has much to offer the wider field of Human-Computer Interaction in terms of creating more intuitive interactions.

In many commercial computer applications, such as word processing and email, users are not pushing computers to the limits of their memory capacity or processing ability/speed. Musicians do, however, "... *push machines to their limits when it comes to expression and performance*" (Kirn, 2013, online). This is particularly true for touch-screen computers. Kirn further states that musicians are the "... *greatest test of every nuance of a touch display, every millisecond of latency, because they don't just use them as an interface: they use them as an instrument*" (Kirn, 2013, online). Roberts, Forbes, and Höllerer (2013, p.3) support this by stating "...*musical applications require the ability to control large parameter spaces concurrently and expressively.*"

In short, musicians are looking to use their computers as instruments, not as tools. According to Tanaka, "*The term tool implies that an apparatus takes on a specific task, utilitarian in nature, carried out in an efficient manner*" (Tanaka, 2000, p.389). Tools should be easy to use and accessible to anyone. A musical instrument, on the other hand, is not meant for use on a single task as a tool is. Rather, it "... *often changes context, withstanding changes of musical style played on it while maintaining its identity*" (Tanaka, 2000, p.390). Additionally, Bertelsen, Berinbjerg, and Pold state that "*Musical instruments are not just functionalistic means to well defined ends; exploring the instrument is an integral part of the creative process*" (Bertelsen, Berinbjerg, and Pold, 2007, p.234). Musical instruments are not supposed to be perfectly efficient tools to accomplish formulaic tasks. Rather, they are intended to allow composers and performers the ability to explore their capabilities for the sake of their artistic output.

Computing devices are becoming more intimately intertwined into the lives of billions of people. These users not only want their devices to be intuitive and easy to use, but require intimate control over a variety of processes and applications. The research conducted as part of this thesis will provide some insight into the ways in which people use computers to interact with music, as well as for the broader field of HCI research.

Chapter 3

Preliminary Study on Gestural Intuitiveness

“Unlike traditional notation, which requires serious study for a long period of time, a child can learn the relationship between drawn gestures and sound in minutes” – Curtis Roads, 2001, p.163

In a 2006 study, Godøy et al. investigated listeners’ associations of gestures with musical sounds. The researchers studied test subjects’ “sound-tracing” gestures, i.e., “...gestures that listeners make with a pen on a digital tablet in response to various sound fragments” (Godøy, Haga, and Jensenius, 2006, p.1). The goal of the 2006 study was similar to that of the study described in this chapter, and both ask test subjects to make gestures they believed “...corresponded well with the sounds they heard” (Godøy et al, 2006, p.3). Unlike the 2006 study, in which subjects used a pen on a digital tablet, subjects instead used the surface of an iPad to enact what they felt was an appropriate gestural response to the test sound.

3.1 Purpose

The purpose of this study was to determine what gestures users create in response to audio. Subjects enacted gestural shapes in response to listening to parameter changes of a granular synthesizer. Their interactions were video recorded, and used to determine how closely the gestures that users draw match the sounds they are listening to.

3.2 Test Overview

Test subjects sat at a desk with a powered-off iPad in front of them. Additionally, there was a computer and headphones for the playback of the audio samples. Subjects wore a pair of headphones to hear the samples. Before taking the test, they were given a hand-out stating the purpose of the test, instructions for taking the test, and a short questionnaire to fill out (see Appendices E and F).

The tester asked each subject if they had any questions as to the purpose of the test or regarding a specific instruction. The instructions given to each subject are listed below:

- You will hear several audio clips played in succession. Each clip will be played three times.
- When you hear the clip for the third time, please pretend that You are the one generating the sound clip by making a gesture on the provided iPad. You are allowed to make any kind of multi-touch gesture, using both hands to generate the gesture if you wish.

3.3 Test Subjects

Subjects were recruited from students and staff members of the University of York Audio Lab, as well as students from other academic departments on campus. The majority of student subjects (fourteen) were from the Department of Electronics; the remaining student subjects were from other departments (Linguistics, Literature, Environment, Education, and the Centre for Women's Studies). Four subjects were staff members of the Department of Electronics, and one subject was neither a student nor a staff member. Further details of the subject demographics are shown below:

- Average age: 28.2
- Age range: 21-47
- Percentage female: 60%
- Percentage male: 40%

Fifteen out of the twenty surveyed subjects owned an iOS device, and eighteen subjects had used an iOS device prior to the test. One subject described their experience with iOS devices as trivial, while only one person had never used an iOS device at all.

Fifteen subjects had some kind of kind of audio or music background. These subjects' backgrounds ranged from primary-grade level instrument instruction to university-level music studies. The remaining five subjects had no audio or music background. Subjects' specific responses are shown below in Table 3.1.

Subject	Response
1	Music Technology researcher, violin
2	"I play and write music on several instruments. Additionally I did an audio based undergraduate degree, and am currently undertaking an audio-based postgraduate degree."
3	"Been a professional musician, also an audio professional."
4	Music studies for BA, MSc, and PhD
5	Piano-Grade8, Clarinet-Grade8, Audio Programming for work as researcher, computer music production as a hobby
6	Performance: Guitar-Grade8, Violin-Grade7; Thoery-Grade5; 5/6 Years Orchestral Experience (amateur), 8 years playing in a band; 2 degrees in Music Tech.
7	Piano-Grade7; Violin-Grade8; Music A-Level

8	MusTech Lectures, PhD Audio Interfaces, Piano, violin, bass, guitar (piano = good standard), Composition for Music & Media
9	Violin, viola, piano, singing, sound design for theatre
10	Played flute for 16 years, bass guitar for 7, other instruments too. Music A-Level, music tech degree
11	No experience
12	Musician (Professional singer BA, MA, PhD in Music Technology)
13	"I am a Sonification researcher working on giving real-time feedback to user while making motions. I play guitar and compose ambient/electronic music myself."
14	Vocal student (classical), piano, guitar, flute (choirs, bands, for fun)
15	Violin, piano, recorder, sing in choirs, choral conductor, work in audio and music technology
16	No experience
17	No experience
18	No experience
19	No experience
20	"I have been in an amateur choir. Have a bit of experience watching conductors. Played the violin many years ago."

Table 3.1 Subjects' Musical Backgrounds

To ensure subject confidentiality, all questionnaires have been kept anonymous. Additionally, the subjects' face and voice is not seen or heard in the final edits of the captured video.

3.4 Test Procedure

After giving the test subject the aforementioned hand-out, the tester waited for the subject to signal that they were ready to begin the test. When the subject was ready to begin, the tester began video recording. Once the subject put on the provided headphones, each sample was played back three times by the tester. For the first two playbacks, the subject listened to the sample. On the third playback, the subject enacted a gesture on the provided iPad's screen, which was simultaneously being video recorded by the tester.

After the conclusion of the listening test, video clips were transferred to a computer for editing and analysis. Each test lasted approximately five minutes.

3.5 Technical Details

All of the samples used in the listening test were synthetically generated using Csound's *partikkel* opcode. This was done so that subjects would focus on their gestural response, rather than being potentially distracted through the use of familiar organic sounds.

The four sound files that the subjects heard were generated via a Csound project (.csd file) created by Oeyvind Brandstegg, *partikkel_softsync.csd*, which was included in *The Csound Book DVD* (Ed. Boulanger, 2000). The score of the project was modified to generate audio clips lasting for six seconds each. Each clip was created to highlight a certain parameter of the *partikkel* opcode. These parameters are listed below:

- *igrainrate* – grain rate
- *igrainsize* – grain size
- *igrainFreq* – fundamental frequency of the grain
- *iosc2Dev* – second *partikkel* instance grain rate deviation factor

The .csd file (*partikkel_softsync.csd*) used for the audio generation can be found in Appendix H. The audio clips (*GrFund.wav*, *GrRate.wav*, *GrSize.wav*, and *Osc2Dev.wav*), can also be found in Appendix H.

A copy of the VLC audio/video player was used for sample playback, and was run on a PC running Windows 7. An MOTU UltraLiteMK3 audio interface and a pair of Beyerdynamic DT990 Pro headphones were used for audio monitoring by the test subjects. A Zoom 3HD video recorder was used to capture the gestural responses of the test subjects for later analysis. These video clips were later edited using a copy of Apple's iMovie software on a 2012 MacBook Pro.

3.6 Analysis

The tester organized the gathered data into two spreadsheets, both of which can be found in Appendix G. The first spreadsheet ("Test Subjects") organizes the demographic data of the test subjects, and the second spreadsheet ("Clip Analysis") organizes the information gathered from analyzing the video clips of the test subjects.

The instructions for the test subjects were purposefully vague, so as to see what gestures they would *spontaneously* enact in response to hearing the sound examples. This included the option to use one or both hands. As is shown in Table 3.2 below, the majority of subjects used two hands when gesturing. The author speculates (and discusses later in this section), that this is so subjects could feel that they are not only *generating* the sound, but also *modifying* it.

Clip	1 Hand	2 Hands	No response
GrFund	55%	55%	N/A
GrRate	25%	70%	5%
GrSize	25%	75%	N/A
Osc2Dev	25%	75%	N/A

Table 3.2 Number of Hands Used

Additionally, participants were not told in which orientation the iPad should be positioned. Again, this was to encourage the test subjects to exercise full control as to how they would gesturally respond after hearing the provided sound clips. The majority of subjects kept the iPad in the vertical orientation (the default position of the iPad that was in front of them when they took the test). This is shown below in Table 3.3. However, there were four subjects that asked the tester questions such as “*Can I flip the iPad on the side?*” to which the tester responded, “*You can make the gesture in whatever way you want*”. Those subjects then turned the iPad to the horizontal position.

(Note: One of the subjects failed to give a response during the playing of the “GrRate” clip. This is reflected in any subsequent tables).

Clip	Vertical	Horizontal	No response
GrFund	80%	20%	N/A
GrRate	75%	20%	5%
GrSize	80%	20%	N/A
Osc2Dev	85%	15%	N/A

Table 3.3 Orientation Positions

As can be seen in Table 3.4 below, the gestures that subjects made stayed within the iPad’s multi-touch sensitive area.

Clip	Went past MT sensitive area	Stayed in MT sensitive area	No response
GrFund	25%	75%	N/A
GrRate	0%	95%	5%
GrSize	5%	95%	N/A
Osc2Dev	5%	95%	N/A

Table 3.4 Did gestures stay in multi-touch area?

Figure 3.1 illustrates the area of the iPad’s screen that is sensitive to multi-touch gestures (signified by the area enclosed by the red lines).

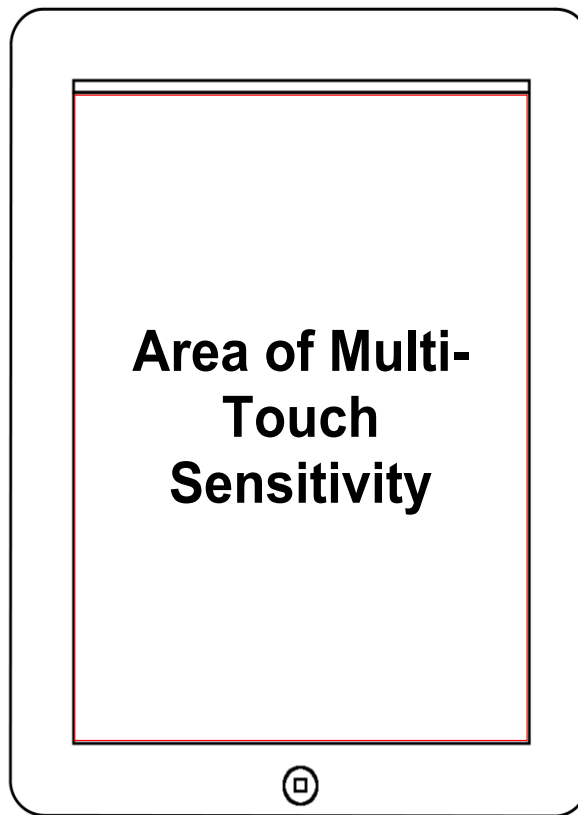


Fig. 3.1
iPad's multi-touch sensitivity area

However, in some cases, subjects enacted gestures that went beyond the multi-touch sensitive area. This is presumed to be because the users may have felt that the iPad's screen was not large enough to accommodate their desired gesture, or, alternatively that they were not aware of the iPad's touch-sensitive boundaries. Examples of this can be seen in the following clips:

- [Subject4 GrFund](#)
- [Subject9 GrFund](#)
- [Subject11 GrFund](#)
- [Subject12 GrFund](#)
- [Subject13 GrFund](#)
- [Subject15 GrFund](#)
- [Subject4 GrSize](#)

3.6.1 Sound Generation and Modification

In many of the subjects' responses, two gestural components were observed: one in which the subject appeared to be simulating the generation of the sound, and one in which they seemed to be modifying the sound. According to Hunt and Kirk (2000), a human operator has to inject energy into an acoustic musical instrument in order for it to operate, and must then continue supplying a certain amount of energy in order to modify the system so that it produces the desired output.

In the case of a violin, the musician injects energy into the system by use of the bow, which generates sound. This energy is modified by "steering" the sound with fingers placed on the neck of the violin. This is shown below in Figure 3.2, sourced from Hunt and Kirk (2000).

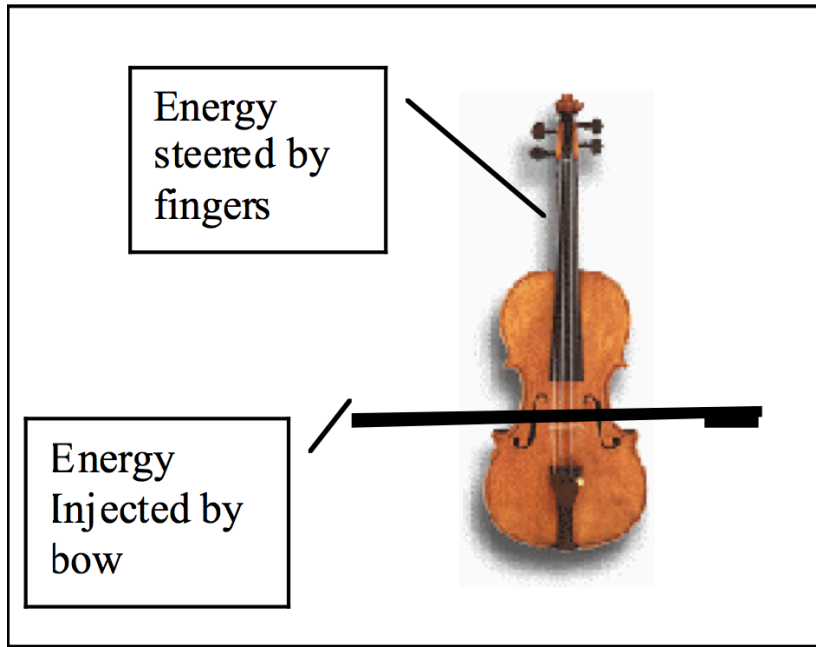


Figure 3.2
Human Energy Input and Control
(Hunt and Kirk, 2000, p.235)

Table 3.5 on the following page illustrates which subjects' gestural responses seemed to indicate an intention of sound generation and/or modification.

Clip	Generation	Modification	No response
GrFund	100%	80%	N/A
GrRate	90%	55%	5%
GrSize	100%	80%	N/A
Osc2Dev	100%	95%	N/A

Table 3.5
Generation and Modification Events

Excluding one subject, who did not give a response to the “GrRate” clip, all subjects responded to the sounds with gestures that seemed to contain an intention of generation, characterized by impulsive, percussive motions. The majority of subjects’ gestures also seemed to contain an intention of sound modification. Smoother, gliding motions across the surface of the iPad’s multi-touch screen generally characterized these interactions. Particularly interesting examples of this can be found in the clips listed below:

- [Subject1 GrSize](#)
- [Subject4 Osc2Dev](#)
- [Subject5 GrRate](#)
- [Subject8 GrRate](#)
- [Subject12 Osc2Dev](#)
- [Subject15 GrFund](#)
- [Subject20 GrSize](#)

3.7 Conclusions

All subjects gesturally responded to sound samples in a manner indicative of sound generation and modification. As such, it is possible that humans, having

developed an innate knowledge of how objects in the world operate on a physical level, are aware that enacting a motion that injects energy into an object will generate a sonic output. Additionally, we seem to be aware that physically adjusting a sounding object will modify the sound it produces.

The tests show that subjects are able to put this knowledge to use on a multi-touch device, such as the Apple iPad. This suggests that a music-based iOS app that makes use of natural mappings between users' inherent knowledge of physical properties and synthesis parameters would be an intuitive tool, making full use of the iPad's multi-touch capabilities.

Chapter 4

Technical Details of User Tests

This chapter gives an overview of the technologies that are part of designing and building the iOS applications used for research into the hypothesis, as well as an overview of the test apps used. It includes descriptions of the technologies involved, and the reasoning behind the choice of certain platforms, programming languages, and synthesis methods. Additionally, specific design and implementation details of the apps are included.

4.1 iOS

iOS was chosen as the development platform because it allowed for the production of apps running on multi-touch capable hardware, specifically the iPad. The Department of Electronics provided access to an iPad, allowing the apps to be extensively tested. Although it is possible to develop on the Android platform, this was deemed impractical due both to the lack of access to hardware for testing, as well the fragmentation of the operating system due to the number of phones on which the platform is available (Velcazo, 2012). Whereas, iOS runs on comparatively few dedicated hardware devices: the iPod Touch, iPhone, iPad, and the iPad Mini (Apple, 2012f).

4.1.1 Multi-touch on iOS

Wessel et al. (2002) list the following features as being essential for a gesture-based musical interface:

- Ability to detect subtle as well as large gestures
- Continuous as well as event-based control
- Low-latency and high bandwidth
- Reliability and portability

A variety of sound processors and instruments have been developed for the iOS platform, some of which have been discussed in Section 2.9. However, only a few of them meet Wessel et al.'s criteria for a gesture-based musical interface, primarily because they rely on skeuomorphic interface elements rather than gestural control.

According to Saffer, there are three stages for an interactive gesture:

- Initiation: how an action begins
- Activation: what happens while an action is occurring
- Updates: what happens when the user has completed an action.

For the user, this is seen as a continuous process. However, the multi-touch system processes each step in turn. This is accomplished through a combination of the multi-touch hardware and the operating system. All multi-touch systems consist of three general components: a sensor, a comparator, and an actuator (Saffer, 2009, p.12). These components are defined below:

Sensor – An electrical or electronic component whose job is to detect changes in the environment.

Comparator – Compares the current state to the previous state or the goal of the system and then makes a judgment. These decisions are then passed to an actuator.

Actuator – Determines the outcome of the comparator's judgments.

In the iOS operating system, the multi-touch screen is the sensor. The screen, which has a resolution of 1024 x 768 pixels (Mark et al., 2011) or a resolution of 2048 x 1536 at 264 pixels per inch if the screen is a Retina display (Apple, 2013c) detects touch events generated by the user, and sends this information to the operating system (the comparator). The operating system then compares the new touch information to previous information, and passes this information to the app the user is interacting with; or the actuator. The app then executes the appropriate actions based on the received touch information.

Figure 4.1 below, sourced from Wilson et al. 2007, illustrates how touch-sensing works on the iPhone/iPad. Once the screen detects a user's touch, the iOS operating system processes this information to determine the coordinates on the screen that the touch was registered.

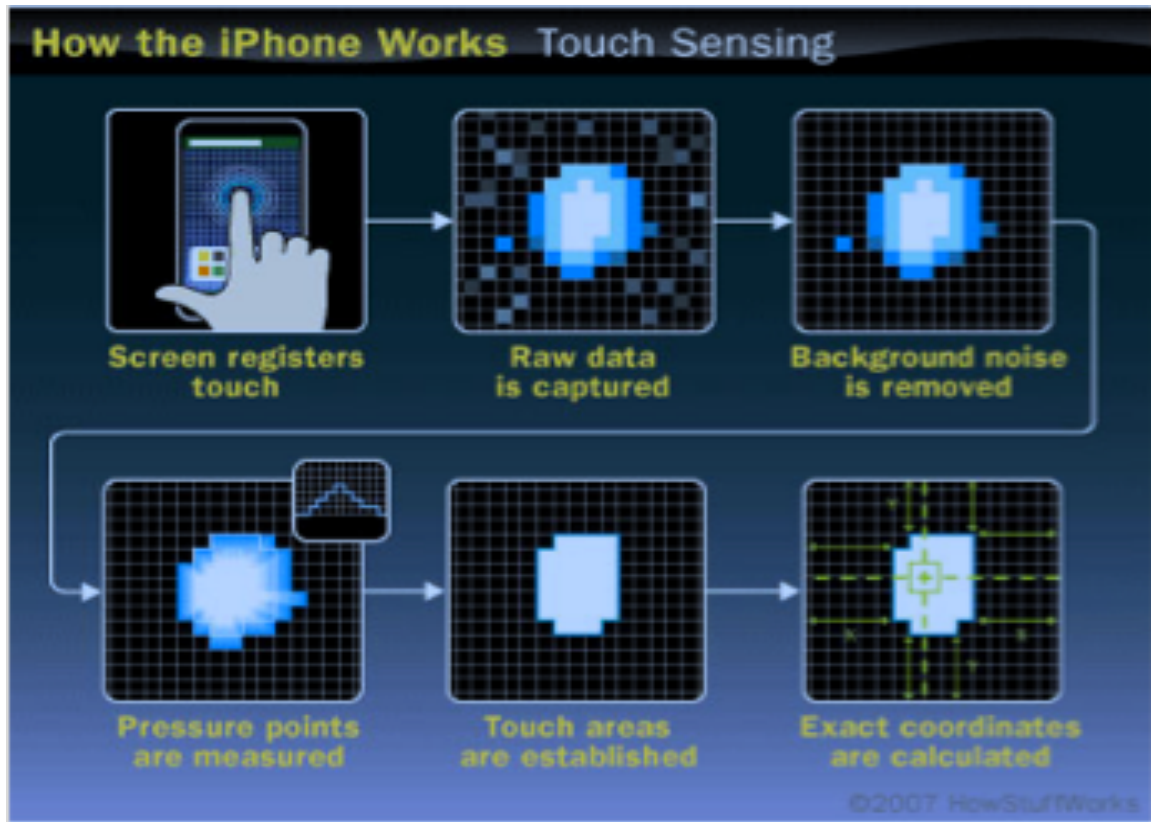


Fig. 4.1
iPhone Touch Detection
(Wilson et al., 2007, online)

Once the touch screen detects a gesture, it needs to be translated to code so that the desired actions of the user can be completed. Saffer (2009, p.133) describes a generic gesture-to-code conversion process as follows:

- Variables to measure: Height, width, depth, speed, duration, and so forth
- Data input: The raw numbers coming in from the sensor readings that populate the variables
- Computation: To figure out the difference between data points
- Patterns: To determine what the sums of the computation mean
- Action: The system action to execute upon finding a pattern.

The gesture recognizers (part of the UIKit Framework) determine the size, shape, and location of areas affected on the screen by the user's touch. This determines what gesture the user performed. Once the gesture type is determined, the corresponding action that the user intended is performed by the app (Wilson et al., 2007). This is done by what Apple refers to as "Gesture Recognizers".

4.1.2 Gesture Recognizers

In its developer documentation, Apple says that gesture recognizers "*...convert low-level event handling code into higher-level actions*" (Apple, 2013b, online). Apple provides several gesture recognizer classes in the *UIKit framework*. Additionally, it is possible for a developer to create their own gesture recognizer designed to accommodate gestures that are not part of Apple's framework of standard gestures.

A gesture is passed through the gesture recognizers inside a series of events (Mark et al., 2011). Each event is triggered when the user makes contact with the multi-touch screen. The operating system detects if a touch has occurred by sensing if a finger has been placed on, dragged across, or lifted from the screen. A tap is recognized if the user touches the screen and then immediately removes their finger from contact with the screen, without moving their finger from the point of initial contact (Mark et al., 2011).

There are two overall types of gestures: discrete and continuous. A discrete gesture occurs once, consisting of one touch event. If a discrete gesture is used, the gesture recognizer will send a single action message to its target (i.e. the code that performs the desired user action). A continuous gesture, however, can consist of multiple touch events that take place over time. If a continuous gesture is used, the

gesture recognizer will send action messages to the target until the multi-touch sequence ends. This process is shown in Figure 4.2 below.

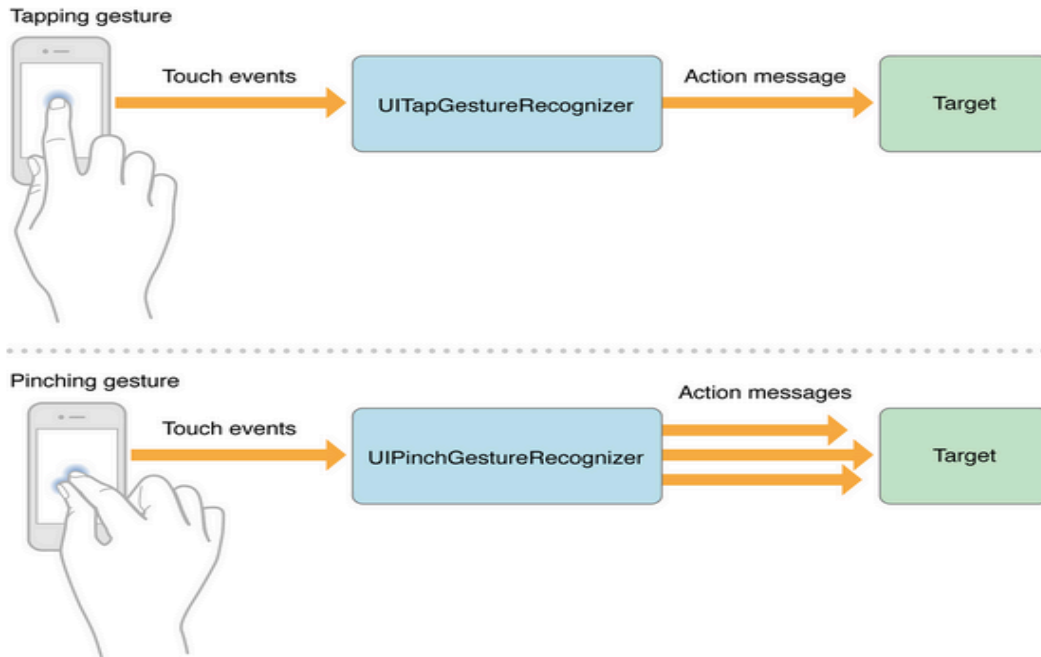


Fig. 4.2
Discrete and Continuous Gestures
(Apple, 2013b, online)

4.1.3 Types of Gestures in iOS

iOS supports a variety of multi-touch gestures, which are listed in Table 4.1 (sourced from Neate, 2012, pp.27-28). The Description column lists how the user's hands perform each gesture, and the Applications column describes how each gesture is traditionally used in a mobile app.



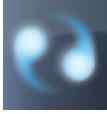


Gesture	Description	Applications
Tap 	A simple short tap on the screen. A program can be implemented to act differently to multiple taps with multiple fingers.	Generally used in the same way a mouse is used, for selecting things, or opening them.
Pinch 	Either pinching so that the fingers go together or spread.	Generally used for zooming in and out.
Rotation 	The movement of the points the fingers are in contact with in a cyclical fashion.	Generally used for rotating elements on the screen, photos, knobs, etc.
Pan 	The movement of 2 or more contact points where they remain approximately the same distance apart.	Normally used for dragging about objects on the screen.
Long Press 	Pressing the screen for longer than a simple tap with one or more fingers.	Generally used to select something on the screen to evaluate its properties, or for deletion.

Table 4.1
Apple iOS Gestures
(Neate, 2012)

Developers are able to use standard gesture recognizer templates in Interface Builder and/or Storyboard files. Additionally, developers may implement gesture recognition programmatically. For more information on iOS gesture recognition, the

reader is encouraged to review Apple’s document “Event Handling Guide for iOS” in the Apple Developer Library (Apple, 2013b). While gesture recognizers are not implemented in this project’s test apps, they are mentioned here to give the reader a broad scope of what is possible in terms of gestural development on iOS devices.

4.1.4 iOS Development in Xcode

As has been established, the main apps for carrying out user tests were developed on Apple’s iOS platform. *Xcode* is Apple’s Integrated Development Environment (IDE) for developing applications for the OSX and iOS platforms (Apple, 2013d). Xcode allows the developer to have seamless integration of code editing, UI design, and testing/debugging inside one window. Xcode’s integrated Interface Builder allows the developer to implement user interfaces that utilize Apple’s built-in interface objects, such as sliders, buttons, and switches. Developers may also implement other open-source or custom user interface objects.

Xcode also allows for easy inclusion of external libraries for implementing extra functionality. One such external library (which the project test apps utilize) is the Mobile Csound Platform, which is discussed in Section 4.2.4.

Simulators for the iPhone and the iPad are included in Xcode, allowing developers the ability to test applications in real-time before deployment onto an actual device. Xcode 4.6.1 was used for project development. A screenshot of the Xcode IDE is shown below in Figure 4.3.

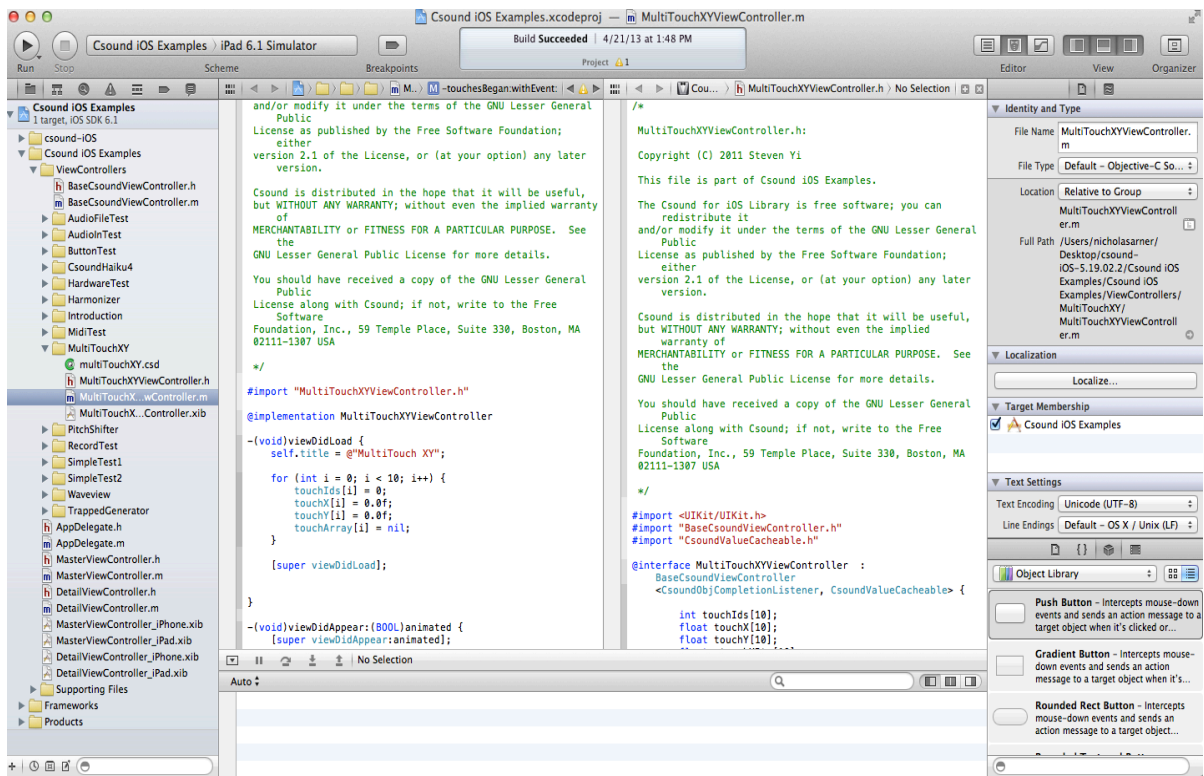


Figure 4.3
Screenshot of Xcode IDE

4.2 Audio Programming for iOS

There are a variety of ways in which developers can implement audio on mobile platforms including iOS. An overview of the available options follows.

4.2.1 Core Audio

Core Audio is a low-level API provided by Apple for implementing digital audio on both the OSX and iOS operating systems (Adamson and Avila, 2012). Given the option of audio development in higher-level synthesis languages (discussed below), it was felt that development in these languages would be better suited for meeting one of the original thesis goals of providing users with the ability to compose high-level musical compositions, and learning these would be a better use of time and resources

than learning Core Audio. This justification carried over to the development of the test apps used for the investigation of the hypothesis.

4.2.2 libpd

Pure Data was also considered for implementing audio synthesis of the app. *libpd* is an API that makes it possible to embed Pure Data into a variety of host platforms, including iOS, Android, and HTML5 (Kirn, 2010). As a result, if a Pure Data patch is created for one device, it can be ported on a host of other devices as well. *libpd* has been used in a variety of commercially successful apps (Kirn, 2013).

4.2.2.1 Pure Data

Pure Data is an open source “...*real time graphical programming environment for audio processing*” (Kriedler, 2009, online). Instead of performing synthesis by writing code in a text-based environment, users connect together visual objects via virtual patch cords. Each object represents a synthesis action or performance, such as an audio input or a simple oscillator. A system of connected objects is known as a *patch*. A screenshot of an example oscillator patch created by the author is shown in Figure 4.4.

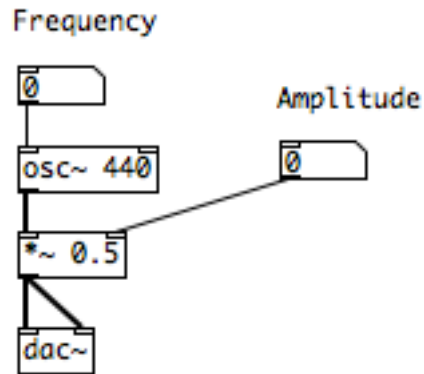


Fig. 4.4
A Pure Data Patch

4.2.3 The Amazing Audio Engine

The Amazing Audio Engine was released in 2013 by developer Michael Tyson (Synthopia.com, 2013). It is built on Core Audio's Remote IO system, and is designed to allow developers to spend more time on specific application development, rather than writing low-level audio code that duplicates previous work.

While it is an attractive option for developing audio on the iOS platform, the Amazing Audio Engine was released too late into project development to spend enough time learning the API. Additionally, the author had already begun research into utilizing the Mobile Csound Platform, which is described in the following section.

4.2.4 Mobile Csound Platform

In 2012, Victor Lazzarini, Steven Yi, and Joseph Timoney announced the *Mobile Csound Platform (MCP)* at the *15th Intl. Conference on Digital Audio Effects* (Lazzarini

et al., 2012a). The MCP allows the developer to develop audio engines in Csound, which can then be deployed on both iOS and Android devices. It is implemented in both Objective-C and Java.

The developers created a new API using Objective-C named “CsoundObj”. The CsoundObj “...controls Csound performance and provides the audio input and output functionality...” (Lazzarini et al., 2012b, p.164). iOS device sensor data is also available to be accessed by Csound using the CsoundObj. In order for iOS to communicate control data and audio signals between Csound, CsoundValueCacheables are added to the CsoundObj. These allow values to be read and written to during each performance cycle. Figure 4.5, taken from *The Mobile Csound Platform* (Lazzarini et al., 2012b, p.164), illustrates this process:

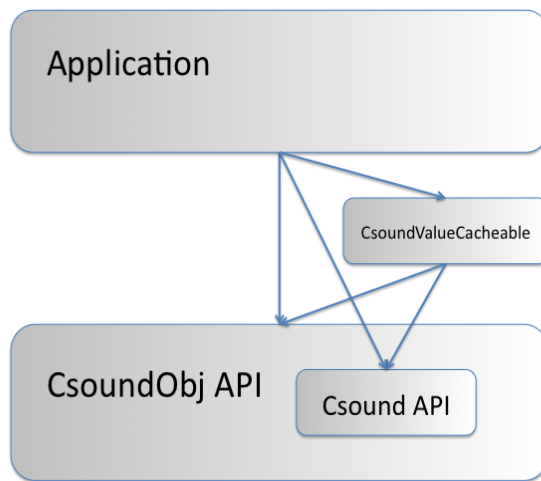


Fig 4.5
Csound for iOS API Relationships
(Lazzarini et al., 2012b, p.164)

More detail on the *Mobile Csound Platform* can be found in “Csound for iOS API-A Beginner’s Guide” (Appendix B), a tutorial on using the API written by the author along with Timothy Neate and Abigail Richardson, fellow colleagues in the University of York Audio Lab. The tutorial was written both for better understanding of the *Mobile Csound Platform* by the authors, as well as to encourage other students and developers in developing iOS applications using Csound as the audio engine.

4.2.4.1 Csound

Csound is a “...*programming language designed and optimized for sound rendering and signal processing*” (Csounds.com, 2012, online). Similar to Pure Data, Csound is freely downloadable software. Composers use Csound to create a wide range of music, including classical, pop, techno, ambient, and experimental music (Csounds.com, 2012).

Csound operates by translating a set of text-based instruments (see Figure 4.6), found in an orchestra file, into a computer data-structure that is machine resident. It then performs these user-defined instruments by interpreting a list of note events and parameter data (known as a score) that the program reads from. The performance can either be played back in real time or written to a disk file (Boulanger, 2000). As Csound has over 1200 different operational codes used for instrument creation, also known as *opcodes*, (Csounds.com, 2012), Csound has the ability to create a wide variety of unique timbres.

```
AudioInput.csd
1 |<CsoundSynthesizer>
2 |<CsOptions>
3 |-o dac --rtmidi=null --rtaudio=null -d --msg_color=0 -M0 -m0 -i adc
4 |</CsOptions>
5 |<CsInstruments>
6 |nchnls=2
7 |Odbfs=1
8 |ksmps=64
9 |sr = 44100
10
11
12 |instr 1
13
14
15 |kgain chnget "gainVal"
16 |asig1,asig2 ins
17 |asig1 = asig1 * kgain
18 |asig2 = asig2 * kgain
19
20 |outs asig1, asig2
21
22 |endin
23
24 |</CsInstruments>
25 |<CsScore>
26 |i      1 0 360000
27
28 |</CsScore>
29 |</CsoundSynthesizer>
```

Figure. 4.6
Screenshot of a block of Csound Code

The author was eager to work with Csound in a way that has not been extensively explored before, as the Csound SDK was released in April 2012 (Kirn, 2012). The Csound for iOS SDK was chosen over *libpd* for this reason. The tutorial in Appendix B is a result of the effort to implement the Csound for iOS SDK.

4.3 History and Overview of Granular Synthesis

One of the ways in which musicians and composers are able to compose with sounds directly, as opposed to notes in traditional notation, is granular synthesis. Instead

of exploration of musical *structures* centred on traditional harmonic and rhythmic theory, timbre is explored instead.

Granular synthesis originated with two publications by Dennis Gabor, “Theory of Communication” and “Acoustic Quanta and the Theory of Hearing” (Gabor, 1944 & 1947). In these papers, Gabor proposes that any sound can be described by a granular, or quantum, representation. Therefore, it would be possible to synthesize both sampled sounds and digital waveforms in terms of granular properties. Each sample is divided up into small ‘grains of sound’ that can be manipulated in real-time. As the threshold of human pitch and amplitude recognition has been estimated to be roughly 50 milliseconds, grain durations are generally between 10-60 milliseconds (Lee, 2000). A pictorial representation of a grain of sound is shown in Figure 4.7.

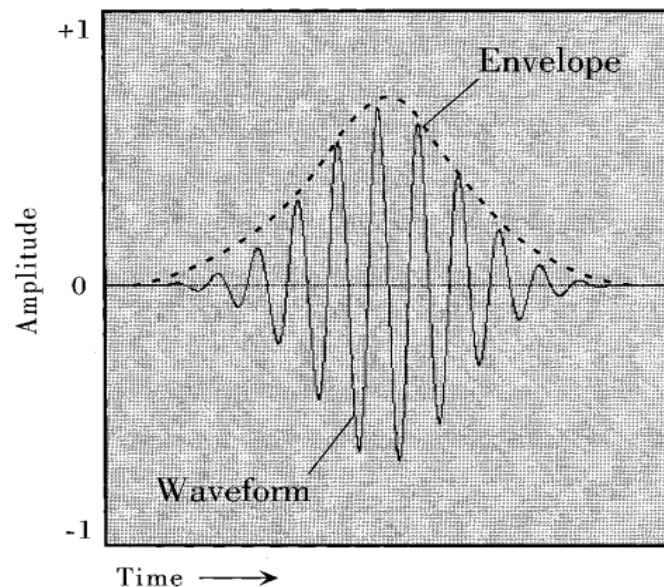


Figure 4.7
View of a grain in the time domain
(Roads, 2001, pg 87)

Amplitude envelopes help form the shape of the grains. Figure 4.8 shows the following commonly used amplitude windowing envelopes, sourced from Roads (2001).

- a) Gaussian
- b) Quasi-Gaussian
- c) Three-stage line segment
- d) Triangular
- e) Sinc function
- f) Expodec
- g) Rexpodec

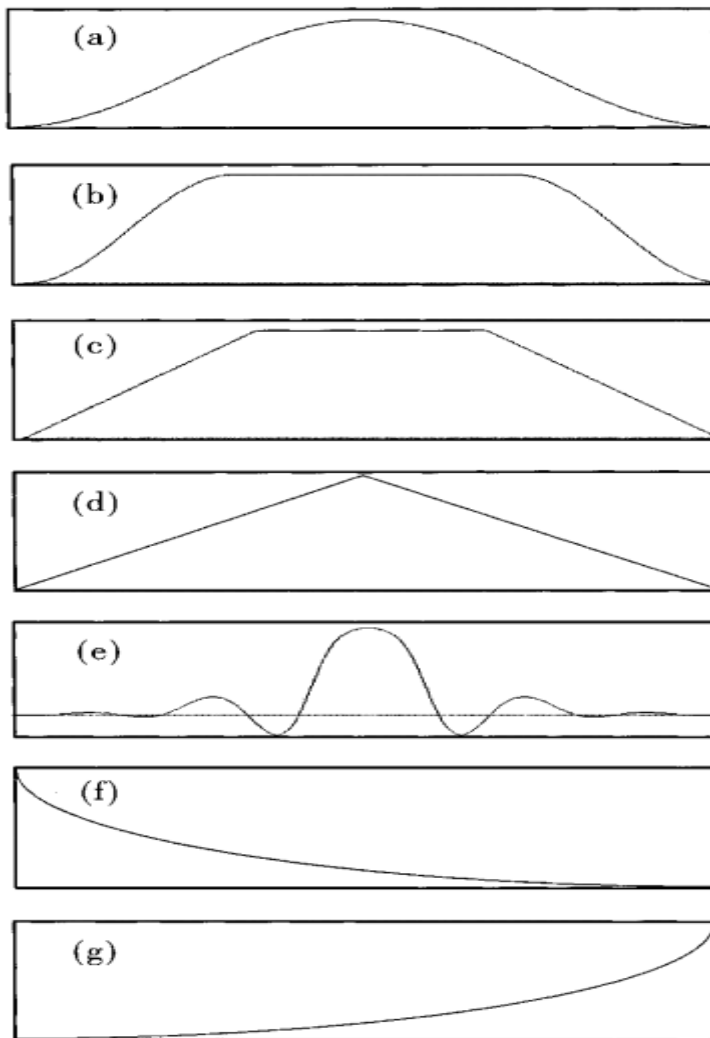


Figure 4.8
Grain Amplitude Envelopes
(Roads, 2001, p.89)

Composer Curtis Roads was the first person to implement granular synthesis in the digital domain. Due to the high number of parameters, Roads “...created an interface that only required the composer to define a beginning set of parameters, after which the program would systematically generate the traits for each individual grain” (Holmes, 2008, p.309). In such a system, the composer determines higher-level compositional characteristics, while the computer system generates sounds according to those defined characteristics.

Roads describes other compositional systems that implement various forms of granular synthesis (Roads, 2001). Additionally, various iOS apps have implemented variations of granular synthesis (see sections 2.9.6, 2.9.9, 2.9.11, 2.9.12, 2.9.13, and 2.9.16).

In “The Computer Music Tutorial”, Roads describes several types of granular synthesis techniques, which are listed below (Roads, 1996).

- ***Fourier/Wavelet Grids***: time domain signal is read in, and frequency versus time content is mapped to a grid. Each grid point is associated with a grain.
- ***Pitch Synchronous***: generation of tones with multiple formant regions in their spectra.
- ***Quasi-synchronous***: multiple streams of grains with a variable delay period between them.
- ***Asynchronous***: grains scattered in regions, called clouds, over a specified duration.
- ***Time Granulation***: envelopes applied to small region of sampled sounds.

Asynchronous granular synthesis was implemented in the test apps using Csound’s *grain* opcode, which is described further in Section 4.4.1.

4.4 Granular Synthesis in Csound

Csound has several granular synthesis opcodes, including *grain*, *granule*, *syncgrain*, *syncloop*, *diskgrain*, *fog*, *fof*, *partikkel*, and *partikkelsync*. (Vercoe et al., n.d.). The *grain* opcode was chosen for use in the test apps due the simplicity of its implementation, while still providing a unique output of granular-based timbres.

4.4.1 The *grain* Opcode

The *grain* opcode randomly reads a portion of the source sound material, which in this case is a square wave. The opcode then outputs a mono audio signal (*al*). The parameters of *grain* are listed in tables 4.2 and 4.3 below, which are modified from an entry in the Canonical Csound Reference Manual, Version 5.13 (Vercoe et al., n.d., p.908).

<i>lgnfn</i>	The <i>f</i> table (a floating point array) number of the grain waveform. This can be just a sine wave or a sampled sound.
<i>iwfn</i>	<i>iwfn</i> - <i>f</i> table number of the amplitude envelope used for the grains
<i>imgdur</i>	Maximum grain duration in seconds.
<i>igrnd</i>	Optional parameter, not used in the project .csd file

Table 4.2
***grain* Initialisation Parameters**

<i>xamp</i>	Amplitude of each grain.
<i>xpitch</i>	Grain pitch.
<i>xdens</i>	Density of grains measured in grains per second. As <i>xdens</i> is controlled through user interface elements and not set to a single value, the synthesizer is asynchronous.
<i>kampoff</i>	Maximum amplitude deviation from <i>xamp</i> . This means that the maximum amplitude a grain can have is <i>xamp</i> + <i>kampoff</i> and the minimum is <i>xamp</i> . If <i>kampoff</i> is set to zero then there is no random amplitude for each grain.
<i>kpitchoff</i>	Maximum pitch deviation from <i>xpitch</i> in Hz. Similar to <i>kampoff</i> .
<i>kgdur</i>	Grain duration in seconds.

Table 4.3
***grain* Performance Parameters**

The Csound .csd file that was used in the test apps for this project was modified from an example project that is included in Chapter 13 of The Csound Book (Lee, 2000). The initialization parameters, as well as control parameters *xamp* and *kampoff*, are coded as constants in the .csd file. The user is able to interact with the remaining control parameters (*xpitch*, *xdens*, *kpitchoff*, and *kgdur*) via rotary knobs, sliders, and multi-touch gestures for each respective test app. These control mechanisms will be discussed further in the Section 4.5.

Two ftables are used to generate and window the square-wave that is the audio source for the *grain* opcode. ftables are arrays of floating point values that are stored in

RAM and are used when Csound is generating sound. They are calculated by Csound GEN routines, a series of ftable generator subroutines (Nelson, 2000). The first ftable uses the *GEN10* subroutine, which generates a table with the size of 16,384 values with one cycle of a square wave. The square wave is comprised of a fundamental and eight harmonics of varying degrees of strength. The second ftable uses the *GEN20* subroutine, which applies a Hanning window with a peak-window value of one in a table with a size of 1,025 values. An illustration of a Hanning window is shown in Figure 4.9 below, which is taken from the Canonical Csound Reference Manual, Version 5.13 (Vercoe et al., n.d.).

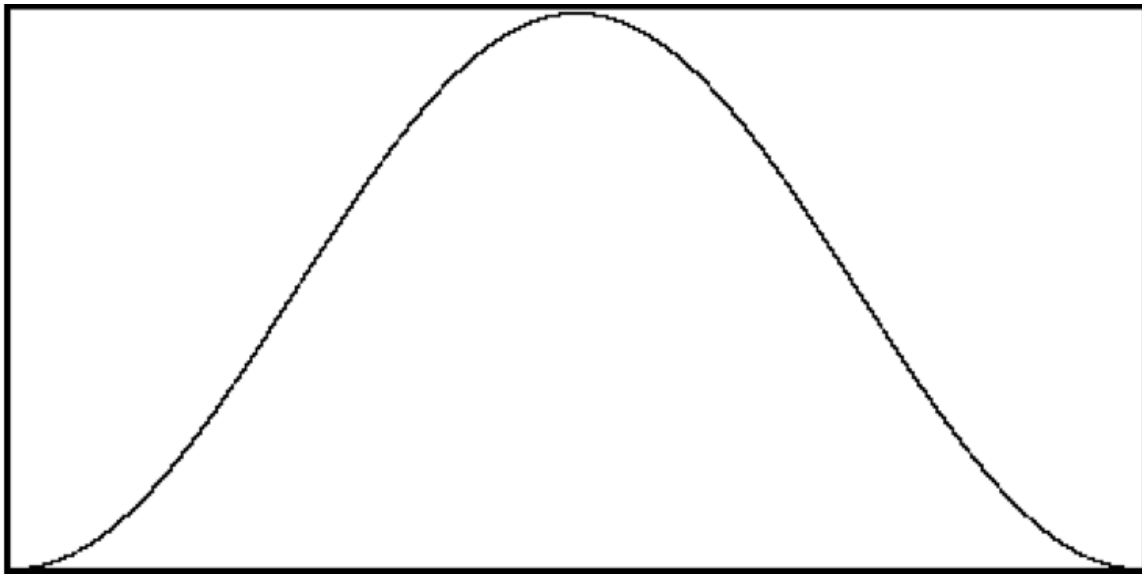


Figure 4.9
A Hanning Window
(Vercoe et al., n.d, p.2617.)

The Csound code for generating the ftables is shown in Figure 4.10 below.

```
; FOR GENERATING 10 SECONDS OF OUTPUT USING THE OPCODE GRAIN
; USING THE GEN10 SUBROUTINE TO GENERATE ONE FULL CYCLE OF
; SQUARE FUNCTION AS SOURCE

f1 0 16384 10 1 0 0.3 0 0.2 0.5 0.14 0 .111

;USING THE GEN20 TO GENERATE A HANNING WINDOW FOR ENVELOPE
f 2 0 1025 20 2 1
```

Figure 4.10
Code for generating a square wave
and applying Hanning window

Figure 4.11 below is a block diagram of the *grain* opcode as implemented in the three test apps. The diagram is adapted from a similar diagram found on page 281 of *The Csound Book* (Lee, 2000). Variables listed as starting with “x” above are able to be implemented as either audio or control rate variables, and are signified accordingly. Any variables set to default values in the .csd file appear with their initialized values.

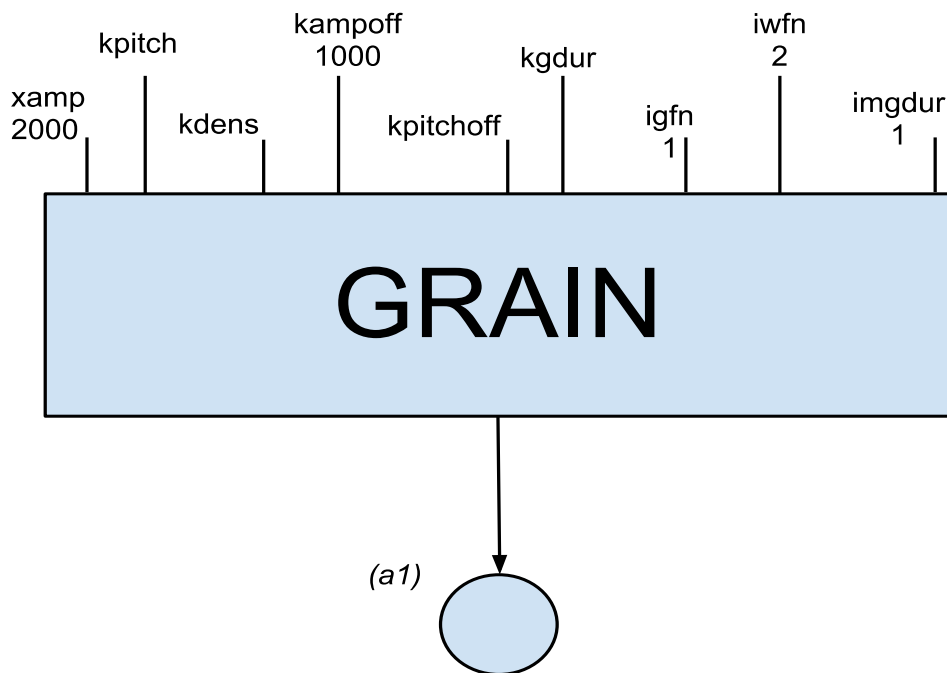


Figure 4.11
Block Diagram of *grain* opcode

In order to make the output of the *grain* synthesizer sound more aesthetically pleasing, an instance of Csound's *reverb* opcode is added. The opcode is simple, requiring input and output audio channels, and the reverb time in seconds (Vercoe et al., n.d.). The reverb time for all test apps is set to 1.5 seconds. An optional parameter, *iskip*, is not implemented in the test examples.

In the .csd files used for the test apps, a global variable, *gal*, is initialized to a value of zero at the start of the file. This is done so that even if the granular synthesizer is not running at the start of the rendering performance, the variable will still exist and have a temporary value until audio is being written to it. At the end of the code for the

reverberation instrument (*instr 1307*), global variable *gal* is reset by assigning it to a value of zero. This is done to prevent the variable from accumulating values from the *grain* synthesizer.

The output of the *grain* synthesizer is fed into the global audio variable *gal*, which is then set to the input of the *reverb* opcode. The audio signal is then applied with reverb, the time of which is set to one and a half seconds. The output is then fed to the iPad speaker. A flowchart of this process is show in Figure 4.12 below.

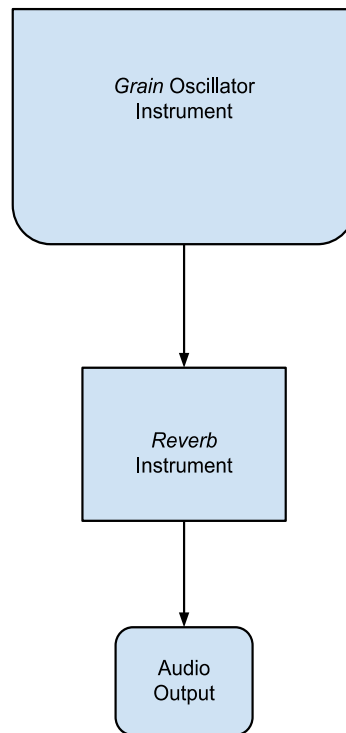


Figure 4.12
.csd Instrument Flowchart

The implementation of this set-up was done with reference to the Csound FLOSS manual chapter on Reverberation (McCurdy, 2010). The following sections detail the test apps that implement user control of the *grain*-based granular synthesizer in three different control settings: rotary knobs, sliders, and multi-touch gestures. The method of sound generation for the Csound synthesizer is exactly the same in all three apps. As such, only the methods of controlling the synthesizer through the application code will be discussed further. Application code for the test projects can be found in Appendix O, and the .csd file used in the projects can be found in Appendix P.

4.5 Design of Test Apps

Three apps were developed to test the hypothesis. Each of the apps allowed users to control the *same four parameters* described in the previous section: grain pitch, grain density, grain pitch offset, and grain duration. Additionally, the first two apps were configured with an On/Off switch to start and stop the rendering of the embedded .csd file. This was to allow test subjects to feel more in control of the app's actions. Without the addition of an On/Off switch, the apps would begin to generate audio output before a subject would have the chance to interact with the user interface elements. This code is shown below in Figure 4.13.

```
//Start rendering if Switch turned on
-(IBAction)toggleOnOff:(UISwitch *)sender
{
    if (sender.on){
        //Locate .csd and assign create a string with its file path
        NSString *tempFile = [[NSBundle mainBundle]
            pathForResource:@"1306-KNOBS" ofType:@"csd"];
    }
}
```

Figure 4.13
Code for rendering control

4.5.1 Rotary Knob Test App

The first app that test subjects interact with is based on parameter control via rotary knobs. Each parameter is assigned to a single knob. The knobs are implemented using tutorial code from developer Tim Bolstad (Bolstad, 2009), as Apple does not provide a default knob class as part of the UIKit. A screenshot of the app is shown below in Figure 4.14.

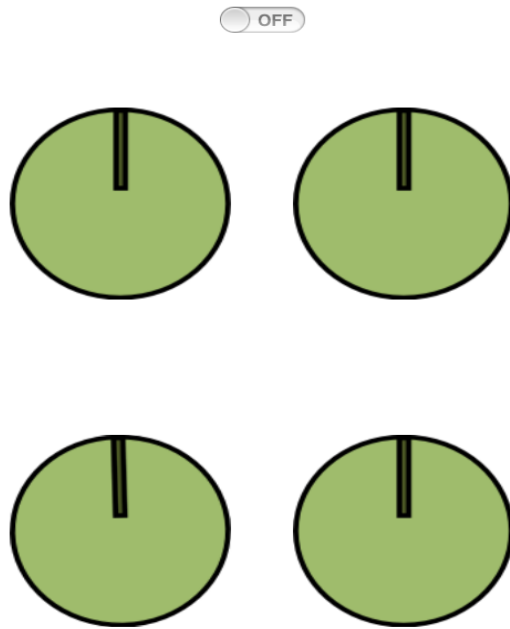


Figure 4.14
Rotary Knob Test App

The starting value and ranges for each rotary knob are set when it is created.

Code for setting the values of Knob 1 as an example is shown in Figure 4.15 below:

```

//Set values of knob1
[knob1 setMinimumValue:100.0];
[knob1 setMaximumValue:1000.0];
[knob1 setValue:550.0 animated:NO]
[knob1 setPrecision:25.0];

```

Figure 4. 15
Setting the value parameters of Knob1

When the user rotates the knob, the value of the knob is sent to variables that Csound will access. The code for this is shown below in Figure 4.16.

```

//Send values of knobs to respective Csound variables
- (void)knobTwist:(TABKnob*)sender
{
    if (sender == knob1)
        grainPitchValue = sender.value;
    else if (sender == knob2)
        pitchOffsetValue = sender.value;
    else if (sender == knob3)
        grainDensityValue = sender.value;
    else if (sender == knob4)
        grainDurationValue = sender.value;
}

```

Figure 4.16
Sending knob values to Csound variables

Csound accesses these variables through strings that are initialized in the application. The strings refer to variables in the application code with value information to control parameters of the Csound audio engine. The code for initializing the strings and connecting them to the necessary variables is shown below in Figure 4.17.

```
kpitch chnget "grainPitchVal"  
kpitchoff chnget "pitchOffsetVal"  
kdens chnget "grainDensityVal"  
kgdur chnget "grainDurationVal"
```

Figure 4.17
Csound code for reading in
variable control information

4.5.2 Faders Test App

The second app that test subjects interact with allows for parameter control via four sliders. Sliders are instances of Apple's *UISlider* class, which allows for "drag and drop" placement. The code for sending slider values to Csound is similar to the code for sending rotary knob values in the previous example. The code for this is shown below in Figure 4.18.


```

//make control variables value of faders
- (IBAction)grainPitchSlider:(id)sender
{
    UISlider *grainPitchSlider = (UISlider *)sender;
    grainPitchValue = grainPitchSlider.value;
}

- (IBAction)pitchOffsetSlider:(id)sender
{
    UISlider *pitchOffsetSlider = (UISlider *)sender;
    pitchOffsetValue = pitchOffsetSlider.value;
}

- (IBAction)grainDensitySlider:(id)sender
{
    UISlider *grainDensitySlider = (UISlider *)sender;
    grainDensityValue = grainDensitySlider.value;
}

- (IBAction)grainDurationSlider:(id)sender
{
    UISlider *grainDurationSlider = (UISlider *)sender;
    grainDurationValue = grainDurationSlider.value;
}

```

Figure 4.18
Code for sending UISlider
values to Csound

The default thumb image for the sliders was replaced by an image included in an example Xcode project from the appcellerator.com forums (Duggal, 2011) in order to make the sliders appear more like analogue faders as found in many pieces of audio equipment, and in computer music apps and applications. Each slider controls a single parameter of the *grain*-based synthesizer. A screenshot of the app is shown below in Figure 4.19.

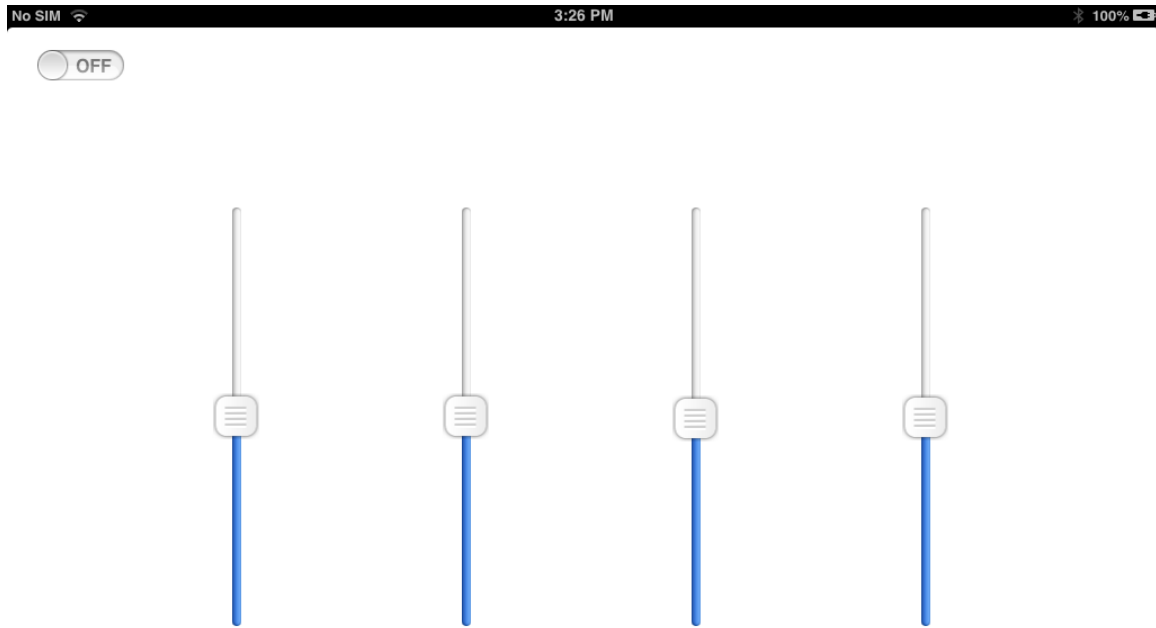


Figure 4.19
Faders Test App

Code for sending control information from iOS to Csound, and the Csound code for retrieving that information, is essentially the same as the previously described Rotary Knob test app.

4.5.3 Multi-Touch Test App

The third and final app that test subjects interact with is based on multi-touch control of the four previously mentioned parameters of the *grain*-based synthesizer. Each synthesis parameter is mapped to a single-finger touch. When the app starts, the output of Csound is muted until the user touches the screen. The user is able to interact

with the app using two fingers. Each finger is able to control a separate mapping zone. These zones are shown below in Figure 4.20.

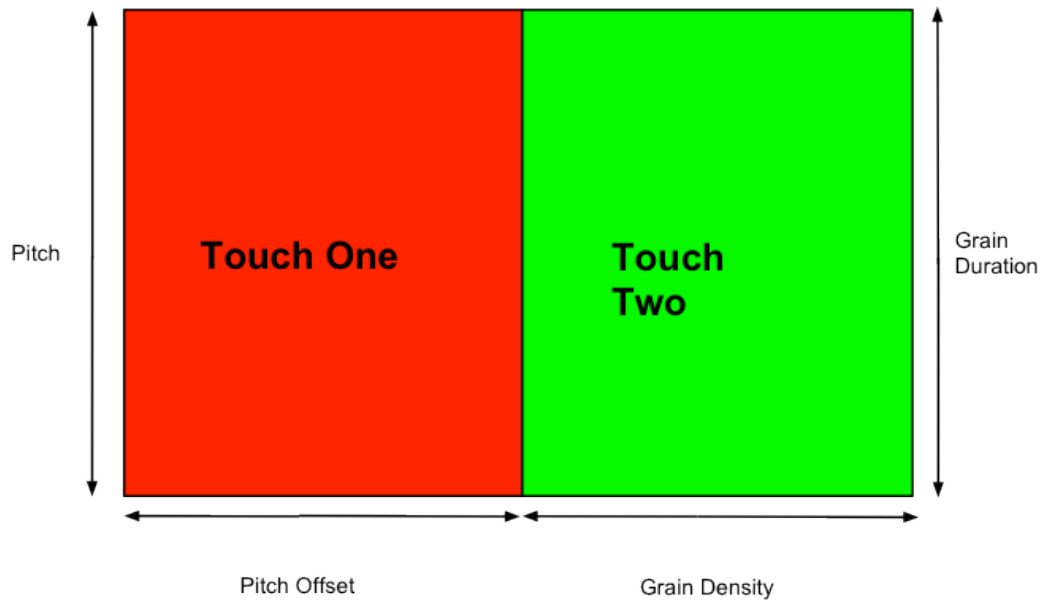


Figure 4.20
Touch Control Zones

As shown above, the red zone on the left allows for control of pitch and pitch-offset. When the user moves a finger along the Y-axis, they are controlling pitch; and when their finger moves along the X-axis, pitch-offset is controlled. When the user adds a second finger in the green zone on the right, two additional variables can be controlled: grain density in the X-axis, and grain duration in the Y-axis. A screenshot of the finished app is shown below in Figure 4.21.

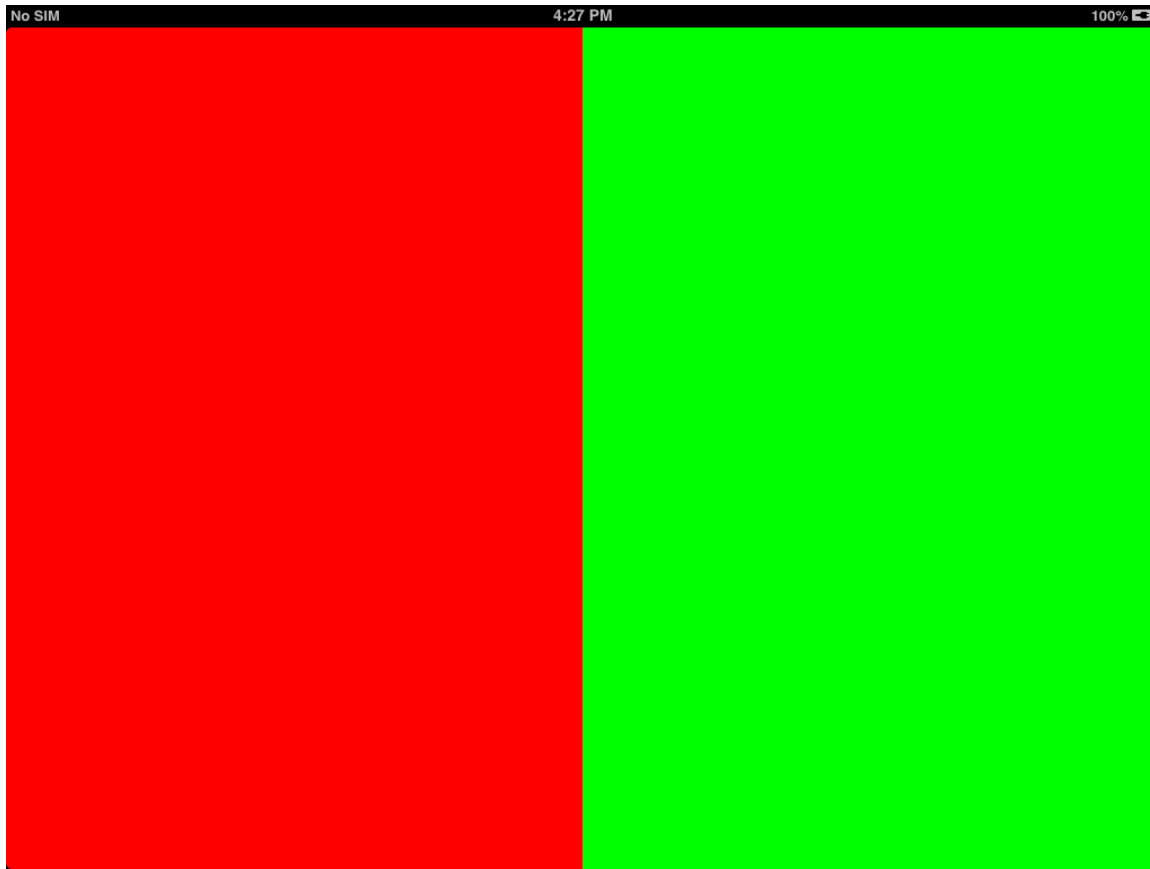


Figure 4.21
Screen shot of Touches App

The red and green frames shown in Figure 4.20 are created using Apple's CGRect function in the ViewController.h file. Each area is assigned a separate touch ID: the red area is assigned to the first touch (UniqueID:1), and the green area is assigned to the second touch (UniqueID:2). The code for this is shown in Figure 4.22 below.

```

//Set size and location of touch areas on screen
CGRect frame1=CGRectMake(0, 0, 512, 768);
CGRect frame2=CGRectMake(512, 0, 512, 768);

//Assign seperate touches to respective screen areas
_firstTouch=[[OneTouch alloc] initWithFrame:frame1
                withColor:[UIColor redColor]
                delegate:self
                uniqueID:1];

_secondTouch=[[OneTouch alloc] initWithFrame:frame2
                withColor:[UIColor greenColor]
                delegate:self
                uniqueID:2];

```

Figure 4.22
Creating Two Touch Areas

The details of the detection and tracking of users' fingers, as well as their mapping to the synthesis parameters are discussed in the next section.

4.5.3.1 Multi-Touch Implementation Details

A class, *OneTouch*, was implemented to track the location of separate touch events and send their coordinate information to the *ViewController*. The *ViewController* then scales the values to appropriate parameter ranges before sending them to Csound for control of each respective synthesis parameter.

The *OneTouch* class makes use of two Apple methods for implementing multi-touch control in iOS, *(void)touchesBegan* and *(void)touchesMoved*. In the *touchesBegan* method, the initial location of each touch is determined. This method is shown below in Figure 4.23.

```

-(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    if([touches count]==1)
    {
        //Tracks initial location of touches
        UITouch *touch=[touches anyObject];
        _initialPoint=[touch locationInView:self];
    }
}

```

Figure 4.23
***touchesBegan* method**

The *touchesMoved* method detects when each touch is moved on the screen, and sends that information to the *ViewController*. Two parameters, valueX and valueY, are defined as floats. This is shown in Figure 4.24 below.

```

-(void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event
{
    Float32 valueY;
    Float32 valueX;
}

```

Figure 4.24
Parameter declaration in
touchesMoved

For each touch that is moved on the screen, the X and Y coordinate points are updated. These points are then scaled to a value between 0 and 1. The code for this process is shown below in Figure 4.25.

```

// For each touch, track point location in the view
UITouch *touch=[touches anyObject];
CGPoint currentPoint=[touch locationInView:self];

// Y value range (scale points to be between 0 and 1)
valueY = 1 - currentPoint.y / _scalingRangeY;
valueY = 1 - abs(currentPoint.y -1)/_scalingRangeY;

// X value range (scale points to be between 0 and 1)
valueX = 1 - currentPoint.x / _scalingRangeX;
valueX = 1 - abs(currentPoint.x -1)/_scalingRangeX;

```

Figure 4.25
X/Y position tracking

For each detected touch (*object: _uniqueID*), a delegate outputs the corresponding X and Y location value that will later be read by the *ViewController*. The delegates are *sendValueYFromOneTouch* and *sendValueXFromOneTouch*. The delegate code is shown in Figure 4.26 below.

```

//Send X and Y values of each TouchID to ViewController
[_delegate sendValueYFromOneTouch:valueY object:_uniqueID];
[_delegate sendValueXFromOneTouch:valueX object:_uniqueID];

```

Figure 4.26
Delegate Code for Sending X/Y Values

In the *ViewController.h* file, the delegate methods of *OneTouch*, *sendValueYFromOneTouch* and *sendValueXFromOneTouch* are called via the code

shown below in Figure 4.27. The View Controller looks for the values sent by the delegates, which is then appropriately modified and sent to Csound in the *.m* file.

```
//Call TouchOne class delegates
-(void)sendValueYFromOneTouch:(Float32)valueY
                                object:(UInt16)objectID;
-(void)sendValueXFromOneTouch:(Float32)valueX
                                object:(UInt16)objectID;
```

Figure 4.27
Calling *TouchOne* Delegates

The code for setting up and communicating with Csound is exactly the same as the previous two test apps; with the exception that Csound is muted when the app first starts. This is done using the Csound for iOS API method “muteCsound”, as shown in Figure 4.28 below.

```
//Mute Csound when app first loads
[self.csound muteCsound];
```

Figure 4.28
***muteCsound* method**

The delegate methods for *OneTouch* are then called. When the first touch (*ObjectID==1*) is detected, Csound is unmuted using the *unmuteCsound* method. For the first touch detected, the *sendValueYFromOneTouch* method sends the value of the Y coordinate to the parameter *pitchValue*. This parameter is scaled to be in a range

between 100 and 1,000 Hz. When the second touch is detected, the value of the Y coordinate for the touch is sent to *densityValue*, which is scaled to be a value between 1 and 100.

Similarly, the delegate method *sendValueXFromOneTouch* sends the X coordinate value of touch 1 to the parameter *offsetValue*, which is scaled to be a value between 0 and 1,000. The X coordinate value of touch 2 is sent to the parameter *durationValue*, which is scaled to be a value between 0.05 and 0.95.

The code for the two delegate methods is shown in Figure 4.29 below.

```

//Csound variables assigned depending on touch location
-(void)sendValueYFromOneTouch:(Float32)valueY
object:(UInt16)objectID
{
    if (objectID == 1)
    {
        //Unmute Csound when user first touches app
        [self.csound unmuteCsound];

        pitchValue = 100 + valueY * 900;
        //printf("pitchValue is %f", pitchValue);
    }
    else if (objectID == 2)
    {
        densityValue = 1 + valueY * 100;
        //printf("densityValue is %f", densityValue);
    }
}

-(void)sendValueXFromOneTouch:(Float32)valueX
object:(UInt16)objectID
{
    if (objectID == 1)
    {
        offsetValue = valueX * 1000;
        //printf("offsetValue is %f", offsetValue);
    }
    else if (objectID == 2)
    {
        durationValue = 0.05 + valueX * 0.95;
        //printf("durationValue is %f", durationValue);
    }
}

```

Figure 4.29
Delegate Method Implementation

Chapter 5

App User Tests

This chapter describes the purpose, design, and implementation of the user tests intended to test the hypothesis. The validity of the hypothesis is explored, as well as subsequent modifications.

5.1 Test Purpose

The apps described in the previous chapter were used in a subject test to help determine the validity of the hypothesis that *users prefer using multi-touch gestures to interact with music as opposed to traditional skeuomorphs*. The three apps allow test subjects to interact with a granular synthesizer implemented in Csound in three different control paradigms: 1) rotary knobs, 2) faders, and 3) multi-touch. The rotary knobs and faders are examples of traditional skeuomorphs commonly employed in many mobile music apps, such as some of those described in Section 2.9. Screenshots of the three test apps are shown Sections 4.5.1-4.5.3.

The prediction is that test subjects will *initially* prefer to interact with the granular synthesizer via faders instead of rotary knobs and multi-touch interaction, however, after repeated interaction with the apps, will grow to prefer the multi-touch option to the faders app. The prediction is also that users will consistently dislike interacting with the rotary knob-based interface. This is due to its inherent physical properties, which are similar to traditional analogue systems.

The first test gathered an overview of how a group of twenty subjects interacted with the test apps, and what their perceptions and opinions of them were. A subset of these subjects (ten of them) later interacted with the same test apps to determine whether their opinions of the apps changed with repeating the test.

5.2 Test Subjects

The user tests were conducted in the University of York, Department of Electronics Audio Lab. Twenty subjects were presented with an iPad, and interacted with the three apps in turn.

Test subjects were recruited from staff and students of the University of York, Department of Electronics' Audio Lab. Additionally, some people from other academic departments on campus were recruited for the test. Eleven of the subjects were from the Department of Electronics; the remaining subjects were from other academic departments at the University of York, including English/Related Literature, Linguistics, Mathematics, Centre for Women's Studies, Computer Science, and Centre for Medieval Studies. Three of the subjects were members of academic staff.

- Average Age: 25.45
- Age Range: 19-41
- Percentage Female: 45%
- Percentage Male: 55%

Thirteen out of the twenty subjects owned an iOS device, and eighteen out of the twenty subjects had used an iOS device prior to the test. Only four subjects had previously used an iOS app to make music. These apps included various synthesizer apps: GarageBand, Propellerhead's Figure App, and MagicPiano. In total, 75% of the

test subjects had some kind of prior musical background. Table 5.1 below shows subjects' responses when asked if they have any sort of prior music or audio background:

Subject	Response
1	MSc/BEng in Music Technology, Previous Performance Experience (Guitar-Grade 8, Violin-Grade 7 (ABRSM))
2	Musicologist, pianist, music teacher, MSc/PhD Music Technology
3	10 years voice, 13 years piano, 1 year guitar, 14 years flute, 1 year University level music study
4	Singer, BA/MA/PhD in Music and Music Technology
5	MusicTech Student, self-taught guitarist
6	No
7	No
8	Piano/clarinet lessons to Grade 8, music technology studies, PhD in Acoustics
9	Played cello in high school
10	Sang in a choir
11	"Have played guitar and used various music generating computer programs."
12	Singing in musicals, Grade 4 piano, Grade 1 trombone
13	No
14	Play guitar/compose electronic music. Used various music production software/hardware. Played with some music apps on iPod.
15	No
16	Professional in both

17	Guitarist and music producer (hobbyist)
18	No
19	Music A-Level, Violin/Piano-Grade 8
20	Musical training in classical/choral singing. Work in audio & music technology.

Table 5.1
Subjects' Music/Audio Backgrounds

Users turned on the sound for each app when they were ready, and were able to interact with the apps for as long as they wished. Before their first test, subjects read a provided hand-out and signed a consent form (see Appendix J). Attached to the consent forms were questionnaires, which can be found in Appendix K. Subjects' hands were video recorded during their interactions with the test apps for potential later use in determining any possible interaction commonalities. All subjects' comments for each app were audio recorded for use in analysing their opinions and perceptions of the test apps. These recordings can be found in Appendix N.

5.3 Test Procedure

Subjects were asked to interact with each of the test apps in turn by adjusting the available audio parameters. Subjects were allowed to interact with each of the test apps for as long as they wished; the longest amount of a time a subject interacted with an app was for approximately 4 minutes and 6 seconds.

Subjects were given no specific instructions on how to interact with the apps, such as what type of sound they should make, or if they should try to reproduce any specific sounds.

The tester was present while the subjects read the provided hand-out. The tester remained present for the duration of the tests, and assisted the subjects in answering any questions they had before, during, and after the tests. Additionally, the tester helped start each app before the users began the test.

At the conclusion of each test, the tester collected the questionnaires for analysis. Additionally, the tester stored all video and audio files from the tests for later analysis.

5.4 Technical Set-Up

The apps used for the subject tests were run on an iPad 2 supplied by the Department of Electronics. Subjects' comments were recorded using an audio recording app on the Tester's iPhone 5. Subjects' interactions with the test apps during both Test 1 and Test 2 were video recorded using a Canon EO5-700D camera. Video footage was then edited using iMovie.

5.5 Test 1 Results

Table 5.2 below shows Test One Subjects' answers when asked which app they prefer, and which app they like the least.

Subject	Most Preferred App	Least Preferred App
1	Touches	Knobs
2	Touches	Faders
3	Faders	Touches
4	Faders/Touches	Knobs
5	Touches	Knobs
6	Faders/Touches	Knobs
7	Faders/Touches	Knobs
8	Touches	Faders
9	Faders	Touches
10	Touches	Knobs
11	Faders	Knobs
12	Faders	Knobs
13	Faders	Touches
14	Touches	Knobs
15	Faders	Touches
16	Touches	Knobs
17	Faders	Touches
18	Faders	Touches
19	Knobs	Touches
20	Touches	Knobs

Table 5.2
Test1 - Subjects' Most and Least Preferred Apps

5.6 Test 2 Results

Ten of the subjects who participated in Test 1 were asked to repeat the test to see if their opinions of the test app would change upon repeated usage. Participants were selected by availability to conduct a repeat test. Subjects were again allowed to play with each app for as long as they wished. Their preferences after completing this repeat test are shown below in Table 5.3.

Subject	Most Preferred App after Test1	Least Preferred App after Test2	Opinion Change Since Test1?
1	Touches	Knobs	No
2	Touches	Sliders	No
3	Knobs	Touches	Yes
6	Sliders	Knobs	No
7	Touches	Knobs	No
8	Touches	Knobs	No
10	Sliders/Touches	Knobs	Yes
13	Sliders	Knobs	Yes
14	Touches	Knobs	No
15	Sliders	Knobs	Yes

Table 5.3
Test2- Subjects' Most and Least Preferred Apps

Four of the subjects in Test 2 did express a change in app preference from one test to the other. These changes are shown in Table 5.4 below.

Subject	Most Preferred-Test 1	Least Preferred-Test 1	Most Preferred-Test 2	Least Preferred-Test 2
3	Faders	Touches	Knobs	Touches
10	Touches	Knobs	Faders/Touches	Knobs
13	Faders	Touches	Faders	Knobs
15	Faders	Touches	Faders	Knobs

Table 5.4
User Change in Preference

5.7 User Comments

After completing the first test, subjects were asked to state any impressions, thoughts and opinions regarding the test apps. Videos of subjects interacting with the apps can be found on the additional YouTube channel at this [link](#).

The Faders app was widely preferred over the Knobs app. This was because users found the Faders app to be much more intuitive and easy to use. Users expected the Knobs app to behave in the exact same manner as a physical rotary-potentiometer knob, i.e., rotating the knob in a circular motion. While such an action could have been implemented in the app, it was decided *not* to do so for one reason: many music-based apps that do implement rotary knobs do *not* behave in such an analogous (rotary) manner. Instead, users are expected to slide their finger up and down in order to move

the knob left and right. In their review of iOS music creation apps, Kell and Wanderley classify knobs that behave in this manner as actually being faders (Kell and Wanderley, 2013). It is unclear why many developers implement rotary knobs in such a counterintuitive manner, apart from it being slightly easier to code. An example of a user struggling with this configuration can be found [here](#).

Faders were deemed to be easier and more intuitive to use because they provided a sense of linear visual feedback, were easy to interact with, and seemed to be conducive to producing a desired sound. Users seemed to appreciate that when they moved a faders up and down, the corresponding parameter was either increased or decreased (for example increasing or decreasing the pitch of the grains). Users described the Faders app as being easy to control with their fingers, and not as cumbersome as the Knobs app.

Although the granular synthesis engine implemented in the apps was generally described as being difficult to control, users generally felt that the Faders app allowed for the best implementation if they wanted to reproduce a sound that they had heard earlier while interacting with the app. As subjects were not given instructions to try to reproduce sounds during the tests, it is possible that they were doing so in order to better understand what effects their interactions with the user interface had on the musical output.

Such reproducibility would be necessary if the user wanted to have fine-grained control over the music creation process. Subjects also described the faders as being more responsive to their touches and easier to control than the rotary knobs. This may be related to the fact that the knobs and faders provide the user with what Norman refers to

as *affordances*, fundamental properties that determine how an object should be used (Norman, 2001). In other words, a knob *affords* the action of turning; a fader *affords* the action of a sliding motion.

Subjects who preferred the Touches app stated that the reasons for their choice were that the app felt “*intuitive, interactive, creative, and playful*”. One subject even stated that the app “*encouraged creativity*”, and that it “*Makes for a more user-friendly and interesting performance experience*”. Another subject stated that they preferred the Touches app because it felt more *tactile*.

While interacting with the app, some subjects seemed to be exploring the timbre space available via the granular synthesizer. While subjects could achieve some sonic reproducibility via repeated movements over appropriate areas of the X/Y axis, those who expressed a preference for the app seemed content to explore the timbral possibilities of the app via moving their finger across the screen, alternatively in fast, [quick swirling](#) and [tapping gestures](#), and more [slow and nuanced](#) gestures.

5.8 Hypothesis Discussion

The original hypothesis is that users prefer multi-touch gestures for interacting with music as opposed to traditional skeuomorphs. In this hypothesis framework, users would consistently prefer to interact with music using multi-touch gestures, regardless of the paradigm of the application or the user’s intent and purpose for interaction.

Based on the results of the tests, users do not show a preference for interacting with music with only multi-touch gestures to validate the initial hypothesis. As such, the hypothesis has been modified.

The modified hypothesis is that if users want an app that will allow them to explore music in a manner they would describe as “*intuitive, interactive, creative, and playful*”, then a multi-touch gestural app is the preferred option. These users desire real-time interaction with music: they want to have an immersive, “flow-like” creative experience when interacting with the app. Such an experience would be similar to Csikszentmihalyi’s concept of “flow”, as discussed in Section 2.6 (Leman, 2010).

However, if users want to be able to intricately modify and edit music or audio, then an app implementing a skeuomorphic design paradigm is the preferred option. Such users want to understand what the individual parameters are and adjust them to reach a certain goal. If the app does use skeuomorphs, then it needs to be intuitive, easy to use, and provide appropriate visual feedback.

The modified hypothesis is therefore: users who want an intuitive and exploratory experience with music prefer an app with multi-touch gestures, but those who want to be able to modify and edit music prefer apps with skeuomorphic UI elements.

The user tests conducted as described in this chapter provide initial evidence for this modified hypothesis. The tests were intended to explore a principle rather than to provide statistical validity.

Given that some users prefer the complex multi-touch interface and some prefer the simpler skeuomorphic interfaces, these preliminary results may help support Hunt and Wanderley’s findings (2002) that users prefer complex interfaces for complex, real-time tasks, and simple interfaces for simple, non-real-time tasks. Simpler interfaces and mappings may be more suited for non-real-time/goal-focused applications.

Additionally, it is observed that multi-touch gestural control is not initially well received by goal-focused users. Given the inherent multi-parametric complexity of multi-touch interfaces, many users may be initially intimidated by a rich multi-touch gestural app, and will avoid its use, even if it meets the needs of their tasks. However, given a long enough period of testing, subjects will begin to prefer the more complex multi-touch gestural app.

The author suspects that if each subject had more opportunities to interact with the apps over a longer time span (i.e., more than the longest time a user spent on a test, which was approximately 4 minutes), those who did not initially enjoy interacting with the Touches app would eventually come to at least appreciate what it is capable of, even if they did not completely prefer or enjoy it.

5.9 Potential Test Improvements

After the tests, five users stated that all three test apps would have been much more enjoyable to use if the synthesized sound output was different. Of the twenty subjects who participated in the tests, only five had any comment on the sound generated by the app, four of these subjects giving a negative opinion of the sound. These subjects stated that the sound was too chaotic, glitchy, and hard to control. One subject stated that due to the nature of the sound, it was hard to tell exactly what they were trying to control. Only one subject (Subject 8) out of the twenty specifically stated that they enjoyed the sound; a video of them interacting with the Touches app during the first test can be found [here](#).

It is possible that if the sound had a more pleasing aesthetic quality, the results of the test would be different. Users who expressed frustration at using the Touches app might have been willing to spend a longer time interacting with it had the sound generated by the apps been more pleasing to their musical tastes, and in doing so may have come to prefer the Touches app over the Knobs and Faders apps.

Additionally, some users stated that they would have enjoyed using the touches app if it implemented visual feedback. As users are able to see the skeuomorphic interface elements (knobs and faders in the app examples) change their positions when they interact with them, users expected the same sort of visual feedback to occur when interacting with the Touches app. The study of visual feedback in multi-touch gestural music apps is a potential further area of research. Interestingly, one user did say that the lack of visual feedback made the Touches app “*more fun somehow*” (Subject 8).

This is consistent with Hunt’s notion of an “*explorative operation*” (Hunt, 2000). Hunt defines an “*explorative operation*” mode of interaction as when “...*the user discovers how to control a device by exploring different input control positions and combinations, thus gaining an immediate response from the system*” (Hunt, 2000, p.102). The user who stated that they preferred a lack of visual feedback on the Touches app may have done so because they were able to enter into an explorative mode of operational interaction. They may have been content to truly explore the musical possibilities of the app, as opposed to relying on visual feedback to reproduce certain specific sounds while in an analytical mode of operation. This analytical mode would be characterized by viewing each individual musical parameters as separate, rather than listening to the combined overall musical result of the individual parameters.

If any future versions of the test described in this chapter are to be carried out, the addition of visual feedback by generating animation or other visual content with each touch movement should be considered.

Chapter 6

Conclusions and Further Work

This chapter states the conclusions drawn to support the modified hypothesis, stated in Section 5.8. Possible future work in the area is also proposed. The chapter concludes with a statement on the significance of the conducted research.

6.1 Conclusions

Before implementing an interactive music app, designers need to make sure that the purpose of the app is clearly defined: for example, is the app supposed to allow for intricate audio editing, or will it allow for interactive, real-time composing and/or performing? Obviously there are many different types of music apps (some of which have been described in Section 2.9); the ‘audio editing’ and ‘performing’ are mentioned merely as differing examples. If the user requires fine-grained, intricate control of individual parameters, then an app implemented with traditional skeuomorphs is more appropriate. However, if the user wishes to explore a wide musical design space, then multi-touch gestures are much more appropriate for encouraging the user to intimately explore and play with the potential musical material.

6.2 Further Work

In addition to the inclusion of visual feedback as discussed in Section 5.8, the author proposes the following projects as potential continuations of the research conducted as part of this thesis.

Haptic Feedback in Multi-touch Interactions

Apple has recently applied for a patent for a “Method and apparatus for localization of haptic feedback” (Campbell, 2013). Depending on whether this is implemented on iOS devices, and if developers are given sufficient ability to write code for this feature, another project could be to determine whether haptic feedback aids multi-touch based musical interaction. Such a system could have particular benefits for visually-impaired individuals. Additionally, a combination of haptic and visual feedback in aiding multi-touch interactions could be investigated.

Investigation of Custom Gestures for Musical Interaction

As discussed in Section 4.1.3, Apple allows developers to create custom gestural interactions. To the best of the author’s knowledge, there are currently no iOS apps that allow *users* to determine which gesture they wish to use to control specific musical parameters. A potential app for investigating user gestural preferences would allow the users to *themselves* set up mappings between specific musical parameters and specific multi-touch gestures.

6.3 Significance of Research

Music interaction and digital musical instrument research bears a certain resemblance to “mainstream” Human-Computer Interaction research. Both are tasked with similar aims: helping users accomplish certain goals that are assisted through the aid of computer technology. Wanderley and Depalle (2004, p.632) describe this by stating that the study of gestural interaction with musical parameters “...can be seen as a highly specialized branch of HCI”. Additionally, music

interaction has offered several contributions to HCI research, some of which are listed in Holland et al., 2013. Wanderely and Orio (2002, p. 74) further support this by stating that “...a bidirectional flow of knowledge between classical HCI research on input devices [...] and the design of new digital musical instruments can lead to substantial improvement in both fields”.

However, music interaction research is a narrower field where the goal is to develop methods and implementations that allow the end user, in this case someone interested in engaging with a musical system, to have the best possible experience of interacting with that system. Holland et al. (2013, p.3) state that “*Music Interaction borrows countless elements from HCI, and in general is held to the same standard as HCI research. But at the same time, the practice of Music Interaction is bound up with the practices of the music community*”. In mainstream HCI research, the scope is much broader; typically the research goal is to develop ways of easily manipulating data/accomplishing some sort of end goal as quickly and efficiently as possible.

In such scenarios, interface designs consisting of WIMP (*Windows Icons, Menus, Pointers*), and skeuomorphic elements such as knobs and sliders are appropriate, as they allow users to quickly and efficiently accomplish a task. This is in a contrast to designing interfaces for musical interaction, in which “...*the design of a new input device for musical performance is generally directed toward the fulfilment of specific and sometimes idiosyncratic musical goals*” (Wanderley and Depalle, 2004, p.637). Complex systems, such as music, often demand complex

interfaces. Such interfaces are often not learnt quickly, and many times users will grow frustrated with them before achieving any satisfactory results. Complex interfaces may at times be appropriate for helping users meaningfully engage with musical material, as shown in Hunt (2000), and McDermott et al. (2013).

Professional musicians spend a lifetime practicing their craft. As long as they have a musical goal to reach, they will keep doing so. Musicians always find a way to keep advancing in their instrumental and performance skills, or ways to keep growing as a composer. In both cases, they are interacting with a system (a physical instrument to produce musical output, or creating musical structures in a novel and creative manner). In both cases, the system they are interacting with is a complex one.

The area in which the field of Music Interaction research can help benefit mainstream HCI research is in the interaction of users with complex systems. Mainstream HCI research, with its focus on speed and efficiency, may at times not present appropriate solutions for the target areas it hopes to solve. Music interaction, with its specific focus of helping people interact with complex systems in novel ways, is in a position to help mainstream HCI researchers in creating immersive, flow-like interactions for users engaging in high-dimensional systems.

This research has shown that when users wish to creatively engage with music on a particular device, they prefer to do so with multi-touch gestures where possible. If they wish to have intricate control over specific musical parameters, however, then skeuomorphic user interface elements are preferred, even where

multi-touch gestures are available. Designers of music-centric apps, or of any app that involves a system as complex as music, should consider implementing multi-touch gestures, as - if wisely implemented - they allow the user to achieve a state of flow and creative exploration.

Appendix A - Composition App UI Sketches

Appendix A consists of sketches created to design the interface and interaction methods of an interactive-composition iPad app, and drawn at the time when this was the intended development goal as part of the MSc.

Sketch 1.A

As the user vocalizes into the microphone, the process of granularization is illustrated through the use of a waveform display / graphics to show the granularization occurring.

Sketch1.B

As further vocalizations are added, they are continually displayed along with those that occurred previously. The waveforms may be pushed/pulled to control the parameters of the granular synthesizer.

Sketch 1.C

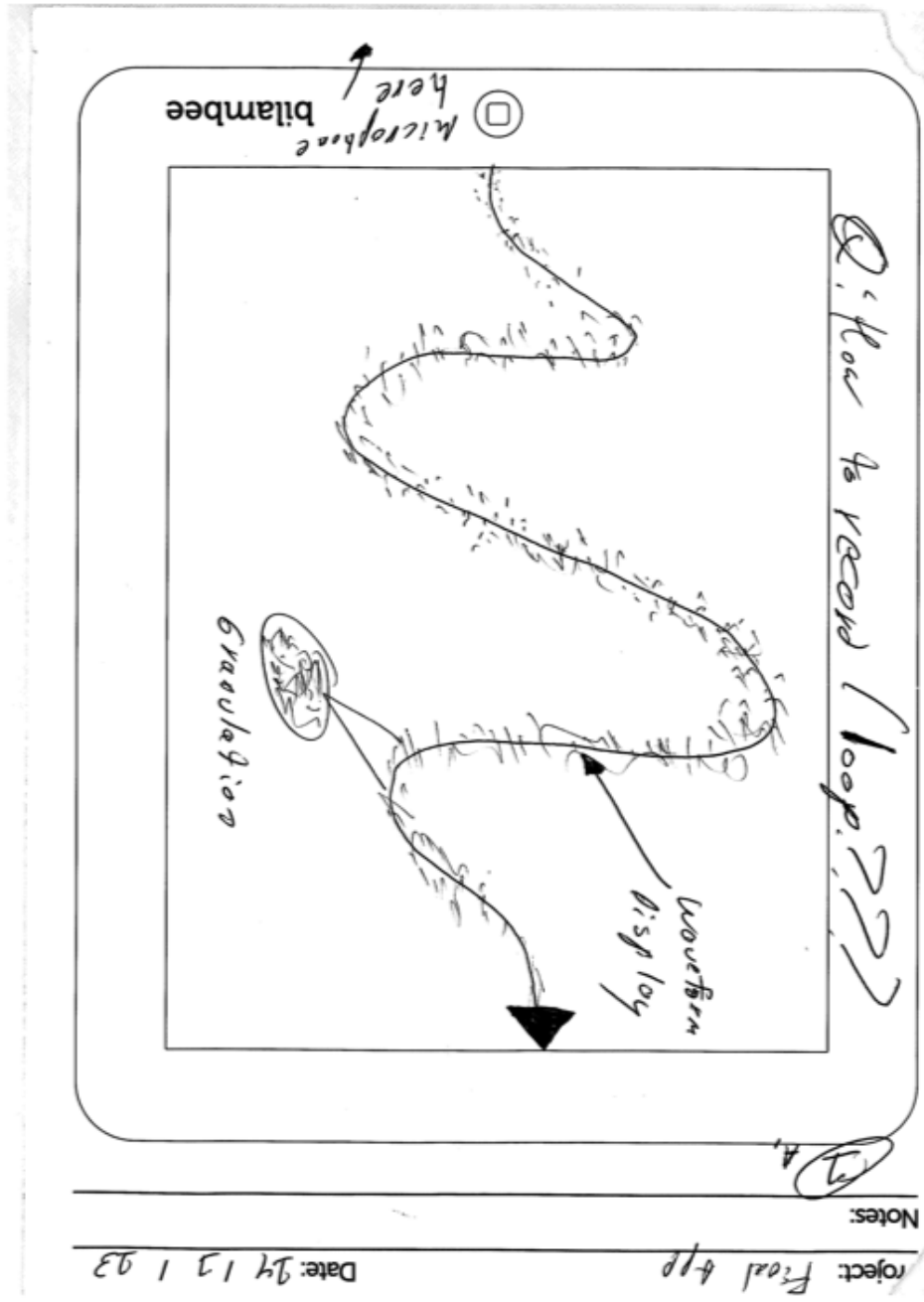
A cellular automata algorithm that affects the granularized visualizations would direct “Ball objects” shown in the sketch. It was not decided what parameters they would be mapped to.

Multi-Touch Mapping Sketch1

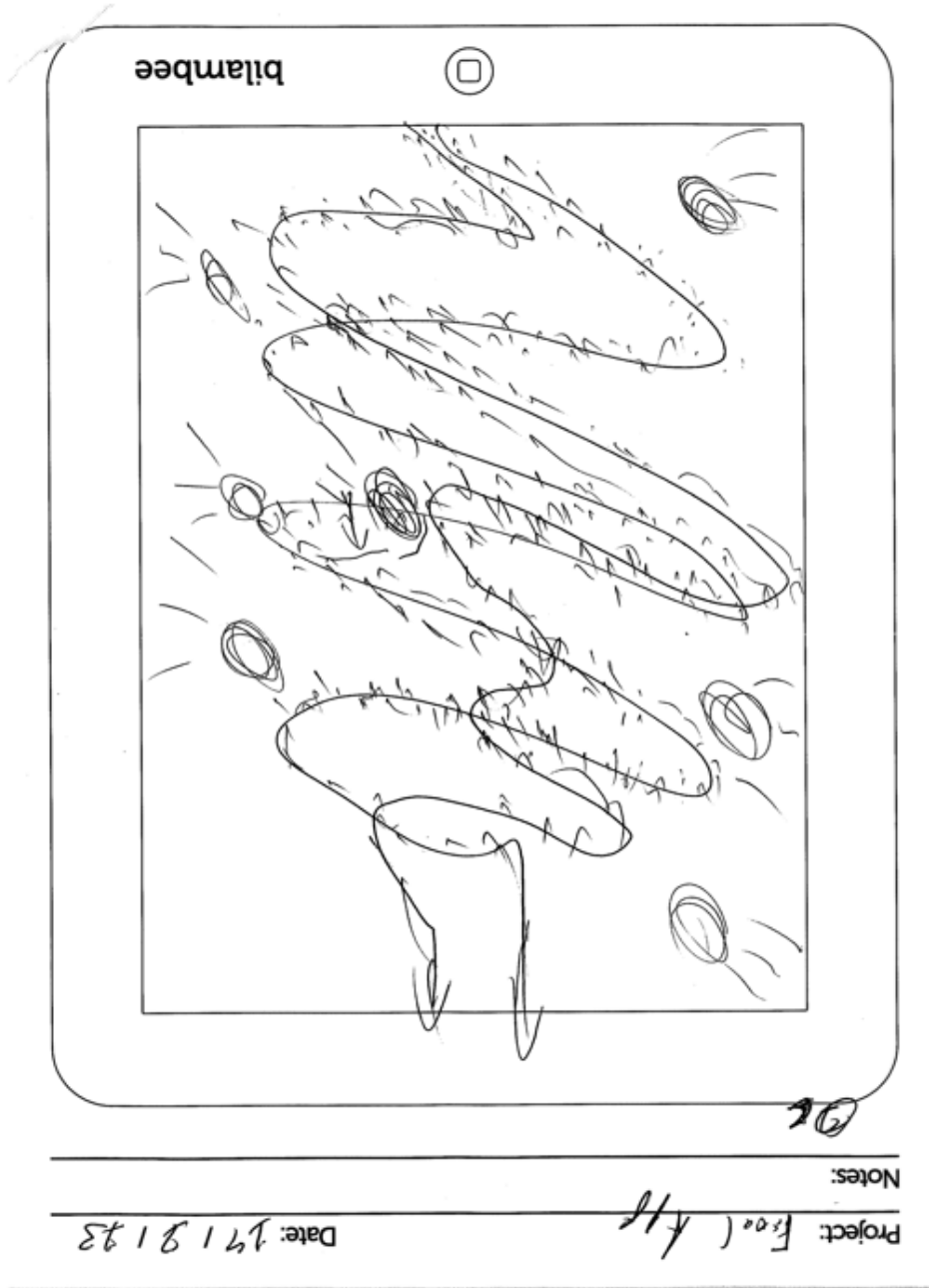
This details an original mapping sketch for the multi-touch test app. Synthesis parameters are controlled by a variety of tapping, pinching, and swiping gestures.

Multi-Touch Mapping Sketch2

This sketch was a second idea for the mapping of the multi-touch test app. Synthesis parameters are controlled by pinching and swiping gestures. Grain pitch is an X/Y control space in the left corner, and pitch offset is an X/Y control space in the right corner.



Sketch 1.A

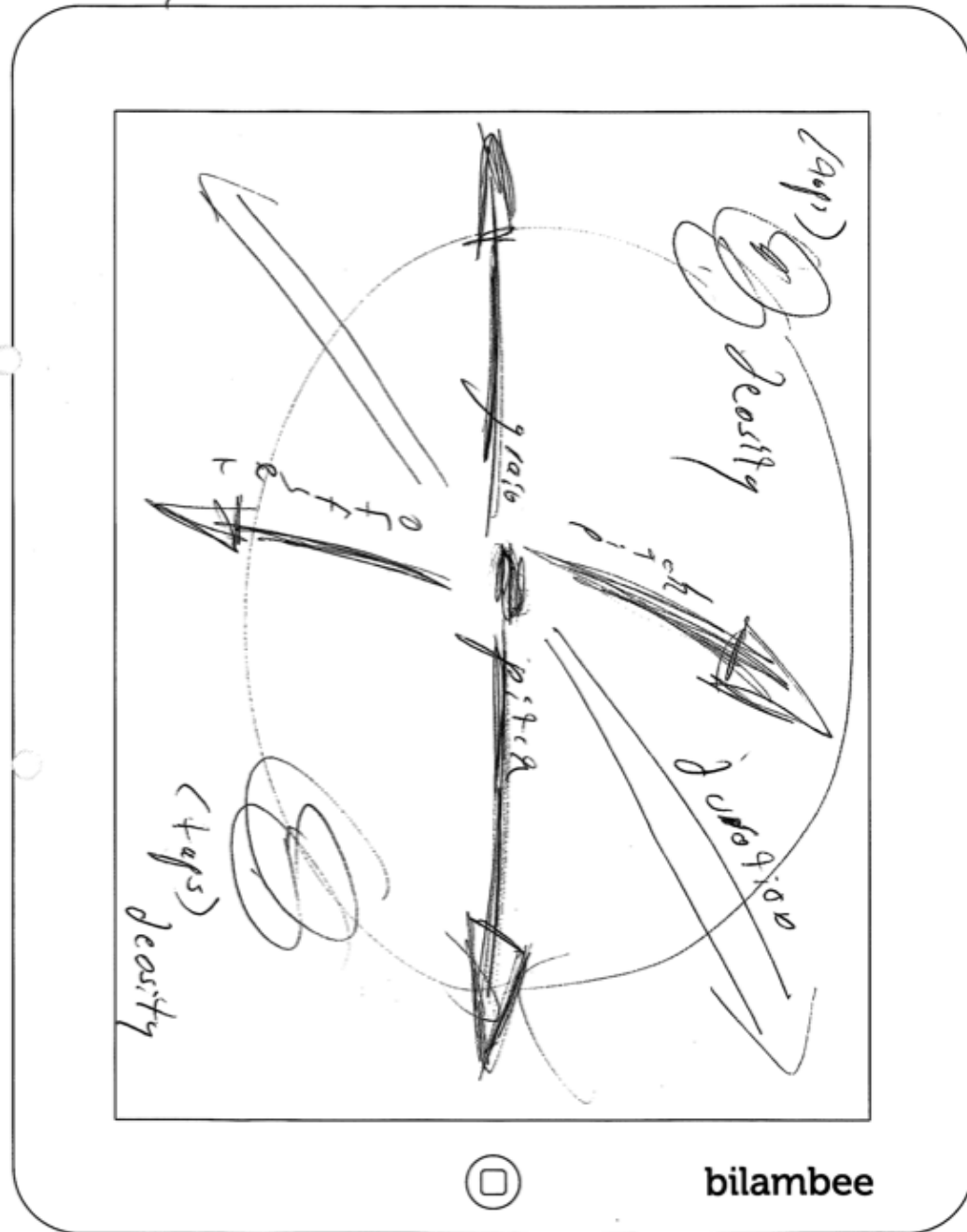


Sketch 1.C

Project: Post App 3 - Multi Touch

Date: / /

Notes: Mapping Sketches

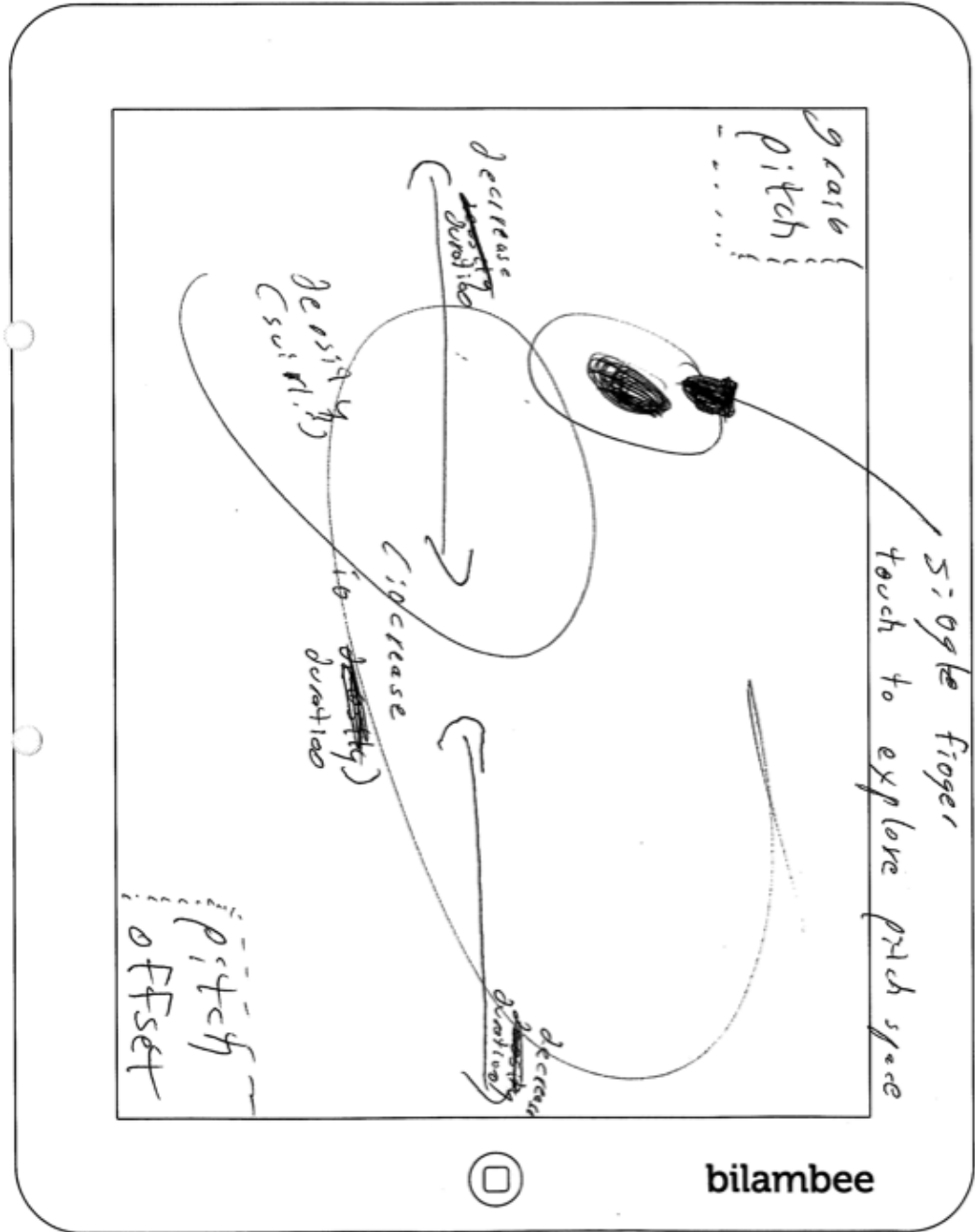


Multi-Touch Mapping Sketch1

Project:

Date: / /

Notes:



Multi-Touch Mapping Sketch2

Appendix B - Csound for iOS Tutorial

In an effort to better understand the Csound for iOS API, the author, along with fellow colleagues Timothy Neate and Abigail Richardson, wrote a tutorial for the API. The tutorial is aimed at those who have some iOS development experience, and are wanting to quickly develop audio for use in their apps.

THE UNIVERSITY *of York*

Department of Electronics

Audio Lab

Csound for iOS API

A Beginner's Guide 1.1

17/2/2013

Timothy Neate, Nicholas Arner & Abigail
Richardson

Abstract

This tutorial aims to help iOS developers with the implementation of the Mobile Csound Platform for iOS. Developers who are looking to incorporate audio into their apps, but do not want to deal with the complexities of Core Audio, will find this particularly useful.

It provides some background information on the API and outlines how to integrate Csound and iOS, and allow them to communicate. The provided example project is then described - outlining the key features of the API. Some common problems that users are likely to encounter are then discussed to troubleshoot potential issues

Acknowledgements

We would like to thank our supervisor, Dr. Andy Hunt, for his support and guidance while working through the Csound for iOS API, as well as working on this tutorial. We would also like to thank Dr. Victor Lazzarini and Steven Yi, the authors of the Csound for iOS API. We would especially like to thank Steven for his extremely helpful responses to our questions regarding the API.

1. Introduction

The traditional way of working with audio on both Apple computers and mobile devices is through the use of Core Audio. Core Audio is a low-level API which Apple provides to developers for writing applications utilizing digital audio. The downside of Core Audio being low-level is that it is often considered to be rather cryptic and difficult to implement, making audio one of the more difficult aspects of writing an iOS app.

In an apparent response to the difficulties of implementing Core Audio, there have been a number of tools released to make audio development on the iOS platform easier to work with. One of these is *libpd*, an open-source library released in 2010. *libpd* allows developers to run Pure Data on both iOS and Android mobile devices. Pure Data is a visual programming language whose primary application is sound processing.

The recent release of the Mobile Csound Platform provides an alternative to the use of PD for mobile audio applications. Csound is a synthesis program which utilizes a toolkit of over 1200 signal processing modules, called opcodes. The release of the Mobile Csound Platform now allows Csound to run on mobile devices, providing new opportunities in audio programming for developers. Developers unfamiliar with Pure Data's visual language paradigm may be more comfortable with Csound's 'C'-programming based environment.

For those who are unfamiliar with Csound, or want to learn more, the FLOSS manuals are an excellent resource, and can be found here:

<http://flossmanuals.net/csound/>

For more advanced topics in Csound programming, the Csound Book (Boulanger ed., 2000) will provide an in-depth coverage.

In order to make use of the material in this tutorial, the reader is assumed to have basic knowledge of Objective-C and iOS development. Apple's Xcode 4.6.1 IDE (integrated development environment) will be used for the provided example project.

Although the Mobile Csound API is provided with an excellent example project, it was felt that this tutorial will be a helpful supplement in setting up a basic Csound for iOS project for the first time, by including screenshots from the project set-up, and a section on common errors the user may encounter when working with the API.

The example project provided by the authors of the API includes a number of files illustrating various aspects of the API, including audio input/output, recording, interaction with GUI widgets, and multi-touch. More information on the example project can be found in the API manual, which is included in the example projects folder.

1.1. The Csound for iOS API

The Mobile Csound Platform allows programmers to embed the Csound audio engine inside of their iOS project. The API provides methods for sending static program information from iOS to the instance of Csound, as well as sending dynamic value changes based on user interaction with standard UI interface elements, including multi-touch interaction.

1.2. Document Structure

This document begins, in Section 2, by describing the example provided by the authors. Section 2 is divided into two further sections: Section 2.1 which describes the functionality of the example application and Section 2.2 which details line by line through the example code how this application works. Section 3 provides a step by step guide to setting up an Xcode project for use with the Mobile Csound API. This section describes how to download the API and include it into the project (Section 3.1) as well as the necessary components of the view controller (Section 3.2) and Csound file (Section 3.3). Section 4 outlines some common problems, which have been found through the creation of this tutorial, and their solutions. Section 5 is a reference of the methods which are available for use in the Mobile Csound API. This section briefly details the functionality of these methods and their method calls. Section 6 provides the authors' conclusions about this tutorial.

NOTE: This tutorial uses Csound 5, and has not been tested with Csound6.

2 Example Walkthrough

This section discusses why the example was made, and what can be learned from it; giving an overview of its functionality, then going into a more detailed description of its code. A copy of the example project can be found at the following link.

<https://sourceforge.net/projects/csoundiosguide/>

2.1 Running the Example Project

Run the provided Xcode project, CsoundTutorial.xcodeproj, and the example app should launch (either on a simulator or a hardware device). A screenshot of the app is shown in Figure 2.1 below. The app consists of two sliders, each controlling a parameter of a Csound oscillator. The top slider controls the amplitude, and the bottom slider controls the frequency.

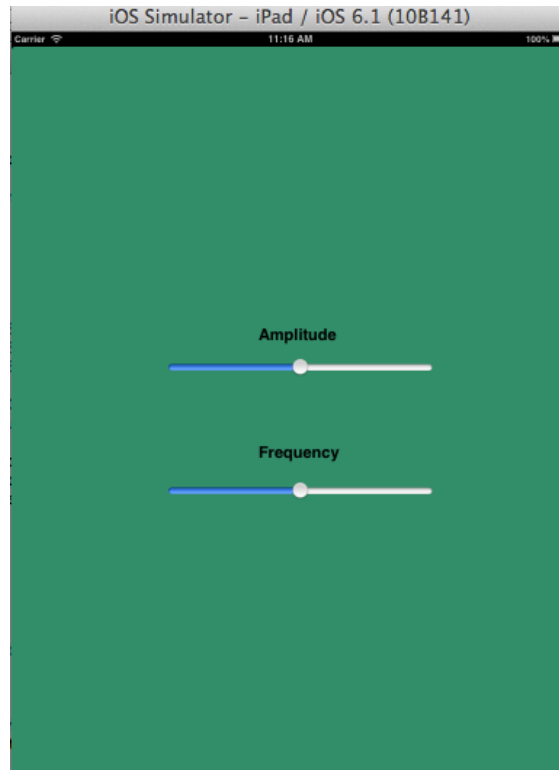


Figure 2.1-App running on iPad simulator

2.2 Oscillator Example Walkthrough

This example outlines how to use the methods in the Csound-iOS API to send values from iOS into Csound. This example was made purposefully simple, with the intent of making its functionality as obvious as possible to the reader. This section begins by giving an overview of both the iOS and Csound implementation, and then describes how this is achieved by breaking down the example code. The code to create this oscillator example was done in the *ViewController.h* and the *ViewController.m* files, which are discussed below in sections 2.2.3.1 and 2.2.3.2. The project is split into Objective-C code, Storyboards for the user interface elements, and a Csound file for the audio engine.

2.2.1 iOS Example Outline

In the Xcode project user interface sliders are used to allow a user to control the Csound audio engine through iOS. Communication begins with iOS requesting some memory within Csound; setting a pointer to this location. It updates this pointer with values from the user interface sliders. Csound references the same memory location by naming it with a string, this named communication link is called a channel. When sending this information, iOS uses methods within the iOS-Csound API to setup this channel name, and update it dependant on the control rate.

2.2.2. Csound Example Outline

In this example, Csound is not aware of iOS. All it knows is that there is a piece of memory assigned for it, and it must retrieve information from here dependent on its control rate. Csound uses the *chnget* opcode to do this. *chnget* searches for some channel with a specific name and retrieves values from it.

2.2.3. The iOS File

This example is implemented across two main files:

The **.h file** is used to include all the necessary classes, declare properties, and allow for user interaction by connecting the interface to the implementation.

The **.m file** is used to implement communication between the interface methods declared in the .h file, and the Csound file. These will now be discussed in more depth, with code examples.

2.2.3.1. The .h File

The imports (discussed in detail in section 3.2.1) are declared:

```
#import <UIKit/UIKit.h>
#import "CsoundObj.h"
#import "CsoundValueCacheable.h"
```

Apart from the standard UIKit.h (which gives access to iOS interface widgets) these ensure that the code written can access the information in the other files in the Csound API.

Next comes the class definition:

```
@interface ViewController : UIViewController
<CsoundObjCompletionListener, CsoundValueCacheable>
```

Every iOS class definition begins with the **@interface** keyword, followed by the name of the class. So our class is called *ViewController*, and the colon indicates that our class inherits all the functionality of the *UIViewController*.

Following this are two Protocol definitions, which are listed between the triangular brackets `< >`. In Objective-C a Protocol is a list of **required** functionality (i.e., methods) that a class needs to implement. In this case there are two Protocols that are defined by the Csound API, that we want our class to conform to: *CsoundObjCompletionListener* and *CsoundValueCacheable*. This will allow us to send data between iOS and Csound, and so is essential for what we are about to do. The required functions that we have to implement are described in the section following this one (2.2.3.2).

The Csound object needs to be declared as a property in the .h file, which is what this next line of code does:

```
//Declare a Csound object
@property (nonatomic, retain) CsoundObj* csound;
```

The next section of code allows for the interface objects (sliders) to communicate with the .m file:

```
- (IBAction)amplitudeSlider:(UISlider *)sender;
- (IBAction)frequencySlider:(UISlider *)sender;
```

Just to the left of each of these IBAction methods, you should see a little circle. If the storyboard is open (MainStoryboard.storyboard) you will see the appropriate slider being highlighted if you hover over one of the little circles.

2.2.3.2. The .m File

The .m file imports the .h file so that it can access the information within it, and the information that it accesses.

At the beginning of the implementation of the ViewController, the *csound* variable which was declared in the .h file is instantiated with `@synthesize` thus:

```
@implementation ViewController
@synthesize csound = mCsound;
```

Note that the Csound object must be released later in the *dealloc* method as shown below:

```
- (void)dealloc
{
    [mCsound release];
    [super dealloc];
}
```

For each parameter you have in iOS that you wish to send to Csound, you need to do the things outlined in this tutorial. In our simple example we have an iOS slider for Frequency, and one for Amplitude, both of which are values we want to send to Csound.

Some global variables are then declared, as shown in Table 2.1, a holder for each iOS parameter's current value, and a pointer for each which is going to point to a memory location within Csound.

Variable	Description
<code>float myFrequency;</code>	This value comes from the frequency slider in the interface. It is a float, as the value to send from iOS to Csound needs to be a floating point number. Its range is 0 – 500.
<code>float myAmplitude;</code>	This value comes from the amplitude slider in the interface. Its range is 0 – 1 because of the way the gain is controlled in the .csd file.
<code>float* freqChannelPtr;</code>	These variables are used in conjunction with the method <i>getInputChannelPtr</i> (described towards the end of this section) to send frequency and amplitude values to Csound.
<code>float* ampChannelPtr;</code>	

Table 2.1-Variables for the .m File

The next significant part of the .m file is the *viewDidAppear* method. When the view loads, and appears in iOS, this iOS SDK method is called. In the example, the following code is used to locate the Csound file:

```
NSString *tempFile = [[NSBundle mainBundle] pathForResource:@"aSimpleOscillator" ofType:@"csd"]
NSLog(@"FILE PATH: %@", tempFile);
```

This code searches the main bundle for a file called *aSimpleOscillator* of the type *csd* (which you will be able to see in Xcode’s left-hand File List, under the folder Supporting Files). It then assigns it to an *NSString* named *tempFile*. The name of the string *tempFile* is then printed out to confirm which file is running.

The methods shown in Table 2.2 are then called:

Method Call	Description
<pre>self.csound = [[CsoundObj alloc] init];</pre>	This instantiates the csound object, which will be our main contact between iOS and Csound. It allocates and initialises some memory to make an instance of the CsoundObj class.
<pre>[self.csound addCompletionListener:self];</pre>	Sets our code (self – i.e. ViewController) to be a listener for the Csound object.
<pre>[self.csound addValueCacheable:self];</pre>	Sets our code (self) to be able to send real-time values to the Csound object.
<pre>[self.csound startCsound:tempFile];</pre>	The Csound object uses the method <i>startCsound</i> to run the file at the string <i>tempFile</i> . Remember how <i>tempFile</i> was set up to point to the Csound csd file (in our case <i>aSimpleOscillator.csd</i>). So, in other words, this line launches Csound with the csd file you have provided.

Table 2.2-Csound API Methods

The methods that allow the value of the slider to be assigned to a variable are then implemented. This is done with both frequency, and amplitude. As shown below for the amplitude slider:

```
- (IBAction)amplitudeSlider:(UISlider *)sender  
{  
    UISlider *ampSlider = (UISlider *)sender;  
    myAmplitude = ampSlider.value;  
}
```

This method is called by iOS every time the slider is moved (because it is denoted as an *IBAction*, i.e. an Interface Builder Action call). The code shows that the *ampSlider* variable is of type *UISlider*, and because of that the current (new) value of the slider is held in *ampSlider.value*. This is allocated to the variable *myAmplitude*. Similar code exists for the frequency slider.

The protocol methods are then implemented. The previous section showed how we set up our class (*ViewController*) to conform to two Protocols that the Csound API provides: *CsoundObjCompletionListener* and *CsoundValueCacheable*.

Take a look at the place where these Protocols are defined, because a Protocol definition lists clearly what methods are required to be implemented to use their functionality.

For *CsoundValueCacheable* you need to look in the file *CsoundValueCacheable.h* (in the folder *valueCacheable*). In that file it's possible to see the protocol definition, as shown below, and its four required methods.

```
#import <Foundation/Foundation.h>

@class CsoundObj;

@protocol CsoundValueCacheable <NSObject>

-(void)setup:(CsoundObj*)csoundObj;
-(void)updateValuesToCsound;
-(void)updateValuesFromCsound;
-(void)cleanup;

@end
```

Every method needs at least an empty function shell. Some methods, such as *updateValuesFromCsound* are left empty, because – for the tutorial example – there is no need to get values from Csound. Other protocol methods have functionality added. These are discussed below.

The *setup* method is used to prepare the *updateValuesToCsound* method for communication with Csound:

```
-(void)setup:(CsoundObj* )csoundObj
{
    NSString *freqString = @"freqVal";
    freqChannelPtr = [csoundObj getInputChannelPtr:freqString];

    NSString *ampString = @"ampVal";
    ampChannelPtr = [csoundObj getInputChannelPtr:ampString];
}
```

The first line of the method body creates a string; *freqString*, to name the communication channel that Csound will be sending values to. The next line uses the *getInputChannelPtr* method to create the channel pointer for Csound to transfer information to. Effectively, iOS has sent a message to Csound, asking it to open a communication channel with the name “*freqVal*”. The Csound object allocates memory that iOS can write to, and returns a pointer to that memory address. From this point onwards iOS could send data values to this address, and Csound can retrieve that data by quoting the channel name “*freqVal*”. This is described in more detail in the next section (2.2.4).

The next two lines of the code do the same thing, for amplitude parameter. This process creates two named channels for Csound to communicate through.

The protocol method *updateValuesToCsound* uses variables in the .m file and assigns them to the newly allocated memory address used for communication. This ensures that when Csound looks at this specific memory location, it will find the most up to date value of the variable. This is shown below:

```
-(void)updateValuesToCsound
{
    *freqChannelPtr = myFrequency;
    *ampChannelPtr = myAmplitude;
}
```

The first line assigns the variable *myFrequency* (the value coming from the iOS slider for Frequency) to the channel *freqChannelPtr* which, as discussed earlier, is of type `float*`. The second line does a similar thing, but for amplitude.

For the other Protocol *CsoundObjCompletionListener* it is possible to look for the file *CsoundObj.h* (which is found in Xcode’s left-hand file list, in the folder called *classes*). In there is definition of the protocol.

```
@protocol CsoundObjCompletionListener
-(void)csoundObjDidStart:(CsoundObj*)csoundObj;
-(void)csoundObjComplete:(CsoundObj*)csoundObj;
```

In this example there is nothing special that needs to be done when Csound starts running, or when it completes, so the two methods (*csoundObjDidStart:* and *csoundObjComplete:*) are left as empty function shells. In the example, the protocol is left included, along with the empty methods, in case you wish to use them in your App.

2.2.4 The Csound File

This Csound file contains all the code to allow iOS to control its values and output a sinusoid at some frequency and amplitude taken from the on-screen sliders. There are three main sections: The Options, the Instruments, and the Score. These are all discussed in more detail in section 4. Each of these constituent parts of the .csd file will now be broken down to determine how iOS and Csound work together.

2.2.4.1 The Options

There's only one feature in the options section of the .csd that needs to be considered here; the flags. Each flag and its properties are summarised in Table 2.3.

Flag	Description
-o dac	Enables audio output to default device
-+rtmidi=null	Disables real-time MIDI Control
-d	Suppress all displays

Table 2.3-Csound Flags

2.2.4.2 The Instrument

The first lines of code in the instrument set up some important values for the .csd to use when processing audio. These are described in Table 2.4, and are discussed in more detail in the Reference section of the Csound Manual.

Line	Description
sr = 44100	This sets the sample rate of Csound to 44100 Hz. It is imperative that the sample rate of the Csound file corresponds with the sample rate of the sound card the code is running on.
ksmps = 64	This defines the control rate. In the example this will determine the speed that the variables in Csound are read. ksmps is actually the number of audio samples that are processed before another control update occurs. The actual control rate equates to sample rate / ksmps (i.e. 44100 / 64 = 689.0625 Hz).
nchnls = 2	This is the number of audio channels. 2 = standard stereo.
Odbfs = 1	This is used to ensure that audio samples are within the appropriate range, between zero and one. Anything greater than one will induce clipping to the waveform.

Table 2.4-Csound .csd Options

The instrument then takes values from Csound using the *chnget* opcode:

```

kfreq chnget "freqVal"
kamp chnget "ampVal"

```

Here, the *chnget* command uses the “*freqVal*” and “*ampVal*” channels previously created in iOS to assign a new control variable. The variables *kfreq* and *kamp* are control-rate variables because they begin with the letter ‘k’. They will be updated 689.0625 times per second. This may be faster or slower than iOS updates the agreed memory addresses, but it doesn’t matter. Csound will just take the value that is there when it accesses the address via the named channel.

These control-rate variables are used to control the amplitude and frequency fields of the opcode *oscil*; the Csound opcode for generating sinusoidal waves. This is then output in stereo using the next line.

```

asig poscil kamp,kfreq,1
outs asig,asig
endin

```

The third parameter of the *oscil* opcode in this case is 1. This means ‘use f-table 1’. Section 3.3 explains f-tables in more depth.

2.2.4.3 The Score

The score is used to store the f-tables the instrument is using to generate sounds, and it allows for the playing of an instrument. This instrument is then played, as shown below:

```
i1 0 10000
```

This line plays instrument 1 from 0 seconds, to 10000 seconds. This means that the instrument continues to play until it is stopped, or a great amount of time passes.

It is possible to send score events from iOS using the method *sendScore*. This is discussed in more depth in section 6.1.

3 Using the Mobile Csound API in an Xcode Project

Section 3 provides an overview of how to set up your Xcode project to utilize the Mobile Csound API, as well as how to download the API and include it into your project.

3.1 Setting up an Xcode Project with the Mobile Csound API

This section describes the steps required to set up an Xcode project for use with the Mobile Csound API. Explanations include where to find the Mobile Csound API, how to include it into an Xcode project and what settings are needed.

3.1.2 Creating an Xcode Project

This section briefly describes the settings which are needed to set up an Xcode project for use with the Mobile Csound API. Choose the appropriate template to suit the needs of the project being created. When choosing the options for the project, it is important that *Use Automatic Reference Counting* is not checked (Figure. 3.1). It is also unnecessary to include unit tests.

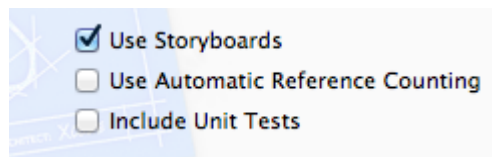


Figure 3.1-Project Set Up

Note: When including this API into a pre-existing project, it is possible to turn off ARC on specific files by entering the compiler sources, and changing the compiler flag to: `-fno-objc-arc`

3.1.3 Adding the Mobile Csound API to an Xcode Project

Once an Xcode project has been created, the API needs to be added to the Xcode project. To add the Mobile Csound API to the project, right click on the Xcode project and select *Add files to <myProject>*. This will bring up a navigation window to search for the files to be added to the project. Navigate to the *Csound-iOS* folder, which is located as shown in Figure 3.2 below.

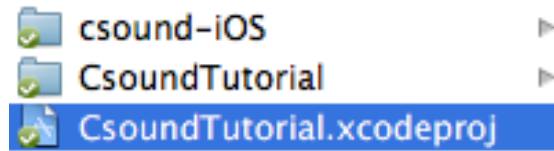


Figure 3.2-Navigating to the API Folder

Select the whole folder as shown and click *add*. Once this has been done, Xcode will provide an options box as shown in Figure 3.3. Check *Copy items into destination group's folder (if needed)*.

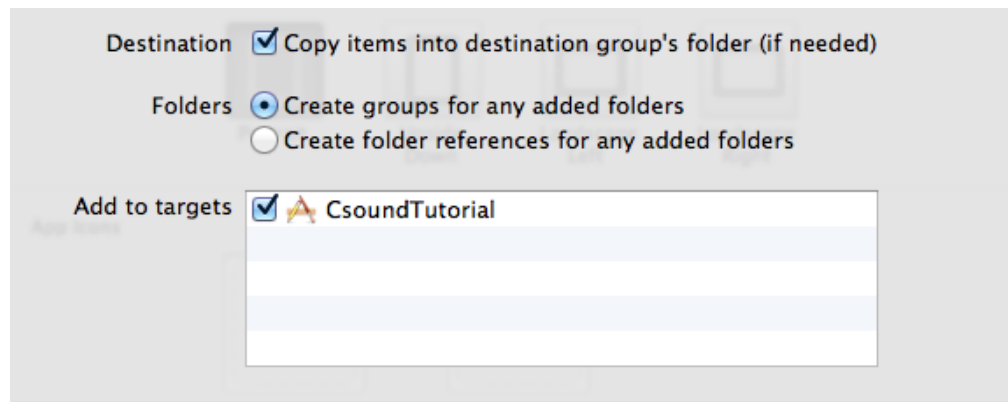


Figure 3.3-Adding the API Folder

The options in Figure 3.3 are selected so that the files which are necessary to run the project are copied into the project folder. This is done to make sure that there are no problems when the project folder is moved to another location - ensuring all the file-paths for the project files remain the same.

Once this addition from this section has been made, the project structure displayed in Xcode should look similar to that in Figure 3.4.

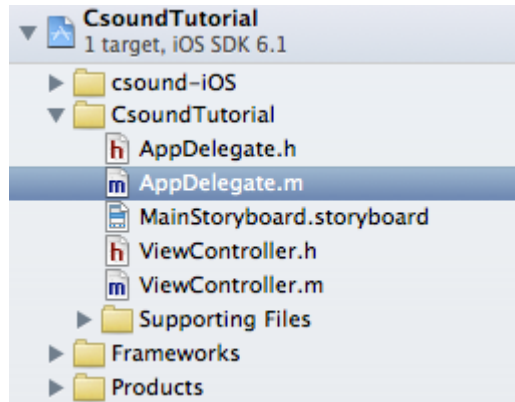


Figure 3.4 - The Main Bundle for the Project

3.1.4 Compiling Sources

A list of compile sources is found by clicking on the blue project file in Xcode, navigating to the *Build Phases* tab and opening *Compile Sources*. Check that the required sources for the project are present in the *Compile Sources* in Xcode. All the files displayed in Figure 3.5 should be present, but not necessarily in the same order as shown.

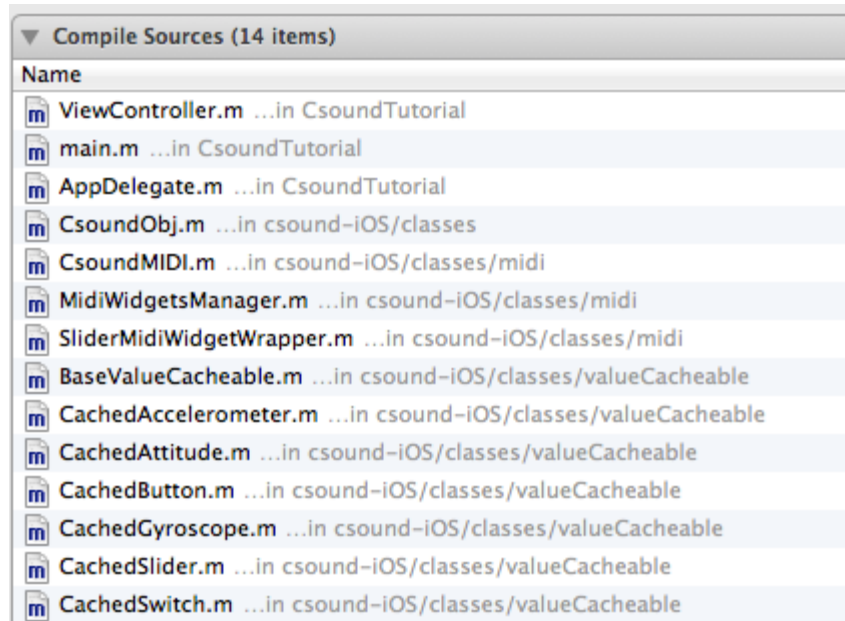


Figure 3.5-View of 'Compile Sources' Window

3.1.5 Including the Necessary Frameworks

There are some additional frameworks which are required to allow the project to run. These frameworks are:

- AudioToolbox.framework
- CoreGraphics.framework
- CoreMotion.framework
- CoreMIDI.framework

To add these frameworks to the project, navigate to the 'Link Binary With Libraries' section of Xcode. This is found by clicking on the blue project folder and navigating to the 'Build Phases' tab, followed by opening 'Link Binary With Libraries'. To add a framework, click on the plus sign and search for the framework required. Once all the necessary frameworks are added, the 'Link Binary With Libraries' should look similar to Figure 3.6 below.

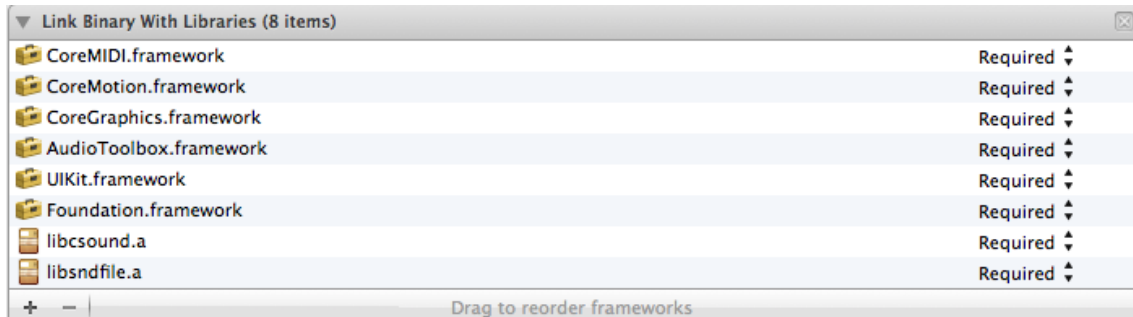


Figure 3.6-Adding Necessary Frameworks

3.1.6 The .csd File

The project is now set up for use with the Mobile Csound API. The final file which will be required by the project is a .csd file which will describe the Csound instruments to be used by the application. A description of what the .csd file is and how to include one into the project is found in *Section 3.3*. This file will additionally need to be referenced appropriately in the Xcode project. A description of where and how this reference is made is available in *Section 2.2.3.2*.

3.2 Setting up the View Controller

This section describes how the *ViewController.h* and the *ViewController.m* should be set up to ensure that they are able to use the API. It will discuss what imports are needed; conforming to the protocols defined by the API; giving a brief overview. This section can be viewed in conjunction with the example project provided.

3.2.1 Importing

So that the code is able to access other code in the API, it is important to include the following imports, along with imports for any additional files required. The three imports shown in Table 3.1 are used in the header file of the view controller to access the necessary files to get Csound-iOS working:

Import	Description
#import "CsoundObj.h"	This is used so that the code is able to access all the key methods of the API.
#import "CsoundValueCacheable.h"	This must be used to access the methods 'updateValuesFromCsound' and 'updateValuesToCsound'. These methods are used to communicate between Csound and iOS.

Table 3.1-Header File Imports

In our example you can see these at the top of *ViewController.h*

3.2.2 Conforming to Protocols

It is imperative that the view controller conforms to the protocols outlined in the `CsoundObj.h` file; the file in the API that allows for communication between iOS and Csound. This must then be declared in the `ViewController.h` file:

```
@interface ViewController : UIViewController
<CsoundObjCompletionListener, CsoundValueCacheable>
```

The API authors chose to use protocols so that there is a defined set of methods that must be used in the code. This ensures that a consistent design is adhered to. They are defined in the `CsoundValueCacheable.h` file thus:

```
@class CsoundObj;

@protocol CsoundValueCacheable <NSObject>

-(void)setup:(CsoundObj*)csoundObj;
-(void)updateValuesToCsound;
-(void)updateValuesFromCsound;
-(void)cleanup;
```

Each of these must then be implemented in the `ViewController.m` file. If it is unnecessary to implement one of these methods, it still *must* appear but the method body can be left blank, thus:

```
-(void)updateValuesFromCsound
{
    //No values coming from Csound to iOS
}
```

3.2.3 Overview of Protocols

When writing the code which allows us to send values from iOS to Csound, it is important that the code conforms to the following protocol methods (Table 3.2):

Protocol methods	Action
-(void)setup:(CsoundObj*)CsoundObj	Set up the necessary channels and pointers to communicate with Csound.
-(void)updateValuesToCsound	Update the values being sent from iOS to Csound.
-(void)updateValuesFromCsound	Collect any values from Csound.
-(void)cleanup	Reset any values used in communication and de-allocate any memory used.
-(void)csoundObjDidStart:(CsoundObj*)csoundObj	This method is called when a Csound object is created. This allows developers to notify the user that Csound is running on iOS.
-(void)csoundObjComplete:(CsoundObj*)csoundObj	Much like the way the 'csoundObjDidStart' method works, this allows developers to notify the user that Csound has stopped running in iOS.

Table 3.2-Protocol methods which must be implemented in your *ViewController*.

3.3 Looking at the Csound ‘.csd’ File

The following section provides an overview of the Csound editing environment, the structure of the .csd file, and how to include the .csd file into your Xcode project.

3.3.1 Downloading Csound

A Csound front-end editor, CsoundQt, can be used for editing the .csd file in the provided example project. It is advised to use CsoundQt with iOS because it is an ideal environment for developing and testing the Csound audio engine – error reports for debugging, the ability to run the Csound audio code on its own, and listen to its results. However, using CsoundQt is not essential to use Csound as an audio engine as Csound is a standalone language. CsoundQt is included in the Csound package download.

In order to use Csound in iOS, the latest version of Csound (*Version 5.19*) will need to be installed.

Csound 5.19 can be downloaded from the following link:

<http://sourceforge.net/projects/Csound/files/Csound5/Csound5.19/>

In order for Xcode to see the .csd file, it must be imported it into the Xcode project. This is done by right-clicking on the ‘Supporting Files’ folder in the project, and clicking on ‘Add files to (*project name*)’ (Figure 3.7).

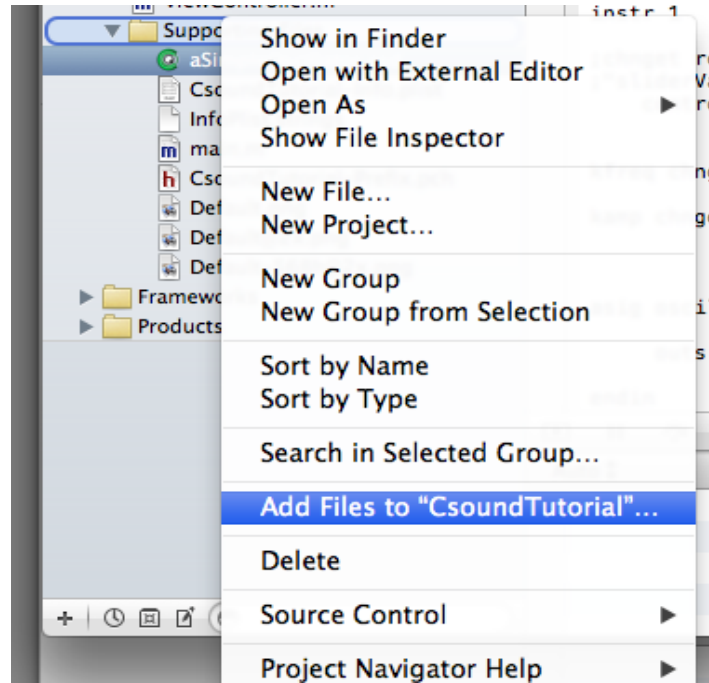


Figure 3.7-Adding the .csd to iOS Project

It is possible to edit the .csd file while also working in Xcode. This is done by right-clicking on the .csd file in Xcode, and clicking on 'Open With External Editor' (Figure 3.8).

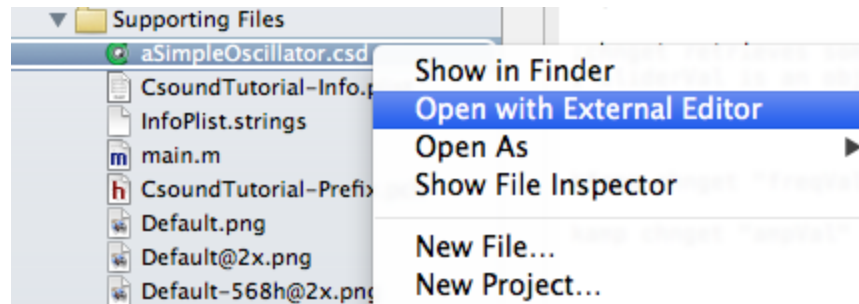


Figure 3.8-Opening the .csd file with an external editor

However, it is important to remember to save any changes to the .csd file before the Xcode project is recompiled.

3.3.2 The .csd File

When setting up a Csound project, it is important that various audio and performance settings configured correctly in the header section of the .csd file. These settings are described in Table 3.3, and are discussed in more detail in the Csound Manual.

Setting	Description
sr	Sample rate
kr	Control rate
ksmps	Number of samples in control period (sr/kr)
nchnls	Number of channels of audio output
0dbfs	Sets value of 0 decibels using full scale amplitude

Table 3.3-Csound .csd Settings

It is important that the sample rate for the Csound project be set to the same sample rate as the hardware it will be run on. For this project, make sure the sample rate set to 44100, as depicted in Figure 3.9. This is done by opening the Audio MIDI Setup, which is easily found on all *Mac* computers by searching in *Spotlight*.

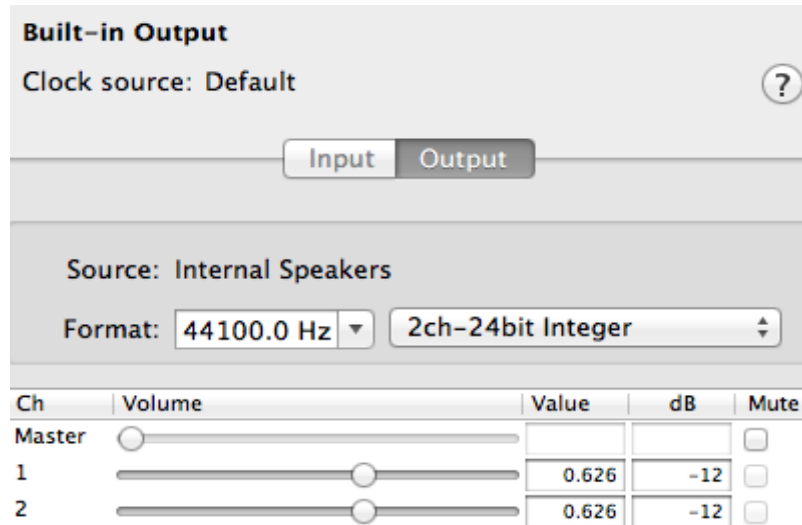
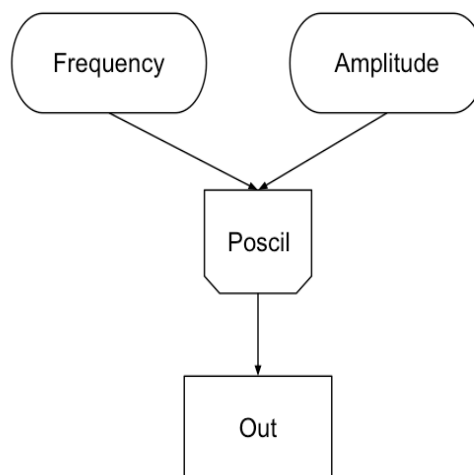


Figure 3.9-Configuring Audio Hardware Settings

3.3.3 Instruments

As mentioned previously, Csound instruments are defined in the orchestra section of the .csd file. The example project provided by the authors uses a simple oscillator that has two parameters: amplitude and frequency, both of which are controlled by UI sliders.

Figure 3.10 on the following page shows a block diagram of the synthesizer we are using in the example project.



3.3.4 Score

The score is the section of the .csd file which provides instruments with control instruction, for example pitch, volume, and duration. However, as the goal here is for users to be able to interact with the Csound audio engine in real-time, developers will most likely opt instead to send score information to Csound that is generated by UI elements in the Xcode project. Details of the instrument and score can be found in the comments of the *aSimpleOscillator.csd* file

Csound uses GEN (f-table generator) routines for a variety of functions. This project uses GEN10, which create composite waveforms by adding partials. At the start of the score section, a GEN routine is specified by function statements (also known as *f-statements*). The parameters are shown below in Table 3.4:

Parameter	Description
f 1	Unique f-table identification number
0	f-statement initialization time expressed in score beats
16384	f-table size
10	GEN routine called to create the f-table
1	strength of ascending partials

Table 3.4-Csound .csd F-Table Parameters

In a Csound score, the first three parameter fields (also known as p-fields) are reserved for the instrument number, the start time, and duration amount. P-fields 4 and 5 are conventionally reserved for amplitude and frequency, however, P-fields beyond 3 can be programmed as desired.

The p-fields used in the example project are shown in Table 3.5.

p-field	1	2	3	4	5
Parameter	Instrument Number	Start	Duration	Amplitude	Frequency

Table 3.5-Csound .csd P-field Parameters

In this project, the first three p-fields are used: the instrument number (i1), the start time (time = 0 seconds), and the duration (time = 1000 seconds). Amplitude and frequency are controlled by UI sliders in iOS.

4 Common Problems

This section is designed to document some common problems faced during the creation of this tutorial. It is hoped that by outlining these common errors, readers can debug some common errors they are likely to come across when creating applications using this API. It discusses each error, describes the cause and outlines a possible solution.

4.1 UIKnob.h is Not Found

This is a problem related to the API. The older versions of the API import a file in the examples that sketches a UIKnob in Core Graphics. This is not a part of the API, and should not be included in the project.

The file in question is a part of the examples library provided with the SDK. It is used in the file ‘AudioIn test’ and is used to sketch a radial knob on the screen. It gives a good insight into how the user can generate an interface to interact with the API.

Solution: Comment the line out, or download the latest version of the API.

4.2 Feedback from Microphone

This is generally caused by the sample rate of a .csd file being wrong.

Solution: Ensure that the system’s sample rate is the same as in the .csd file. Going to the audio and MIDI set-up and checking the current output can find the computer’s sample rate. See section 3.3.2 for more information.

4.3 Crackling Audio

There are numerous possible issues here, but the main cause of this happening is a CPU overload.

Solution: The best way to debug this problem is to look through the code and ensure that there are no memory intensive processes, especially in code that is getting used a lot. Problem areas include fast iterations (loops), and code where Csound is calling a variable. Functions such as *updateValuesToCsound* and *updateValuesFromCsound* are examples of frequently called functions.

An example: an NSLog in the *updateValuesToCsound* method can cause a problem. Say, the *ksmps* in the .csd is set to 64. This means that the Csound is calling for iOS to run the method *updateValuesToCsound* every 64 samples. Assuming the sample rate is 44.1k this means that this CPU intensive NSLog is being called ~689 times a second; very computationally expensive.

4.4 Crackling from amplitude slider

When manipulating the amplitude slider in iOS, a small amount of clicking is noticeable. This is due to the fact that there is no envelope-smoothing function applied to the amplitude changes. While this would be an improvement on the current implementation, however; it was felt that the current implementation would be more conducive to learning for the novice Csound user. This would be implemented by using a port opcode.

5 Csound Library Methods

This section will present and briefly describe the methods which are available in the Manual.

5.1 Basic API Methods

Name	Method Call	Description
startCsound	<pre>-(void) startCsound: (NSString*) csdFilePath;</pre>	Provides the location of the .csd file which is to be used with the Csound object.
	<pre>-(void) startCsound: (NSString *) csdFilePath recordToURL: (NSURL *) outputURL;</pre>	Provides the location of the .csd file which is to be used with the Csound object and specifies a URL to which it will record.
startCsoundToDisk	<pre>-(void) startCsoundToDisk: (NSString*) csdFilePath outputFile: (NSString*) outputFile;</pre>	Provides the location of the .csd file which is to be used with the Csound object and specifies a file to which it will record. This does not occur in realtime, but as fast as possible to the disk. This method is useful for batch rendering.
stopCsound	<pre>-(void) stopCsound;</pre>	This uses the Csound object's method 'stopCsound' to stop the instance of CsoundObj that it is called on.
muteCsound	<pre>-(void) muteCsound;</pre>	Mutes all instances of Csound
unmuteCsound	<pre>-(void) unmuteCsound;</pre>	Unmutes all instances of Csound
recordToURL	<pre>-(void) recordToURL: (NSURL *) outputURL;</pre>	Begins recording to a specified URL. This can be defined at a later point in the code, even after Csound has been started.
stopRecording	<pre>-(void) stopRecording;</pre>	Stops recording to URL

5.2 UI and Hardware Methods

Name	Method Call	Description
addSwitch	<pre>(id<CsoundValueCacheable>) addSwitch: (UISwitch*) uiSwitch forChannelName: (NSString*) channelName;</pre>	<p>Adds a switch to the Csound object. The method requires a switch which already exists as part of the user interface and a name for the channel which will provide information about this switch to the .csd file. For more information about channels of information between Xcode and Csound see section 5.</p>
addSlider	<pre>(id<CsoundValueCacheable>) addSlider: (UISlider*) uiSlider forChannelName: (NSString*) channelName;</pre>	<p>Adds a slider to the Csound Object. The method requires a slider and a channel name.</p>
addButton	<pre>(id<CsoundValueCacheable>) addButton: (UIButton*) uiButton forChannelName: (NSString*) channelName;</pre>	<p>Adds a button to the Csound Object. The method requires a button and a channel name.</p>
enableAccelerometer	<pre>(id<CsoundValueCacheable>) enableAccelerometer;</pre>	<p>Enables the accelerometer for use with the Csound object.</p>
enableGyroscope	<pre>(id<CsoundValueCacheable>) enableGyroscope;</pre>	<p>Enables the gyroscope for use with the Csound object.</p>
enableAttitude	<pre>(id<CsoundValueCacheable>) enableAttitude;</pre>	<p>Enables attitude to allow device motion to be usable with the Csound object.</p>

5.3 Communicating between Xcode and Csound

Name	Method Call	Description
addValueCacheable	<pre>-(void) addValueCacheable: (id<CsoundValueCacheable>) valueCacheable;</pre>	Adds to a list of watched objects so that they can update every cycle of ksmps.
removeValueCacheable	<pre>-(void) removeValueCaheable: (id<CsoundValueCacheable>) valueCacheable;</pre>	Removes a cacheable value from the Csound Object.
sendScore	<pre>-(void) sendScore: (NSString*) score;</pre> <p>Eg:</p> <pre>[self.csound sendScore: [NSString stringWithFormat: @"i1 0 10 0.5 %d", myPitch,]];</pre> <p>(sends a score to instrument 1 that begins at 0 seconds, stops at 10 seconds, with amplitude 0.5 and a pitch of the objective-C variable 'myPitch').</p>	Sends a score as a string to the .csd file. See section 4 for formatting a Csound score line.
addCompletionListener	<pre>-(void) addCompletionListener: (id<CsoundObjCompletionListener>) listener;</pre>	Adds a listener for the Csound Object which waits for an action to be performed that the Csound object needs to react to.

5.4 Retrieve Csound-iOS Information

Name	Method Call	Description
getCsound	<code>-(CSOUND*) getCsound;</code>	Returns the C structure that the CsoundObj uses. This allows developers to use the Csound C API in conjunction with the Objective-C CsoundObj API.
getInputChannelPtr	<code>(float*) getInputChannelPtr: (NSString*) channelName;</code>	Returns the float of an input channel pointer.
getOutputChannelPtr	<code>(float*) getOutputChannelPtr: (NSString*) channelName;</code>	Returns the float of an output channel pointer.
getOutSamples	<code>-(NSData*) getOutSamples;</code>	Gets audio samples from Csound.
getNumChannels	<code>-(int) getNumChannels;</code>	Returns the number of channels in operation.
getKsmpls	<code>-(int) getKsmpls;</code>	Returns ksmpls as defined in the .csd file.
setMessageCallback	<code>-(void) setMessageCallback: (SEL) method withListener:(id) listener;</code>	Sets up a method to be the callback method and a listener id.
performMessageCallback	<code>(void) performMessageCallback: (NSValue *) infoObj;</code>	Performs the message callback.

6 Conclusions

This tutorial provided an overview of the Csound-iOS API, outlining its benefits, and describing its functionality by means of an example project. It provided the basic tools for using the API, equipping iOS developers to explore the potential of this API in their own time.

APIs such as this one, as well as others including *libpd* and *The Amazing Audio Engine* provide developers with the ability to integrate interactive audio into their apps, without having to deal with the low-level complexities of Core Audio.

6.1 Additional Resources

Upon completion of this tutorial, the authors suggest that the reader look at the original Csound for iOS example project, written by Steven Yi and Victor Lazzarini.

This is available for download from:

<http://sourceforge.net/projects/csound/files/csound5/iOS/>

About the Authors

The authors are Masters students at the University of York Audio Lab. Each one is working on a separate interactive audio app for the iPad, and has each been incorporating the Mobile Csound API for that purpose. They came together to write this tutorial to make other developers aware of the Mobile Csound API, and how to utilize it.

The motivation behind this tutorial was to create a step by step guide to using the Mobile Csound API. When the authors originally started to develop with the API, they found it difficult to emulate the results of the examples that were provided with the API download. As a result, the authors created a simple example using the API, and wanted others to learn from our methods and mistakes. The authors hope that this tutorial provides clarity in the use of the Mobile Csound API.

Appendix C - Csound for iOS Tutorial Code

The Xcode project described in the Csound for iOS tutorial is included on the additional DVD. Additionally, the project can be downloaded from the following SourceForge link:

<https://sourceforge.net/projects/csoundiosguide/>

Appendix D - Publication and Recognition of Csound for iOS Tutorial Guide

The following is a link to a blog entry regarding the Csound for iOS Tutorial Guide: [CreateDigitalMusic](#)

The tutorial is also referenced on the Csound for iOS SourceForge WIKI, which can be found here: [Csound for iOS SourceForge WIKI](#)

Additionally, the author was asked to submit the contents of the tutorial to the open-source FLOSS manual section on Csound. The chapter can be found at the following link: [Csound for iOS Chapter in the FLOSS Manual](#)

The following page is an email that was sent by Dr. Richard Boulanger, editor of the *The Csound Book* and developer of apps utilizing the Csound for iOS API, to the Csound users mailing list regarding the Csound for iOS Tutorial.



Nicholas Arner <na705@york.ac.uk>

Csound for iOS Tutorial

Dr. Richard Boulanger <rboulanger@berklee.edu>

24 May 2013 16:00

Reply-To: csound@lists.bath.ac.uk

To: "csound@lists.bath.ac.uk" <csound@lists.bath.ac.uk>

Dear Nicolas, Timothy, and Abigail,

Really nice work. I look forward to sharing it with my students and my team at Boulanger Labs. A wonderful and helpful contribution to the Csound community. You should consider adding it (or some version of it) to the Floss Manual.

Hope to see you and your team at the Csound Conference in October and have you showcase your app(s) there.

Thanks so much. Good luck on your studies, your dissertation, and your "apps".

- Dr.B.

Dr. Richard Boulanger 

Professor of Electronic Production and Design

Professional Writing and Music Technology Division

Office @ 161 Mass Ave - 4th Floor

617-747-2485 (office) 774-488-9166 (cell)

<http://csounds.com/boulanger> <http://boulangerlabs.com> <http://csoundforlive.com>Create your free signature: [CLICK HERE!](#)**Dr. Richard Boulanger, Ph.D.**

Professor of Electronic Production and Design

Professional Writing and Music Technology Division

Office @ 161 Mass Ave - 4th Floor

617-747-2485 (office) 774-488-9166 (cell)

<http://csounds.com/boulanger> <http://csounds.com/mathews><http://boulangerlabs.com> <http://csoundforlive.com> <http://csounds.com>

[Quoted text hidden]

Appendix E - Gestural Intuitiveness Study Participant Handout

Appendix E consists of the participant handout given to subjects partaking the *Preliminary Study on Gestural Intuitiveness*, discussed in Chapter 3.

Participant Handout

Nicholas Arner, MSc Music Technology

University of York Audio Lab, Department of Electronics

Thesis Project App

The author is working to develop an iOS application that assists users in composing music. It is planned that the user will use multi-touch gestures to modify the sound in real-time to produce thick sonic textures, allowing the user to interactively compose pieces of music.

Purpose/Application

The purpose of the study is to determine how closely the gestures that users draw match the sounds they are listening to. Listeners will draw gestural shapes in response to listening to parameter changes of a granular synthesizer similar to the one used in the author's project app.

The listeners will use the touch-screen of an iPad (that is turned off) to generate their gestural responses, which will be captured using a video recorder.

The conclusions drawn from the analysis will be used to help determine the mapping of multi-touch gestures to granular synthesis parameters in the project iPad app.

Instructions

- You will hear several audio clips played in succession. Each clip will be played three times.
- When you hear the clip for the third time, please pretend that *You* are the one generating the sound clip by making a gesture on the provided iPad. You are allowed to make any kind of multi-touch gesture, using both hands to generate the gesture if you wish.
- Enjoy some cookies for your time!

Participant Questionnaire

What is your age? _____

Please list your gender _____

What is your nationality? _____

Are you a university student (Yes/No)? _____

If yes, what is your year of study? _____

Are you a member of the Academic staff? _____

If yes, what department are you based in? _____

Do you own an iOS device (Yes/No)? _____

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

If you are a musician, have you ever used an iOS device to make music (Yes/No)? _____

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

Appendix F - Gestural Intuitiveness Study Participant Questionnaires

Appendix F consists of the participant questionnaires that subjects filled out after completing the study on gestural intuitiveness. PDF copies of the completed questionnaires can be found on the additional DVD.

Participant Questionnaire

What is your age? 22

Please list your gender m

What is your nationality? Dutch

Are you a university student (Yes/No)? Yes

If yes, what is your year of study? 2nd year Master

Are you a member of the Academic staff? no

If yes, what department are you based in? /

Do you own an iOS device (Yes/No)? No

If no, have you ever used an iOS device? Hardly

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Yes, - music technology researcher
- violin

If you are a musician, have you ever used an iOS device to make music (Yes/No)? No

If yes, please describe the type of music and the apps that you use below.

/

What music do you enjoy listening to?

classical, rock, gothic metal, latin-american
music

Participant Questionnaire

What is your age? 22

Please list your gender Male

What is your nationality? British

Are you a university student (Yes/No)? Yes

If yes, what is your year of study? 4 (Research student)

Are you a member of the Academic staff? No

If yes, what department are you based in? Electronics

Do you own an iOS device (Yes/No)? Yes

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

I play and write music on several instruments. Additionally I did an audio based undergraduate degree, and am currently undertaking an audio-based postgraduate degree.

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? Yes

If yes, please describe the type of music and the apps that you use below.

Garageband, Alchemy (camel audio), Figure and other anomalous instrument applications.

What music do you enjoy listening to?

Progressive rock, jazz fusion, post-rock.

Participant Questionnaire

What is your age? 32

Please list your gender M

What is your nationality? BRITISH

Are you a university student (Yes/No)? YES

If yes, what is your year of study? 2

Are you a member of the Academic staff? NO

If yes, what department are you based in? -

Do you own an iOS device (Yes/No)? YES

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Been a professional musician, also an audio professional.

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? NO

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

ALL GENRES

Participant Questionnaire

What is your age? 28.

Please list your gender female

What is your nationality? Greek

Are you a university student (Yes/No)? Yes

If yes, what is your year of study? 4th

Are you a member of the Academic staff? No

If yes, what department are you based in? Electronics

Do you own an iOS device (Yes/No)? No

If no, have you ever used an iOS device? Yes

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Yes, ~~Music~~ Music Studies for
BA, MSc and PhD

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? Yes

If yes, please describe the type of music and the apps that you use below.

as part of a Master Assignment..

What music do you enjoy listening to?

Basically everything...

Participant Questionnaire

What is your age? 32

Please list your gender MALE

What is your nationality? BRITISH

Are you a university student (Yes/No)? NO

If yes, what is your year of study? _____

Are you a member of the Academic staff? YES

If yes, what department are you based in? AUDIO LAB (ELECTRONICS)

Do you own an iOS device (Yes/No)? YES

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

PIANO - GRADE 8

CLARINET - GRADE 8

AUDIO PROGRAMMING FOR WORK AS RESEARCHER

COMPUTER MUSIC PRODUCTION AS A HOBBY

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? ~~NO~~ YES

If yes, please describe the type of music and the apps that you use below.

ELECTRONIC DANCE MUSIC

MOOG-ANIMOOG, 303 EMULATOR, DRUM MACHINES

What music do you enjoy listening to?

MOSTLY ELECTRONIC DANCE MUSIC,

CLASSICAL

Participant Questionnaire

What is your age? 24

Please list your gender MALE

What is your nationality? BRITISH

Are you a university student (Yes/No)? YES

If yes, what is your year of study? 1ST YEAR Ph.D.

Are you a member of the Academic staff? No

If yes, what department are you based in? N/A

Do you own an iOS device (Yes/No)? No

If no, have you ever used an iOS device? YES

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Performance → guitar - Grade 8, violin Grade 7,
Theory grade 5. 5/6 years orchestral
experience (amateur). 8 Years playing in a band.
2 Degrees in MusTech.

If you are a musician, have you ever used an iOS device to make music (Yes/No)? No

If yes, please describe the type of music and the apps that you use below.

N/A.

What music do you enjoy listening to?

METAL.

Participant Questionnaire

What is your age? 25

Please list your gender Female

What is your nationality? British

Are you a university student (Yes/ No)? _____

If yes, what is your year of study? 1st year PhD

Are you a member of the Academic staff? No

If yes, what department are you based in? _____

Do you own an iOS device (Yes/ No)? _____

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Piano - Grade 7
Violin - Grade 8
Music A-Level

If you are a musician, have you ever used an iOS device to make music

(Yes/ No)? _____

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

Lots of genres!
Pop, Indie, Motown, Electronic...

Participant Questionnaire

What is your age? 47

Please list your gender M

What is your nationality? BRITISH

Are you a university student (~~Yes~~/No)? NO (staff)

If yes, what is your year of study? /

Are you a member of the Academic staff? YES

If yes, what department are you based in? ELECTRONICS.

Do you own an iOS device (Yes/No)? YES

If no, have you ever used an iOS device? /

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

- Mustech lectures
- PhD Audio Interfaces
- Piano, violin, bass, guitar. (Piano = good student)
- Composition for music & media.

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? YES

If yes, please describe the type of music and the apps that you use below.

My own apps - in youth music situations.

What music do you enjoy listening to?

Everything but hard to class.

Participant Questionnaire

What is your age? 22

Please list your gender Female

What is your nationality? British

Are you a university student (Yes/No)? Yes.

If yes, what is your year of study? 5.

Are you a member of the Academic staff? No.

If yes, what department are you based in? Electronics.

Do you own an iOS device (Yes/No)? Yes.

If no, have you ever used an iOS device? /

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Violin, Viola, Piano, singing
Sound Design for Theatre.

If you are a musician, have you ever used an iOS device to make music (Yes/No)? No.

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

Large Variety

Participant Questionnaire

What is your age? 21

Please list your gender Female

What is your nationality? British

Are you a university student (Yes/No)? Yes

If yes, what is your year of study? 3

Are you a member of the Academic staff? No

If yes, what department are you based in? Electronics

Do you own an iOS device (Yes/No)? Yes

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Played Flute for 16 years, Bass guitar for 7, other instruments too. Music A level, music tech degree.

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? No

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

Everything? Really into Electronic stuff atm.

Participant Questionnaire

What is your age? 27

Please list your gender Male

What is your nationality? Chinese

Are you a university student (Yes/No)? Yes

If yes, what is your year of study? 2nd

Are you a member of the Academic staff? No

If yes, what department are you based in? Electronic

Do you own an iOS device (Yes/No)? Yes

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

No

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? No

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

Electronic & Pop

Participant Questionnaire

What is your age? 31

Please list your gender female

What is your nationality? British

Are you a university student (Yes/No)? No

If yes, what is your year of study? _____

Are you a member of the Academic staff? yes

If yes, what department are you based in? Electronic

Do you own an iOS device (Yes/No)? Yes

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Yes. Musician (professional singer BA, MA, PhD
music tech)

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? yes

If yes, please describe the type of music and the apps that you use below.

playing with a cat keyboard. No serious
music...

What music do you enjoy listening to?

Pop (60s 80s modern) Opera (classical/romantic)
Indie, Rock, orchestral
everything except R&B.

Participant Questionnaire

What is your age? 27

Please list your gender M

What is your nationality? CHINESE

Are you a university student (Yes/No)? YES.

If yes, what is your year of study? 2nd.

Are you a member of the Academic staff? NO.

If yes, what department are you based in? _____

Do you own an iOS device (Yes/No)? YES.

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

YES. I'm a ~~data~~ ^{sonification} researcher working on giving real-time sonic feedback to user while making meetings.

I play guitar and compose ambient/electronic music myself.

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? YES.

If yes, please describe the type of music and the apps that you use below.

Ambient Soundscape, ~~that~~ 8 bit game boy music.

What music do you enjoy listening to?

Electronic, Ambient, Jazz, Glitch, IDM.

Participant Questionnaire

What is your age? 23

Please list your gender female

What is your nationality? USA

Are you a university student (Yes/No)? Yes No

If yes, what is your year of study? MA (1yr.)

Are you a member of the Academic staff? No.

If yes, what department are you based in? _____

Do you own an iOS device (Yes/No)? Yes No

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Yes - vocal student (classical), piano, guitar, flute
(choirs, bands + for fun)

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? Yes No

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

Everything...

Participant Questionnaire

What is your age? _____

Please list your gender Female

What is your nationality? British

Are you a university student (Yes/No)? No

If yes, what is your year of study? —

Are you a member of the Academic staff? Yes

If yes, what department are you based in? Electronics

Do you own an iOS device (Yes/No)? No

If no, have you ever used an iOS device? yes

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

yes - play violin piano recorder, sing
in choirs, choral conductor
work in audio and music
technology

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? No

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

anything except thrash metal
and Dido

Participant Questionnaire

What is your age? 27

Please list your gender FEMALE

What is your nationality? JORDANIAN

Are you a university student (Yes/No)? Yes No

If yes, what is your year of study? LITERATURE

Are you a member of the Academic staff? NO

If yes, what department are you based in? -

Do you own an iOS device (Yes/No)? Yes No

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

NONE . -

If you are a musician, have you ever used an iOS device to make music (Yes/No)? _____

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

HOUSE, ROCK/INDIE ROCK. R&B

Participant Questionnaire

What is your age? 30

Please list your gender female.

What is your nationality? Canadian

Are you a university student (Yes/No)? Yes.

If yes, what is your year of study? Master's.

Are you a member of the Academic staff? No.

If yes, what department are you based in? Environment Dept.

Do you own an iOS device (Yes/No)? Yes.

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

No.

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? No

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

Alternative, ~~rock~~ folk rock, hip hop, electronic, classical.

Participant Questionnaire

What is your age? 26

Please list your gender FEMALE

What is your nationality? POLISH

Are you a university student (Yes/No)? YES

If yes, what is your year of study? POSTGRAD

Are you a member of the Academic staff? -

If yes, what department are you based in? -

Do you own an iOS device (Yes/No)? NO

If no, have you ever used an iOS device? YES

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

I PLAYED PIANO WHEN I WAS A CHILD
AND GUTAR IN HIGH SCHOOL. ALSO SANG IN
SCHOOL CHOIR IN HIGH SCHOOL.

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? -

If yes, please describe the type of music and the apps that you use below.

-

What music do you enjoy listening to?

A LITTLE BIT OF EVERYTHING.

Participant Questionnaire

What is your age? 26

Please list your gender Female

What is your nationality? Chinese

Are you a university student (Yes/No)? No

If yes, what is your year of study? _____

Are you a member of the Academic staff? No

If yes, what department are you based in? _____

Do you own an iOS device (Yes/No)? Yes

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

No

If you are a musician, have you ever used an iOS device to make music (Yes/No)? _____

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

Pop, country, rock-roll.

Participant Questionnaire

What is your age? 28

Please list your gender Female

What is your nationality? British

Are you a university student (Yes/No)? _____

If yes, what is your year of study? 1st year PhD - Centre for Women's Studies

Are you a member of the Academic staff? No

If yes, what department are you based in? N/A

Do you own an iOS device (Yes/No)? _____

If no, have you ever used an iOS device? No

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

I have been in an amateur choir. Have a bit of experience watching conductors. Played the violin very briefly many years ago.

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? N/A

If yes, please describe the type of music and the apps that you use below.

N/A

What music do you enjoy listening to?

Broad taste. Some classical (mainly baroque, some modern), bands like Radiohead, rock music, bit of pop, Motown, independent solo artists (singer/songwriter).

Appendix G - Gestural Intuitiveness Study Spreadsheets

Excel spreadsheets used for data analysis in Chapter 3 of this thesis, *Preliminary Study on Gestural Intuitiveness*, are included on the additional DVD.

Appendix H - Csound Code/audio files for Gestural Intuitiveness Test

The Csound .csd file and the audio clips generated from it that were used in the *Preliminary Study on Gestural Intuitiveness* are included in the additional DVD.

Appendix I - Video Clips from Gesture Intuitiveness Test

Video clips taken during the Preliminary Study for Gesture Intuitiveness are included on a YouTube channel that can be found at the links below:

[Gesture Test Videos](#)

Appendix J - App User Test Participant Handout

Appendix J consists of the participant handouts given to subjects partaking in both the initial and subsequent App User Tests, described in Chapter 5. The first two pages of the handouts for both tests are identical in content. The third page of each test handout (for users who took the repeat test) is signified accordingly.

Participant Handout (InteractionTest-Short)

Nicholas Arner, MSc Music Technology

University of York Audio Lab, Department of Electronics

Test Goal

The purpose of this test is to determine which types of interactions users prefer when using a multi-touch device to create music.

Purpose/Application

You will be given three different test apps to use, each focusing on separate interaction styles: rotary knobs, sliders, and multi-touch gestures. Each app will consist of a granular synthesizer that you are able to interact with through various interaction styles.

We are asking you to determine which app you prefer. Additionally, your comments regarding the test will be audio recorded for later analysis. When you interact with the third app, the test will be recorded with video.

Instructions

- The tester will present you with an iPad that will have three apps for you to test. You are allowed to spend as much time as you like interacting with each app before moving on to the next one.
- At the end of the test, please fill out the questionnaire on the back of this page.
- Additionally, the tester will make an audio recording of any comments, questions, or criticisms you may have regarding the tested apps.

Statement of Consent

By signing this form, the subject agrees to have their comments audio recorded at the conclusion of the test, as well as to have their interaction with the third test app video recorded. The audio recordings will be kept anonymous, and will only be used for the author's research purposes. Additionally, the video recordings will only be of the subject's hands, and will also be kept anonymous. These videos will be used for the author's research purposes, and will potentially be publically presented as part of this research. Both the audio and video recordings can be deleted upon the subject's request.

Name _____

Signature _____

Date _____

What is your age? _____

Please state your gender _____

What is your nationality? _____

Are you a university student (Yes/No)? _____

If yes, what is your year of study? _____

Are you a member of the Academic staff? _____

What Department are you based in? _____

Do you own an iOS device (Yes/No)? _____

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? _____

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

Which app did you prefer? _____

Why? _____

Which app did you like the least? _____

Why? _____

Subsequent Tests

Which app did you prefer? _____

Why? _____

Which app did you like the least? _____

Why? _____

Did your opinions of the apps change since the last time? _____

If yes, why do you think that is so? _____

Appendix K - App User Test Questionnaires

Appendix K consists of subjects' responses after they completed the App User Test(s). Completed questionnaires from both sets of tests are included. PDF copies of the completed questionnaires can be found on the additional DVD.

What is your age? 24

Please state your gender MALE

What is your nationality? BRITISH.

Are you a university student (Yes/~~No~~)? _____

If yes, what is your year of study? 1st (PLD)

Are you a member of the Academic staff? No.

What Department are you based in? DEPT. OF ELECTRONICS (AUDIO LAB)

Do you own an iOS device (Yes/~~No~~)? _____

If no, have you ever used an iOS device? YES.

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

YES, MSc & BENG IN MUSIC TECH
PREVIOUS PERFORMANCE EXPERIENCE (GUITAR
GRADE 8, VIOLIN GRADE 7 (ABRSM))

If you are a musician, have you ever used an iOS device to make music

(Yes/~~No~~)? _____

If yes, please describe the type of music and the apps that you use below.

N/A.

What music do you enjoy listening to?

METAL OF THE POWER VARIETY.

Which app did you prefer? THE 3RD APP.

Why? FAR MORE USER-FRIENDLY, INTUITIVE & INTERESTING.

Which app did you like the least? RADIAL DIALS.

Why? _____

P.T.O
P.T.O

3RD APP WAS THE BEST, WHY:

- 1) I AM supportive of the idea of controlling more with less. This is an idea that is found in most performance instruments. So adjusting many parameters with two touches worked for me.
- 2) In general, this app. encouraged creativity → there was no suggestion of anything being "turned up/down". Instead the user could explore the "same terrain" given on the user interface.
- 3) Interesting sounds were accomplished more easily → interesting point, maybe!!!
- 4) Was just generally more fun! It was the most captivating

1ST APP WAS THE WORST, WHY:

- 1) SENSITIVITY of radial dials was too high.
- 2) Controls were difficult to operate with multiple touches.
- 3) Not a good environment for creativity.
- 4) Harder to control or regain a ^{specific} sound.
particular.

What is your age? 28

Please state your gender f

What is your nationality? Green

Are you a university student (Yes/No)? Yes

If yes, what is your year of study? 4th PhD

Are you a member of the Academic staff? No

What Department are you based in? Electronics

Do you own an iOS device (Yes/No)? No

If no, have you ever used an iOS device? (10') maybe 20'

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Yes, Musicologist Pianist
Music Teacher, MSc and (maybe PhD)
In Music Tech

If you are a musician, have you ever used an iOS device to make music (Yes/No)? No

If yes, please describe the type of music and the apps that you use below.

No

What music do you enjoy listening to?

Everything - Classic + Classic Rock
more

Which app did you prefer? 3rd > 1st > 2nd.

Why? not thinking how to use the "knobs" ①

Which app did you like the least? 2nd

Why? ~~wasn't~~ wasn't sure about the use of ~~knobs~~ faders at the moment I was feeling confident that I know what I'm doing, ~~the~~ everything was changing, and I had no idea what was going on.

① more interactive, I did whatever I felt to..

What is your age? 24

Please state your gender female

What is your nationality? USA

Are you a university student (Yes / No)? postgraduate

If yes, what is your year of study? MA single year

Are you a member of the Academic staff? No.

What Department are you based in? English + Related Lit.

Do you own an iOS device (Yes / No)? iPhone

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Yes, ^{10 yrs.} voice / ^{~13 yrs. ~1 yr.} piano / guitar / ^{1 yrs.} flute + 1 yr of Uni. level music study

If you are a musician, have you ever used an iOS device to make music

(Yes / No)? _____

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

Anything... but usually stuff I can sing...

Which app did you prefer? #2

Why? easier to interact with + least annoying

Which app did you like the least? #3

Why? seemed unresponsive + could not tell what was going on

What is your age? 31

Please state your gender female

What is your nationality? British

Are you a university student (~~Yes~~/No)? _____

If yes, what is your year of study? _____

Are you a member of the Academic staff? yes

What Department are you based in? Electronics

Do you own an iOS device (Yes/~~No~~)? _____

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Yes. Singer BA, MA PhD Music & Music Technology.

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? _____

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

Opera, Choral orchestral, Folkrock
Acoustic some pop, rock... 1950s
NOT RAP.

Which app did you prefer? 2 & 3

Why? 3 - Not fun to creative but less control over specific parameters than 2.

Which app did you like the least? 4

Why? Couldn't control the movement of the buttons - very frustrating

What is your age? 21

Please state your gender MALE

What is your nationality? BRITISH

Are you a university student (Yes/No)? _____

If yes, what is your year of study? JUST FINISHED 3rd YEAR

Are you a member of the Academic staff? No

What Department are you based in? Elec.

Do you own an iOS device (Yes/No)? _____

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

MUSTECH STUDENT

SELF TAUGHT GUITARIST

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? _____

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

ROCK, JAZZ, CLASSICAL

Which app did you prefer? 3rd ONE

Why? MORE INTERESTING AND SATISFYING THAN THE OTHER TWO - WORKING OUT THE PARAMETRIC MAPPING IS ~~BETTER~~ BETTER THAN THE MORE OBVIOUS

Which app did you like the least? SLIDERS WHEN DEALING WITH 'RANDOM' SYNTHNOISE

Why? ROTARY KNOBS

HAD TO SLIDE FINGER UP & DOWN TO MOVE CIRCULAR KNOBS SO ~~NOT~~ NOT QUITE INTUITIVE ENOUGH.

What is your age? 28
Please state your gender Female
What is your nationality? Taiwanese
Are you a university student (Yes/No)? Yes
If yes, what is your year of study? 2 (PhD)
Are you a member of the Academic staff? Yes? (0 hr. contract)
What Department are you based in? Women's Studies
Do you own an iOS device (Yes/No)? No
If no, have you ever used an iOS device? Yes
Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Not really.

If you are a musician, have you ever used an iOS device to make music (Yes/No)? _____

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?
Metal. Classic. Jazz. Rock.

Which app did you prefer? 2nd - more straightforward, so more effective,
Why? 3rd - more fun, I get to explore different possibilities.
Which app did you like the least? 1st.
Why? Not sure what I was doing, but the design seems to be suggesting I should...

What is your age? 27

Please state your gender Female

What is your nationality? Jordanian

Are you a university student (Yes/No)? Yes

If yes, what is your year of study? Postgraduate MA

Are you a member of the Academic staff? No

What Department are you based in? English

Do you own an iOS device (Yes/No)? Yes

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

No

If you are a musician, have you ever used an iOS device to make music (Yes/No)? _____

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

Indie, Alternative, House, Trance

Which app did you prefer? TEST 2 + TEST 3

Why? Easier to interact with and move my fingers.

Which app did you like the least? TEST 1

Why? Could not understand how to move the pointers.

What is your age? 33

Please state your gender M

What is your nationality? BRITISH

Are you a university student (Yes/No)? NO

If yes, what is your year of study? _____

Are you a member of the Academic staff? YES

What Department are you based in? AUDIO LAB

Do you own an iOS device (Yes/No)? YES

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

YES - PIANO + CLARINET LESSONS TO GRADE 8
MUSIC TECHNOLOGY AT UNIVERSITY
PHD IN ACOUSTICS

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? YES

If yes, please describe the type of music and the apps that you use below.

SYNTHS, PREFERABLY WITH (FUN + SIMPLE INTERFACE)
+ DRUM MACHINES.

What music do you enjoy listening to?

CLASSICAL, ELECTRONIC DANCE MUSIC, SOFT
ROCK AND HIP HOP

Which app did you prefer? 3RD APP (GREEN + RED)

Why? WAS FELT MORE INTUITIVE, FELT LIKE I WAS USING THE WHOLE SCREEN

Which app did you like the least? SLIDER

Why? FELT A BIT BORING, NOT TAKING ADVANTAGE OF THE HARDWARE

LACK OF VISUAL
FEED BACK MADE IT MORE
FUN SOMEHOW!

What is your age? 27

Please state your gender MALE

What is your nationality? DUTCH

Are you a university student (Yes/No)? YES

If yes, what is your year of study? 2nd YEAR

Are you a member of the Academic staff? NO

What Department are you based in? ELECTRONICS

Do you own an iOS device (Yes/No)? NO

If no, have you ever used an iOS device? YES

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

USED TO PLAY THE CELLO IN HIGH SCHOOL
(PRE-UNIVERSITY DAYS)

• If you are a musician, have you ever used an iOS device to make music

(Yes/No)? NO

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

ALL GENRES, IT IS HIGHLY DEPENDENT
ON MY MOOD AT THE TIME

Which app did you prefer? SLIDER

Why? EASY TO USE, KNEW EXACTLY WHAT I WAS DOING.

Which app did you like the least? MULTI-TOUCH (COLOUR-INTERFACE)

Why? DIFFICULT TO DETERMINE WHAT I
WAS DOING WHEN PRODUCING MUSIC

What is your age? 28

Please state your gender female

What is your nationality? British

Are you a university student (Yes/No)? Yes

If yes, what is your year of study? 1st Year PhD

Are you a member of the Academic staff? No

What Department are you based in? Centre for Womens Studies

Do you own an iOS device (Yes/No)? No

If no, have you ever used an iOS device? No

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

No / Been in a choir but no experience of production

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? No

If yes, please describe the type of music and the apps that you use below.

N/A

What music do you enjoy listening to?

Wide variety - alternative, rock, pop, Motown/60s, classical, etc...

Which app did you prefer? The last app (colours)

Why? More fun, intuitive.

Which app did you like the least? The first one (dials)

Why? Awkward to use, hard to remember what did what.

What is your age? 19

Please state your gender Male

What is your nationality? British

Are you a university student (Yes/No)? Yes

If yes, what is your year of study? First

Are you a member of the Academic staff? No

What Department are you based in? Maths

Do you own an iOS device (Yes/No)? Yes

If no, have you ever used an iOS device? Yes

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

I have played guitar and used various music generating computer programmes.

If you are a musician, have you ever used an iOS device to make music (Yes/No)? Yes

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?
Mostly Rock and metal

Which app did you prefer? 2nd
Why? Easier to control and understand

Which app did you like the least? 1st
Why? Difficult to understand initially.

What is your age? 22

Please state your gender Female

What is your nationality? British

Are you a university student (Yes/No)? _____

If yes, what is your year of study? master's

Are you a member of the Academic staff? No

What Department are you based in? N/A

Do you own an iOS device (Yes/No)? ipod touch

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

yes, singing in musicals, grade 4 piano,
grade 1 trombone.

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? _____

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

Finnish music & heavy metal

Which app did you prefer? App 2 - sliders

Why? easier to identify which slider and what

Which app did you like the least? App 1 - circles

Why? hard to manipulate the control controls.

What is your age? 23
Please state your gender female
What is your nationality? American
Are you a university student (Yes/No)? yes
If yes, what is your year of study? masters
Are you a member of the Academic staff? Nope
What Department are you based in? Center for Medieval Studies
Do you own an iOS device (Yes/No)? yes
If no, have you ever used an iOS device? _____
Do you have any sort of prior audio or music background? If yes, please briefly explain below.

No

If you are a musician, have you ever used an iOS device to make music (Yes/No)? _____
If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?
all kinds - Gregorian chant, Bagpipes, Classical, jazz, post-hardcore, acid jazz, EDM, Rock and roll, pop
Which app did you prefer? the slider one
Why? easier to visualize the amount of sound
Which app did you like the least? red + green
Why? felt hard to control.

What is your age? 27

Please state your gender M

What is your nationality? CHINESE

Are you a university student (Yes/No)? Y

If yes, what is your year of study? 2nd PHD

Are you a member of the Academic staff? N

What Department are you based in? Elec.

Do you own an iOS device (Yes/No)? Y

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Yes. I play guitar and compose electronic music. I have used various music production software and hardware. Also, I ~~use~~ play with some musical apps on my iPad mini occasionally.

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? No

If yes, please describe the type of music and the apps that you use below.

Synthesizer apps.

What music do you enjoy listening to?

Ambient. Electronics.

Which app did you prefer? 3rd.

Why? 3D. Less likely to mistouch.

Which app did you like the least? 1st.

Why? ~~It~~ They look like knobs but don't act like knobs.

What is your age? 27

Please state your gender male

What is your nationality? Chinese

Are you a university student (Yes/No)? Yes

If yes, what is your year of study? Second PHd.

Are you a member of the Academic staff? No

What Department are you based in? Electronics

Do you own an iOS device (Yes/No)? Yes

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

NO

If you are a musician, have you ever used an iOS device to make music (Yes/No)? _____

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?
pop . Country .

Which app did you prefer? sliders one

Why? easy to control, the change can be seen ~~extremely~~

Which app did you like the least? nutti - touch.

Why? need more time to get use to it. Difficult to control.

What is your age? 22

Please state your gender Male

What is your nationality? British

Are you a university student (Yes/No)? Yes

If yes, what is your year of study? MSc

Are you a member of the Academic staff? No

What Department are you based in? Electronics

Do you own an iOS device (Yes/No)? Yes

If no, have you ever used an iOS device? /

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Guitarist and music producer (hobbyist)

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? Yes

If yes, please describe the type of music and the apps that you use below.

MorphWiz, Figure, Garageband.

What music do you enjoy listening to?

Progressive Rock, Jazz fusion, Metal, Electronic, 'modern', etc.

Which app did you prefer? 2

Why? Visual feedback - No remembering the parameter's levels.

Which app did you like the least? 3

Why? Easy to forget what 'pitch', 'pitch offset' etc means when there's > 1 dimension of spatial freedom.

What is your age? 22

Please state your gender Female

What is your nationality? Russian

Are you a university student (Yes/No)? Yes

If yes, what is your year of study? MSc

Are you a member of the Academic staff? No

What Department are you based in? Computer Science

Do you own an iOS device (Yes/No)? Yes

If no, have you ever used an iOS device? N/A

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

No

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? No

If yes, please describe the type of music and the apps that you use below.

N/A

What music do you enjoy listening to?

Not pop music

Which app did you prefer? Second

Why? It has clear and obvious way of changing the parameters

Which app did you like the least? Third

Why? Same reason - no way of knowing what you're controlling.

What is your age? 25.

Please state your gender Female.

What is your nationality? British.

Are you a university student (Yes/No)? _____

If yes, what is your year of study? 1st Year Ph.D.

Are you a member of the Academic staff? No.

What Department are you based in? Linguistics.

Do you own an iOS device (Yes/No)? _____

If no, have you ever used an iOS device? _____

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

Music A-level
Violin, Piano player. -Grade 8/7.

If you are a musician, have you ever used an iOS device to make music

(Yes/No) - _____

If yes, please describe the type of music and the apps that you use below.

Magic Piano.

What music do you enjoy listening to?

All sorts! Pop, Indie, mainly.

Which app did you prefer? 1st ~~1~~

Why? Multiple things to control, could try & work out what each part was doing

Which app did you like the least? 2nd

Why? Bit boring - felt just like a volume control.

What is your age? Ask Damian

Please state your gender Female

What is your nationality? British

Are you a university student (Yes/No) No

If yes, what is your year of study? —

Are you a member of the Academic staff? Yes

What Department are you based in? Electronics

Do you own an iOS device (Yes/No)? no

If no, have you ever used an iOS device? no

Do you have any sort of prior audio or music background? If yes, please briefly explain below.

musical training in classical and choral
singing
work in audio + music technology

If you are a musician, have you ever used an iOS device to make music

(Yes/No)? No

If yes, please describe the type of music and the apps that you use below.

What music do you enjoy listening to?

1980s trash,
anything written before 1750
choral music

Which app did you prefer? 3

Why? it was more fun

Which app did you like the least? 1

Why? I can't control twiddly knobs

Subsequent Tests

Which app did you prefer? TEST-3.

Why? MORE INTERACTION WITH ~~GOOD~~ SOUND
BECAUSE OF SYNCHRONY BETWEEN IT AND
FINGER MOVEMENT

Which app did you like the least? TEST-1.

Why? DIFFICULTY IN ALTERING AND UNDERSTANDING
POINTERS' MOVEMENTS.

Did your opinions of the apps change since the last time? NO

If yes, why do you think that is so? _____

Subsequent Tests

Which app did you prefer? THE 3RD APP.

Why? More intuitive -> easier to be creative with. Makes for a more user-friendly and interesting performance experience.

Which app did you like the least? THE 1ST APP.

Why? Difficult to control, least intuitive choice. Just wasn't exciting!

Did your opinions of the apps change since the last time? No.

If yes, why do you think that is so? N/A.

Subsequent Tests

Which app did you prefer? 1st

Why? Once I understood how to move the knobs consistently it was easier because it was just one motion to consider.

Which app did you like the least? 3rd

Why? couldn't get the same range of tones + had trouble pinpointing effective movements to change the sound...

Did your opinions of the apps change since the last time? yes

If yes, why do you think that is so? I'm more awake + found them more understandable this time around.

I still don't get how the 3rd works - ~~the~~ I couldn't get into a lower tone register and that was frustrating...

Subsequent Tests

Which app did you prefer? Sliders

Why? I felt in control and had a better visual of what sounds I could make with different combinations.

Which app did you like the least? Knobs

Why? felt hard to stay in control / get them to rest where I wanted them to.

Did your opinions of the apps change since the last time? yes and NO

If yes, why do you think that is so? I still preferred the slider app, but I appreciated the red/green app more this time. I still don't like the knobs

Subsequent Tests

Which app did you prefer? 2nd one.

Why? I think it's because I "feel" that I knew what's going on and can make the sounds to... more of what I like. (The third is still fun, experimental

Which app did you like the least? 1st one. ^{but} more indirect).

Why? No idea of what is going on... ish... Compare to the first time I did this experiment I dislike it less, but I'm still not sure what I can make out of it.

Did your opinions of the apps change since the last time? Not much.

If yes, why do you think that is so? _____

I think doing the research again gives me more idea about what's happening, so my feeling is not as strong as last time, and I go for a more pragmatic approach to ~~see what~~ focus more on what I can achieve with the apps, rather than just liking them or not!

Subsequent Tests

Which app did you prefer?

Why?

Third, still it's more interactive. I don't have to think about what ~~for~~ each knob/fader is doing.

Which app did you like the least?

Why?

Second. I ~~can't~~ did understand this time what each fader was doing, because this time I did the connection in my head, each knob → 1

Did your opinions of the apps change since the last time?

If yes, why do you think that is so?

No BUT 2
② it I feel that it depends on the purpose of application of the apps. If I wanted to be creative (as how), I would choose the third one, but if I wanted to ~~more~~ achieve a specific sound effect, I would choose the second ③

I had the correlated fader. But the faders were more "boring" for me to play with.

③ However, the first one could combine both purposes. (creative + specific sound effect)

Subsequent Tests

Which app did you prefer? slider one

Why? I produced some music like ~~some~~ saved finally.

Which app did you like the least? Rotary knobs.

Why? It is hard to use multiple fingers.

Did your opinions of the apps change since the last time? Yes

If yes, why do you think that is so? I spend more time on practice, so

~~to~~ maybe I will know how to use it if I have more

time.

Subsequent Tests

Which app did you prefer? 3rd.

Why? Higher success rate in touching. Extra interaction of pointing rather than moving.

Big tangible area for control. 2-D instead of 1-D.

Which app did you like the least? 1st.

Why? Look like a ~~pen~~ but don't act like one.

Did your opinions of the apps change since the last time? Yes.

If yes, why do you think that is so? _____

Mainly for the third app. I discovered that I can pointer at the pad to make an instant parameter change.

Subsequent Tests

Which app did you prefer? APP 3

Why? SEEMED MORE FUN TO PLAY WITH THAN THE OTHERS. INVITED ME TO USE BOTH HANDS. MADE FULL USE OF THE IPAD SCREEN AREA

CAN JUMP FROM ONE PARAMETER VALUE TO ANOTHER

Which app did you like the least? APP 1 + 2 EQUALLY

Why? CAN ONLY SLIDE FROM ONE PARAMETER VALUE TO ANOTHER. A LOT OF SCREEN WAS WASTED. DIDN'T SEEM SUITABLE FOR THIS TYPE OF SOUND SYNTHESIS

Did your opinions of the apps change since the last time? NO

If yes, why do you think that is so? _____

Subsequent Tests

Which app did you prefer? Second for finer tuning, third for fun
Why? See below.

Which app did you like the least? First

Why? The graphics for the knobs are counter-intuitive. They don't work with the movements you'd expect. Also bad that they're not side-by-side

Did your opinions of the apps change since the last time? No - strengthened opinion
If yes, why do you think that is so? _____

With the third, it's more tactile - it's like playing with things in infant's school. Novel and something you don't normally do. Not sure how long I'd keep liking it once the novelty had faded but I do like the tactile approach.

The second is easier than the first not just because of the awkward knobs, but because it's easier when you can see the levels side-by-side.

Appendix L - App User Test Spreadsheets

Excel spreadsheets used for data analysis in the App User Tests, discussed in Chapter 5, are included on the additional DVD.

Appendix M - Video Clips from App User Tests

Video clips taken during the App User Tests can be found on a YouTube channel, located at the following link:

[Test1](#)

[Test2](#)

Appendix N - Subject Audio Clips

Audio recordings of subjects' responses after taking the App User Tests are included on the additional DVD.

Appendix O - App User Test Xcode Projects

The Xcode projects used to create the three apps used in the App User Tests are included on the additional DVD. The following screen-captures are of the *.h* and *.m* files of the respective Test Projects.

Pages 260-264 contains the *.h* and *.m* files for the Rotary Knobs Test App.

Pages 265-269 contains the *.h* and *.m* files for the Faders Test App.

Pages 270-274 contains the *.h* and *.m* files for the Touches Test App.

```
//
// ViewController.m
// Test1-RotaryKnobs
//
// Nicholas Arner
// MSc by Research-Music Technology
// University of York Department of Electronics
// Audio Lab
// 2013

#import <UIKit/UIKit.h>
#import "TABKnob.h"
#import "BaseCsoundViewController.h"
#import "CsoundObj.h"
#import "CsoundValueCacheable.h"

@interface ViewController : BaseCsoundViewController
<CsoundObjCompletionListener, CsoundValueCacheable>
{
    TABKnob *knob1;
    TABKnob *knob2;
    TABKnob *knob3;
    TABKnob *knob4;
}

//Switch for rendering on/off
- (IBAction)toggleOnOff:(UISwitch *)sender;

@property (nonatomic, retain) TABKnob *knob;

- (void)knobTwist:(TABKnob*)sender;

//Declare a Csound object
@property (nonatomic, retain) CsoundObj* csound;

@end
```



```

//
// ViewController.m
// Test1-RotaryKnobs
//
// Nicholas Arner
// MSc by Research-Music Technology
// University of York Department of Electronics
// Audio Lab
// 2013

#import "ViewController.h"
@interface ViewController ()
@end
@implementation ViewController

//Initialise control variables
float grainPitchValue = 550.0;
float* grainPitchPtr;

float pitchOffsetValue = 500.0;
float* pitchOffsetPtr;

float grainDensityValue = 50.0;
float* grainDensityPtr;

float grainDurationValue = 0.525;
float* grainDurationPtr;

- (void)viewDidLoad {

    //Define screen parameters
    int height = 1024;
    int width = 768;
    int size = 200;
    int halfSize = size / 2;

    //Load image for all knobs
    UIImage *knobImage = [UIImage imageNamed:@"greenknob.png"];

    // Create the knob object and give it an image
    knob1 = [[TABKnob alloc] initWithFrame:CGRectMake(width / 3 - halfSize,
        height / 3 - halfSize, size, size)];
    [knob1 setThumbImage:knobImage forState:UIControlStateNormal];
    // Assign a target and action.
    [knob1 addTarget:self action:@selector(knobTwist:) forControlEvents:
        UIControlEventValueChanged];
    // Place in a view
    [self.view addSubview:knob1];
    //Set values of knob1
    [knob1 setMinimumValue:100.0];
    [knob1 setMaximumValue:1000.0];
    [knob1 setValue:550.0 animated:NO];
    [knob1 setPrecision:25.0];
}

```

```

//Knob2
knob2 = [[TABKnob alloc] initWithFrame:CGRectMake(2 * width / 3 -
    halfSize, height / 3 - halfSize, size, size)];
[knob2 setThumbImage:knobImage forState:UIControlStateNormal];
[knob2 addTarget:self action:@selector(knobTwist:) forControlEvents:
    UIControlEventValueChanged];
[self.view addSubview:knob2];
//Set values of knob2
[knob2 setMinimumValue:0.0];
[knob2 setMaximumValue:1000.0];
[knob2 setValue:500.0 animated:NO];
[knob2 setPrecision:28.0];

//Knob3
knob3 = [[TABKnob alloc] initWithFrame:CGRectMake(width / 3 - halfSize,
    2 * height / 3 - halfSize, size, size)];
[knob3 setThumbImage:knobImage forState:UIControlStateNormal];
[knob3 addTarget:self action:@selector(knobTwist:) forControlEvents:
    UIControlEventValueChanged];
[self.view addSubview:knob3];
//Set values of knob3
[knob3 setMinimumValue:1.0];
[knob3 setMaximumValue:100.0];
[knob3 setValue:50.0 animated:NO];
[knob3 setPrecision:2.5];

//Knob4
knob4 = [[TABKnob alloc] initWithFrame:CGRectMake(2 * width / 3 -
    halfSize, 2 * height / 3 - halfSize, size, size)];
[knob4 setThumbImage:knobImage forState:UIControlStateNormal];
[knob4 addTarget:self action:@selector(knobTwist:) forControlEvents:
    UIControlEventValueChanged];
[self.view addSubview:knob4];
//Set values of knob4
[knob4 setMinimumValue:0.05];
[knob4 setMaximumValue:1.0];
[knob4 setValue:0.525 animated:NO];
[knob4 setPrecision:0.025];
}

//Send values of knobs to respective Csound variables
-(void)knobTwist:(TABKnob*)sender
{
    if (sender == knob1)
        grainPitchValue = sender.value;
    else if (sender == knob2)
        pitchOffsetValue = sender.value;
    else if (sender == knob3)
        grainDensityValue = sender.value;
    else if (sender == knob4)
        grainDurationValue = sender.value;
}

-(void)viewDidAppear:(BOOL)animated
{

```

```

    [super viewDidLoadAppear:animated];
}

//Start rendering if Switch turned on
-(IBAction)toggleOnOff:(UISwitch *)sender
{
    if (sender.on){
        //Locate .csd and assign create a string with its file path
        NSString *tempFile = [[NSBundle mainBundle] pathForResource:@"1306-
        KNOBS" ofType:@"csd"];

        // Print the file path and name
        NSLog(@"File opened: %@", tempFile);

        //Allocate some memory
        self.csound = [[CsoundObj alloc] init];

        //Tell Csound to add a completion listener
        [self.csound addCompletionListener:self];

        //Tell Csound to add value cacheables for talking to iOS
        [self.csound addValueCacheable:self];

        //Tell csound to start the file
        [self.csound startCsound:tempFile];
    } else {
        [self.csound stopCsound];
    }
}

#pragma mark CsoundObjCompletionListener

-(void)csoundObjDidStart:(CsoundObj *)csoundObj
{
}

-(void)csoundObjComplete:(CsoundObj *)csoundObj
{
}

#pragma mark ValueCacheable

//Sets up communication with Csound
-(void)setup:(CsoundObj*) csoundObj
{
    NSString *grainPitchString = @"grainPitchVal";
    grainPitchPtr = [csoundObj getInputChannelPtr:grainPitchString];

    NSString *pitchOffsetString = @"pitchOffsetVal";
    pitchOffsetPtr = [csoundObj getInputChannelPtr:pitchOffsetString];

    NSString *grainDensityString = @"grainDensityVal";

```

```

    grainDensityPtr = [csoundObj getInputChannelPtr:grainDensityString];

    NSString *grainDurationString = @"grainDurationVal";
    grainDurationPtr = [csoundObj getInputChannelPtr:grainDurationString];
}

//Communicates values to Csound
-(void)updateValuesToCsound
{
    *grainPitchPtr = grainPitchValue;
    *pitchOffsetPtr = pitchOffsetValue;
    *grainDensityPtr = grainDensityValue;
    *grainDurationPtr = grainDurationValue;
}

-(void)updateValuesFromCsound
{
    //No values coming from Csound to iOS
}

-(void)cleanup
{
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
}

- (void)dealloc
{
    [knob1 release];
    [knob2 release];
    [knob3 release];
    [knob4 release];
    [super dealloc];
}

@end

```

```
//
// ViewController.h
// Test2-Faders
//
// Nicholas Arner
// MSc by Research-Music Technology
// University of York Department of Electronics
// Audio Lab
// 2013

#import <UIKit/UIKit.h>
#import "BaseCsoundViewController.h"
#import "CsoundObj.h"
#import "CsoundValueCacheable.h"

#define degreesToRadians(x) (M_PI * x / 180.0)

@interface ViewController : BaseCsoundViewController
<CsoundObjCompletionListener, CsoundValueCacheable>{
}

//Declare a Csound object
@property (nonatomic, retain) CsoundObj* csound;

//Switch for rendering on/off
- (IBAction)toggleOnOff:(UISwitch *)sender;

//For interaction with sliders
- (IBAction)grainPitchSlider:(id)sender;
- (IBAction)pitchOffsetSlider:(id)sender;
- (IBAction)grainDensitySlider:(id)sender;
- (IBAction)grainDurationSlider:(id)sender;

@property (retain, nonatomic) IBOutlet UISlider *grainPitchSlider;
@property (retain, nonatomic) IBOutlet UISlider *grainOffsetSlider;
@property (retain, nonatomic) IBOutlet UISlider *grainDensitySlider;
@property (retain, nonatomic) IBOutlet UISlider *grainDurationSlider;

@end
```

```

//
// ViewController.m
// Test2.1-Faders
//
// Nicholas Arner
// MSc by Research-Music Technology
// University of York Department of Electronics
// Audio Lab
// 2013

#import "ViewController.h"
@interface ViewController ()
@end
@implementation ViewController;

//Initialize control variables
float grainPitchValue = 550.0;
float* grainPitchPtr;

float pitchOffsetValue = 500.0;
float* pitchOffsetPtr;

float grainDensityValue = 50;
float* grainDensityPtr;

float grainDurationValue = 0.525;
float* grainDurationPtr;

- (void)viewDidLoad
{
//Define screen parameters
int height = 1024;
int width = 768;
int halfSize = 80;

//Rotate sliders and set their positions
CGAffineTransform trans1 = CGAffineTransformMakeRotation(M_PI * 1.5);
_grainPitchSlider.transform = trans1;
_grainPitchSlider.center = CGPointMake(height/5, width/2+halfSize);

CGAffineTransform trans2 = CGAffineTransformMakeRotation(M_PI * 1.5);
_grainOffsetSlider.transform = trans2;
_grainOffsetSlider.center = CGPointMake(height/5*2, width/2+halfSize);

CGAffineTransform trans3 = CGAffineTransformMakeRotation(M_PI * 1.5);
_grainDensitySlider.transform = trans3;
_grainDensitySlider.center = CGPointMake(height/5*3, width/2+halfSize);

CGAffineTransform trans4 = CGAffineTransformMakeRotation(M_PI * 1.5);

```

```

    _grainDurationSlider.transform = CGAffineTransformMakeScale(1, 1);
    _grainDurationSlider.center = CGPointMake(height/5*4, width/2+halfSize);

    [super viewDidLoad];
    //Set thumb image for all sliders
    [[UISlider appearance] setThumbImage:[UIImage imageNamed:@"custom-
        slider-handle.png"] forState:UIControlStateNormal];
}

-(void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
}

//Start rendering if Switch turned on
-(IBAction)toggleOnOff:(UISwitch *)sender
{
    if (sender.on){
        //Locate .csd and assign create a string with its file path
        NSString *tempFile = [[NSBundle mainBundle] pathForResource:@"1306-
            SLIDERS" ofType:@"csd"];

        // Print the file path and name
        NSLog(@"File opened: %@", tempFile);

        //Allocate some memory
        self.csound = [[CsoundObj alloc] init];

        //Set up the Csound engine
        [self.csound addCompletionListener:self];
        [self.csound addValueCacheable:self];
        [self.csound startCsound:tempFile];
    }
    else
    {
        [self.csound stopCsound];
    }
}

//make control variables value of faders
-(IBAction)grainPitchSlider:(id)sender
{
    UISlider *grainPitchSlider = (UISlider *)sender;
    grainPitchValue = grainPitchSlider.value;
}

-(IBAction)pitchOffsetSlider:(id)sender
{
    UISlider *pitchOffsetSlider = (UISlider *)sender;
    pitchOffsetValue = pitchOffsetSlider.value;
}

```

```

- (IBAction)grainDensitySlider:(id)sender
{
    UISlider *grainDensitySlider = (UISlider *)sender;
    grainDensityValue = grainDensitySlider.value;
}

- (IBAction)grainDurationSlider:(id)sender
{
    UISlider *grainDurationSlider = (UISlider *)sender;
    grainDurationValue = grainDurationSlider.value;
}

#pragma mark CsoundObjCompletionListener

-(void)csoundObjDidStart:(CsoundObj *)csoundObj
{
}

-(void)csoundObjComplete:(CsoundObj *)csoundObj
{
}

#pragma mark ValueCacheable

//brief set up communication with Csound

-(void)setup:(CsoundObj* )csoundObj
{
    NSString *grainPitchString = @"grainPitchVal";
    grainPitchPtr = [csoundObj getInputChannelPtr:grainPitchString];

    NSString *pitchOffsetString = @"pitchOffsetVal";
    pitchOffsetPtr = [csoundObj getInputChannelPtr:pitchOffsetString];

    NSString *grainDensityString = @"grainDensityVal";
    grainDensityPtr = [csoundObj getInputChannelPtr:grainDensityString];

    NSString *grainDurationString = @"grainDurationVal";
    grainDurationPtr = [csoundObj getInputChannelPtr:grainDurationString];
}

//Communicates values to Csound
-(void)updateValuesToCsound
{
    *grainPitchPtr = grainPitchValue;
    *pitchOffsetPtr = pitchOffsetValue;
    *grainDensityPtr = grainDensityValue;
    *grainDurationPtr = grainDurationValue;
}

```



```
    }

    -(void)updateValuesFromCsound
    {
        //No values coming from Csound to iOS
    }

    -(void)cleanup
    {
    }

    -(void)didReceiveMemoryWarning
    {
        [super didReceiveMemoryWarning];
    }

    -(void)dealloc
    {
        [_grainPitchSlider release];
        [_grainOffsetSlider release];
        [_grainDensitySlider release];
        [_grainDurationSlider release];

        [super dealloc];
    }

@end
```

```
//
// ViewController.h
// Touches
//
// Nicholas Arner
// MSc by Research-Music Technology
// University of York Department of Electronics
// Audio Lab
// 2013

#import <UIKit/UIKit.h>
#import "OneTouch.h"
#import "BaseCsoundViewController.h"
#import "CsoundObj.h"
#import "CsoundValueCacheable.h"

@interface ViewController : BaseCsoundViewController <OneTouchProtocol,
CsoundObjCompletionListener, CsoundValueCacheable>
{
    OneTouch *_firstTouch;
    OneTouch *_secondTouch;

    Float32 pitchValue;
    Float32 offsetValue;
    Float32 densityValue;
    Float32 durationValue;
}

-(id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)
nibNameOrNil;

//Call TouchOne class delegates
-(void)sendValueYFromOneTouch:(Float32)valueY
    object:(UInt16)objectID;
-(void)sendValueXFromOneTouch:(Float32)valueX
    object:(UInt16)objectID;

@end
```

```

//
// ViewController.m
// Touches
//
// Nicholas Arner
// MSc by Research-Music Technology
// University of York Department of Electronics
// Audio Lab
// 2013

#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

//Declare pointers for communication with Csound
float* grainPitchPtr;
float* pitchOffsetPtr;
float* grainDensityPtr;
float* grainDurationPtr;

-(id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)
nibNameOrNil{

self=[super initWithNibName:nibNameOrNil bundle:nibNameOrNil];

if(self)
{
//Set size and location of touch areas on screen
CGRect frame1=CGRectMake(0, 0, 512, 768);
CGRect frame2=CGRectMake(512, 0, 512, 768);

//Assign separate touches to respective screen areas
_firstTouch=[[OneTouch alloc] initWithFrame:frame1
withColor:[UIColor redColor]
delegate:self
uniqueID:1];

_secondTouch=[[OneTouch alloc] initWithFrame:frame2
withColor:[UIColor greenColor]
delegate:self
uniqueID:2];

//Add touches to view
[self.view addSubview:_firstTouch];
[self.view addSubview:_secondTouch];

#pragma Csound set-up

//Locate .csd and assign create a string with its file path

```

```

NSString *tempFile = [[NSBundle mainBundle] pathForResource:@"1300-
MULTITOUCH" ofType:@"csd"];

// Print the file path and name
NSLog(@"File opened: %@", tempFile);

//Stops all previous instances of Csound running.
[self.csound stopCsound];

//Allocate some memory
self.csound = [[CsoundObj alloc] init];

//Tell Csound to add a completion listener
[self.csound addCompletionListener:self];

//Tell Csound to add value cacheables for talking to iOS
[self.csound addValueCacheable:self];

//Mute Csound when app first loads
[self.csound muteCsound];

//Tell csound to start the file
[self.csound startCsound:tempFile];

}
return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
}

//Csound variables assigned depending on touch location
-(void)sendValueYFromOneTouch:(Float32)valueY object:(UInt16)objectID
{
    if (objectID == 1)
    {
        //Unmute Csound when user first touches app
        [self.csound unmuteCsound];

        pitchValue = 100 + valueY * 900;
        //printf("pitchValue is %f", pitchValue);
    }
    else if (objectID == 2)
    {
        densityValue = 1 + valueY * 100;
        //printf("densityValue is %f", densityValue);
    }
}

-(void)sendValueXFromOneTouch:(Float32)valueX object:(UInt16)objectID
{

```

```

    if (objectID == 1)
    {
        offsetValue = valueX * 1000;
        //printf("offsetValue is %f", offsetValue);
    }
    else if (objectID == 2)
    {
        durationValue = 0.05 + valueX * 0.95;
        //printf("durationValue is %f", durationValue);
    }
}

#pragma mark CsoundObjCompletionListener

-(void)csoundObjDidStart:(CsoundObj *)csoundObj
{
}

-(void)csoundObjComplete:(CsoundObj *)csoundObj
{
}

#pragma mark ValueCacheable

//Sets up communication with Csound
-(void)setup:(CsoundObj*) csoundObj
{
    NSString *grainString = @"grainPitchVal";
    grainPitchPtr = [csoundObj getInputChannelPtr:grainString];

    NSString *offsetString = @"pitchOffsetVal";
    pitchOffsetPtr = [csoundObj getInputChannelPtr:offsetString];

    NSString *densityString = @"grainDensityVal";
    grainDensityPtr = [csoundObj getInputChannelPtr:densityString];

    NSString *durationString = @"grainDurationVal";
    grainDurationPtr = [csoundObj getInputChannelPtr:durationString];

    //Set initial parameter values
    pitchValue = 550.0;
    offsetValue = 500.0;
    densityValue = 50.0;
    durationValue = 0.525;
}

//Communicates values to Csound
-(void)updateValuesToCsound
{
    *grainPitchPtr = pitchValue;
    *pitchOffsetPtr = offsetValue;
    *grainDensityPtr = densityValue;
}

```

```
    *grainDurationPtr = durationvalue;
}

-(void)updateValuesFromCsound
{
    //No values coming from Csound to iOS
}

-(void)cleanup
{
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
}

- (void)dealloc
{
}

@end
```

Appendix P – App User Test .csd File

The .csd file used in all three of the test apps created for this thesis is included on the additional DVD. Although the label for the .csd says “1306-KNOBS.csd”, the .csd is exactly the same as the ones used in all three test apps.

Appendix Q - Literature Repository

All available electronic literature that is cited in this thesis is included on the additional DVD. Literature that was read by the author during the research process, but not used in the thesis, is included as well.

Appendix R - Fraunhofer MSc Presentation

Appendix R on the DVD includes a PowerPoint presentation given by the author regarding the work done as part of this MSc thesis while on an internship at Fraunhofer IIS in Erlangen, Germany.

References

Abrams, S., et al. (1999). Higher-level Composition Control in Music Sketcher: Modifiers and Smart Harmony. In: *International Computer Music Conference 1999*. Beijing. [Online]. Available at: <http://openmuse.org/noncpl/Higher-levelCompositionControlinMusicSketcher.pdf>. [Accessed 29 January 2013].

Abrams, S., et al. (2001). Qsketcher: An Environment for Composing Music for Film. In: *International Computer Music Conference 2001*. Havana. 29 January 2013. Available at: <http://openmuse.org/noncpl/QSketcherICMC2001Paper.pdf>. [Accessed 29 January 2013].

Adamson, C. and Avila, K. (2012). *Learning Core Audio*. Upper Saddle River: Addison Wesley.

AffinityBlue. (2013). NodeBeat. [Online]. Available at: <http://nodebeat.com/>. [Accessed 27 January 2014].

Apple (2011a). CP1919. [Online]. Available at: <https://itunes.apple.com/us/app/cp-1919/id422919711?mt=8>. [Accessed 19 November 2012].

Apple (2011b). *Filtatron*. [Online]. Available at: <https://itunes.apple.com/en/app/filtatron/id396776418?mt=8> mt. [Accessed 19 November 2012].

Apple (2012a). *Animoog*. [Online]. Available at: <https://itunes.apple.com/gb/app/animoog-for-iphone/id490169960?mt=8>. [Accessed 19 November 2012].

Apple (2012b). *Figure*. [Online]. Available at: <https://itunes.apple.com/us/app/figure/id511269223?mt=8>. [Accessed 19 November 2012].

Apple (2012c). *GrainProc*. [Online]. Available at: <https://itunes.apple.com/us/app/grainproc/id572380905?mt=8>. [Accessed 19 November 2012].

Apple (2012d). *iKaossilator*. [Online]. Available at: <https://itunes.apple.com/en/app/korg-ikaossilator/id452559831?mt=8>. [Accessed November 2012].

Apple (2012e). *Human Interface Principles*. [Online]. Available at: [http://developer.apple.com/library"/ios/#documentation/userexperience/conceptual/mobilehig/Principles/Principles.html](http://developer.apple.com/library). [Accessed 6 December 2012].

Apple (2012f). *iOS*. [Online]. Available at: <http://www.apple.com/ios/what-is/>. [Accessed 15 October 2012].

Apple (2012g). *iMaschine*. [Online]. Available at: <https://itunes.apple.com/gb/app/imaschine/id400432594?mt=8> [Accessed 19 November 2012].

Apple (2012h). *MegaCurtisFREE*. [Online]. Available at: <https://itunes.apple.com/us/app/megacurtis-free/id317498757?mt=8&ign-mpt=uo%3D4>. [Accessed 19 November 2012].

Apple (2012i). *NodeBeat*. [Online]. Available at: <https://itunes.apple.com/gb/app/nodebeat/id428440804?mt=8>. [Accessed 19 November 2012].

Apple (2012j). *Reactable Mobile*. [Online]. Available at: <https://itunes.apple.com/en/app/reactable-mobile/id381127666?mt=8>. [Accessed 19 November 2012].

Apple (2012k). *SynthStation*. [Online]. Available at: <https://itunes.apple.com/gb/app/synthstation/id373969724?mt=8> [Accessed 19 November 2012].

Apple (2013a). Apple Press Info: Apple's App Store Marks Historic 50 Billionth Download. [Online]. Available at: <http://www.apple.com/pr/library/2013/05/16Apples-App-Store-Marks-Historic-50-Billionth-Download.html>. [Accessed 8 November 2013].

Apple (2013b). *Event Handling Guide for iOS*. [Online]. Available at: http://developer.apple.com/library/ios/#documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/GestureRecognizer_basics/GestureRecognizer_basics.html#//apple_ref/doc/uid/TP40009541-CH2-SW2. [Accessed 8 April 2013].

Apple (2013c). *iPad*. [Online]. Available at: <http://www.apple.com/uk/ipad/specs/>. [Accessed 9 March 2013].

Apple (2013d). *Xcode4* [Online]. Available at: <https://developer.apple.com/xcode/>. [Accessed 6 April 2013].

Atkins-Wakefield, J. (2012). Research into Novel Interfaces for Sonification on the iPad. MEng Thesis. University of York, 2012.

barefoot-coders.com (2012). Dandy. [Online]. Available at: <http://www.barefoot-coders.com/>. [Accessed 9 November 2013].

Bachl, S., et al. (2010). Challenges for Designing the User Experience of Multi-touch Interfaces. In: *Proceedings of Workshop on Engineering Patterns for Multi-Touch Interfaces*. [Online]. Available at: <http://deco.inso.tuwien.ac.at/wp-content/uploads/mt-challenges.pdf>. [Accessed 16 November 2012].

- Beaudouin-Lafon, M. (2004). Designing Interaction, not Interfaces. In *Proceedings of the working conference on Advanced visual interfaces (AVI '04)*. ACM, New York, NY, USA, 15-22. DOI=10.1145/989863.989865 [Online]. Available at: <http://doi.acm.org/10.1145/989863.989865>. [Accessed 3 October, 2013].
- Bencina, R. (2006). "Implementing Real-Time Granular Synthesis," in Greenbaum & Barzel (Eds.), *Audio Anecdotes*. III A.K. Peters, Natick. Draft. [Online]. Available at: <http://www.rossbencina.com/static/code/granular-synthesis/BencinaAudioAnecdotes310801.pdf>. [Accessed 25 February 2013].
- Bergman, E., and Johnson, E. (1997). Towards Accessible Human-Computer Interaction. In Nielson, J. (Ed.). *Advances in Human-Computer Interaction, Vol.5*. Ablex Publishing Corporation, Norwood, New Jersey. Pp. 87-112.
- Bertelsen, O., Breinbjerg, M., & Pold, S. (2007). Instrumentness for Creativity Mediation, Materiality, and Metonymy. In: *ACM SIGHCI Sixth Creativity and Cognition Conference*. [Online]. Available at: <http://dl.acm.org/citation.cfm?id=1254992>. [Accessed 8 July 2013].
- Bolstad, T. (2009). *iOS Rotary Slider Controls-AKA Knobs*. [Online]. Available at: <http://timbolstad.com/2010/09/19/ios-rotary-slider-controls-aka-knobs/>. [Accessed 8 June 2013].
- Bongers, B. (2000). Physical Interfaces in the Electronic Arts: Interaction Theory and Interfacing Techniques for Real-time Performance. In Wanderley, M. and Battier, M. (Eds.), *Trends in Gestural Control of Music*. Paris: Ircam-Centre Pompidou. pp. 231-258. [Online]. Available at: http://hapticity.net/pdf/nime2005_192-works_cited/10.1.1.90.4997.pdf. [Accessed 25 October 2013].
- Boulanger, A. (2004). *Autism, New Music Technologies, and Cognition*". M.S. Thesis. Massachusetts Institute of Technology. [Online]. Available at: <http://dspace.mit.edu/handle/1721.1/37390>. [Accessed 10 October 2012].
- Boulanger Labs (2012). *CsGrain*. [Online]. Available at: <http://www.boulangerlabs.com/products/csgrain/>. [Accessed 17 October 2012].
- Boulanger, R. Introduction to Sound Design in Csound. In Boulanger, R. (Ed.). *The Csound Book*. Cambridge, MIT Press. pp. 5-65.
- British Association for Music Therapy (2012a). *What is Music Therapy?* [Online]. Available at: <http://www.bamt.org/music-therapy.html> [Accessed 8 December 2012].
- British Association for Music Therapy (2012b). *Who Can Benefit?* [Online]. Available at: <http://www.bamt.org/music-therapy/who-can-benefit.html>. [Accessed 9 December 2012].
- Brunner, C. (2009). A Cultural Approach to the Notion of the Instrument. In: *International Computer Music Conference, Montreal, 2009*. [Online]. Available at: <http://quod.lib.umich.edu/cgi/t/text/text-idx?c=icmc;view=toc;idno=bbp2372.2009.079>. [Accessed 2 July 2013].

Burland, K. and Magee, W. (2012). Developing identities using music technology in therapeutic settings. *Psychology of Music*. [Online]. Available at: <http://pom.sagepub.com/content/early/2012/11/15/0305735612463773.abstract>. [Accessed 5 December 2013].

Cadoz, W. and Wanderley, M. (2000). Gesture – Music. In Wanderley, M and Battier, M. (Eds.). *Trends in Gestural Control of Music*. Paris: IRCAM-Centre Pompidou. pp. 71-94. [Online]. Available at: [http://www.vigliensoni.com/McGill/CURSOS/2009_09/MUMT620/READINGS/2/2_Gesture-Music%20\(Cadoz-Wanderley\).pdf](http://www.vigliensoni.com/McGill/CURSOS/2009_09/MUMT620/READINGS/2/2_Gesture-Music%20(Cadoz-Wanderley).pdf). [Accessed 25 October 2012].

Campbell, M. (2013). Apple awarded patent for more accurate haptic feedback system. *Apple Insider*. Available at: <http://appleinsider.com/articles/13/02/19/apple-awarded-patent-for-more-accurate-haptic-feedback-system>. [Accessed 3 October 2013].

Carlson, C. and Wang, G. Borderlands: An Audiovisual Interface for Granular Synthesis. In: *NIME 2011, Ann Arbor, 21-23 May*. [Online]. Available at: http://www.modulationindex.com/borderlands_paper.pdf. [Accessed 25 February 2013].

Casciato, C. and Wanderley, M. (2007). Lessons from Experienced Gestural Controller Users. In: *ENACTIVE/07, Paris, 19-22 November*. [Online]. Available at: https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CDAQFjAA&url=http%3A%2F%2Fwww.idmil.org%2F_media%2Fwiki%2Fenactive07_casciatowanderley.pdf%3Fid%3Dpublications%26cache%3Dcache&ei=Epd_Uu2XD6m84ATGyIGYDA&usq=AFQjCNEAnRcUTCPO04NKzZFKJiKcqU7LyQ&sig2=oH1GecapktRYLW-JGtH1Hw&bvm=bv.56146854,d.bGE&cad=rja. [Accessed 21 October 2013].

Castro, D. (2008). Digital Quality of Life: Accessibility for People with Disabilities. [Online]. Available at: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1284647. [Accessed 2 October 2008].

Chadabe, J. (1984). Interactive Composing: An Overview. *Computer Music Journal*. 8(1), pp. 22-27. [Online]. Available at: <http://www.jstor.org/stable/3679894>. [Accessed 3 November 2012].

Chadabe, J. (1977). Some Reflections on the Nature of the Landscape within which Computer Music Systems are Designed. *Computer Music Journal*. 1(3). Pp. 5-11. [Online]. Available at: <http://www.jstor.org/stable/3679605>. [Accessed 3 November 2012].

Challis, B. and Smith, R. (2012). Assistive Technology and Performance Behaviours in Music Improvisation. *Lecture Notes for the Institute of Computer Sciences, Social Informatics, and Telecommunications Engineering*. pp 63-70., 101. [Online]. Available at: http://link.springer.com/chapter/10.1007/978-3-642-33329-3_8?null#page-1. [Accessed 1 December 2012].

Cleveringa, W., et al. (2009). Assisting Gesture Interaction on Multi-Touch Screens. [Online]. Available at: http://openexhibits.org/wp-content/uploads/papers/Cleveringa_2009_AGI.pdf. [Accessed 6 March 2013].

Cole, B. MIDI and Communalilty. *Organised Sound*. 1(1). pp 51-54. 1996.

Cook, P. (2001). Principles for Designing Computer Music Controllers. In: *NIME 2001, Seattle, 1-2 April*. [Online]. Available at: http://www.cs.princeton.edu/sound/publications/prc_chi2001.pdf. [6 July 2013].

“Composition”. Harvard Dictionary of Music. 4th ed. 2003.

Corêa, A., et al. (2009). Computer Assisted Music Therapy: a Case Study for an Augmented Reality Musical System for Children with Cerebral Palsy Rehabilitation. In: *Proceedings of the Ninth IEEE International Conference on Advanced Learning Technologies*. [Online]. Available at: http://celstec.org.uk/system/files/file/conference_proceedings/icalt2009/data/3711a218.pdf. [Accessed 21 November 2012].

Csounds.com. (2012). *About Csound*. [Online]. Available at: <http://csounds.com/about>. [Accessed 11 May 2013].

Di Scipio, A. (2003). Sound is the interface: from interactive to ecosystemic signal processing. *Organised Sound*. 8(3). Pp. 22-27. [Online]. Available at: http://www.ak.tu-berlin.de/fileadmin/a0135/Unterrichtsmaterial/Di_Scipio/Sound_is_the_interface.PD. [Accessed 16 November 2012].

Duggal, V. (2011). "Ti.UI.Slider.thumbImage." *appcellerator network, Posting to Titanium Mobile*. [Online]. Available at: <https://jira.appcelerator.org/browse/TIMOB-6595?page=com.atlassian.jira.plugin.system.issuetabpanels:changehistory-tabpanel>. [Accessed 19 July 2013].

Essl, G., et al. (2008). Developments and Challenges turning Mobile Phones into Generic Music Performance Platforms. In: *Mobile Music Workshop 2008, Vienna, 13-15*. [Online]. Available at: http://web.eecs.umich.edu/~gessl/georg_papers/os09-mobileinteractivity.pdf. [Accessed 3 November 2012].

Farbood, M. (2001). Hyperscore: A New Approach to Interactive, Computer-Generated Music. M.S. Thesis. Massachusetts Institute of Technology. [Online]. Available at: http://opera.media.mit.edu/papers/Masters_Mary.pdf. [Accessed 6 November 2012].

Fels, S., Gadd, A., Mulder, A. (2002). Mapping Transparency Through Metaphor: Towards More Expressive Musical Instruments. *Organised Sound*. 7(2). [Online]. Available at: http://hct.ece.ubc.ca/research/metamuse/papers/FelsGaddMulder_os2002.pdf. [Accessed 26 March 2013].

Gabor, D. (1944). Theory of Communication. In: *J.IEE*, 93(26). Pp. 429-457. Available at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05298517>. [Accessed 28 June 2013].

Gabor, D. (1947). Acoustic Quanta and The Theory of Hearing”. *Nature*, 159(4044). Pp. 591-594. [Online]. Available at:

<http://www.nature.com/nature/journal/v159/n4044/pdf/159591a0.pdf> . [Accessed 28 June 2013].

Geiger, G. (2006). Using the Touch Screen as a Controller for Portable Computer Music Instruments. In: *NIME 2006, Paris, 4-8 June*. [Online]. Available at: http://www.nime.org/proceedings/2006/nime2006_061.pdf . [Accessed 16 November 2012].

Giordano, M. and Wanderley, M. (2013). Perceptual and Technological Issues in the Design of Vibrotactile-Augmented Interfaces for Music Technology and Media. In: *2013 Haptics-Audio Interaction Design Workshop*. [Online]. Available at: <http://www.idmil.org/publications>. [Accessed

Godøy, R., Haga, E., Jensenius, A. (2006). Exploring Music-Related Gestures by Sound-Tracing: A Preliminary Study. [Online]. Available at: https://www.duo.uio.no/bitstream/handle/10852/26899/Godxy_2006b.pdf?sequence=1. [Accessed 5 June 2013].

Gómez, D., et al. (2007). A Look at the Design and Creation of a Graphically Controlled Digital Musical Instrument. In: *NIME 2007, New York, 6-10*. [Online]. Available at: <http://dl.acm.org/citation.cfm?id=1279740.1279811>. [Accessed 8 March 2013].

Gorman, M., et al. (2007). A Camera-Based Music-Making Tool for Physical Rehabilitation. *Computer Music Journal*, 32(2). pp39-53.

GRAINPROC (2012). *Granular Synthesis for the iPad*. [Online]. Available at: <http://grainproc.e7mac.com/>. [Accessed 21 November 2012].

Gross, S. (2012). Material Skeuomorphism and the Relationship of Form to Function. In: *CHI'12, May 5-10, 2012, Austin*. [Online]. Available at: http://www.shadgross.com/wp-content/uploads/2012/08/Materiality.Gross_.pdf . [Accessed 28 March 2013].

Haueisen, J. and Knösche, T. (2001). Involuntary Motor Activity in Pianists Evoked by Music Perception. *Journal of Cognitive Neuroscience*, 13(6). pp 786-792. [Online]. Available at: http://www.brainmusic.org/EducationalActivitiesFolder/Haueisen_pianists2001.pdf. [Accessed 14 July 2013].

Hewett, et al. (1992). ACM SIGCHI Curricula for Human Computer Interaction. *SIGCHI.org*. [Online]. Available at: <http://old.sigchi.org/cdg/cdg2.html>>. [Accessed 3 October 2013].

Holland, S., et al. (2013). Music Interaction: Understanding Music and Human-Computer Interaction. In S. Holland, K. Wilkie, P. Mulholland, A. Seago, (Eds.). *Music and Human-Computer Interaction*. London: Springer, 2013. pp.1-28. [Online]. Available at: <http://link.springer.com/book/10.1007/978-1-4471-2990-5/page/1>. [Accessed 10 August 2013].

- Holmes, T. (2008). *Electronic and Experimental Music*. 3rd ed. New York: Routledge.
- Hözl, M., et al., (2009). Constraint-Muse: A Soft-Constraint Based System for Music Therapy. In Goos, Gerhard, et al. (Eds.). *Lecture Notes in Computer Science*, Volume 5728, pp.423-432. Available at: http://link.springer.com/chapter/10.1007%2F978-3-642-03741-2_29?LI=true#. [Accessed 21 November 2012].
- Hughes, N. (2013). *Apple has sold 170M iPads to date, implying sales near 15M in Sept. quarter*. [Online]. Available at: <http://appleinsider.com/articles/13/10/23/apple-has-sold-170m-ipads-to-date-implying-sales-near-15m-in-sept-quarter>. [Accessed 8 November 2013].
- Hunt, A. (2000). *Radical user-interfaces for real-time musical control*. PhD Thesis. University of York. [Online]. Available at: http://www-users.york.ac.uk/~adh2/Andy_Hunt_Thesis.html. [Accessed 14 November 2012].
- Hunt, A., and Kirk, R. (2000). Mapping Strategies for Music Performance. In Wanderley, M and Battier, M. (Eds.). *Trends in Gestural Control of Music*. Paris: Ircam-Centre Pompidou. pp. 231-258. [Online]. Available at: http://vigliensoni.com/BUP/Dropbox_10_11_15/proyectoMarco/Info/9_Mapping%20Strategies%20for%20Musical%20Performance_Hunt.pdf. [Accessed 25 October 2012].
- Hunt, A. and Kirk, R. (2003). MidiGrid: Past, Present, Future. In: *NIME, Montreal, 2003*. pp135-139.
- Hunt, A., Kirk, R., and Neighbour, M. (2004). Multiple Media Interfaces for Music Therapy. In: *IEEE MultiMedia*, 11(3). pp. 50-58. [Online]. Available at: <http://eprints.whiterose.ac.uk/654/1/hunta1.pdf>. [Accessed 14 November 2012].
- Hunt, A and Wanderley, M. (2002). Mapping performer parameters to synthesis engines. *Organised Sound*, 7(2). pp.97-108. [Online]. Available at: <http://digitalmusicstudio.wikispaces.asu.edu/file/view/Mapping+Performer+Parameters.pdf>. [Accessed 14 November 2012].
- Hunt, A., Wanderley, M., and Paradis, M. (2002). The Importance of Parameter Mapping in Electronic Instrument Design. In: *NIME 2002, Dublin, Ireland, May 24-26*. [Online]. Available at: <http://dl.acm.org/citation.cfm?id=1085171.1085207>. [Accessed 25 October 2012].
- Hunt, A., Wanderley, M., and Kirk, R. (2002). Towards a Model for Instrumental Mapping in Expert Musical Interaction. In: *International Computer Music Conference, Berlin, 2002*. [Online]. Available at: http://recherche.ircam.fr/equipements/analyse-synthese/wanderle/Gestes/Externe/Hunt_Towards.pdf. [Accessed 25 October 2012].
- Ingram, A., et al. Towards the Establishment of a Framework for Intuitive Multi-touch Interaction Design. In: *AVI '12: Proceedings of the International Working Conference on Advanced Visual Interfaces, Capri Island, 22-25 May 2012*. [Online]. Available at: <http://coitweb.uncc.edu/~xwang25/pubs/Amy-AVI2012.pdf>. [Accessed 25 March 2013].

Jordà, S. (2003). Interactive Music Systems for Everyone: Exploring Visual Feedback as a Way For Creating More Intuitive, Efficient and Learnable Instruments. In: *Stockholm Music Acoustics Conference, August 6-9, 2003*. [Online]. Available at: [http://infodate.nctu.edu.tw/teaching/techart/assi/94/9342804%E9%9F%B3%E6%A8%82%E6%96%87%E7%8D%BB\(%E9%83%AD%E6%80%A1%E5%A9%B7\)/Interactive%20Music%20Systems%20for%20%20Everyone.pdf](http://infodate.nctu.edu.tw/teaching/techart/assi/94/9342804%E9%9F%B3%E6%A8%82%E6%96%87%E7%8D%BB(%E9%83%AD%E6%80%A1%E5%A9%B7)/Interactive%20Music%20Systems%20for%20%20Everyone.pdf). [Accessed 12 July 2013].

Jordà, S. (2005). Instruments and Players: Some thoughts on digital lutherie. *Journal of New Music Research*. 33(3) pp.321-341.
Available at: <http://www.tandfonline.com/doi/pdf/10.1080/0929821042000317886>. [Accessed 11 July 2013].

Jordà, S., et al. (2007). The reacTable: Exploring the Synergy between Live Music Performance and Tabletop Tangible Interfaces. In: *Conference on Tangible and Embedded Interaction, Baton Rouge, February 15-17 2007*. [Online]. Available at: <http://dl.acm.org/citation.cfm?id=1226998>. [Accessed 19 November 2012].

Khooshabeh, P., et al. Gestural Musical Improvisation and Programming. In: *2005 Symposium on Visual Languages and Human-Centric Computing, Dallas, 20-24 September*. [Online]. Available at: [http://ieeexplore.ieee.org/xpl/tocresult.jsp?sortType%3Dasc_p_Sequence%26filter%3DAND\(p_IS_Number%3A32326\)%26rowsPerPage%3D50&pageNumber=2](http://ieeexplore.ieee.org/xpl/tocresult.jsp?sortType%3Dasc_p_Sequence%26filter%3DAND(p_IS_Number%3A32326)%26rowsPerPage%3D50&pageNumber=2). [Accessed 30 October 2012].

Kell, T. and Wanderley, M. (2013). A Quantitative Review of Mappings in Musical iOS Applications. In: *Proceedings of the Sound and Music Computing Conference 2013*. pp. 473-480. [Online]. Available at: <http://smcnetwork.org/node/1749>. [Accessed 3 February 2014].

Kirn, P. (2010). libpd: Put Pure Data in Your App, On an iPhone or Android, and Everywhere, Free. *Create Digital Music*. [Online]. Available at: <http://createdigitalmusic.com/2010/10/libpd-put-pure-data-in-your-app-on-an-iphone-or-android-and-everywhere-free/>. [Accessed 15 October 2012].

Kirn, P. (2012). csGrain Gets Granular Goodness on iPad 2/3; Vanguard of Multi-Platform Csound Renaissance. *Create Digital Music*. [Online]. Available at: <http://createdigitalmusic.com/2012/04/csgrain-gets-granular-goodness-on-ipad-23-vanguard-of-multi-platform-csound-renaissance/>. [Accessed 17 October 2012].

Kirn, P. (2013). As Touch and Laptops Converge, Finally Potential for Music Making? *Create Digital Music*. [Online]. Available at: <http://createdigitalmusic.com/2013/06/as-touch-and-windows-converge-potential-for-performing-music-live-djing/>. [Accessed 3 June 2013].

Korg (2013). iKaossilator. [Online]. Available at: <http://www.korg.com/ikaossilator>. [Accessed 9 November 2013].

Kriedler, J. (2009). Programming Electronic Music in Pd. [Online]. Available at: <http://www.pd-tutorial.com/english/index.html>. [Accessed 15 October 2012].

Lazzarini, V., et al. (2012a). “Digital Audio Effects on Mobile Platforms” *15th International Conference on Digital Audio Effects, DAF 2012*. Ed. J.Wells, York Department of Electronics Audio Lab, York UK, 2012. Pp. 287-293.

Lazzarini, V., et al. (2012b). The Mobile CSound Platform” *International Computer Music Conference: Non Cochlear Sound. Ljubljana, 2012*. [Online]. Available at: <http://quod.lib.umich.edu/cgi/p/pod/dod-idx/mobile-csound-platform.pdf?c=icmc;idno=bbp2372.2012.031>. [Accessed 28 November 2012].

Lee, A.S.C. (2000). Granular Synthesis in Csound. In Boulanger, R. (Ed.). *The Csound Book*. Cambridge, MIT Press. pp. 281-292.

Leman, M. (2010). Music, Gesture, and the Formation of Embodied Meaning. In Godøy and Leman, (Eds.). *Musical Gestures: Sound, Movement, and Meaning*”, Eds Godøy and Leman. Routledge, New York. pp. 126-153.

Leman, M., and Godøy, R. (2010). Why Study Musical Gesture? In Godøy and Leman, (Eds.). *Musical Gestures: Sound, Movement, and Meaning*”, Routledge, New York. pp.3-11.

Lee, M. and Wessel, D. (1992). Connectionist Models for Real-Time Control of Synthesis and Compositional Algorithms. In: *International Computer Music Conference 1992, San Jose*. [Online]. Available at: http://cnmat.berkeley.edu/system/files/attachments/Connectionist_Models.pdf. [Accessed 29 March 2013].

Machover, T. Instruments, Interactivity, and Inevitability. In: *NIME 2002, Dublin, May 24-26*. [Online]. Available at: http://www.nime.org/proceedings/2002/nime2002_115.pdf. [Accessed 11 July 2013].

Magee, W. (2006). Electronic technologies in clinical music therapy: A survey of practice and attitudes. *Technology and Disability*, 18(3). pp.139-146.

Magee, W. and Burland, K. (2008). Using electronic music technologies in music therapy: opportunities, limitations, and clinical indicators. *British Journal of Music Therapy*, 22(1) pp. 3-15.

Magnusson, T. and Mendieta, E. The Acoustic, The Digital and The Body: A Survey of Musical Instruments. In: *NIME 2007, New York, June 6-10*. [Online]. Available at: <http://dl.acm.org/citation.cfm?id=1279757>. [Accessed 11 March 2013].

Mark, D. et al. (2011). *Beginning iOS Development*. Apress Media LLC, New York.

McDermott, James, et al. (2013). Should Music Interaction Be Easy? In S. Holland, K. Wilkie, P. Mulholland, A. Seago, (Eds.). *Music and Human-Computer Interaction*. London: Springer. pp29-48. [Online]. Available at: <http://link.springer.com/book/10.1007/978-1-4471-2990-5/page/1>. [Accessed 10 August 2013].

Miranda, E and Wanderley, M. (2006). *New Digital Musical Instruments: Control and Interaction Beyond the Keyboard*. Middleton, WI: A-R Editions, Inc.

McKay, G and Higham, B. (2011). *Community Music: History and Current Practice, its Constructions of 'Community', Digital Turns and Future Soundings*. [Online]. Available at: http://www.academia.edu/1066617/Community_Music_History_and_Current_Practice_its_Constructions_of_Community_Digital_Turns_and_Future_Soundings. [Accessed 8 December 2012].

McCurdy, I. (2010). REVERBERATION in “The Csound FLOSS Manual”. [Online]. Available at: http://en.flossmanuals.net/csound/ch035_e-reverberation/. [Accessed 17 June 2013].

Merril, D and Raffle, H. (2013). *Semiacoustic Sound Exploration with the Sound of Touch*. In Franinović, K and Serafin, S. (Eds.). *Sonic Interaction Design*. Massachusetts: The MIT Press. pp 213-224.

MIDIcreator User Manual. *MIDIcreator Resources*. [Online]. Available at: <http://www.midicreator-resources.co.uk/midicreator/resources/Manuals/manual.pdf>. [Accessed 8 December 2012].

MIDIcreator Sensors. *MIDIcreator Resources*. [Online]. Available at: <http://www.midicreator-resources.co.uk/midicreator/resources/Manuals/manual.pdf>. [Accessed 8 December 2012].

Nagler, J. (2011). *Music Therapy Methods with Hand-Held Music Devices in Contemporary Clinical Practice: a commentary*. *Music and Medicine*, 3(3). pp.196-199. [Online]. Available at: <http://mmd.sagepub.com/content/3/3/196.full.pdf+html>. [Accessed 1 November 2012].

Nash, C. (2011). “Supporting Virtuosity and Flow in Computer Music. PhD Thesis”. University of Cambridge. [Online]. Available at: http://www.academia.edu/2470149/Supporting_virtuosity_and_flow_in_computer_music. [Accessed 27 August 2013].

Neate, T. (2012). “The Interactive Sonification of Electromyography Data for iOS”. BSc Project Report. University of York.

Nelson, J.C. (2000). *Understanding and Using Csound's GEN Routines*. In Boulanger, R. (Ed.). *The Csound Book*. Cambridge: Massachusetts Institute of Technology Press, pp. 65-90.

Norman, Donald. (2001). *The Design of Everyday Things*. London: The MIT Press.

Oh, J., et al. (2010). *Evolving The Mobile Phone Orchestra*. In: *NIME 2010, Sydney, 15-18 June*. [Online]. Available at: <https://ccrma.stanford.edu/~jorgeh/assets/publications/mophonime2010.pdf>. [Accessed 16 November 2012].

Overholt, D. (2009). The Musical Interface Technology Design Space. *Organised Sound* 14(2). pp. 217-226. [Online]. Available at: <http://journals.cambridge.org/action/displayAbstract;jsessionid=F8388B8F794998202D0C3CFB5078384C.journals?fromPage=&aid=5882432>. [Accessed 16 November 2012].

Roads, C. (1988). Introduction to Granular Synthesis. *Computer Music Journal*, 12(2). Pp. 281-282. [Online]. Available at: <http://www.jstor.org/stable/3679937>. [Accessed 16 November 2012].

Roads, C. (1996). *The Computer Music Tutorial*. Cambridge: Massachusetts Institute of Technology.

Roads, C. (2001). *Microsound*. Cambridge: Massachusetts Institute of Technology.

Roberts, C., Forbes, A., & Höllerer, T. (2013). Enabling Multimodal Mobile Interfaces for Interactive Musical Performance. In: *NIME'13, Daejeon, May 27-30, 2013*. [Online]. Available at: http://www.mat.ucsb.edu/Publications/Enabling_MMI_for_IMP.pdf. [Accessed 9 November 2013].

Rogers, Y., Sharp, H., and Preece, J. (2011). *Interaction Design: Beyond Human – Computer Interaction*. Chichester. John Wiley & Sons Ltd.

Rudi, J. (1997). Computer Music Composition for Children. *IEEE Signal Processing Magazine*, 143. pp. 140-143. [Online]. Available at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04117938>. [Accessed 30 October 2012].

Saffer, D. (2009). *Designing Gestural Interfaces*. Sebastapool: O'Reilly Media.

Schlei, K. (2012). TC-11: A Programmable Multi-Touch Synthesizer for the iPad. In: *NIME 2012, Ann Arbor, June 15-18, 2012*. [Online]. Available at: http://www.eecs.umich.edu/nime2012/Proceedings/papers/230_Final_Manuscript.pdf. [Accessed 2 April 2013].

Selker, T. (2008). Touching the Future *Communications of the ACM*, 51(2). pp.14-16. [Online]. Available at: http://u.cs.biu.ac.il/~ariel/download/mm664/resources/interfaces_technologies/touch/touching_the_future.pdf. [Accessed 12 February 2013].

Settel, Z., and Lippe, C. Convolution Brother's Instrument Design. In: *NIME 2003, Montreal, May 22-14*. [Online]. Available at: <http://dl.acm.org/citation.cfm?id=1085761>. [Accessed 13 July 2013].

Synthtopia (2013). *Amazing Audio Engine' Streamlines iOS App Development*. [Online]. Available at: <http://www.synthtopia.com/content/2013/03/20/amazing-audio-engine-streamlines-ios-music-app-development/>. [Accessed 17 May 2013].

Tanaka, A. "Musical Performance Practice on Sensor-based Instruments". In M. Wanderley and M. Battier, (Eds.). *Trends in Gestural Control of Music*. Paris: Ircam-Centre Pompidou.

pp. 389-406. [Online]. Available at:
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.74.2844&rep=rep1&type=pdf>.
[Accessed 8 July 2013].

“The Reactable”. Photo. Reactable. *Music Technology Group*.
[Online]. Available at: <http://mtg.upf.edu/project/reactable>. [Accessed 4 August 2013].

Treadaway, C. Hand E-craft: an Investigation into Hand Use in Digital Creative Practice. In: *Seventh ACM conference on Creativity and Cognition, New York, 2009*. [Online]. Available at:
<http://dl.acm.org/citation.cfm?id=1640233&picked=prox&CFID=359827105&CFTOKEN=95641065> [Accessed 10 March 2013].

Wanderely, M. and Orio, N. (2002). Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI. *Computer Music Journal*, 26(3). pp. 62-76. [Online]. Available at:
<http://www.mitpressjournals.org/doi/abs/10.1162/014892602320582981?journalCode=comj>
[Accessed 24 March 2013].

Wanderley, M. (2004). SensorWiki. [Online]. Available at:
<http://www.sensorwiki.org/doku.php/>. [Accessed 25 November 2012].

Wanderley, M. and Depalle, P. (2004). Gestural Control of Sound Synthesis. *J.IEE*, 92(4). pp. 632-644. [Online]. Available at:
<http://kkothman.iweb.bsu.edu/oldTeaching/mumet440/papers/wanderly-gestcontrol.pdf>.
[Accessed 21 October 2013].

Wessel, D. et al. Intimate Musical Control of Computers with a Variety of Controllers and Gesture Mapping Metaphors. In: *NIME 2002, Dublin May 24-26*. [Online]. Available at:
http://www.nime.org/proceedings/2002/nime2002_192.pdf. [Accessed 8 March 2013].

Wessel, D. and Wright, M (2002). Problems and Prospects for Intimate Musical Control of Computers. *Computer Music Journal*, 26(3). pp.11-12. [Online]. Available at:
<http://dl.acm.org/citation.cfm?id=1245198>. [Accessed 24 March 2013].

Weinberg, Gil. (1999). *Expressive Digital Musical Instruments for Children*. M.S. Thesis, Massachusetts Institute of Technology. [Online]. Available at:
<http://dspace.mit.edu/handle/1721.1/62942>. [Accessed 10 October 2012].

Winkler, T. Making Motion Musical: Gesture Mapping Strategies for Interactive Computer Music. In: *International Computer Music Conference Proceedings 1995, Banff Centre for the Arts*. [Online]. Available at: <http://www.cin.ufpe.br/~fcac/tg-pesquisa280520100951/Making%20motion%20musical-Gesture%20mapping%20strategies%20for%20interactive%20computer%20music.pdf>.
[Accessed 5 March 2013].

Wilkie, K., et al. (2010). What Can the Language of Musicians Tell Us About Music Interaction Design? *Computer Music Journal*, 34(4) pp. 34-39. [Online]. Available at:

http://www.mitpressjournals.org/doi/abs/10.1162/COMJ_a_00024?journalCode=comj. [Accessed 30 March 2013].

Wilson, T., et al. (2007). How the iPhone Works. *HowStuffWorks.com* <<http://electronics.howstuffworks.com/iphone2.htm>> [Accessed 09 February 2014].

Wöldecke, B, et al. (2012). ANTracks 2.0 – Generative Music on Multiple Multitouch Devices. In: *NIME'12, Ann Arbor, June 15-18, 2012*. [Online]. Available at: http://www.nime.org/proceedings/2010/nime2010_348.pdf. [Accessed 23 March 2013].

Woojijuce (2012). *Grain Science: Advanced granular synthesis for iOS*. [Online]. Available at: <http://www.woji-juice.com/products/grain-science/>. [Accessed 9 October 2013].

Wright, J., et al. CyberBand: A “Hands-On” Music Composition Program. In: *International Computer Music Conference 1997*. Thessaloniki. [Online]. Available at: http://openmuse.org/noncpl/ICMC1997_CB.PDF. [Accessed 29 January 2013].

Velcazo, C. (2012). *3,997 Models: Android Fragmentation As Seen By The Developers of OpenSignalMaps*. TechCrunch. [Online]. Available at: <http://techcrunch.com/2012/05/15/3997-models-android-fragmentation-as-seen-by-the-developers-of-opensignalmaps/>. [Accessed 15 October 2012].

Vercoe, B. et al., n.d. The Canonical Csound Reference Manual, Version 5.13.

Xambnaó, A., et al. (2011). “Multi-touch interaction principles for collaborative real-time music activities: towards a pattern language.” *International Computer Music Conference Proceedings, Huddersfield, 2012*. Web. [Online]. Available at: <http://oro.open.ac.uk/28757/>. [Accessed 16 November 2012].