PERCEPTUALLY COHERENT MAPPING SCHEMATA FOR VIRTUAL SPACE AND MUSICAL METHOD

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF MUSIC
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Robert Hamilton

June 2014

This dissertation is online at: http://purl.stanford.edu/ts761kv5081

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Chris Chafe, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Jonathan Berger

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Ge Wang

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Our expectations for how visual interactions *sound* are shaped in part by our own learned understandings of and experiences with objects and actions, and in part by the extent to which we perceive *coherence* between gestures which can be identified as "sound-generating" and their resultant sonic events. Even as advances in technology have made the creation of dynamic computer-generated audio-visual spaces not only possible but increasingly common, composers and sound designers have sought tighter integration between action and gesture in the visual domain and their accompanying sound and musical events in the auditory domain. Procedural audio and music, or the use of real-time data generated by in-game actors and their interactions in virtual space to dynamically generate sound and music, allows sound artists to create tight couplings across the visual and auditory modalities. Such procedural approaches however become problematic when players or observers are presented with audio-visual events within novel environments wherein their own prior knowledge and learned expectations about sound, image and interactivity are no longer valid.

With the use of procedurally-generated music and audio in interactive systems becoming more prevalent, composers, sound-designers and programmers are faced with an increasing need to establish low-level understandings of the crossmodal correlations between visual gesture and sonified musical result both to convey artistic intent as well as to present communicative sonifications of visual action and event. For composers and designers attempting to build evocative and expressive procedural sound and music systems, when the local realities of any given virtual space are completely flexible and malleable, there exist few to no dependable locale-specific models upon which to base their choices of mapping schemata.

This research focuses jointly on the creative and technical concerns necessary to build procedurally-generated crossmodal musical interactions, as well as on the perceptual issues surrounding specific mapping schemata linking interactions with sound and music. A software solution and methodology are presented to facillitate the mapping of parameters of action, motion and gesture from virtual space to sound-generating process, allowing composers and designers to repurpose real-time data as drivers for compositional and sound-related process. Creative and technical examples drawn from a series of multimodal musical experiences are presented and discussed, exploring a variety of potential mapping schemata as well as the inner workings of the presented codebases.

To assess the perceived coherence between motion and gesture in the visual modality and generated sound and musical events in the auditory modality, this research also details a user-study measuring the impact of audio-visual crossmodal correspondences between low-level attributes of motion and sound. Subjects taking part in a controlled study were presented with multimodal examples of musically sonified motion in a pairwise comparison task and asked to rate the perceived fit between visual and auditory events. Each example was defined as a composite set of simple motion and sound attributes. Study results were analyzed using the Bradley-Terry statistical model, effectively calculating the relative contribution of each crossmodal attribute within each attribute pairing to the perceived coherence or 'fit' between audio and visual data. The statistical analysis of correlated motion/sound mappings and their relative contributions to the perceptual coherence of audio-visual interactions lay the groundwork towards the establishment of predictive models linking attributes of sound and motion to perceived fit.

This work is dedicated to my family,
without whom none of this would matter.

Acknowledgements

I feel indeed privileged and humbled to have been able to carry out the work presented in this dissertation within a community as genuinely supportive and inspiring as the one at CCRMA. The vision and character of the denizens of this place are truly amazing and have on more than one occasion invigorated my flagging spirits through their camaraderie, humor and random acts of genius.

This work would not have been at all possible without the gentle guiding hand of my advisor, Chris Chafe. His wisdom and encouragement know no bounds and his ability to infuse our lives and work with his own creative passions, wit and dedication have been a true inspiration.

I would like to give sincere thanks to the members of my thesis committee: to Ge Wang, whose energy and artistic madness have fueled so many unforgettable experiences and to Jonathan Berger whose support and enthusiasm are never wavering. Thanks also to the members of my oral defense committe: Takako Fujioka for her support and guidance during the experimental and analytic aspects of this work and Paul Demarinis for his insight.

I'd like to thank my friend Chris Platz for his amazing collaboration and staggering artistic vision. Chris built nearly every visual environment, avatar and animation on display in this body of work and each one has been more mind blowing than the one before. I look forward to many more incredible journeys both physical and virtual working together.

I am grateful to have been able to collaborate with incredible minds and talents like Juan Pablo Cáceres, traveling the globe only to unite performers and audiences over network pipes. Thank you to Fernando Lopez-Lezcano for all his support over the years, be it technological, musical or moral. His steadfast vision and dedication to doing things the right (if not the easy) way have always been an enormous inspiration. And for her incredible efforts keeping me on track at Stanford for almost 10 years, I'd like to give a huge thank you to Debbie Barney.

Thank you to all the CCRMA and Department of Music faculty and staff for their friendship and support over the many years including Carr Wilkerson, Julius Smith, Jay Kadis, Jonathan Abel, John Chowning, Eleanor Selfridge-Field, Nette Worthey, Craig Sapp, Sasha Leitman, Dave Berners, Mario Champagne, Bill Verplank, Mark Applebaum and Jarek Kapuscinski. A very special thanks to Max and Marjorie Mathews whose wit and wisdom we miss on a daily basis.

I would like to thank all my Stanford classmates, friends and colleagues over the years including Spencer Salazar, Jieun Oh, Jorge Herrera, Edgar Berdahl, Jeff Smith, Nick Bryant, Hongchan Choi, Juhan Nam, John Granzow, Romain Michon, Turgut Ercetin, Kurt Werner, Woon Seung Yeo, Michael Gurevich, Hiroko Terasawa, Bruno Ruviaro, Luke Dahl, Michael Berger, Juan Cristobal Cerillo, Per Bloland, Nick Kruge, Aaron Trumm, Turner Kirk, Phaedon Sinis, Eoin Callery, Stephen Henderson, Reza Payami, Mike Gao, Alex Chechile, Myles Borins, June Holtz, Roberto Morales Manzares, Rocco Di Pietro, Maureen Chowning, Henrik Bennetsen, Jeffrey Schnapp, Daniel Horn, Daniel Miller and Dave Kerr.

For their patience in helping me unravel the mysteries of statistical analysis, much thanks to Chiara Sabatti, Blair Kaneshiro and Cameron Turner. Thanks to Jeannie, Jeff, Perry, Mark, Turner, alltom, Amanda, Jodi, Lynn, Tony, Shae, Stephanie, Jessica, Michael, Sunil, Lindsey, Tricia, Ryan, John, Stefan, Elon, Parag, Alex, Loreen, Erika and the entire team at Smule for providing so many interesting distractions over so many years. And thanks to my teachers and friends from Dartmouth and and Peabody, especially Jon Appleton, McGregor Boyle and Geoff Wright for their inspiration, guidance and support over the years.

Most of all I'd like to thank my family to whom this dissertation is dedicated. To my parents for their unwavering support over my lifetime of learning. To my brother Bill whose memory is with me always. To my wife Chrysi who inspires me everyday to not only keep exploring new grounds but also to remember to enjoy the precious moments and those whom we hold most dear. And to Miltos, who just wants to fly the bird.

Contents

Abstract iv

Acknowledgements vii

1 Introduction 1

1.1 Motivation . . . 1
1.2 Problem Statement . . . 2
1.3 Musical Data . . . 3
1.3.1 Musical Sonification . . . 3
1.3.2 Crossmodal Mapping . . . 3
1.3.3 Music and Games . . . 4
1.3.4 Procedural Music . . . 5
1.4 Contributions . . . 5
1.4.1 UDKOSC . . . 6
1.4.2 Dataset of Crossmodal Sonifications . . . 6
1.4.3 User Study of Perceived Crossmodal Coherence . . . 6
1.4.4 OSC Toolkit . . . 7
1.5 Outline . . . 8

2 Background 9

2.1 Metaphor, Sound and Music . . . 10
2.2 Crossmodal Correspondence and Binding . . . 11
2.3 Perceptual Coherence . . . 12
2.4 Gesture and Motion . . . 14

2.5 Presence, Immersion and Virtual Space . . . 15
2.6 Computer Gaming and Music . . . 16
2.7 Procedural Audio and Music . . . 17
2.8 Interactive Sound and Music . . . 19
2.9 Networked Musical Environments . . . 19
2.10 Virtual Space and the Representation of Sound . . . 21
2.11 Summary . . . 22

3 Music in Virtual Worlds 23
3.1 Early Procedural Musical Environments . . . 24
3.1.1 q3apd . . . 25
3.1.2 *Maps and Legends* . . . 25
3.1.3 q3osc . . . 28
3.1.4 *nous sommes tous Fernando...* . . . 30
3.1.5 Sirikata . . . 31
3.1.5.1 *Dei Due Mondi* . . . 32
3.1.5.2 *In C* . . . 34
3.1.5.3 *Canned Bits Mechanics* . . . 36
3.1.5.4 *Dialoghi* . . . 37
3.2 UDKOSC . . . 37
3.2.1 Unreal Development Kit (UDK) . . . 38
3.2.2 System Overview . . . 38
3.2.3 Output Data. . . . 40
3.2.4 Input Data. . . . 40
3.3 *Tele-harmonium* . . . 41
3.3.1 Instrument Constructs . . . 42
3.3.2 Projectiles . . . 43
3.3.3 Compositional Elements and Mapping . . . 43
3.4 *ECHO::Canyon* . . . 45
3.4.1 Background . . . 45
3.4.2 Musical Sonification in *ECHO::Canyon* . . . 46

3.4.3 Environment . . . 47
3.4.4 Crystal Emitters . . . 48
3.4.5 Canyons . . . 48
3.4.6 Characters . . . 49
3.4.6.1 Valkordia . . . 50
3.4.6.2 Trumbruticus . . . 50
3.4.6.3 Shelltapper . . . 51
3.5 Composing Musical Interactions . . . 52
3.5.1 Musical Projectiles . . . 53
3.5.2 Swarming and Flocking behaviors . . . 54
3.5.3 Musical Pathing . . . 55
3.5.3.1 Compositional Mapping in *Maps and Legends* . . . 56
3.5.3.2 Actor Paths in *In C* . . . 56
3.5.3.3 Canyon walls and floors in *ECHO::Canyon* . . . 57
3.5.4 Skeletal Tracking of Avatar Physiologies . . . 58
3.5.4.1 Wing and Trunk Motion . . . 60
3.5.4.2 Posing States . . . 61
3.5.4.3 Creature Calls . . . 63
3.6 Spatialization Techniques . . . 64
3.6.1 Spatial Super-imposition . . . 65
3.6.1.1 Correlated speaker positioning . . . 66
3.6.1.2 Virtual Replicas . . . 66
3.7 Summary . . . 69

4 Crossmodal Dataset **70**
4.1 Overview . . . 70
4.2 Visual Stimuli . . . 71
4.2.1 OSCControl Scripting . . . 71
4.2.2 Avatar . . . 72
4.2.3 Camera Position and Angle . . . 73
4.2.4 Acceleration . . . 74

4.2.5 Deceleration . . . 76
4.2.6 Continuous Rotation . . . 78
4.2.7 Discrete Rotation . . . 80
4.2.8 Jump Event . . . 82
4.2.9 Circle . . . 85
4.3 Audio Stimuli . . . 87
4.3.1 Instrument: STK Clarinet . . . 87
4.3.2 Parameter Mappings . . . 88
4.3.3 Parameter Mapping Ranges . . . 88
4.4 Summary . . . 89

5 Evaluating Perceived Coherence 90
5.1 Study Overview . . . 92
5.2 Research Questions . . . 93
5.3 Study Procedures . . . 94
5.3.1 Amazon Mechanical Turk . . . 94
5.3.2 Participants . . . 95
5.3.3 Study Format . . . 95
5.3.4 Data Validation . . . 97
5.3.5 Attribute Descriptors . . . 99
5.4 Data Analysis . . . 101
5.4.1 Bradley-Terry Model . . . 101
5.5 Results . . . 102
5.5.1 Ranked Fit . . . 103
5.5.2 Attributes as predictors of fit . . . 106
5.5.3 Directional attribute pairs as predictors of fit . . . 107
5.5.3.1 Mean fit for Speed . . . 108
5.5.3.2 Mean Fit for Rotation . . . 109
5.5.3.3 Mean Fit for Height . . . 110
5.5.3.4 Mean Fit for Frequency . . . 111
5.5.3.5 Mean Fit for Amplitude . . . 112

5.5.3.6 Mean Fit for Breath Pressure . . . 113
5.5.4 Inverse and Directional Pairings . . . 114
5.5.5 Symmetrical Pairings . . . 117
5.5.6 Contour Matching . . . 118
5.5.7 Similarity . . . 121
5.6 Discussion . . . 124
5.6.1 Assessing Perceived Coherence and Fit . . . 124
5.6.2 The Role of Perceptual Invariance . . . 125
5.6.3 Symmetrical Mappings . . . 126
5.6.4 Potential Issues . . . 128
5.7 Summary . . . 130

6 Conclusions and Future Directions **131**
6.1 Review . . . 131
6.2 Contributions . . . 132
6.3 Future Work . . . 134
6.4 Concluding Remarks . . . 137

Appendices **139**

A UDKOSC Installation Guide **139**
A.1 Download and Install the UDK . . . 140
A.2 Download UDKOSC . . . 140
A.3 Install UDKOSC . . . 141
A.3.1 OSC .dll . . . 141
A.3.2 UDKOSC Class structure . . . 142
A.3.3 UDKOSC Config files . . . 142
A.3.4 Maps and Resources . . . 143
A.4 Configure and Build UDKOSC Project . . . 144
A.5 Test OSC Output and Input . . . 145

B OSC Toolkit **147**
B.1 OSCControl 147
B.1.1 Setup 148
B.1.2 Running OSCControl 148
B.1.3 Actor Controls 149
B.1.4 Camera Controls 151
B.1.5 Console Commands 152
B.1.6 Wait Command 152
B.1.7 Command Blocks 154
B.2 OSC Recording and Playback 155

Bibliography **157**

List of Tables

4.1 Parameter mapping ranges for the STK Clarinet instrument. 89

5.1 Attributes of Motion and Sound. 99
5.2 Directional attributes for Motion and Sound. 100
5.3 Top five examples ordered by ranked fit. 102
5.4 Bottom five examples ordered by ranked fit. 104
5.5 Perceived fit rankings in descending order 105
5.6 Predictive ability of single-modality attributes. 106
5.7 Grouped example rankings ordered by Estimate difference. 116
5.8 Symmetrical relationships for acceleration examples. 117
5.9 Top twenty-five most similar example pairings. 122
5.10 Average similarity for each attribute of motion and sound. 123

B.1 OSCControl Console Commands. 153

List of Figures

2.1 Spore and Electroplankton integrated music with in-game action. . . 18

3.1 OSC control data flow schematic. 24
3.2 Compositional map for *Maps and Legends.* 26
3.3 Distance drives spatialization in *Maps and Legends.* 27
3.4 Bi-directional data flow between q3osc and ChucK audio engine. . . . 29
3.5 GtkRadiant model of the CCRMA Listening Room. 30
3.6 Lizard performers in *nous sommes tous Fernando...* 32
3.7 Sirikata environment created for *Dei Due Mondi.* 33
3.8 Sirikata environment and performers from *In C.* 34
3.9 Networking schema for the *Due Serrata in Sirikata* concerts. 35
3.10 Pianist and avatar interaction in *Canned Bits Mechanics.* 36
3.11 Sirikata environment and performers from *Dialoghi.* 37
3.12 OSC and control data schematic for UDKOSC. 39
3.13 Tele-harmonium model in UDKOSC. 41
3.14 Tele-harmonium organ console. 42
3.15 Sound-generating and controlling constructs within the Tele-harmonium. 43
3.16 UDKOSC environment for *ECHO::Canyon.* 45
3.17 Crystal emitters in *ECHO::Canyon.* 47
3.18 Horizontal and vertical ray-traces in UDKOSC. 49
3.19 Valkordia avatar in flight. 50
3.20 Trumbruticus avatar. 51
3.21 Shelltapper avatar. 52
3.22 Projectile patterns in q3osc. 54

3.23 Musical pathways from *In C.* 57
3.24 Skeletal meshes for Trumbruticus and Valkordia avatars. 58
3.25 Composed pathways in *ECHO::Canyon.* 59
3.26 Range of motion for Valkordia wing in flight and posing state. 60
3.27 Sonification of Valkordia wing-tip bones. 61
3.28 Range of motion for Valkordia wings during posing state. 62
3.29 Inverse kinematic effectors controlling Valkordia wing position. 62
3.30 Trumbruticus avatar with extended trunk. 63
3.31 Valkordia wing tracking driving "call" sound. 64
3.32 Spatial super-imposition aligns virtual and physical listening spaces. . 65
3.33 Speaker layouts for *nous sommes tous Fernando....* 67
3.34 Virtual replicas of physical performance spaces. 68

4.1 Humanoid avatar used in user study. 72
4.2 Sequence showing avatar foward acceleration. 74
4.3 Acceleration raw OSC data plot and series of sonograms. 75
4.4 Avatar forward deceleration. 76
4.5 Deceleration raw OSC data plot and series of sonograms. 77
4.6 Avatar forward motion with continuous rotation. 78
4.7 Continuous rotation raw OSC data plot and series of sonograms. . . . 79
4.8 Avatar forward motion with discrete turn event. 80
4.9 Discrete rotation raw OSC data plot and series of sonograms. 81
4.10 Avatar forward motion with jump event. 82
4.11 Jump speed raw OSC data plot and series of sonograms. 83
4.12 Jump height raw OSC data plot and series of sonograms. 84
4.13 Avatar forward motion with full circle. 85
4.14 Circle rotation raw OSC data plot and series of sonograms. 86
4.15 Reference spectrogram of the STK Clarinet model. 87
4.16 Supercollider instrument definition for the STK Clarinet instrument. 88

5.1 Excerpt from Turk study interface. 96
5.2 Interaction plot for speed mean of fit 108

5.3 Interaction plot for rotation mean of fit 109
5.4 Interaction plot for height mean of fit 110
5.5 Interaction plot for frequency mean of fit 111
5.6 Interaction plot for amplitude mean of fit 112
5.7 Interaction plot for breath pressure mean of fit 113
5.8 Individual parameter contours. 118
5.9 Perceived fit for matched and unmatched contours. 119
5.10 Sound contour mean of fit grouped by motion type. 120
5.11 Similarity histogram and plots of example counts. 121

A.1 Unreal Frontend screen-capture. 143

Chapter 1

Introduction

1.1 Motivation

Our relationships with sound and space are complex, bounded on one side by the inflexible laws of physics and on the other by human cognition and perception. Through experience, exposure and experimentation we each develop a very personal cognitive understanding of our sound world. In doing so we are learning implicitly, continually building and revising internal models that describe how we expect sound - both environmental as well as musical sound - to accompany certain real-world interactions [36, 114, 67]. The strike of hammer on steel, the rumble of a passing train, or the cheering of a frenzied crowd all are familiar enough sounding events that most of us could agree that our internal representations of these sounds share a great number of commonalities. And while each one of us views and hears the world through different eyes and ears, our shared experiences within reality's relatively consistent sound worlds lead us to expect and predict certain sight-sound interactions in a similar way.

With the introduction of rendered immersive graphical computer environments, humans have the ability to completely reorient their visual and auditory systems, allowing a generated reality to take precedence over a physical one. In these created spaces, motion and gesture can act as direct extensions of our own physical actions or can be abstracted into forms which would be difficult if not impossible to recreate within the confines of the physical world. No commonalities of crossmodal interaction

are guaranteed when stepping across the digital frontier and no limits exist to the potential mappings of sight to sound (and sound to sight) other than the creativity and whimsy of designer and developer. As such our personal internal representations must reorient themselves within each new virtual experience, often requiring us to relearn and readjust our expectations while continuously reforming our own predictive models.

1.2 Problem Statement

If the interactions of action and sound in rendered space can be seen as one gateway to new internal models of sonic representation, how should we consider music? Music, with its loose affiliation of time and frequency-based structures painted with varying degrees of rigor and haphazard freedom already poses distinct challenges to the idea of a commonly held perception and internal representation for all but its most basic elements. Musical form and function as well as method and meaning vary widely from composer to composer, not to mention from listener to listener, across years of history and miles of geography. In many cases music serves as an external representation of a composer or performer's internal sonic world, an abstraction of any number of ideas, influences and goals into an audible construct. Our own personal internal representations of music and musical sound are thusly influenced not only by the sounds we hear in space but also by the proposed intentions of composer and performer, whether we consciously understand them or not. Adding another level of abstraction to such an already rich and personal set of representations is a daunting task, but one we must investigate when bringing together action and gesture from the visual modality with sound and musical expression in the auditory modality.

1.3 Musical Data

1.3.1 Musical Sonification

The understanding and analysis of musical sound provides us a low-level entry point from which we can engage this problematic. There exists a rich body of physics-based musical interaction and gesture in the history of instrument design and performance practice. From the drawing of bow on tuned string to the strike of hand on drumhead, to the arc of conductor's baton in space, musical gesture has over time evolved into a number of basic archetypes that have shaped many of our own internal representations about how musical sound is created and controlled. And while these archetypes vary from culture to culture and from age to age, their basic grounding in the physics of the real-world again reinforces an inherent commonality in how humans perceive and internalize musical action and gesture. But when attempting to create novel sonic and musical events within rendered environments, there are no requirements on how interaction and generated sound must relate. Musical sound and the gestures and interactions that create it can be explored and modified in rendered space by mapping any data generating process to any aspect of sound, anywhere along the continuum from direct low-level parameter mapping all the way to high-level control over elements of musical structure or abstract musical process.

1.3.2 Crossmodal Mapping

For composers and media artists seeking to present their own creative intentions and internal sonic representations to the outside world, the problem becomes one of crossmodal mapping: how to best marry sound generating processes and elements of musical form to visual occurrence within the rendered environment. In doing so we encounter a new musical problematic: when motion and action in space can directly create and control sound and music from low-level sounding events to high-level compositional structures, how does one decide upon the "right" cross modal mapping schemata? Turning towards the interest areas of psychology and cognition, by better understanding mechanisms which drive our memory of and expectation

for the sonic outputs of perceived interactions, physical or virtual, composers and designers can better generate creative and musical outputs that simply "make sense" to their audiences.

1.3.3 Music and Games

Computer-based representations of space and action have over the last five decades become both increasingly sophisticated and commonplace. The computer gaming industry has bloomed into a multi-billion dollar industry, offering rich graphic content and dynamic control interfaces on hardware platforms ranging from desktop and laptop computer systems to home gaming-consoles and mobile devices. The advent of ubiquitous fast network connections and an explosion in modern society's use of social media have supported the rise of popular massively multiplayer online environments - distributed server-based virtual spaces capable of engaging large numbers of clients at any given time. Communities built around socialization and communication like Second Life have become wildly popular alongside strictly game-based environments. And while there do exist large-scale music-based social gaming networks and gaming franchises focused on interactive musical gameplay, the majority of gaming titles still primarily use music as a supporter for game narrative and mood.

Even as graphics processing used in systems have become faster, more realistic and more sophisticated with the increase in power available to most computing devices, the allocation of cpu cycles to sound and music concerns in most commercial gaming software pipelines has remained relatively small by comparison. One result of such limitation is that the vast majority of sound assets used in software systems still consist of pre-recorded samples, artfully combined by composers and sound designers to create dynamic soundscapes and musical experiences. As such, of primary concern for composers working with interactive music systems for games and similar environments is the avoidance of undue repetition and the integration of thematic musical material with experiences unfolding in the software. Techniques such as horizontal arrangement and resequencing - where game data drives the playback of subsequent

thematic sections of a prerecorded composition - or vertical arrangement and orchestration - in which the density and orchestration of musical material in the form of layered recorded materials is driven by game data - allow composers to dynamically adjust their composed materials in alignment with game progress and narrative. And through the use of dynamic digital signal processing coloring both game audio and music, a greater degree of interactivity coupled with high-quality composed material can be achieved.

1.3.4 Procedural Music

Data rich systems found in computer gaming or virtual reality platforms, offer an opportunity to couple motion and action in virtual spaces to sound and music generating processes. By creating music and sound in virtual environments procedurally, that is by creating and controlling sounds through the mapping of parameters of motion, action or state to sound producing systems, the sonic experience presented to users can be tightly integrated with the visual and narrative experiences on which successful cognitive immersion in game environments is based. The same user control systems that control motion or action in space can control sound and music. By coupling user action in direct as well as abstracted fashion, rich artistic environments can be created. In this way, the virtual environment itself can become both an active and reactive participant in the sonic experience.

1.4 Contributions

The research described in this dissertation presents artistic and analytic techiques for mapping action and motion in virtual space to musical sound generating processes. The contributions detailed within consist of the UDKOSC framework for building procedural music systems, a toolkit to aid in the creation, recording and playback of OSC messages, a user study measuring the perceived coherence of crossmodal mapping schemata and an audio-visual dataset consisting of examples of sonified avatar motion.

1.4.1 UDKOSC

In order for artists and researchers to freely explore new mapping schemata and musical interactions between gesture and motion in rendered space and musical construct, there is a need for a framework that facillitates the coupling of data to sound generating process in a manner that is accessible without the need for low-level programming access. UDKOSC [51, 52] builds upon the Unreal Development Kit, a freely available game programming environment, exposing parameters of motion and action from within the game environment using the Open Sound Control (OSC) protocol [126, 127]. In this manner, artists and researchers can easily couple data streams representing action and motion from within the game environment with sound-generating processes or instruments using standard musical programming environments such as Supercollider, ChucK, Pure Data and Max/MSP. The source code for UDKOSC is freely available in a public github repository.

1.4.2 Dataset of Crossmodal Sonifications

A dataset of crossmodal sonifications procedurally-generated by mapping attributes of avatar motion to parameters of a physically modeled clarinet instrument was recorded using UDKOSC and the OSC Toolkit. The dataset consists of video excerpts, audio recording and recorded data streams tracking multiple parameters of avatar motion in rendered space. The entire body of video and audio examples are available on the CCRMA website for researchers to use for futher study and analysis.

1.4.3 User Study of Perceived Crossmodal Coherence

A user study was carried out measuring the perceived coherence between attributes of motion tracked in video examples and a series of generated musical sonifications. Rankings of users' subjective assessment of the coherence or fit between crossmodal examples and their attributes was measured using the Bradley-Terry model for bimodal pairwise analysis. Additional consideration and analysis was given to the similarity perceived between crossmodal video pairings.

1.4.4 OSC Toolkit

The artistic and analytic work explored in this body of research for the most part takes place within a framework wherein parameters and control data are passed between software-based game-environments and sound engines using the Open Sound Control data protocol. To facillitate the generation, recording and processing of parameter data, a set of software tools were created. More information on each of these tools can be found in this document's Appendix B. The toolkit is made up of three primary helper applications:

OSCControl OSCControl is a ruby application that converts and outputs syntactically simple commands into properly formatted OSC messages or bundles. OSCControl currently builds OSC namespaces that correspond to the set of actor, camera and environment commands that UDKOSC is capable of understanding.

OSCRecorder OSCRecorder is a ruby script that listens to a given port and records incoming OSC messages as binary data using the YAML markup language.

OSCPlayer OSCPlayer is a ruby script that reads a YAML file recorded with OSCRecorder and plays back the timed OSC data, sending to a target IP address and port.

1.5 Outline

The research presented in this dissertation roams freely between the worlds of musical composition, game design and development, procedural audio and music, perception and auditory cognition.

In Chapter 2, an overview of gaming history and technologies, interactive audio within virtual environments and perceptual systems is presented with special attention given to games and interactive experiences that show tightly coupled cross modal interactions. Background material on the cognitive implications of measuring cross modal correspondence and perceived coherence between visual and auditory stimuli is also discussed.

Chapter 3 discusses and dissects a series of artistic works and software platforms built specifically to explore mapping schemata between virtual motion and gesture and musical sonification.

In Chapter 4 a dataset consisting of video examples showing crossmodal sonified avatar motions is described and discussed. Each example was prepared using UDKOSC and the OSC Toolkit.

Chapter 5 details a series of user studies designed to measure observers' subjective assessments of the relative perceived coherence between audio and visual elements of short video examples of sonified avatar motion. Carried out using the Amazon Mechanical Turk platform, data from user responses to a pairwise comparison task was analyzed using the Bradley-Terry statistical model. By examining participants' subjective assessments of perceived fit between the audio and visual modalities with special attention given to a set of low-level attributes making up each modal example, a ranked order of influence for individual and paired attributes is generated. Additional consideration is given to the perceived similarities between examples, as well as to the role of contour in perceived crossmodal fit.

In Chapter 6, conclusions reached for both the artistic and analytic components of this work are discussed, in addition to future work in both interest areas.

Chapter 2

Background

An exciting intersection sits between the areas of music, virtual visual environments, and interactive control systems that necessitates considerable attention from practitioners specializing in each of these fields. The perceptual concerns connecting physical and virtual space, both visual and auditory, can be approached from multiple angles and through multiple disciplines. Creative designers, artists and musicians can build interactive music systems and game environments that translate avatar motion and interaction into dynamic sound and music systems. Researchers can explore the cognitive connections between visual and auditory modalities, analyzing human perception through the lenses of crossmodal correlation and perceived coherence. This chapter aims to explore supporting material across these disparate fields, to help contextualize the creative and analytic approaches investigated in the rest of this dissertation.

2.1 Metaphor, Sound and Music

"You can place metaphor in a different space"

Muhal Richard Abrams

There is a human tendency to imagine and describe abstract thought and condition in terms of conceptual and linguistic metaphor [72]. The use of language and more specifically linguistic metaphor to describe our perception of attributes of sound and music is a common way that we communicate such abstract experiences, albeit one shaped strongly by our own cultural biases [26, 129]. It is through metaphor that we feel comfortable describing sensory percepts from one domain or modality in terms of another, using cognitive correspondences between the abstract and the familiar to communicate sensory percepts through language [73].

Johnson has proposed that the nature of such crossmodal metaphors is rooted in our own embodied physical patterns of experience, themselves made up of what he describes as “image schemata”, essentially our internal experiential building blocks linking figurative cognitive concept with more literal linguistic descriptors [66]. Image schemata can be thought of as supporting our internal structural representations of abstract concepts such as “near” vs “far”, a “part” of a “whole”, or even the base understanding of what “interaction” itself could mean. Thought of in this way image schemata form the basis of our very human ability to metaphorically conceptualize abstract or ephemeral systems, and allow us to communicate characteristics of such systems using our own language systems.

In Western tradition, there has been a strong crossmodal association between motion in space and specific characteristics of sound and music, for instance, vertical position as a metaphor for pitch or frequency. Following Lakoff and Turner’s principle of Invariance [117], we can look to the strong perceived correlation between the continuum that represents verticality and that which represents pitch to understand how and why these linguistic metaphors feel so appropriate and transparent. However when viewed across diverse cultures, this association breaks down. Image schemata favored by various ethnic and geographically-isolated populations use

quite different metaphors to represent musical properties, replacing “high/low” with “sharpness/heaviness” by the ancient Greeks, “small/large” in Java and Bali, and “young/old” by the Suyá of the Amazon river basin [128]. This suggests there is nothing intrinsic about any one metaphoric mapping, just a highly correlated association between our internal cognitive understandings or image schemata and the base attributes of music and sound to which we are perceiving and attending .

Challenges to an overly metaphorical interpretation of motion’s attributation to music can be found in ecological approaches derived from Gibson’s theories on affordance and direct perception [44, 46]. Clarke looks to musical articulations and attributes such as “attack point, timbre, dynamic and pitch” as holding perceptual cues, allowing auditory stimuli and sequence to specifying motion in a virtual space, putting forth the notion that “the perception of motion and gesture in music relies on the detection of motional and gestural invariants in sound sequences which may be quite poor approximations to their real-world counterparts” [24]. Such ecological views can be supported by the compositional grouping of musical sound and sound sources into ’virtual’ sound sources, an idea posited first by Stephen McAdams [86] and furthered by Bregman [18], capable of posessing identity and even intention.

2.2 Crossmodal Correspondence and Binding

The observation of correspondences between attributes of multiple modalities has long been an area of rich interest and exploration. Crossmodal correspondences have been observed between a number of sensory modalities including touch, sound, smell, vision and taste (for a survey of the literature see [110]). Specifically concerning sound and visual crossmodal correspondences, Köhler’s early observations on phonetic or sound symbolism showed that the rounded and angular linguistic characteristics from the nonsense words “Baluma” (or later “Maluma”) and “Takete” were reliably matched to visual shapes with respectively rounded and angular features [70, 71]. Early investigations of frequency have shown that higher-pitched sounds have been shown to reliably correspond to crossmodal stimuli ranging from elevated locations in space [90, 97, 103] to increased brightness [80, 123]. Increases in volume have been shown to

exhibit strong crossmodal correspondences with increases in brightness [15, 112]. And numerous studies using speeded classification methods have shown improved response times when audio-visual stimuli are presented that exhibit congruence or consistency with observed correspondences [81] for unisensory vs. bisensory stimuli [13], direct vs indirect tasks [33], and crossmodal similarity with metaphor [82].

Successful crossmodal bindings, that is the ability for events from different modalities to be seen as corresponding, have been shown to be linked to the perceived plausibility of events sharing common traits including spatial coincidence, temporal synchrony and gestalt-related principles such as common fate [111]. Such plausibility also is influenced by subjects' prior experiences with events sharing similar attributes from each modality [69, 68]. When visual gesture and auditory response share these and other associations that increase the perceived commonality between multimodal events, the chance for a perceptual crossmodal binding are greatly increased.

2.3 Perceptual Coherence

For crossmodal stimuli which are strongly perceived as belonging to or originating from a shared event, one hypothesis has been that human perception of individual modal events collapses into a single crossmodal percept [68, 69]. The strong crossmodal binding effectively results in a joint perception across modalities, creating a single perceptual event that can be said to be *perceptually coherent*. In defining perceptual coherence, Handel states that "perceiving in all sensory domains is finding structure in the energy flux and that deriving equivalences among the domains can deepen our understanding of how we create the external world"[55]. Such a view is complemented by the self-organizing operational principles defined in Gestalt psychology, particularly in reference to principles such as good continuation, common fate, proximity and reification, as well as a conceptual nod towards psychophysical isomorphism, where correlations are drawn between experiential and mental activities [74].

Coherence can perhaps also be understood as the extent to which the stimuli at hand fit existing perceptual models for similar interactions and their resultant generated sound. Gaver discusses the methods by which we perceive auditory events within familiar ecologies, that is within the reality in which we commonly exist [43]. An understood sound event generated from an understood ecological interaction (such as a stick striking a hollow drum) could be described as being strongly coherent in that the resultant sound artifact was in line with an expected result. If the environment within which visual action and auditory response is generated distorts or even completely disregards the fundamental principles upon which our understanding of crossmodal relationships is based, then existing ecological models can't accurately predict the sonic outcome. Without access to a reliable model of sonic interactions for a novel perceptual environment, the perceived coherence between action and sonic result will themselves be calculated on-the-fly, perhaps even conflicting with expected results from prior experiences. In this case, coherence itself may serve as the primary measure of how and where users should attend to and understand crossmodal audio-visual gestures.

The research and artistic practices discussed in this dissertation are primarily concerned with the perception of correspondences which are created as coupled sets, mapping parameters from one modality to those of another, with the intention of being perceived as crossmodal coherent events. When stimuli are presented with the intent of creating strong crossmodal bindings, one measure of the success or failure of individual data mappings can be measured by the amount of coherence or *fit* which is perceived by subjects or audiences. If a given crossmodal binding is perceived as highly coherent in the context of a given artistic practice, that provides a significant understanding that can influence the creation of future mapping schemata.

2.4 Gesture and Motion

The role of gesture in musical performance can be both structural and artistic. In traditional musical performance, physical gesture has been necessary to generate sufficient energy to put instrumental resonating systems into motion. Musical performers and dancers at the same time convey intention, performative nuance and expressivity through their gestures, shaping the performance experience for performers and audiences alike by communicating without language [61, 28, 65].

The influence of visual gesure and the extent to which gesture could influence the perceived duration of marimba notes by audience members was the focus of a key series of investigations by Schutz and Lipscomb [106, 105]. An audio-visual dataset of "long" and "short" notes as performed by a professional marimba player was recorded. By presenting the audio and video components to subjects, both in their original form as well as with inverted audio-visual pairings, they showed that longer performative visual gestures increased the perceived length of musical notes. These results strongly indicate that gesture from the visual modality had a direct influence on the perceived duration of events in the auditory modality.

Recent work by Eitan and Granot explores the formation of audio-visual cross-modal bindings linking motion in space and auditory stimuli by requiring subjects to visualize sequences of directional motion for imaginary cartoon-like avatars upon hearing melodic auditory stimuli [32]. Results from these experiments suggest directional asymmetries for a series of musical parameters including dynamic, pitch and tempo changes. In one study, dimenuendo exhibited a significant association with motion in the vertical dimension, but the inverse or crescendo interestingly did not. In another study, crescendi manifested imagery increasing in speed, while diminuendi did not manifest the inverse, that is, imagery of decreasing speed. And while pitch decreases strongly suggested descending vertical motion, increases in pitch did not suggest strong ascending vertical motion.

2.5 Presence, Immersion and Virtual Space

The simulation and control of motion and action in rendered computer space have become common experiences in modern computing. Powerful graphics processors can render gorgeously textured immersive environments at high frame-rates while allowing dynamic motion and real-time exploration without noticable artifact or loss of quality. Fluid and intuitive control schemata, designed and refined in the commercial gaming marketplace, allow users to perform intricate and subtle motions and actions in virtual space. The human experience, one with which we are all intimately familiar, can be extended or co-opted in such environments, allowing us to engage new realities with new capabilites and technologically-enabled senses in ways never before experienced.

One result of such technological advances is that computer-rendered virtual spaces are capable of strongly influencing users' cognitive sense of presence to the point where user reaction and sense of self can be said to exhibit strong immersion [108, 124]. For virtual space and environment, an oft-cited measure of effectiveness or goal in and of itself can be seen in how immersive an experience is and how fully a user's sense of space and place or presence can be co-opted [125]. Witmer further defines presence as "a psychological state characterized by perceiving oneself to be enveloped by, included in, and interacting with an environment that provides a continuous stream of stimuli and experiences." User engagement or involvement in virtual environments and game-based activity is often experienced at quite a high or immersive level [59], potentially suggesting strong correlations between engagement levels and states of presence or immersion within both musical and virtual systems.

While initial discussions of factors contributing to successful immersion or telepresence were first introduced in the context of video-aided tele-operation of robotic machinery [89] the quality of visual representation in modern virtual environments has reached a level where immersive behaviors in users of virtual environments can be seen as equal to if not stronger than any such behaviors in operators of physical teleoperation systems [125], with a measurable relation to quality of visual projection [58], field of vision [98] and intent or meaningfully-arranged stimuli [59]. In the cases of gaming or interactive media systems, it has been argued that attending to

generated virtual environments with an engaging storyline or task-based interaction model requiring cognitive involvement actually heightens a sense of immersion or telepresence. Loomis attributes such engagement naturally to human tendencies towards distal attribution of presence, stating "most of our perceptual experience, though originating with stimulation of our sense organs, is referred to external space beyond the limits of the sensory organs." [76].

2.6 Computer Gaming and Music

The earliest video gaming systems trace their evolutions back to military defense systems of the 1940s, specifically to the introduction of *Spacewar!*, a space-combat simulation developed at the Massachusetts Institute of Technology for the DEC PDP-1 computer. The subsequent widespread cultural adoption of computer-based gaming systems began in earnest in the early 1970s with the introduction of the *Galaxy Game* system at Stanford University and the commercial success of *Computer Space* by Atari founders Nolan Bushnell and Ted Dabney [79]. Since that time, the use of gaming technologies has spawned a massive commercial industry with a market worth greater than $90 billion in 2014 [14] with an estimated 58% of Americans reported as playing video games [7].

Virtual environments and the ecosystems of action and reaction developed to function within such environments are rich and complex. These dynamic systems are equally capable of creating realistic simulations of existing physical location and modes of interaction or fantastical interpretations of space and a user's interaction with that space. Contextual sound is commonly used to heighten a users situational awareness, to convey direct verbal meaning or to reinforce or suggest visual cues and stimuli [87]. It can be argued that the role of sound and music systems in modern commercial and research-grade virtual environments - task or narrative-based simulations or games - tend to be supportive in nature. When compared to current state-of-the-art graphics rendering and visual presentation systems as a whole, it can be said that most gaming systems offer comparatively limited methods for generating and manipulating sound. That being the case, sound and music are often relegated

to the goal of reinforcing actions and atmospheres presented by a system's visual projection.

Balancing creative intention with hardware and storage constraints, for sound artists working with commercial gaming systems the dominant sound generation techniques have predominantly been based around pre-recorded sample playback, with comparatively simple real-time processing techniques available in-engine such as artificial reverberation and filtering [95]. While fully interactive and real-time-based generative sound synthesis has yet to become a common methodology in computer game sound and music, both the exporting of in-game gesture and motion to be used as dynamic generative control data for the purpose of driving real-time testing of soundscapes [42], and audio effects [34] as well as real-time sound generation with interactive computer-music languages for commercial entertainment titles have been pursued in a number of commercial game titles [54, 120, 29].

2.7 Procedural Audio and Music

The use of procedural techniques linking game data directly to the generation of sound and music systems in games and virtual environments is not new. In the early days of computer gaming, before computer memory space was large enough to allow for the storage and playback of large pre-recorded audio files and samples, the majority of sound effects and game soundtracks were commonly synthesized in real time using Programmable Sound Generators (PSGs) driving simple synthesis processes [25]. As gaming systems became larger and more complex, the role of sound and music generation shifted significantly away from real-time data-driven synthesis and control towards pre-recorded music and sound, more in the style of a motion picture.

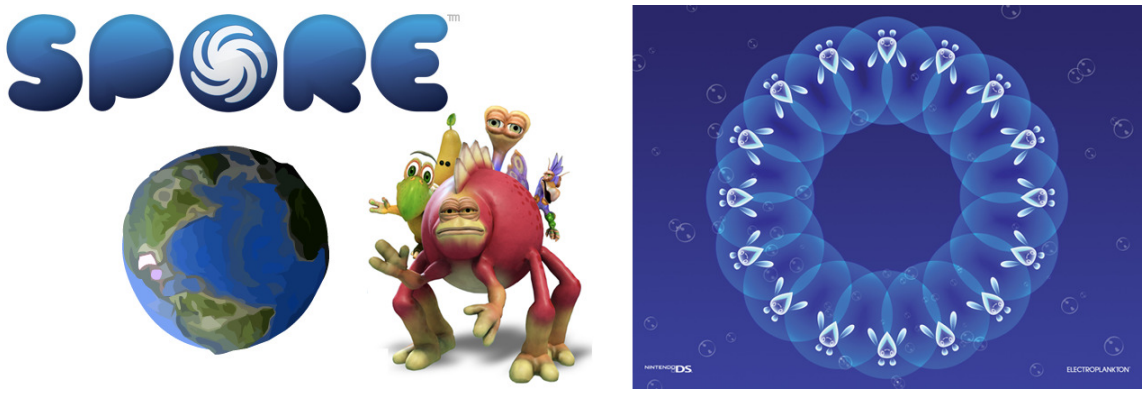


Figure 2.1: Commercial games like EA's *Spore* and Nintendo's *Electroplankton* showcased generative music systems, mapping parameters of in-game action to musical attributes such as melody and rhythm.

Andy Farnell described this paradigm shift in gaming audio and music as such:

> These computers seemed alive. There was such low latency and responce from the machine, and the sounds that they made were generated synthetically. There was a tight coupling... it was like an instrument. I think sound samples came along and sound became dead. And it stayed dead for a long time. [35]

While many commercial video games embraced the use of sound samples and more static soundtracks, the growth of a more casual gaming practice - featuring less complex game systems often based around portable devices like the Nintendo DS gaming system or mobile devices - often showcased innovative musical game systems. Designer Toshio Iwai created *Electroplankton* in 2005 with a dynamic musical system that was tightly integrated into the game itself [64]. The more recent introduction of projects such as libPD [19], a wrapper for Pure Data enabling its use as an embedded audio engine, the use of process-driven audio and music in commercial gaming systems has increased. The high-profile game *Spore* used Pure Data to drive procedural music for its in-game character creator, mapping user action to parameters influencing the musical score in real-time [29].

2.8 Interactive Sound and Music

There exist an abundance of interactive control systems and methodologies that can be used to map physical motion and gesture to sound and music-generating process, ranging from Max Mathews' early work with the GROOVE ("General Real-time Output Operations on Voltage-controlled Equipment") system [85], the Radio Baton and the Conductor program [85, 84, 16], to more recent commodity interfaces such as Nintendo's Wiimote [91, 17] and Microsoft's Kinect motion-tracking system [107]. Such systems can employ direct mapping schema like traditional instruments, mapping user-directed motion directly to sound generating process. However unlike traditional instruments, such methodologies are capable of abstracting control data across multiple parameters of a given instrument, driving complex parameter sets and non-linear dynamic processes simultaneously across multiple systems. Computer-mediated control systems can span the continuum from wholly deterministic instrumental control to completely generative or process-driven sound and music, connecting any aspect of user motion to virtually any aspects of sound and music generation and control.

2.9 Networked Musical Environments

The use of networked/multi-user video game engines for music and sound generation has become increasingly common as generations of musicians who have grown up with readily accessible home video game systems, internet access and personal computers seek to bring together visually immersive graphical game-worlds, wide-spanning networks and interactive control systems with musical systems. Though its graphical display is rendered in two dimensions, *Small_Fish* by Kiyoshi Furukawa, Masaki Fujihata and Wolfgang Muench [39] is a game-like musical interface which allows performers/players to create rich musical tapestries using a variety of control methods. *Auracle* [37], allows networked users to collaborate and improvise using vocal gesture. And *Fijuu2* [96] created a fully rendered three-dimensional instrument that users controlled with a standard game-pad, to tightly marry idomatic game controls to instrumental musical output.

While the transmission of music across communication systems has only become truly ubiquitous within the last few decades, the history of network-based musical transmission dates back to the end of the previous century with the introduction of Gray's "Musical Telegraph" and Cahill's "Telharmonium" [23]. While both instruments were designed to generate electronic sound and to transmit that sound over telephone wires, the Telharmonium was established as perhaps the world's first networked musical subscription service, selling musical performances transmitted in real-time across telephone wires to paying establishments. The ability to create sound in one location and broadcast it for performance in another location is a uni-directional method of communication; there is no musical interplay or bi-directionality involved, relegating any audience member to the role of passive listener. Musical networks of particular interest in the context of this dissertation can be defined as necessarily bi or poly-directional connected musical streams each representing one or more voices of live musical performance data; networks can exist within a single physical location on a local network or in discrete and disparate physical locations and spaces, making use of both commercial and research-grade internet access. As such, there currently exists an increasingly well-populated community of network musicians and an evolving performance practice for networked ensemble performance.

Early networked performances by The Hub [47] stand out as rich examples of the complex musical constructs formed through communal composition, improvisation and real-time performance. Stanford's SoundWIRE group [109] utilizes multiple channels of uncompressed streaming audio over its JackTrip software [20] to superimpose performance ensembles and spaces alike, with performances of note including networked concert performances with the ensembles Tintinnabulate at RPI (NY) and VistaMuse at UCSD, as well as with performers at Beijing University in 2008's "Pacific Rim of Wire" concert [21]. Both the Princeton Soundlab's Princeton Laptop Orchestra (PLOrk) [115] as well as the Stanford Laptop Orchestra (SLOrk) [122] have displayed the powerful possibilities of collaborative networked compositional form using local area networks for synchronization and communication and distributed point-source spatialization.

2.10 Virtual Space and the Representation of Sound

In this research, interactive systems promoting a combination of first and third-person views of virtual space have been primarily utilized, as engagement and immersive cognitive behaviors in first-person based visual and auditory systems have proven to be more prevalent than in more abstract visual systems [125]. In such systems, in which a user's self is represented virtually in the form of an avatar capable of interacting with virtual constructs and actions, the focus of both visual and auditory representations presented to each user has commonly been user-centric. A user is presented with an illusory visual and sonic representation of the virtual environment from the viewpoint of that user's avatar, complete with signal processing designed to strengthen the related illusions of space, distance and motion. As users listen to audio output through headphones, stereo speakers, or an industry-standardized multi-channel configuration such as 5.1 or 8.1, all audio processing done in game-engines attempts to create realistic illusions of motion for one user sitting in the sound-system's centralized "sweet-spot". Such individualistic and user-centric presentation by its very nature restricts the communal sensory experience fostered in the virtual environment from existing anywhere except within the game-world itself.

In a context where live musical presentations are attended by a viewing and listening audience, a user-centric approach to video and audio presentation proves difficult, as a system optimized towards a single point of attention is suddenly forced to deliver visual and audio cues to a group spread within a viewing and listening space. By inverting these traditional models of sound-presentation and by focusing on a space-centric model of sound projection for game-environments, a communal listening experience can be fostered inclusive of all listeners within a shared physical space, including game-users and audience members alike. Of particular interest in the context of this research are environments and technologies sufficiently capable of rendering high-quality visual realizations of virtual space with enough clarity of presentation and consistency of detail to immerse and engage users' concept and identity of presence while at the same time affording expressive and varied virtual motion and

gesture. With these criteria in mind, three virtual environments have been to date leveraged and customized in the context of this research: ioquake3 (an open-source branch of the Quake III gaming engine) [4, 62], the Sirikata extensible virtual world [60], and the Unreal Development Kit [40].

2.11 Summary

This chapter has provided a review of appropriate background material supporting the cognitive, musical and technological areas of interest explored in the context of this dissertation. The role of metaphor, crossmodal correlation and perceptual coherence in the human understanding of causal multimodal events has been discussed, as have representative examples from the histories of interactive and networked music systems, as well as procedural music and computer gaming.

Chapter 3

Music in Virtual Worlds

There have traditionally been necessary synergies between motion and action in space and the production and manipulation of musical sound. For most pre-digital musical systems, physical activation was an inherent component of instrumental performance practice. The introduction of computer-based musical systems has removed the necessity of such couplings, allowing abstracted data-analysis or algorithmic process to both instigate and manipulate musical output. However artists seeking to retain some level of control within these digital contexts often develop and employ mapping schemata linking real-time control data to parameters of musical form and function. Such mappings provide an interface between human intention and digital process and can range from the simple to the complex, from the distinct to the abstract.

Within virtual game worlds or interactive social environments - complex audio-visual rendered spaces ranging from fast-paced game engines to expansive platforms for e-commerce and networked interaction - the same data flows that drive visual presentation and user interaction can be tightly integrated with sound and music systems. This chapter outlines a series of software engines and multimodal musical works designed to facillitate the procedural generation of sound and music by mapping parameters of action and motion to sound-generating processes.

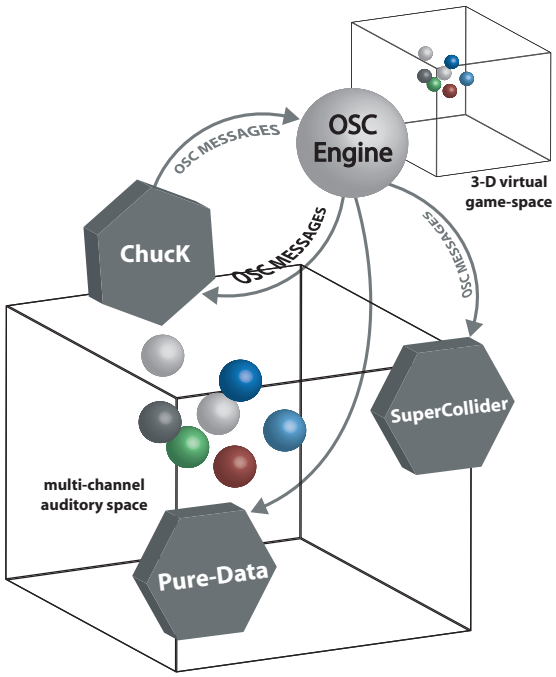


Figure 3.1: Control data from gaming environments can be passed from one computer process to another, encapsulated as UDP packets within an Open Sound Control (OSC) namespace. For systems such as q3osc and UDKOSC, the Oscpack OSC implementation was used to pass data from one IP address to another, sonifying avatar actions and motions in virtual space with sound engines written in ChucK and Supercollider.

3.1 Early Procedural Musical Environments

In the context of real-time interactive musical realization and performance, the ability to render high-quality musical sound that is dynamic and potentially timbrally and temporally complex is crucial. Overlooking the hardware and software constraints faced by current commercially-shipping game and entertainment products, powerful and flexible systems can be built using available technologies and communication protocols. For much of the research and creative practice discussed in this dissertation, control parameters extracted from game engines using the Open Sound Control protocol are transmitted encapsulated as UDP packets over wired or wireless networks to software-based sound servers, generating and spatializing audio and music based on the confluence of composer intent and user-controlled virtual motion and gesture.

3.1.1 q3apd

q3apd [94, 22] was a modification to the open-source 1.32 Quake 3 gaming engine which streamed user coordinate, view and player data from the game engine into Pure Data (PD) [99] using PD's internal FUDI protocol, an OSC-like string protocol used by PD to link GUI and DSP systems. Oliver describes q3apd as "an experimental work that takes the events and spatial dynamics of combat and uses this information to make live computer music." [93] By tracking four in-game combatants controlled by artificially intelligent processes and mapping parameters of movement, location, actor health, view angle and item status to sound generating processes in PD, q3apd sought to create "an auralisation of activity in the arena: a musical re-presentation of the flows and gestures of artificial life in combat." [94]

q3apd was presented as an installation in which the Quake 3 artificial intelligence or "bot" algorithms were the sole source of control. The sonified battle raged for two weeks without human control or interference during a 2006 exhibition at the Lovebytes festival in Sheffield, UK [6]). The work's compositional structure and orchestration were directly tied to the architecture of the game environment: "For this reason game-objects and architectural elements were carefully positioned so that the flow of combat would produce common points of return (phrases) and the orchestration sounded right overall." [94]

3.1.2 *Maps and Legends*

For the 2008 musical work *Maps and Legends* [48], the q3apd codebase was coupled with Pure Data to track user motion around specially-designed compositional maps. In-game actors controlled by human performers followed a bright yellow pathway deliniated by large yellow directional arrows, triggering modular components of a multi-channel composition by entering pre-determined trigger zones. Pre-composed computer-generated musical cells were assigned to each in-game performer and were triggered, processed and controlled through performers' motion through the rendered environment. Eight large speaker representations were arranged around the periphery of the square compositional map for *Maps and Legends*, matching speaker locations

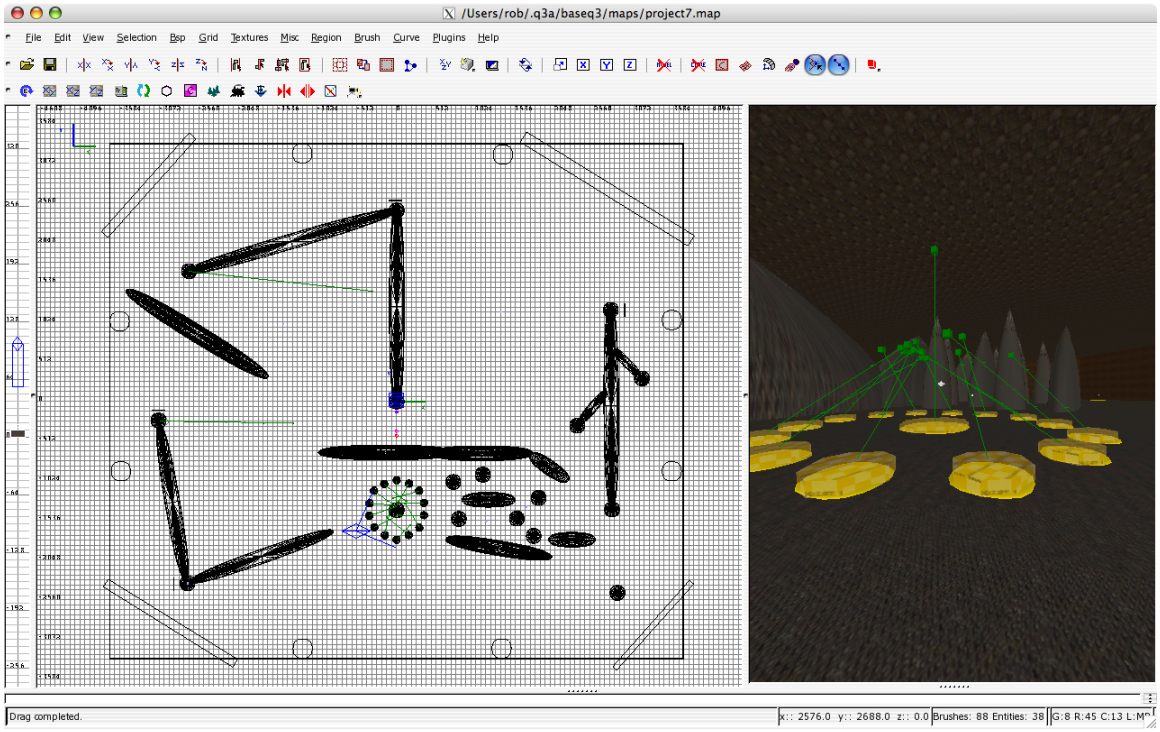


Figure 3.2: The map created for 2008's *Maps and Legends* featured bright yellow pathways, guiding actors around the rendered space. To explore the vertical dimension, a series of circular jump-pads were placed in the level, which would launch actors upwards when stepped on. The map for *Maps and Legends* was created using the GtkRadiant open-source .bsp building tool.

in the horizontal plane of CCRMA's listening room[1]. By correlating avatar location with respect to each of eight speaker representations to the amplitude of triggered musical events in an eight-channel surround soundfield, an overlay of the compositional map was created in physical space, bringing together the experience of action in two environments.

Maps and Legends was designed as a fully-rendered three-dimensional compositional map built using the GtkRadiant game-level editing software [1]. While clearly-visible pathways, directional arrows and active directional jump-pads were built into the map to encourage or force performer motion in certain predefined or composed

[1]The Listening Room is a sound studio located in CCRMA's facility at Stanford University. It is a quiet heptagonal listening space surrounded by a three-dimensional multi-channel speaker array.

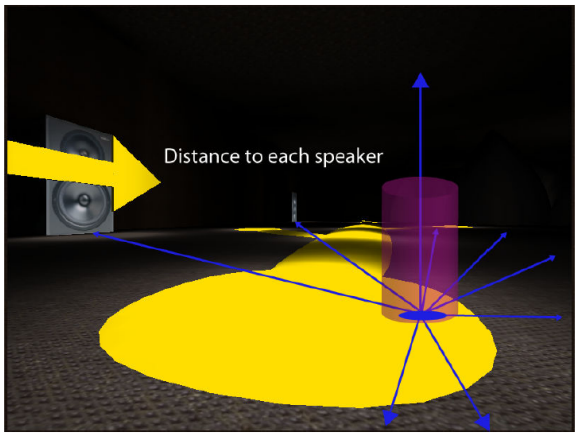


Figure 3.3: The Euclidean distance between the avatars in *Maps and Legends* and each of eight speaker representations was tracked in real-time and used to control the amplitude of each performer's triggered soundfiles. As avatars moved closer to each speaker construct, their associated soundfiles were panned to the corresponding speaker in the eight-channel soundfiled.

directions, each performer retained a high-level of independence and improvisatory flexibility, allowing for spontaneous new interpretations of the pre-composed materials. Users running the game-client software and connected over a standard high-speed network controlled their avatars - and through them the musical work - using a combination of computer-keyboard controls for motion and a mouse for view-angle.

Multiple game-clients could connect to a single host game-server which in turn streamed FUDI-formatted data reflecting each player's actions and coordinates to a sound-server in the performance venue running Pure Data. Sound generation and processing for each independent performer were handled by and spatialized within an 8-channel PD patch, circumventing the sound-system of the game itself and substituting the composer's musical environment for Quake III's stock in-game sounds and music.

Basic control values supplied by q3apd such as player-motion and XYZ position, were used to calculate constantly distance vectors from a performer's current position to pre-defined virtual speaker locations within the compositional map. Similarly, distance between multiple performers was calculated and mapped to sound events - complementary or disruptive depending on composer-defined performance states

- in an effort to support or discourage performers from moving too close or too far from one another. To clarify the relationship between individual performers and their respective sound sets, triggered sound files were spatialized based on a performer's distance from each of eight defined speaker locations (see Figure 3.3). Sounds followed the performers' motions through space, spatialized between multiple speakers at any given time. Speaker locations were defined in PD as XY coordinate pairs, with independent gain levels for each performer/speaker pairing determined by a simple distance function. In this manner, multiple speaker configurations for multi-channel output were easily configured without any changes to the compositional map itself.

3.1.3 q3osc

q3osc [49] is a heavily modified version of the open-sourced ioquake3 gaming engine featuring an integrated Oscpack [12] implementation of Open Sound Control for bi-directional communication between a game server and one or more external audio servers. By combining ioquake3's internal physics engine and robust multiplayer network code with a full-featured OSC packet manipulation library, the virtual actions and motions of game clients and previously one-dimensional in-game weapon projectiles can be repurposed as independent and behavior-driven OSC emitting sound-objects for real-time networked performance and spatialization within a multi-channel audio environment. This section details the technical and aesthetic decisions made during the development and initial implementations of q3osc and introduces specific mapping and spatialization paradigms used for sonification.

q3osc is a musical and visual performance environment which makes use of these paradigms of communal networked conflict and collaboration to re-purpose virtual entities and traditional game control methodologies into agents and actions capable of affecting musical response in physical auditory space. By extending the source-code of a multi-user video game engine to include Open Sound Control input and output libraries, data from within a game-engine's virtual environment can be encapsulated and transmitted to one or more external sound-servers with the intent of sonifying virtual gesture and motion into a musically rich and satisfying aural experience. In

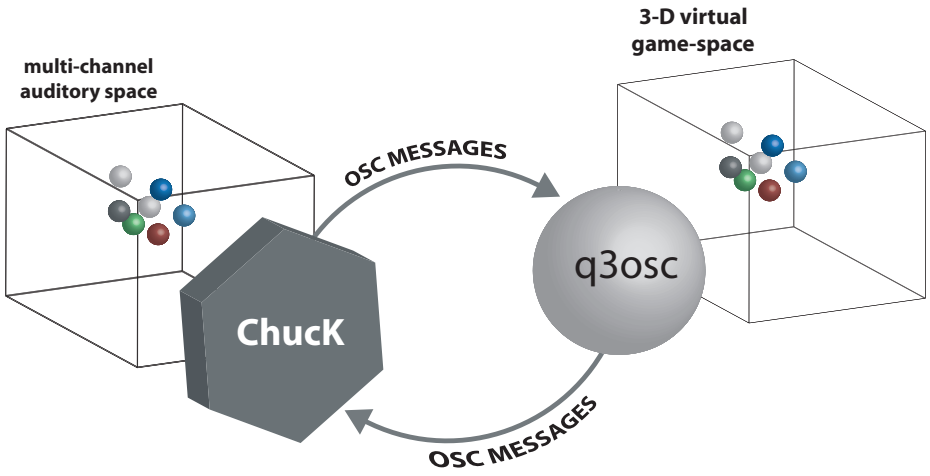


Figure 3.4: For q3osc, the location of projectiles moving in game-space was sent over OSC to a ChucK sound engine, where the impact and location of each projectile was sonified using a tuned filter. OSC messages could also be passed into the game-engine from ChucK, controlling environment parameters such as gravity and game speed.

turn, data generated by external performative controllers, potentially extracted from the auditory environment or generated using algorithmic processes can be routed back into the virtual environment as control data for the modification of environment parameters and the control of individual game entities.

Custom modifications made to traditional game elements such as the targeting methods of "weapon" projectiles and global world-level variables such as gravity and avatar speed allow for striking and dynamic change in the visual and performative characteristics of both client behaviors and the virtual environment itself. By empowering game-players with the ability through gesture and motion to generate complex evolving visual and musical structures, relatively simple modifications in pre-existing game code can have profound and far-reaching musical and performative effects.

q3osc was initially used to drive sound in a three-dimensional model of the CCRMA Listening Room (see Figure 3.5). Sonified projectiles moving around the rendered space were spatialized according to their position in the environment. Pitched sound events for each projectile were positioned around the Listening Room's speaker field using a simple power panning algorithm. The crossmodal mapping between

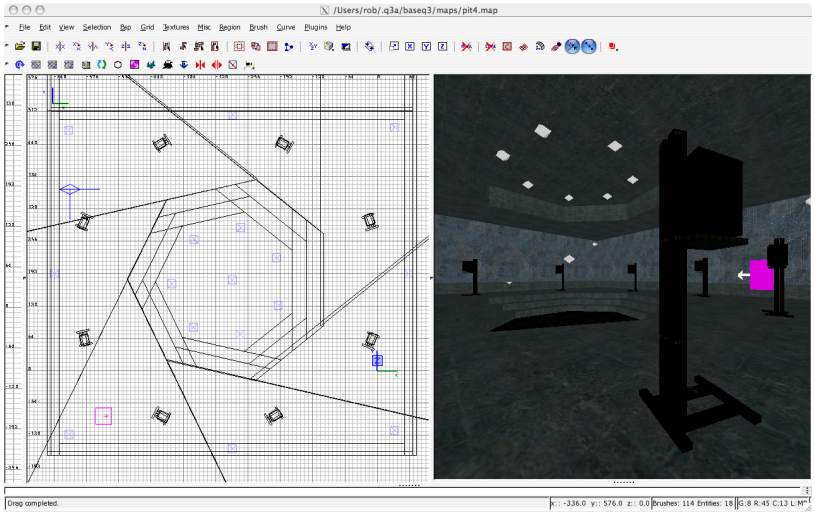


Figure 3.5: To test the panning of sound sources around the CCRMA Listening Room, a model of the heptagonal space was created in GtkRadiant with the position of each speaker construct mapped to the location of the actual speakers in the physical room.

matched environments (physical and virtual) was reported to create feelings of disorientation amongst listeners and observers, perhaps due to the strangeness of separating the visual and auditory perspectives for each event.

3.1.4 *nous sommes tous Fernando...*

Premiered in May, 2008 by the Stanford Laptop Orchestra, the ensemble piece *nous sommes tous Fernando...* [50] is a flexible performance environment comprised of a series of richly textured performance maps for laptop ensembles of various sizes ranging from 5 virtual speaker locations to twenty. In its most basic form, the map for *nous sommes tous Fernando...* utilizes a series of five hemispherical-shaped objects placed on the floor in a cross-pattern in virtual space. Laptop-stations and connected hemispheres are placed in an analagous pattern in the performance space with a subwoofer connected to the center speaker and performers sitting at each point of the cross. Audience members sitting around and within the speaker configuration watch the output from one laptop connected to a video projector, acting as a virtual camera.

In SLOrk performances, OSC messages carrying position data for projectiles fired by each performer are sent from the Linux game server to each of the Mac OS X client machines. The bounces of in-game projectiles are then sonified across the multi-hemisphere soundfield using the ChucK programming language [119], correlating bounce positions in the virtual environment with scaled amplitudes across the physical speaker layout using a simple distance function. Performers sit at each speaker station with additional performers as desired connecting via client machines not necessarily connected to sound-servers. Frequency and resonance coefficients for a dynamically allocated array of tuned filters are mapped to additional distance and coordinate data, creating a dynamically shifting sound world controlled by the projectile firings of in-game performers.

Performers in *nous sommes tous Fernando...* control green lizard avatars, moving through an environment where all surfaces save the floor are made up of blocks of random angles and sizes. In this manner, any projectile bouncing off the walls or ceiling will be reflected in a completely unpredictable manner, minimizing the chances of simple periodic or infinitely repetitive note sequences. Each performer is allowed to affect changes on a given set of environmental parameters such as each client's speed of motion, the gravity of the environment and the speed of fired projectiles. Additionally all performers can control whether all projectiles in the map persist or whether they are destroyed. And while *nous sommes tous Fernando...* is at heart an improvisatory work, the virtual camera operator also plays the role of conductor, typing messages to performers in the game engine while attempting to shape the ensemble performance into a coherent form.

3.1.5 Sirikata

Sirikata [60] is an open source platform designed for the creation of multi-user games and virtual worlds designed and developed at Stanford University in the Department of Computer Science. To showcase the Sirikata technology, an evening-length concert event [53] was designed by a cross-disciplinary team including researchers from Computer Science, the Stanford Humanities Lab and Stanford's Center for Computer

Figure 3.6: In the SLOrk work *nous sommes tous Fernando...,* members of the laptop orchestra controlled lizards moving through space, firing sonified projectiles at hemisphere-shaped speaker constructs. The physical location of the orchestra's actual speakers was mapped to the relative location of the in-game speaker constructs.

Research in Music and Acoustics (CCRMA). The result of this collaboration was a series of evening-long concert events at the 2009 Milano-Torino (MiTo) festival in Milan, Italy. All virtual environments created for the *Due Serrata in Sirikata* concerts were created by digital artist Chris Platz.

To integrate Sirikata with sound servers written in music programming languages like ChucK and Supercollider [88], an implementation of Open Sound Control was compiled into the Sirikata codebase. Custom control and object interactions, as well as multi-user networking and a basic physics engine were implemented, allowing client avatars to interact with each other and the environment itself. Users within the Sirikata engine could move through the created environments, controlling parameters of sound generation and spatialization through their avatar's motion in space or by simply interacting with specific objects in the environment itself.

3.1.5.1 *Dei Due Mondi*

In the work *Dei Due Mondi* by Robert Hamilton and Juan-Pablo Caceres, eight Sirikata performers act as spatializing agents for an eight-channel pre-composed fixed-media composition, with avatar location in virtual space mapped to a panning location in listening space around the audience. Rendered environments representing the Milano concert hall and the connected concert space in California were displayed, with

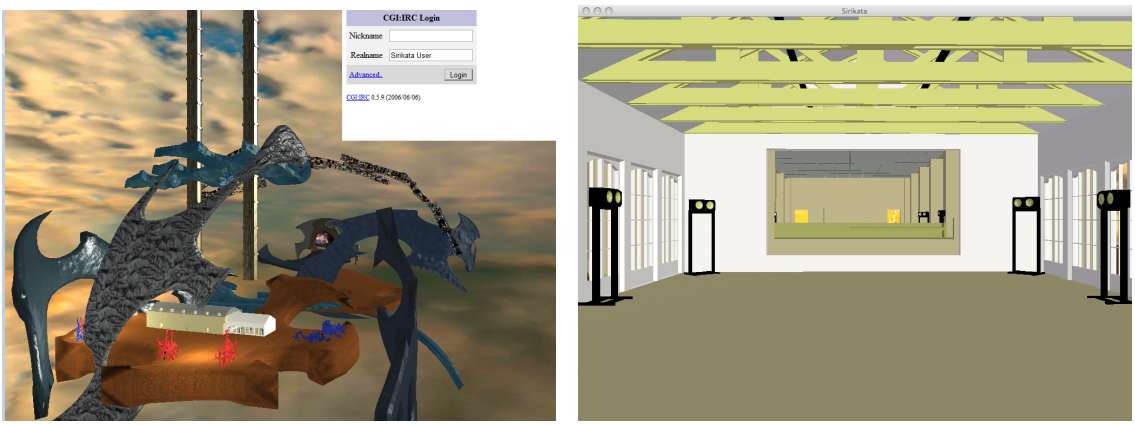


Figure 3.7: Within the Sirikata environment used in *Dei Due Mondi*, replicas of both the CCRMA Stage and the performance space in the Polytechnica de Bovisa in Milano were explored by performers located in both physical locations. Speaker constructs in each of the realistic room models were designed to be mapped to the physical speakers in both halls, panning each performers' soundfiles to the closest speaker.

avatars freely able to move between the two spaces, bringing their sound with them. Virtual rendered speaker images within both the Milan and the Stanford environments were designed to match the physical speaker layout in each concert hall. Performers created patterns of motion throughout each concert space, effectively traveling between the two spaces through their virtual representation. As user avatars move through three-dimensional coordinate space, each avatar's X, Y and Z location data is streamed over OSC to a sound-server. Performers following a loosely-arranged choreography of motion control amplitude levels and sounding direction of their assigned part. An additional set of speaker locations, spread "outside" the representations of the concert halls in a fantastical world was used as well.

In this manner, performers could control sound with a direct mapping (within their own relative concert space), with a remote mapping (within the other concert hall) or with an abstract mapping.

Figure 3.8: For the MiTo performance of *In C,* performers in Milan and in California moved their avatars down three musical paths. As the avatars moved over each section of the path, physically modeled STK instruments played cells from Riley's composition, moving through all 53 cells in order.

3.1.5.2 *In C*

Terry Riley's *In C*, a metronomic piece composed for an indeterminate and unspecified number of musicians, presents a unique set of challenges in the context of distributed performance [**?**]. The pulse of the piece is calculated and distributed to match network delay between connected sites, ensuring that the rhythmic patterns that comprise the piece are tightly synchronized, albeit shifted a number of metric beats, ensuring the musical result is different at each end of the performance [21].

For a 2009 performance of *In C* at the Settembre Musica/Milano-Torino (MiTo) Festival [2], the ensemble was distributed between Milan, Stanford California and Missoula Montana. While previous performances have included traditional instruments and laptop performers, this new rendition added virtual-world avatar performers with custom visual art and a novel method of moving one's instrument through Riley's composed musical pathways. The orchestra consisted of piano and avatar/laptop performers in Milan, avatar/laptop performers and violin in Stanford and Celleto and Electric Violin in Missoula. In keeping with earlier networked performances of *In C*, the ensemble was synchronized through a network-metronome generating audio and OSC messages.

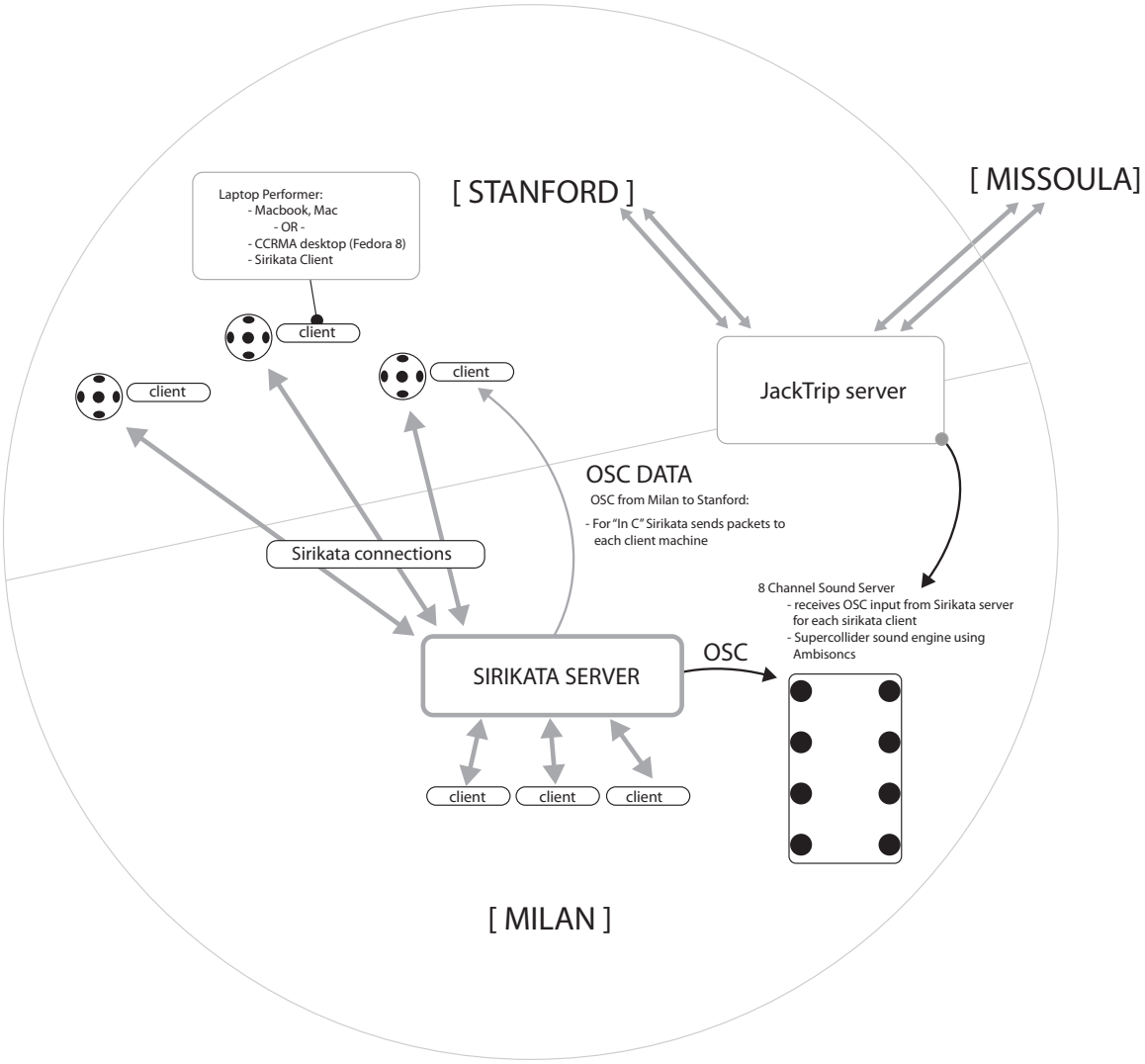


Figure 3.9: A complex networking scheme was necessary to link the audience and performers in Milan with performers in California and Montana. Jack Trip connections routed real-time audio between locations while OSC data and Sirikata networking packets were sent over UDP.

Figure 3.10: Pianist Chrysi Nanou interacts in real-time with an avatar during a performance of *Canned Bits Mechanics.* The work functioned as a duet between virtual and live musicians.

3.1.5.3 *Canned Bits Mechanics*

Canned Bits Mechanics by Juan-Pablo Caceres and Robert Hamilton, is a performance work for piano, virtual performer and networked player-pianos inspired in part by the 100-year anniversary of Marinetti's 1909 publication of the Futurist Manifesto [78]. In keeping with the turn-of-the-century futurist aesthetic, the visual and sonic components of the piece are metallic and mechanical, including recordings of prepared player-pianos and piano rolls. In addition to a local pianist performing from Milan, two remote Yamaha Disklaviers located in California were controlled using interactive real-time algorithms from the concert stage in Milan. Avatar motion and speed were mapped to soundfile playback speed, amplitude and direction, allowing the virtual performer to control a wide variety of sounds through rapid motion. The aural results depended on the speed of motion of the avatar and its position, giving great flexibility to interact with the other performers. Flying gestures created simultaneous visual and aural sweeps, creating an engaging experience for performer and audience alike.

Figure 3.11: The virtual enviroment built for *Dialoghi* showed elements from *In C, Dei Due Mondi* and *Canned Bits Mechanics* scattered around a field of space debris. As an avatar moved through the space, triggering sounds from each piece, performers connected by Jack Trip improvised over the network.

3.1.5.4 *Dialoghi*

These same sample-based motion gestures were used in *Dialoghi* a networked improvisation combining four remote instrumental performers, the local piano and Sirikata avatar performer. Designed as a culminating piece for the concert, the visual setting consisted of "cosmic debris" from each the previous pieces in the concert (*In C*, *Canned Bits Mechanics*, *Dei Due Mondi*) serving as a visual and musical deconstruction of the event itself. To musically illustrate this deconstruction the avatar's speed and direction of flight controlled altered samples selected from *In C, Canned Bits Mechanics* and *Dei Due Mondi* that had been heard in the previous three pieces of the concert.

3.2 UDKOSC

Introduced in 2011, UDKOSC is a modification to the Unreal Development Kit (UDK) gaming engine featuring OSC input and output streams for the controlling, tracking and sonification of in-game actions and motions. Customized game functionalities built into UDKOSC are applicable to modalities including real-time musical performance and composition, dynamic local and networked performance, procedural

music/audio and rapid-prototyping of interactive game-sound design. UDKOSC was designed to support the creation of immersive mixed-reality musical performance spaces as well as to serve as a rapid prototyping workflow tool for procedural/adaptive game audio professionals [95][118]. Data points tracked in UDKOSC include actor, projectile and static mesh coordinate positions in world-space, in-game events such as collision, and real-time ray tracing from player actors to identify interaction with specific object types and classes.

3.2.1 Unreal Development Kit (UDK)

The Unreal Development Kit (UDK) [9] is a next-generation professional development framework for creating visually rich networked gaming and simulation environments. Built upon the Unreal Engine 3 - the engine powering current commercial gaming titles such as *Gears of War*, *Bioshock* and *Unreal Tournament* - the UDK offers tools for building fully-rendered complex virtual architectures within the Windows operating system, designing AI and animation workflows, a robust multi-user networking layer and a powerful object-oriented scripting language called UnrealScript. The UDK is available at no cost for educational and non-commercial use and was updated monthly with new features ranging from enhancements to the lighting and modeling tools to integration with Apple's iOS, allowing works created in the UDK to be published to mobile devices such as the iPhone, iPad and iPod.

3.2.2 System Overview

UDKOSC is a set of UDK custom class files, which when compiled, take the form of a custom game-type within the UDK. UDKOSC was brings together real-time procedural sound synthesis, spatialization and processing techniques from the realm of computer music with the visually immersive networked multi-player environments of commercial-grade gaming engines. Gestures, motions and actions generated by actors in game-space are analyzed and transformed in real-time into control messages for complex audio and musical software systems.

While the UDK explicitly restricts developers from accessing the Unreal Engine's

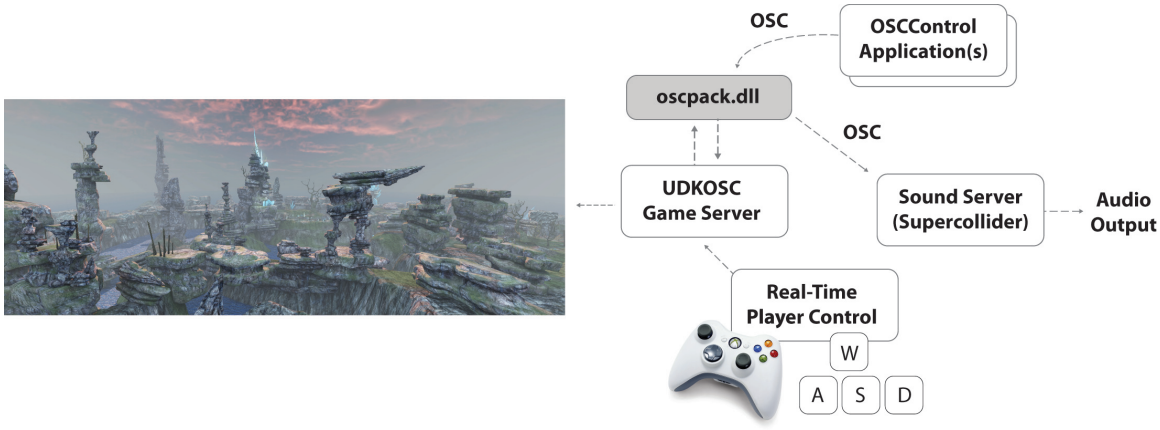


Figure 3.12: OSC data generated by UDKOSC can drive external sound engines such as the Supercollider engine written for *ECHO::Canyon*. OSC data generated by OSCControl can be used to control avatar and camera motion, alongside more traditional gaming control systems like the Microsoft XBox game-pad.

core C++ engine, it exposes UnrealScript as a higher-level scripting language. UnrealScript allows developers to bind Windows Win32 .dll's to UnrealScript classes, enabling external blocks of code to interact with the scripting layer. Using this Dll-Bind functionality, UDKOSC binds a customized version of Oscpack to a series of UnrealScript classes and mirrored data structures, passing bidirectional data both into and out from the game engine. Real-time game data can be streamed over UDP to any given IP-address and port combination. At the same time control messages and data from external processes can be streamed into the game engine.

OSC input can be used to drive actor velocity and rotation in three-dimensions, and can be targeted towards numerous entities individually or simultaeously through the use of OSC bundles. Input data for the control of game pawn, player or camera actors is designed to offer detail and flexibility in the manner in which actors are moved and controlled within the game engine. In this way, detailed choreographed positioning and action can be carried out, such as the "flocking" actions of flying actors or intricate camera motion and cut scenes.

Actors within the UDK, or characters moving through and engaging with the environment, can be controlled by human performers - called "Pawns" - or controlled by game artificial-intelligence or pathing algorithms - called "Bots". These actors

are separate entities from the "Camera", essentially a projected viewpoint within the environment which is displayed on screen. UDKOSC adds the ability for Pawns, Bots and Cameras to be controlled via commands received from externally-generated OSC messages.

3.2.3 Output Data.

Currently, UDKOSC tracks a number of in-game parameters for each actor and exports them using OSC including:

- Pawn and Bot unique identifier within the UDK
- Pawn and Bot Cartesian location (X, Y and Z coordinates), rotation (pitch, yaw and roll) and current speed
- Camera view rotation
- Projectile Cartesian location and collision event (triggered when a projectile touches a solid entity within the environment)
- Interp Actor, Trigger and Static mesh Cartesian location and collision events
- Location and rotation for individual bones from a Pawn's Skeletal Mesh

3.2.4 Input Data.

UDKOSC can receive commands over OSC including:

- Pawn and Bot movement speed, direction vector and rotation
- Projectile target location
- Camera cartesian location, rotation and speed
- Camera mode: fixed location, manual rotation, first and third-person modes

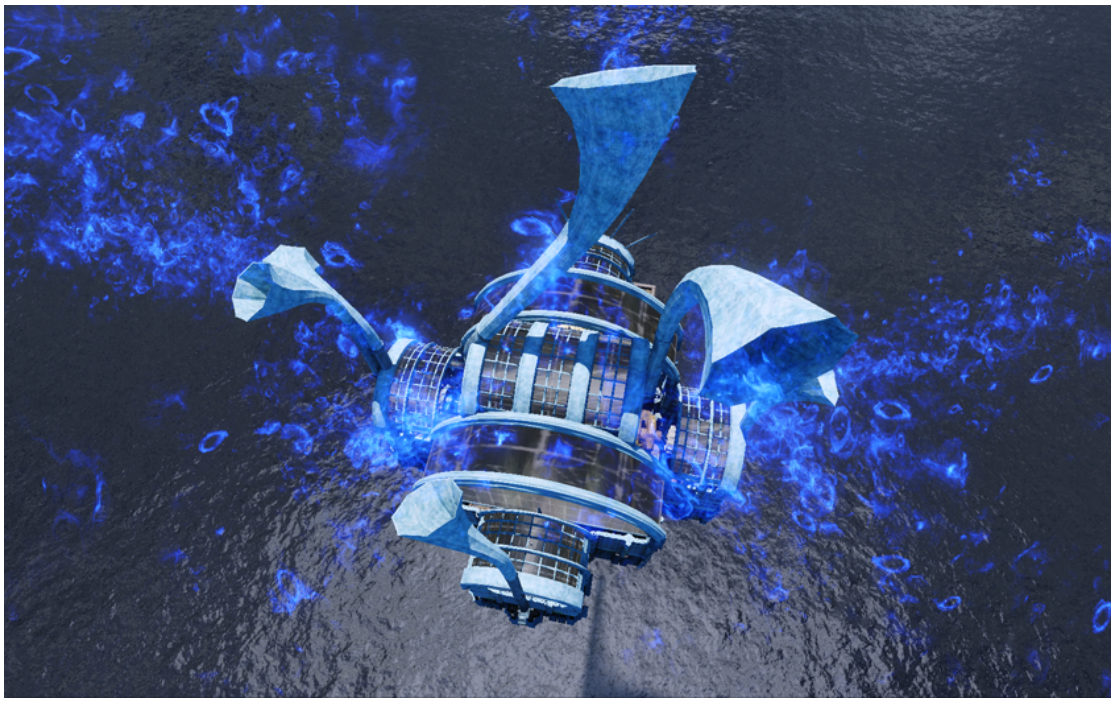


Figure 3.13: The Tele-harmonium is a building-shaped instrument filled with mechanically-driven clock-like gears and horn structures. The building itself bears a strong resemblance to a music-box.

3.3 *Tele-harmonium*

Tele-harmonium (2010) by Robert Hamilton and Chris Platz for live piano and virtual performer was the first work written using UDKOSC and was designed to incorporate the architecture of the rendered environment itself as an instrument capable of performing a duet with a live pianist. Visually, the Tele-harmonium was conceived as a giant machine-building, similar in concept to an immersive music box with rotating gears and mechanisms to mechanically reproduce sound and music. In *Tele-harmonium*, the primary character gestures focused on a robot actor's motion around an environment as well as the flight and homing patterns of glowing projectiles.

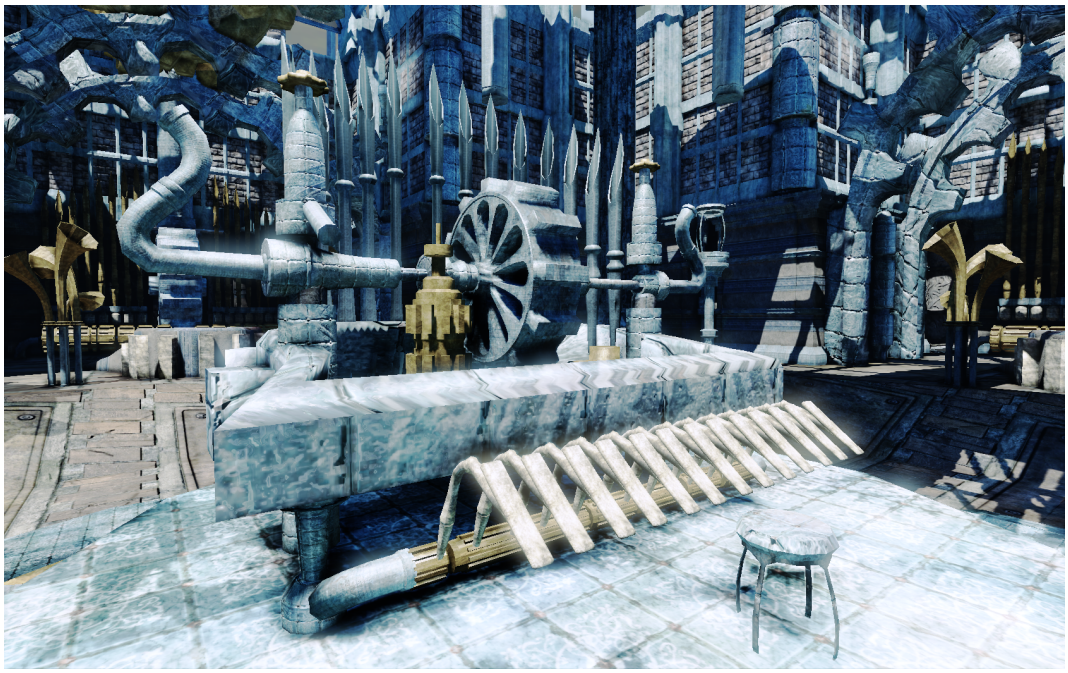


Figure 3.14: At the center of the Tele-harmonium sits an organ console, itself representative of the 'missing' performer (in this case, the live pianist).

3.3.1 Instrument Constructs

The storyline of the work describes the building as an instrument capable of playing back musical memories and as such, the virtual performer is seen travelling, entering and interacting with structures within the building to create and control sound and music. At the center of the space is a rendered organ-like console, intended as the virtual manifestation of the live pianist, though perhaps more appropriately, the pianist should be seen as the live representation of the virtual instrument. Horn-shaped structures fill the environment, with each construct being capable of sending OSC messages upon avatar or projectile collision or interaction. In this manner, objects in the environment become playable "instrumental" constructs. Specific constructs in the environment are designed as attractors for projectiles based on control modes set by performers in real-time.

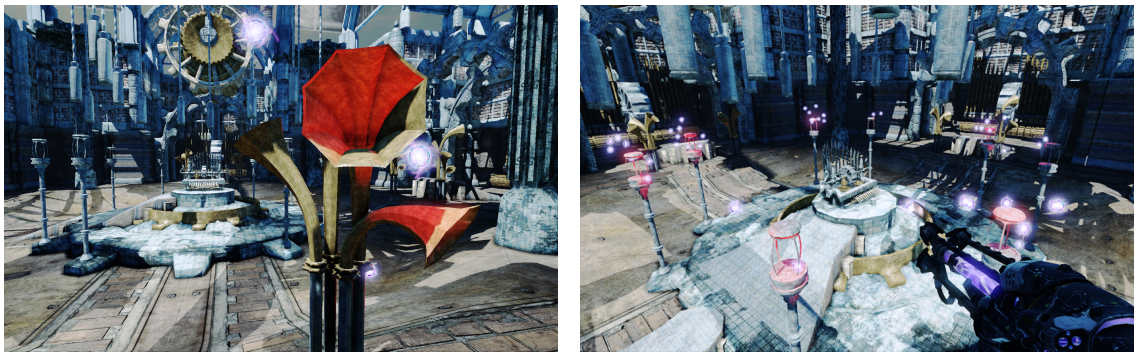


Figure 3.15: Inside the Tele-harmonium, the mechanical nature of the building is evident. The sounding horns (left), the organ-console in the center of the space (center) and eight-sounding posts (right) each allow users to control sound and spatialization during a performance.

3.3.2 Projectiles

Around the center of the Tele-harmonium space are a series of poles and hanging bells, each of which acts as an attractor. When projectiles collide with the bell constructs, specific pitches are sounded, resulting in the formation of a series of tuned bells. When projectiles are triggered to be attracted to the standing poles, continuous pitches are generated based on each projectile's measured distance to the center of the environment, with the coordinate point [0,0,0] resting at the center of the keyboard console construct. Continuous sounds are created using a modified version of the projectile object, mapping spatial coordinates to frequency, amplitude and specific timbral characteristics of a simple synthesized instrument. Swarming and following behaviors are extremely effective in generating rich soundfields when mapped to a simple continuous sound source, with slight fluctuations in pitch and timbre working together to create a timbrally rich musical sound.

3.3.3 Compositional Elements and Mapping

Musically, *Tele-harmonium* is loosely built upon the harmonic and melodic structure of the Sonata for keyboard in D minor, K. 213 (L. 108) "The Lover" by Dominico Scarlatti. Excerpts from the Sonata are heard both in manipulated recording as well

as in live performance by the pianist. Additional musical materials that comprise the piano score were composed as derivative and allegorical variations on particular phrases and structures found in the original work. The virtual performer controls the structure of the composition as a whole, moving though specific locations and triggering recorded excerpts of the original Scarlatti in conjunction with the live pianist's playing.

To allow the virtual performer the ability to successfully duet with an instrument as melodically and harmonically flexible as a piano, both projectile collision and continuous OSC data streams were connected to sound-generating virtual instruments in Supercollider. As projectiles collide with specific objects in the rendered environment, simple pitched sounds are generated using bursts of noise passed through tuned filter banks. Settings for filter frequency/pitch and amplitude are extracted from the UDKOSC stream: Cartesian distance from the central keyboard structure was mapped to frequency while the size of the projectile itself was mapped to amplitude.

Continuous sounds are created using a modified version of the projectile object, mapping spatial coordinates to frequency, amplitude and specific timbral characteristics of a simple synthesized instrument. Swarming and following behaviors are extremely effective in generating rich soundfields when mapped to a simple continuous sound source, with slight fluctuations in pitch and timbre working together to create a dynamic and rich musical sound. As each individual projectile is tracked through coordinate space over OSC, their relative locations within virtual space can be corrolated to locations in physical space as rendered through an Ambisonic decoder. Musical spatial gestures can be created in this manner, allowing sound to move throughout a listening space, creating an immersive auditory experience.

Figure 3.16: The environment of *ECHO::Canyon* is a huge outdoor space featuring rocky peaks, sonified crystal structures and carved tunnels and canyon pathways.

3.4 *ECHO::Canyon*

The paradigms of avian flight, biologically-inspired kinesthetic motion and manually-controlled avatar skeletal mesh components through inverse kinematics are used in the musical performance work *ECHO::Canyon* to control real-time synthesis-based instruments within a multi-channel sound engine. This section discusses gestural and control methodologies as well as specific mapping schemata used to link virtual actors with musical characteristics.

3.4.1 Background

ECHO::Canyon (2013) by Robert Hamilton and Chris Platz is an interactive musical performance piece built within UDKOSC. Premiered on April 25, 2013 at Stanford University's Center for Computer Research in Music and Acoustics, *ECHO::Canyon* creates a reactive musical environment within which the idiomatic gestures and motions of flight are mapped to musical sound-producing processes.

3.4.2 Musical Sonification in *ECHO::Canyon*

During the piece performers control virtual actors moving through a fully-rendered outdoor landscape using a computer keyboard and mouse or commercial game-pad controller. Each actors' location and rotation in game-space, as well as other parameters describing their interactions with objects within the environment are streamed in real-time to sound-servers using Open Sound Control. The environment itself is sculpted in such a way as to allow performers the freedom to control musical interactions by moving above, around and through the topography. In this way the process of environment design complements the process of musical composition, with sections of virtual hills, canyons and valleys acting as musical pathways through the environment.

While *ECHO::Canyon* is built within a gaming engine, unlike many commercial games where audio and music play a supporting role to displays of rich visual content, the role of music and sound within the work are intended to occupy a perceptual role equal to the presented visual modality. Sonifications used in *ECHO::Canyon* are designed to be musical and performative in nature, and are fundamentally presented as foreground constructs, rather than as background or more associative "sound-effect" constructs. To that end traditional approaches for game sound design are replaced instead by sets of composed interactions.

Within Supercollider, data representing avatar positioning, rotation and action is mapped to specific parameters within instances of synthesized instruments. In *ECHO::Canyon*, the flight of a player-controlled Valkordia pawn through the environment is sonified in real-time. At any given moment of the piece each pawn's speed, rotation, absolute Z-location or height, height relative to the "ground," side proximity to solid environment structures and a Euclidean distance to a series of "crystal" objects in the environment all serve as parameters driving real-time synthesis. Alongside one or multiple human-controlled pawns, flocks of OSC-controlled Valkordia bots, themselves driving separate synthesis processes, are controlled with pre-composed OSC-emitting scripts. During flight as well as during a specially-designed "posing state," the location of bones in the bird-skeletons' wings are tracked and mapped to their own synthesis processes.

Figure 3.17: Crystal emitters dot the landscape of ECHO::Canyon and are some of its strongest visual and harmonic elements. As player avatars move closer to any crystal location, they engage a series of sound-generating processes mapped to their avatar's distance from each crystal .

Musical output for *ECHO::Canyon* is spatialized across multi-channel speaker systems by a Supercollider sound server making use of stereo output, simple 4-channel panning or higher-order ambisonics [77] as the performance space allows. When ambisonic output is selected events are placed in the soundfield using a mapping schema uncommon in standard video-game audio where coordinate locations in the environment are mapped to static corrolary locations within the listeners' soundfield. When stereo or simple 4-channel panning is employed, location-based sound events are placed in a more conventional actor-centric perspective, with their amplitudes scaled proportionally to the distance between actor and sound-emitting location.

3.4.3 Environment

The world of *ECHO::Canyon* is situated in a fantastical open-air virtual environment featuring rich flora, jagged volcanic rock outcroppings and various species of megafauna. Isolated on all sides from outside interference by a massive ocean, ECHO::Canyon exists as a bizarre island oasis wherein biological evolution has progressed in novel directions, unaffected by the rest of the world. Musical sound has similarly evolved, facilitating communication not only between creatures on the island but between creatures and the island itself, specifically through the use of massive energy crystals situated at key locations around the environment.

3.4.4 Crystal Emitters

As seen in Figure 3.17, jutting out from high-peaks towering above the ECHO::Canyon environment are sound-generating clusters of crystal, capable of resonating at various pitches when a creature approaches. These crystals form the harmonic structure of *ECHO::Canyon* and draw their resonant frequencies and filter coefficients from a combination of avatar location and randomly generated seed pitches established at the launch of the piece. Each crystal is positioned to break the large environment down into regions, within which performers can congregate, to interact with one another, or across which performers can spread to create an immersive sound field sounding around and throughout the audience.

Musically, each crystal cluster generates a polyphonic wash of pitches scaled in part based on the coordinate location of the crystal itself and in part based on a random value, created and seeded with each launching of the game engine. In Supercollider, each crystal location drives a separate instance of the Gendy1 dynamic stochastic synthesis UGen. An actor's distance from each of these crystals controls both the Gendy1's amplitude as well as the frequency of a ResonZ two pole filter.

When multiple actors approach a single crystal instance, each of their location coordinates are used to modulate a different control parameter of the instrument, creating a group interaction schema which differs signficantly from the individual user schema. This results in a collaborative control system in which each member of an ensemble can directly control one or more attributes of the synthesis process.

3.4.5 Canyons

Spiraling outwards from a central peak extend a series of low-lying canyons with sharply sloping walls. For creatures moving quickly through these canyons, a variety of musical sounds can be produced by skimming horizontally close to the canyon walls. Ridges carved into the terrain and outcroppings of rock and crystal create composed variations of amplitude and timbre as performers move past them. By sculpting repeated patterns into canyon walls the designers are able to "compose" rhythmic elements with which the performers can choose to engage.

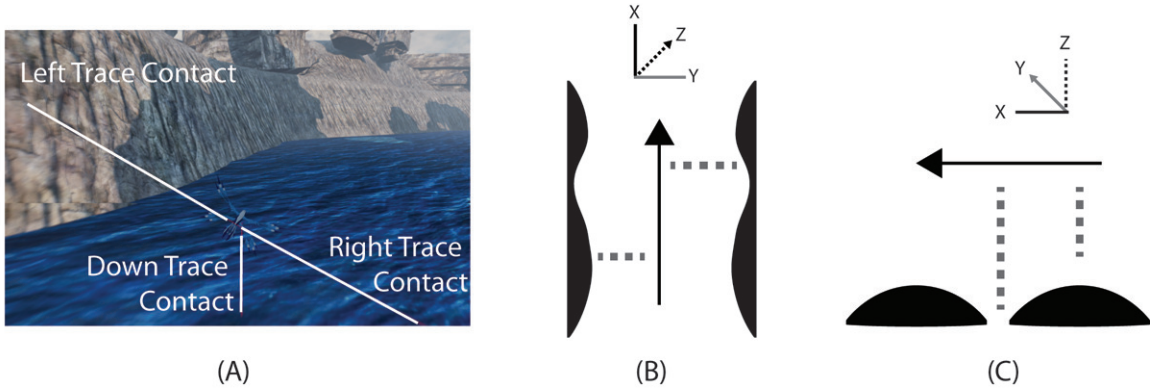


Figure 3.18: Real-time horizontal and vertical ray traces are used to monitor the contours around each avatar in the *ECHO::Canyon* environment. Ray traces visualized as vertical and horizontal lines in (A) track the distance between the Skeletal Mesh and objects and contours to its right and left sides (B), as well as directly below (C).

The relationship between a flying Valkordia actor and its environment was a key gestural component in the shaping of *ECHO::Canyon* both literally and figuratively. Valleys, mountains and caves were sculpted with articulated shapes to accentuate specific features of synthesis processes. Figure 3.18 shows vertical and horizontal ray traces tracking the relative distance between a flying actor, the ground, and the walls of a valley. In this example, the ray trace distances were used to control amplitude and filter frequencies of separate synthesis processes, as well as panning for the horizontal traces.

3.4.6 Characters

The island of ECHO::Canyon is populated by three races of creatures, each capable of generating unique musical sounds through gesture and interaction with aspects of the environment, as well as with other creatures. Each creature type can be controlled by performers of the piece, allowing for variety in performance and weighting in orchestration.

Figure 3.19: The flying Valkordia avatar features a four-winged articulated skeleton and flight animations modeled from the behaviors of insects and birds. A highlighted bone seen in the right-front wing-tip above is tracked in real-time to drive sound processes.

3.4.6.1 Valkordia

The Valkordia are a race of four-winged birdlike creatures with a unique bone structure that creates a variably pitched sound when air passes over and through specialized openings on each wing. Valkordia are highly agile and speedy creatures which generate unique polyphonic vocal cries to communicate. As part of their mating or fighting rituals, males and females alike will often adopt a vertical pose, keeping themselves aloft by rapidly beating their lower/rear wings. While posing, each Valkordia can generate musical phrases, both challenging and enticive, using their beating wings and voices alike.

3.4.6.2 Trumbruticus

The Trumbruts are a race of elephant-like megafauna that roam freely around the *ECHO::Canyon* environment. Equipped with a musical prehensile trunk, as well as two posterior-facing rear trumpet horns, herds of Trumbruts communicate using musical blasts from their three main horns, and are capable of producing loud and low tones, as well as higher harmonic whistling clusters. Users controlling a Trumbruticus model can independently move each creature's head and trunk using inverse kinematic mappings. Like the Valkordia, each Trumbruticus is capable of producing a complex

Figure 3.20: The Trumbruticus avatar features a user-controllable trunk as well as two rear-facing “trumpet-bell” appendages.

“call” as well as a more persistent low frequency tone.

3.4.6.3 Shelltapper

The Shelltappers are an ancient race of dinosaur-like creatures sporting a thick carapace and a mane of antenna-like "tappers". The scales that make up the Shelltapper’s thick shell are tuned to musical notes and resonate with a sound reminiscent of a stuck modal bar when struck with the creature’s tappers. As they walk through ECHO::Canyon foraging for food, these creatures create pitched rhythmic patterns, modulated by their speed of motion and distance from a batch of crystals. The further Shelltappers are from any crystal, the less pitched their shells are, resulting in a timbral range from flat percussive sounds to highly resonant bell-like timbres.

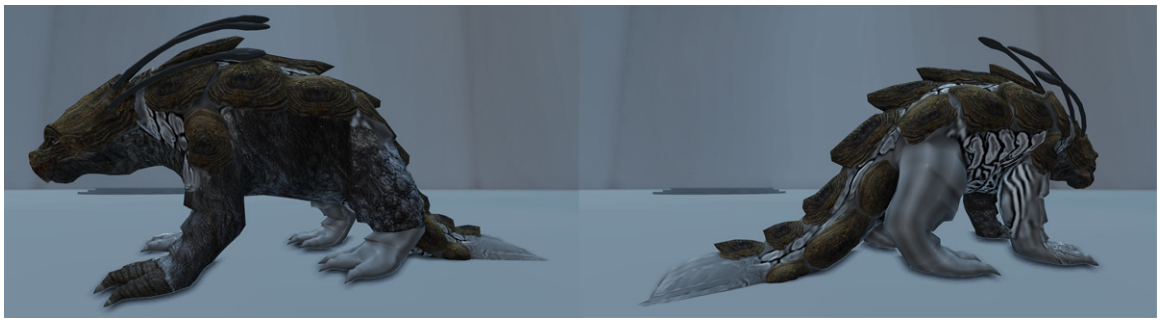


Figure 3.21: The Shelltapper is a slow-moving dinosaur-like creature. Scales on the back of the Shelltapper trigger a modal synthesis process when struck by the creature's antennae.

3.5 Composing Musical Interactions

The performance works described in this chapter explore a number of mapping schemata linking user-controlled activity with sound-generating process. In these works, the environment often takes on an intermediary role, acting as an indeterminate stochastic filter of sorts. Users' intentions are modulated by unpredictable variances as they interact with the environment. During each work's design process, unique and idomatic control schemata were devised, taking into account these environment-based unpredictabilities. The unique nature of this kind of intermediary layer drove the compositional process for these works away from the crafting of determinate musical forms and towards the composition of musical interactions themselves.

Choreographies of music and action found in dance and film commonly make use of a reactive association between gesture and sound. Dancers' reactions - spontaneous or choreographed - to a musical event or sequence of events often form physical motions or gestures with direct temporal correspondence to the onset, duration or contour of a sounding event [65]. Similarly, events in static visual media such as film, music video and some computer games are often punctuated by the synchronization of visual elements with unrelated auditory or musical cues, linking the audio and visual in our perception of the event without any causal relationship existing between the two modalities.

The tracking of actor motion and action within interactive virtual environments offers another potential approach to the mapping of physiological gesture to parameters of sound and music for multimodal presentation. As avatars within three-dimensional space are wholly-digital constructs, there exists a massive amount of data readily available that represents their internal and external state, their ongoing relationship to other objects in the surrounding environment, and the state of the environment itself. This data can drive complex dynamic musical and sound-generating systems while preserving a causal link between the visual gesture and the resultant audio gesture.

With virtual actors, the contours of motion in virtual space - both *macro*, such as motion along a three-dimensional Cartesian vector, or *micro*, such as the relative articulation of individual bones within an avatar skeletal mesh - can be tracked and used as control data for computer-based musical systems. In this manner, the gesture or motion itself drives and controls the sound-generating process, an inversion of a more common reactive model and very much in line with traditional models of instrumental performance. By pairing macro and micro avatar motions with real-time musical sonification, composers and designers repurpose elements of model physiology and structure, as well as the topographies of virtual space itself, into components of musical gesture. Multiple modalities of interaction can then be combined to create performance works wherein the interactions between virtual actor and virtual environment drive any number of parameters of computer mediated musical sound, structure and space.

3.5.1 Musical Projectiles

Projectile Bounces A direct and effective mapping technique used in both q3osc and UDKOSC is the sonification of projectile bounces, or collisions between projectile entities and environment surfaces. At the point of contact between a given projectile and environment surface, the projectile bounces off the surface, setting and transmitting a "bounce" OSC flag. To best accommodate a massive number of bouncing

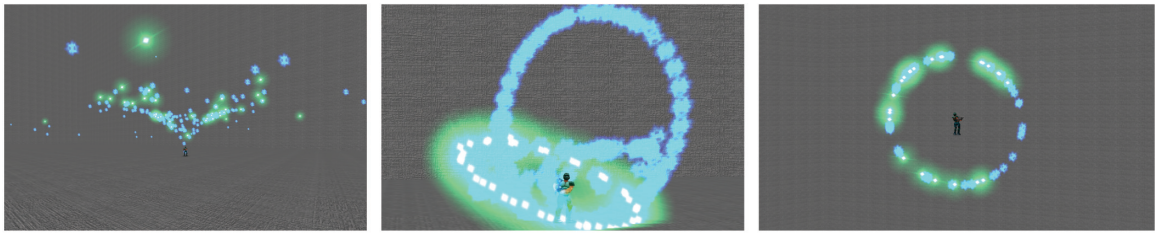


Figure 3.22: Sonified projectiles in q3osc can be directed to track specific actors or locations. By modifying an avatar's motion while being tracked in this way, patterns and shapes of swarms of projectiles can be created and controlled.

projectiles, in q3osc an extremely simple sonification running on a ChucK sound-server of an impulse routed through a resonant filter has proven to be extremely efficient and robust. By mapping coordinate data for a single plane - say the X axis - received in the same OSC message as the bounce flag to a desired musical scale, each bounce event can be made to trigger simple notes. As a possible example of additional non-spatial bounce mappings, speed of the projectile can be mapped to velocity or amplitude of each bounce-triggered note-event, Y-axis coordinate data can be mapped to note-duration and Z-axis coordinate data can be mapped to length of decay on the reverb. At any such point of impact, data points such as angle-of-incidence, distance-traveled, and duration of existence all are easily extracted.

Continuous Projectile Output As coordinate data for q3osc and UDKOSC projectiles is transmitted continuously over OSC, sonification of projectile motion across coordinate space can be introduced in addition to each projectile's discrete bounce events. Towards this end, individual functions can be instantiated for each projectile, spatializing the projectile's motion across speakers within the same array or through osc analysis, across distributed speaker arrays.

3.5.2 Swarming and Flocking behaviors

Our cognitive abilities to group and associate like motions of active objects into single cohesive units can bring disparate dynamic elements together into one unified mass gesture [74]. The sonification of such behaviors with simple sound sources can create

dynamic musical textures through similar motion and position of each source [31]. Projectiles in both q3osc and UDKOSC can be dynamically sent to follow individual avatars or specific locations in space. Such follow behaviors can give rise to interesting patterns and shapes, as seen in Figure *3.22*.

In a similar fashion, *ECHO::Canyon* features flocks of Valkordia pawns, controlled over OSC or via in-game "follow" states, with a simple mapping of their Z-coordinate to a pitched oscillator and their distance from the player actor to the oscillator's amplitude. When driven by OSC input, each bird in the flock tracks a target effector which is moved in pre-composed patterns through the game-space. When acting under the direction of an AI Controller - an in-game logic sequence that allows each actor to follow a pre-determined set of rules - each bird in the flock can be set to follow individual performers or positions in space. The sonic result of either control schema is a shifting grain-like cloud of pitched oscillators.

3.5.3 Musical Pathing

One compositional methodology that has been used in a number of works detailed in this chapter revolves around the design of virtual environments to guide performers through and around constructs that will drive specific musical interactions. These compositional pathways are inherently flexible, designed to generate musical reaction by bringing performers into contact or proximity with specific structures or regions of the environment. While not rigid or deterministic sequences that must be followed in a singular linear progression, these musical pathways allow a composer/designer's intent regarding musical form or structure to be made available to the performers, while allowing for significant flexibility in performance and interpretation. The following examples show how musical pathing has been used across this body of multimodal musical works.

3.5.3.1 Compositional Mapping in *Maps and Legends*

Maps and Legends presents performers with a very clear pathing structure in the form of a bright-yellow path with floating yellow directional arrows, guiding performers through the composition. At key points in the compositional map, circular trigger regions lay in the suggested path of motion. Precomposed musical materials were triggered when performers moved over these coordinate spaces. In this manner, separate sets of mono sound files, all complementary parts of the composition, were assigned to individual performers and triggered in PD based on each performers' position. These sets of sound files, a mixture of synthesized and sampled sound, made up the bulk of pre-composed materials for *Maps and Legends*. Other basic mappings included a light chorus and reverb processing applied to each performer's currently playing sound file when moving over a highlighted pathway - an aural incentive of sorts for moving along the pre-composed paths - as well as a longer reverb and longer delay applied when the user's Z-coordinate indicates that they were fairly "high" in the map's vertical dimension - a state that could be triggered more often by lowering server-side parameters like virtual gravity.

3.5.3.2 Actor Paths in *In C*

In C was designed in a manner that lets musical performers move linearly through composed musical cells, choosing repetition and cell-onset timings while remaining strictly locked to the underlying metronomic pulse. The virtual environment artwork was created as an allegory of the work, sequentially following a set of patterns or cells though motion down a rendered pathway. A clear linear pathing model was used, so that each cell of the modular music was like-wise represented visually as the cell of a pathway. The scale of the path and the visual affordance was clearly laid out in the lattice railways that lead from one area of the environment to the next.

As multiple performers were connected from disparate global locations, separate paths were created to allow variation in the musical pathing. The journey took the audience through a surreal floating city landscape with an end tower clearly in sight for all three paths. This conforms with the instructions of the piece where

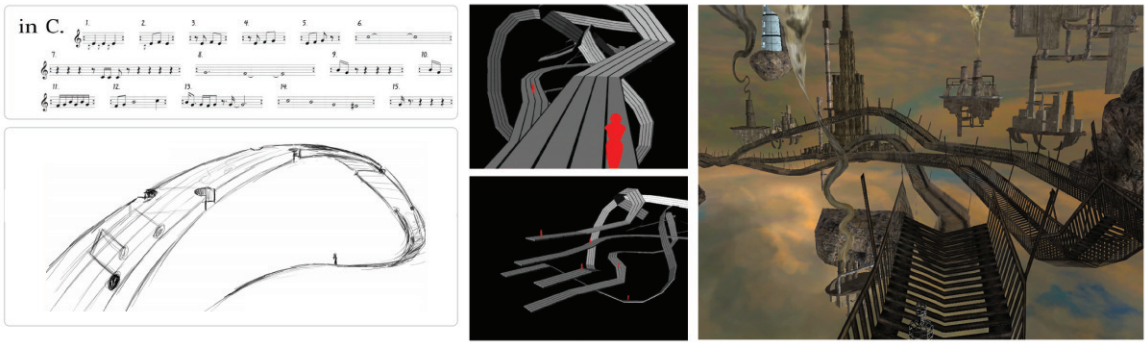


Figure 3.23: Riley's *In C* routes performers down a musical path comprised of 53 sections. In the multimodal performances of *In C,* avatar motion through rendered paths triggers the playback of musical cells. These paths are shown here first as an excerpt from the musical score (top left), as a design sketch of the "musical path" (bottom left), as an artist's early design drafts (center) and as fully rendered paths in Sirikata (right).

at the end the whole ensemble "lands" on the last pattern. This rendition of *In C* adds different layers of musical and visual meaning that conform with the original intention of the piece as a multimodal expansion of existing repertoire. The idea of local synchronization is expanded to distributed musical synchronization. The idea of advancing through a series of musical cells is not only illustrated in the virtual world, but literally becomes a performance practice: physically advancing means musically advancing in the cells.

3.5.3.3 Canyon walls and floors in *ECHO::Canyon*

While *ECHO::Canyon* was designed to be a fundamentally open environment, much care was put into the design of pathways and routes to guide performers in specific musical directions. As seen in Figure 3.18, horizontal and vertical ray-tracing allows performers controlling the Valkordia avatar to instigate rhythmic pulses or phrases by flying through these pre-composed pathways. Ridges carved into the sides of a cliff or bumps laid out across a pathway each create sounding patterns when flown over. By controlling the Valkordia's speed of motion, performers can play each figure at a faster or slower tempo. Figure 3.25 shows a series of these pathways highlighted

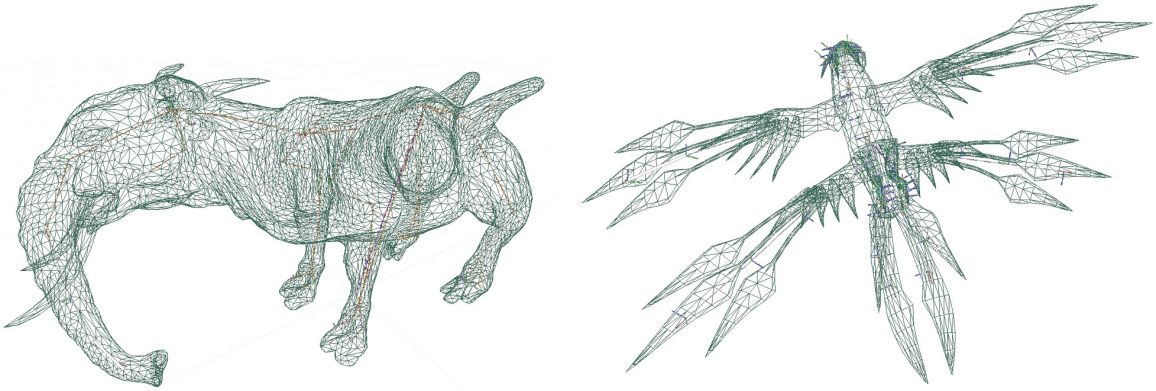


Figure 3.24: Creatures in the *ECHO::Canyon* world were sculpted and rigged with articulated skeletal meshes, allowing them to move with gestures appropriate for their body type, speed and style of motion. Here the skeletal meshes of the Trumbruticus and Valkordia avatars are exposed, with their chain of bones visible as the red line within each body.

with red and green markings.

3.5.4 Skeletal Tracking of Avatar Physiologies

One technique for building realistic digital models of anthropomorphic bodies and motion sequences is the use of a virtual skeleton or skeletal mesh upon which forces and physics-based effectors can act. The addition of constraints on motion and rotation for individual joints and bones allows a given skeletal mesh to act and react dynamically to control systems and the environment without the need for specific code to account for every potential interaction. For the actors used in UDKOSC works like *ECHO::Canyon*, complex skeletal meshes were created and rigged to simulate creature limb movements inspired by the paradigms of avian flight and prehensile trunk motion. By tracking the position and rotation of individual virtual bones that make up a character's skeletal mesh, data derived from the skeletal meshes can be used as a driver for real-time sonification using UDKOSC.

With the intention of drawing audience and performer attention to the actor itself and away from the environment, micro-scale gestures are comprised of motions and articulations of a given avatar's virtual physiology. By mapping the subtle motions of

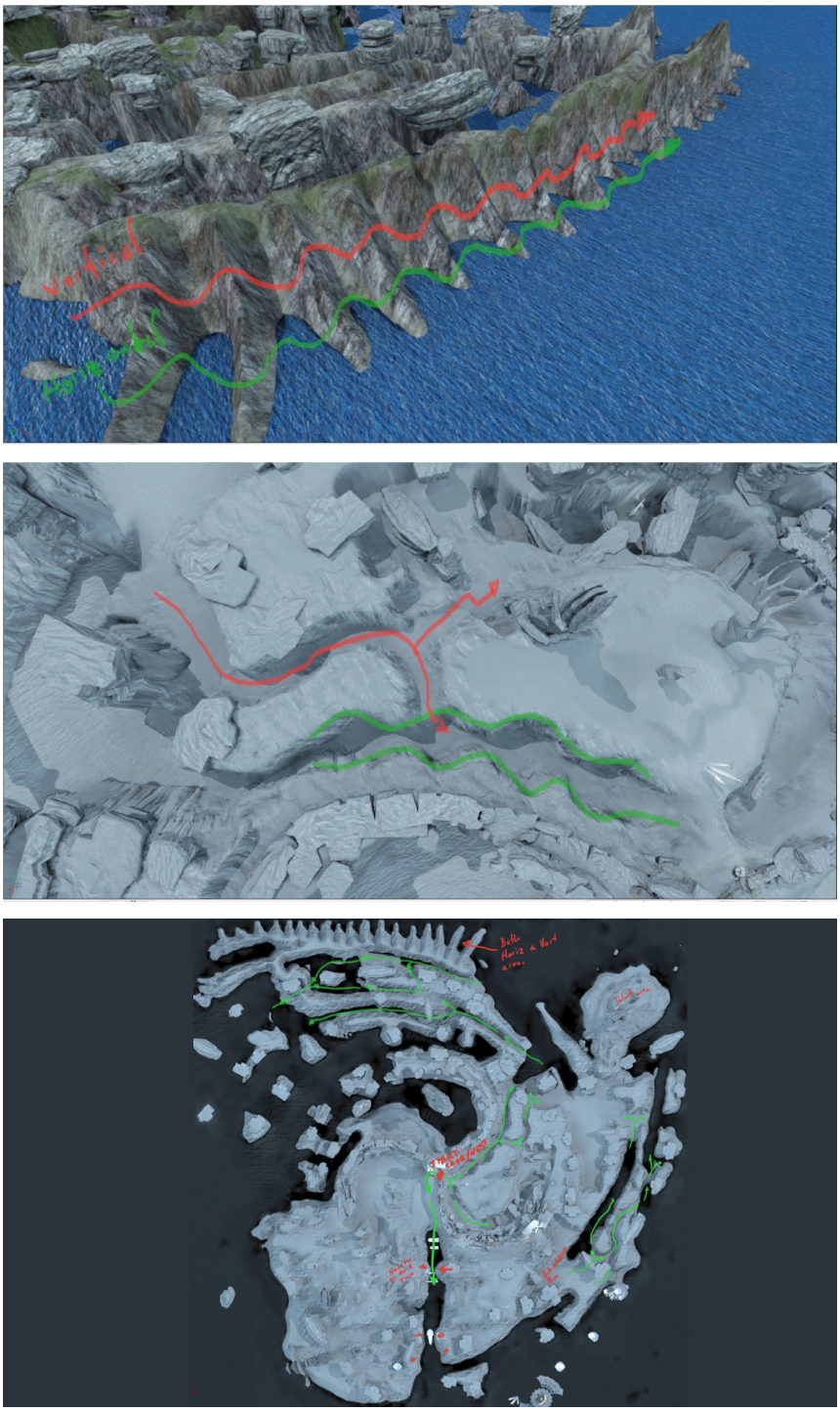


Figure 3.25: Paths throughout the *ECHO::Canyon* environment were "composed" to guide performers around the virtual space in patterns that both generate engaging sound and also contribute to the structure of the environement as a whole. Ridges carved into the side of a cliff create rhythmic musical elements when flown over. Canyon runs emanating out from a central tower shape both the visual and sonic landscape*s*.

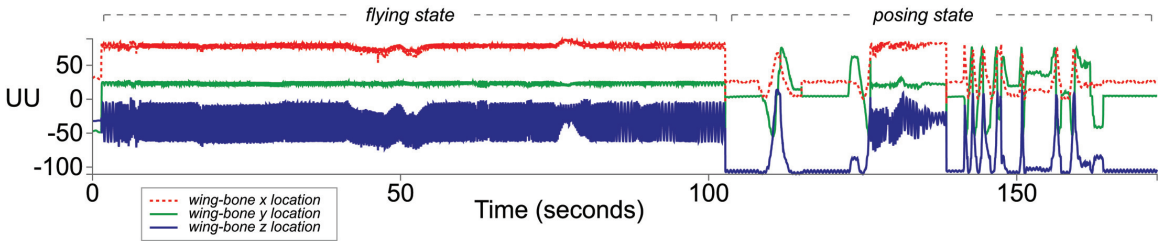


Figure 3.26: The patterns and ranges of motion in three dimensions for a Valkordia's articulated wing-tip. On the left, data during a flight sequence is displayed, while on the right, the same wing-tip bone is tracked during a manual posing session.

bones within an actor's skeletal mesh to both dynamic control systems and evocative musical processes, the micro-scale gestures within *ECHO::Canyon* provide a vastly different viewing and listening experience than the work's macro-scale gestures.

3.5.4.1 Wing and Trunk Motion

For *ECHO::Canyon*, the theme of avian flight is a central component to the musical sonification, animation and control schemata created and used for the piece. The Valkordia character model fuses physical characteristics and idiomatic movements from both bird and insect-like creatures. Bones found in the model's right and left wings (see Figure 3.24) are tracked to drive a pair of pitched oscillators when the avatar was in flight. As users increase or decrease their speed of flight, the animation sequence causing the avatar's wings to beat increases or decreases, speeding up or slowing down the oscillator's pitched motion.

A similar technique was used to sonify the motion of the Trumbruticus avatar's trunk. By tracking a bone located at the tip of the trunk, itself a skeletal chain of nine individual bones, the animated swaying of the trunk during the slow lumbering motion of the avatar is translated into a low-frequency pitch, slightly varying as the height or z-coordinate of the trunk bone changes. The overall effect for Trumbruticus motion is more subtle than the constant pulsing of the Valkordia's wings, giving each creature its own gentle accompanying drone as it traverses the landscapes of *ECHO::Canyon*.

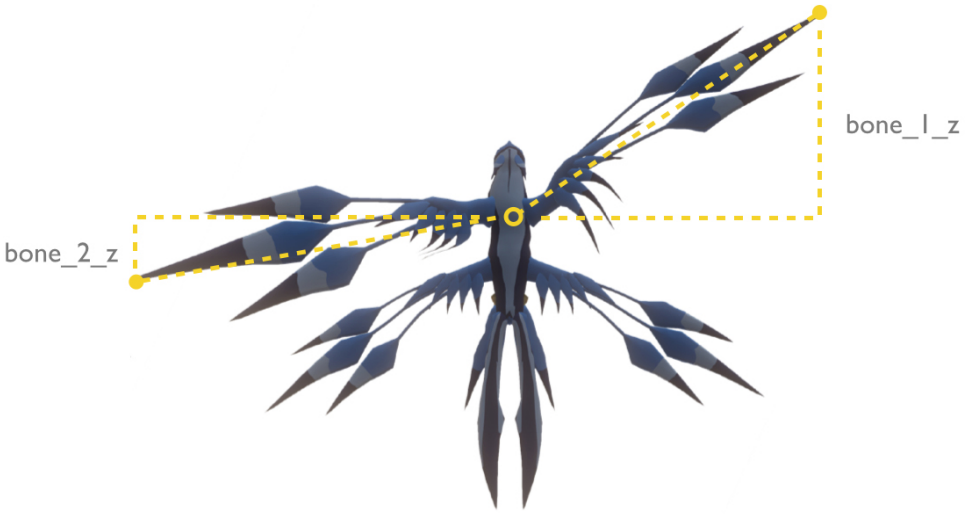


Figure 3.27: The sonification of a Valkordia's wing-tip data in *ECHO::Canyon* tracks the distance between each wing bone and a central point on the avatar skeleton, and maps that value to the frequency of an oscillator.

3.5.4.2 Posing States

Performers in *ECHO::Canyon* enter their posing state by toggling a key on the gamepad or computer keyboard. Upon entering the state, actors controlling a Valkordia avatar no longer fly through the environment; instead their avatar model interpolates into a nearly vertical pose and control over the actor's front two wings are directly mapped to each of the gamepad's two two-dimensional analog joystick controls. Users control the forward, side and back rotation of each wing independently by rolling the analog joysticks around in circular patterns, mimicking the rotation of arms or wings in shoulder sockets. A series of wing poses can be seen in Figure 3.28.

Rather than mapping pre-composed wing animations to output from the joystick controllers, each wing instead tracks an end effector using inverse kinematics [11]. The location of each effector is controlled in 3D space by the joystick output and acts upon a Cyclic-Coordinate Descent or CCD Skeletal Controller component of the UDK. Effectors can be visualized as points in space towards which the chain of bones from the tip of each wing to the shoulder socket are reaching (see Figure 3.29).

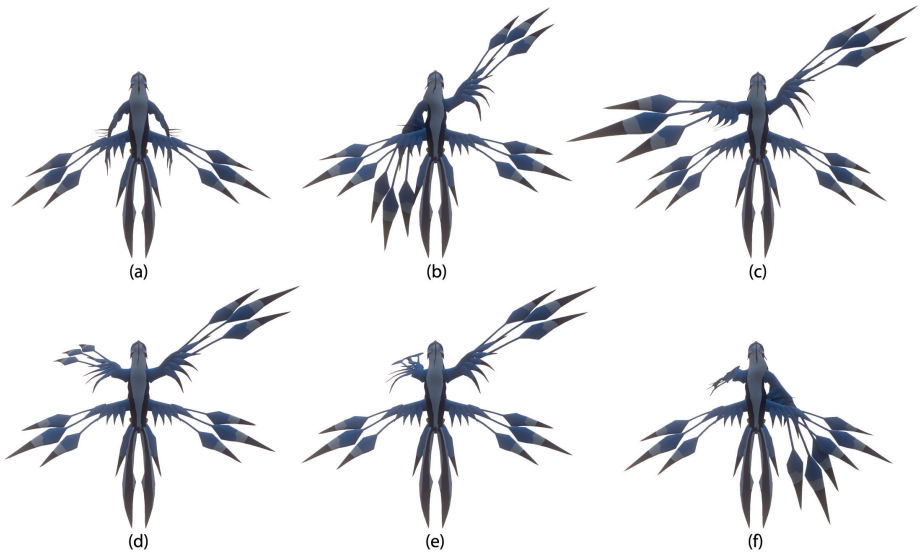


Figure 3.28: Six stages of wing motion during the manual posing state of a Valkordia avatar are displayed, showing the constrained range of wing motion available for user control.

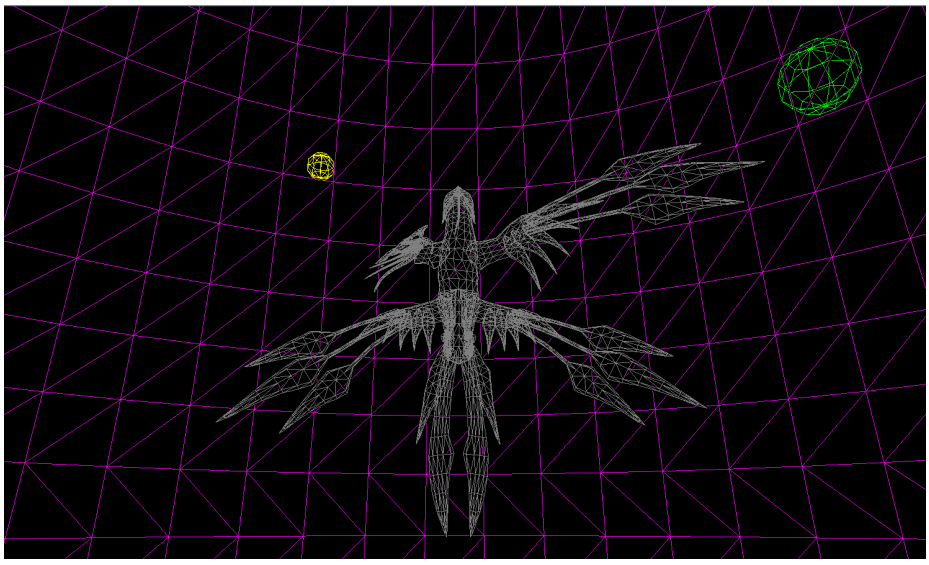


Figure 3.29: The effectors towards which a Valkordia model's wings are reaching are displayed as yellow (left) and green (right) spheres.

Figure 3.30: A Trumbruticus avatar in a posing state extends its trunk while being manually controlled by a performer.

A similar dual effector approach was used to control the motion of the Trumbruticus avatar's head and trunk. When entering the posing state, the Trumbruticus actor simply stands still, with control of its head and trunk driven by the primary joystick controllers. With one inverse kinematic effector driving the central head-bone of the creature and the second effector controlling a bone at the tip of the creature's trunk, performers can smoothly move the avatar's head and trunk through a smooth and wide range of motion. The motion of the trunk remains idiomatic, with the ability to curl and twist, while the head motion acts as a gross modifier to the trunk's overall position.

3.5.4.3 Creature Calls

Each creature in *ECHO::Canyon* was given a singular "call" as a metaphorical way to allow creatures in the narrative to "speak" to one another. The basic call instrument can be described as a cascading set of enveloped oscillators, resulting in a densely textured pitched sound with constant variance. The call is driven by a combination of parameters taken in real-time from the environment and avatar, creating a hybrid model of control. For the Valkordia avatar, the base frequency of the call is set by the avatar's height or z-coordinate in the environment. That frequency is modulated

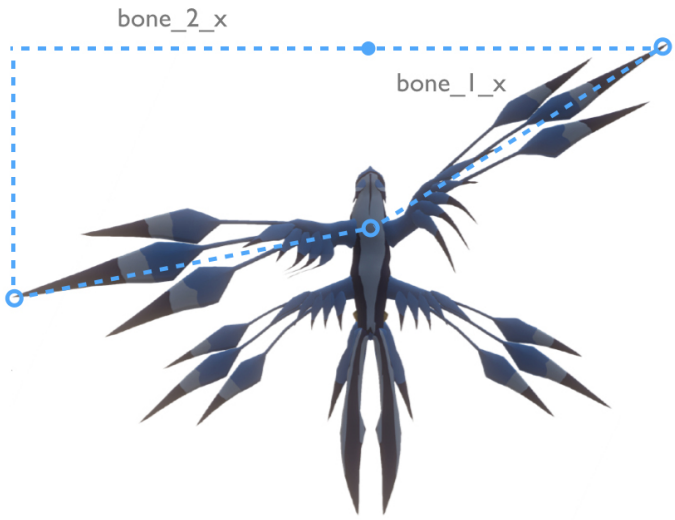


Figure 3.31: During the triggering of a Valkordia avatar's "call" sound, the frequency of each oscillator is modulated by the calculated distance between bones located at each wing-tip.

by the total distance from each wing-tip to a node positioned at the center of the avatar's chest (see Figure 3.31). For the Trumbruticus avatar, the call is generated similarly, though the base frequency is set to a much lower pitch and restricted to a smaller range, and the modulation is controlled by a single bone attached to the tip of the trunk.

3.6 Spatialization Techniques

Actions taking place in specific locations in virtual space can be spatialized around and (perceptually) throughout a multi-channel speaker environment. Throughout many works described in this chapter, a number of spatialization techniques have been used to create immersive audio fields to complement the immersive visual environments. While early works like *Maps and Legends* used relatively crude panning techniques to move sound around an eight-channel speaker environment, a number of more sophisticated techniques were used in later works involving creative positioning of hemispherical speakers and ambisonic spatialization.

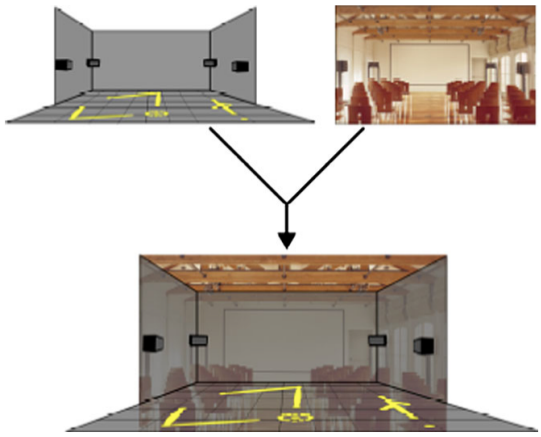


Figure 3.32: By super-imposing the location of sounding events in virtual space onto physical locations in listening spaces, an immersive experience can be created. The creation of visually-similar structures and layouts that share characteristics of "real-world" and virtual spaces is explored in a number of works described in this section.

3.6.1 Spatial Super-imposition

In video gaming, the "First-Person" viewpoint is a common style of game play, allowing users to tele-operate or control an avatar in rendered space while viewing the environment through the eyes of the avatar. In such systems where a user's self is represented virtually in the form of an avatar capable of interacting "physically" with virtual constructs and actions, the focus of both visual and auditory representations presented to each user has commonly been wholly user-centric: a user in his or her real-world seat is presented with an illusory visual and sonic representation of the virtual environment from the viewpoint of that user's avatar. The illusion is heightened through the application of audio signal processing techniques designed to enhance the related first-person illusions of space, distance and motion. As users listen to audio output through headphones, stereo speakers or an industry-standard multi-channel configuration such as 5.1 or 8.1, audio processing done in game-engines tends to attempt to create realistic illusions of motion and spaces for one user sitting in the sound-system's centralized "sweet-spot." Such individualistic and user-centric presentation by its very nature restricts the communal sensory experience fostered in the virtual environment from existing anywhere except within the game-world itself.

In a context where live musical presentations are attended by a viewing and listening audience, the use of user-centric video and audio presentation to the audience itself proves problematic. User-centric systems are optimized for a single point of attention and don't translate to multiple points of attention when suddenly co-opted into delivering visual and audio cues to a group spread within a viewing and listening space. By inverting these traditional models of sound presentation and by focusing on a spatio-centric model of sound projection for game-environments, a communal listening experience can be fostered that is inclusive of all listeners within a shared physical space, including game-users and audience members alike (Fig. 3.32).

3.6.1.1 Correlated speaker positioning

Working with the Stanford Laptop Orchestra's array of hemispherical speakers, it was possible to correlate speaker placement in physical space with speaker-representations in virtual space, creating a physical analogue to a virtual construct. When q3osc works were performed in a smaller chamber-style ensemble, four or five performers would control avatars moving through the environment, firing sound-projectiles which bounced or tracked individual performers, creating sound events at every bounce or collision with a surface of the environment. Virtual speaker placements represented by hemispherical structures could be laid out in a specific pattern in the environment (see Figure 3.33), with hemispherical speakers in the real-world laid out in a similar configuration. Sound events were then spatialization based on each event's Euclidean distance to each speaker structure.

3.6.1.2 Virtual Replicas

The use of site-specific virtual environments, designed to match the environment in which they are presented has been an effective way to present many of these multimodal works using spatial super-imposition. A series of concert halls and studios have been recreated for performance works and for studying spatialization techniques within a given space.

The CCRMA Listening room was the first physical space modeled and represented

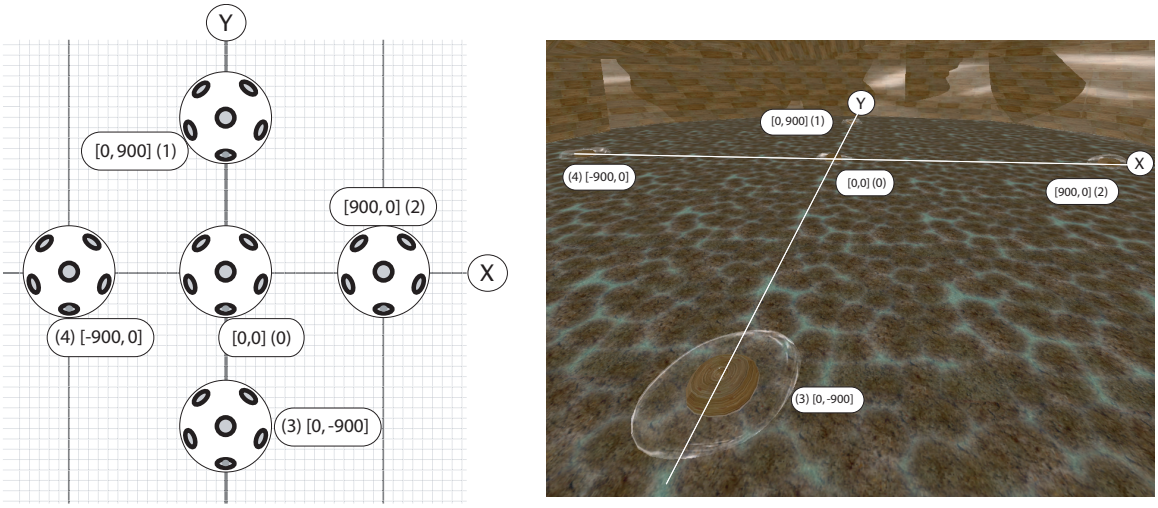


Figure 3.33: The layout of hemispherical speakers used for a chamber performance of *nous sommes tous Fernando...* positioned virtual speaker representations in the same pattern as the speakers in the physical concert hall. Sounding events located near each virtual speaker were subsequently panned to their real-world corrolaries.

in this way during the course of this research (Figure 3.34, row 1). Sonified projectiles moving throughout the virtual model were spatialized around the sound field mapping virtual speaker locations onto the physical location of each of the eight physical speakers in the room. Avatar motion in space was mapped to spatialization of sound around the listening space, forcing listeners and viewers to come to terms with parallel representations of the physical, visual, virtual and shared auditory spaces.

For a large-ensemble performance of *nous sommes tous Fernando...* held outdoors in the CCRMA courtyard by the Stanford Laptop Orchestra, a virtual replica of the courtyard's walkway was built. Virtual speaker locations were positioned throughout the environment to correlate with the layout of SLOrk stations and performers across the courtyard. Figure 3.34 (row 2) shows both a screen capture from within the environment as well as a photograph of one SLOrk laptop displaying the map location that correlates to its own physical location.

In *Dei Due Mondi*, the position of avatars moving through models of the Milan concert hall as well as the CCRMA Stage drove the spatialization of soundfiles in each of the physical concert halls (see Figure 3.34, row 3). To visualize the network link

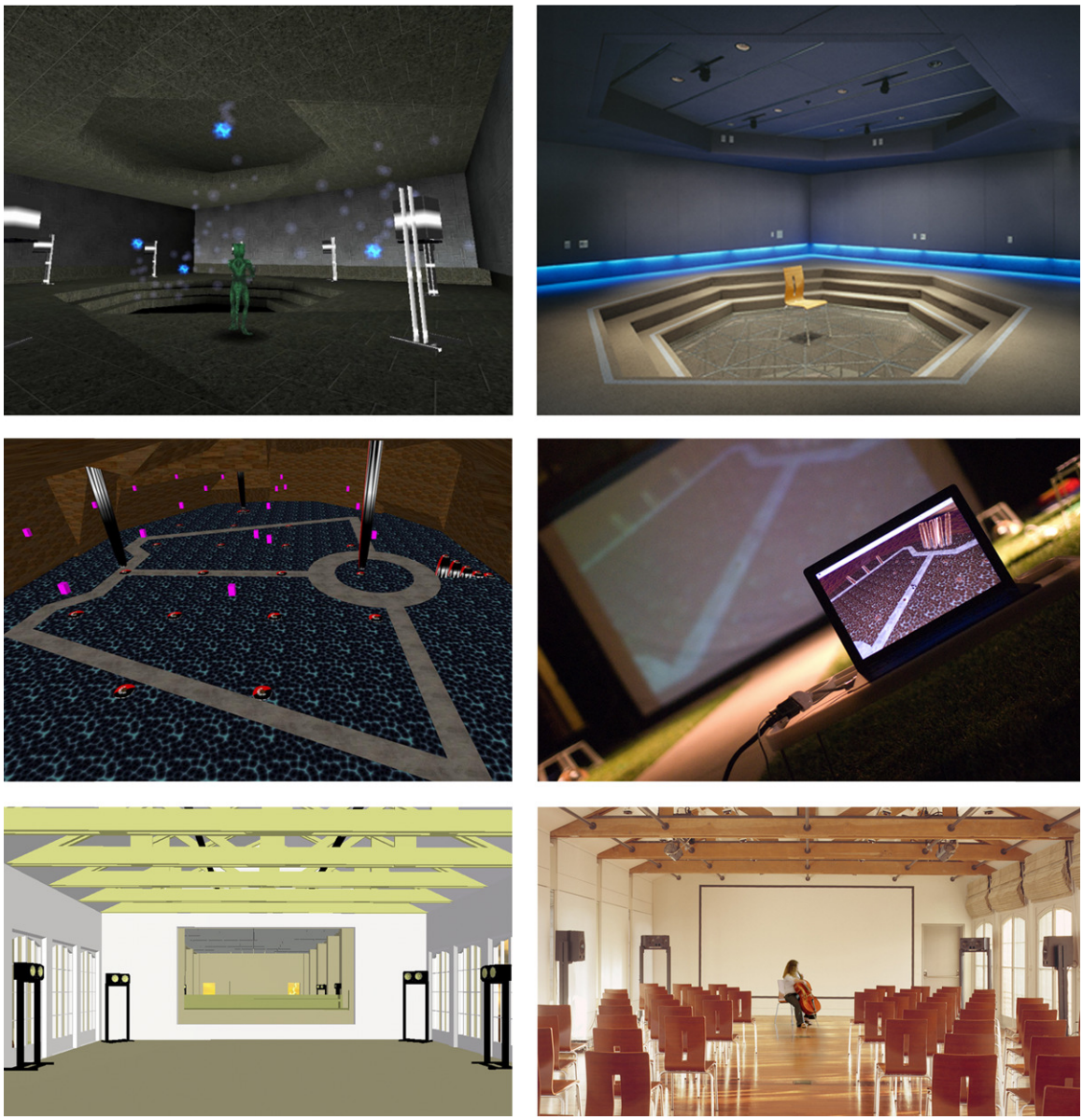


Figure 3.34: Virtual replicas and real-world performance spaces are paired, showing in descending order (1) the CCRMA Listening Room, (2) the CCRMA Courtyard from *nous sommes tous Fernando...* and (3) the CCRMA Stage from *Dei Due Mondi*.

between the hall in Milan and the Stage at CCRMA, virtual replicas of both spaces were included in the *Dei Due Mondi* environment, linked by an opening where the video screens were located in each space. Virtual performers could then move freely between the two rendered spaces, engaging other performers in either virtual space. This spatio-centric approach meant that members of the audience were immersed in an audio rendering designed to both fill the space and show the 1:1 relationship between avatar motion in virtual space and sound motion in physical space.

3.7 Summary

A series of multimodal software environments and interactive musical works has been profiled in this chapter. The q3osc project which integrated Open Sound Control messaging with the Quake III gaming environment was presented alongside the performance work *nous sommes tous Fernando....* The Sirikata project was used to create a series of musical performance works for a 2009 networked concert event connecting real and virtual performers in Milan, Italy with real and virtual performers in California and Montana, USA. The UDKOSC project was discussed with special attention given to the recent performance work *ECHO::Canyon* as well as the interaction schema, character avatars and data mapping techniques used in that work. General techniques for the sonification of projectile, avatar and skeletal mesh data were discussed as well as techniques relating to the spatial super-imposition of sound objects within virtual representations of actual physical spaces.

Chapter 4

Crossmodal Dataset

To examine potential crossmodal correlations between parameters of motion and parameters of sound, an audio-visual dataset consisting of musically sonified avatar interactions was recorded, using UDKOSC to both control avatar motion and record parameter data. Attributes of motion including avatar speed, rotation and height were mapped to parameters of a physically-modeled clarinet. Motion attribute data was linearly scaled for each mapped instrument parameter, so that a noticable change in the parameter would be experienced by subjects viewing and listening to the examples. The attributes of sound driven by motion data include Frequency, Breath Pressure and Amplitude.

4.1 Overview

Short motion sequences were programmed using OSCControl and streamed in real-time into UDKOSC. Visual output from UDKOSC was captured to file using a Black-Magic Intensity Extreme HDMI digital video capture card while OSC messages representing the avatar's motion and action were recorded as binary timestamped packets embedded into YAML markup using OSCRecorder. Sonifications were recorded by playing back recorded YAML data with OSCPlayer and streaming that data into Supercollider. Audio files were recorded in Supercollider and subsequently combined with the recorded video footage using a FFMPEG batch process [5]. Syncronization

between video and audio examples was handled using an inline ChucK script to calculate the onset time of the recorded audio event using UAna [121] RMS analysis and then trimming the audio excerpt to begin at that onset time. A shell script running FFMPEG merged each audio and visual example and created both .mp4 and .webm formatted video files to accomodate HTML5 video playback across a variety of user web browsers.

4.2 Visual Stimuli

Six examples of avatar motion were recorded depicting the same humanoid avatar running in various patterns across a simple room (Fig. 4.1). The primary attributes of motion exhibited in these examples were speed, rotation, and coordinate height. The scene was lit in such a way as to show depth of field in an otherwise feature-sparse environment. In the center of the space was a simple pedestal construct, similarly used to establish depth of field. For all visual examples used in this study, a static camera position was chosen to frame the entire sequence of motion without changing a viewer's position or angle of perspective. For each example detailed below, the units of speed used can be thought of as Unreal-Units/second. An Unreal-Unit (UU) is a constant unit of measurement in the game environment and represents approximately 0.02 meters in scale [41].

4.2.1 OSCControl Scripting

Each motion sequence was scripted using OSCControl's simple syntax for defining camera and avatar movements over time. Scripts are written into simple text files and included in a command-line call to the OSCControl ruby script. For example, to run the scripted jump sequence described in Figure 4.10 and detailed in Figure 4.12, passing OSC output to a locally-networked IP address (10.0.1.101) on Port number 7001, the following command would be used:

```
$ ruby osccontrol.rb 10.0.1.101 7001 ./jump.oc
```

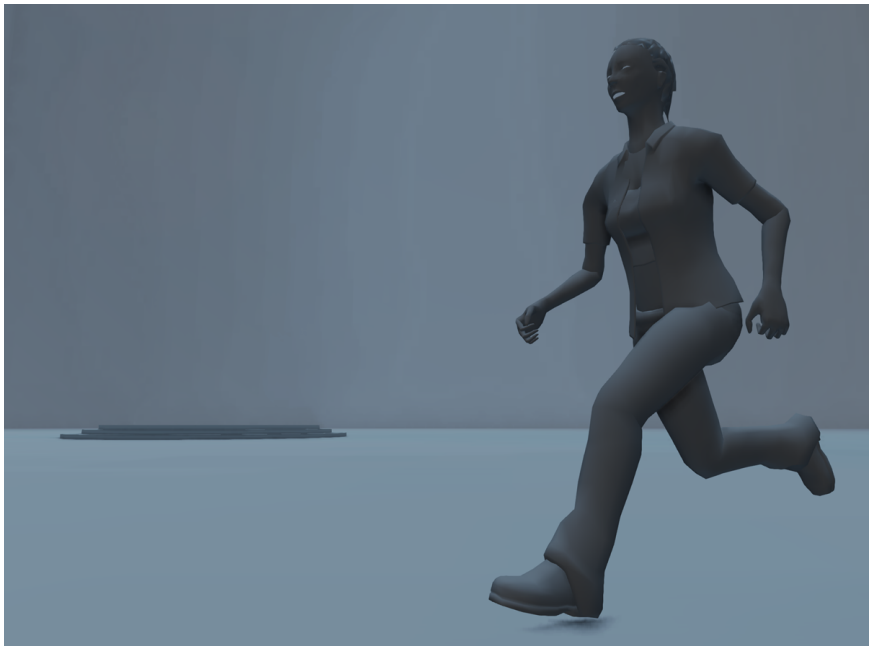


Figure 4.1: A human-like avatar with walking and running animations was used to generate each crossmodal example.

A more detailed description of OSCControl and the control methods it supports can be found in Appendix B.

4.2.2 Avatar

A humanoid avatar complete with walk and run animation sequences was used in this study with the intent of presenting a visual actor similar to those used in many computer game environments. The figure used was simply styled and colored so as not to distract user attention with excess detail or feature. An assumption was made that by using a humanoid avatar complete with simple yet decidedly human-like animations for limb motion and gait, the motion of individual arms and legs would be understood as a natural process for locomotion, and would not cause undue concern for subjects attending to the examples. Rather than focus these studies on simply-shaped avatars (such as spheres or cubes) which are not representative of the majority of avatars used in virtual games and social environments, the decision was

made to use a simple yet articulated skeletal mesh complete with basic animations.

4.2.3 Camera Position and Angle

When viewing motion in three-dimensional space without the ability to dynamically control the position or angle of view, the effect on actor size and scale should be considered. For example, avatar motion away from the camera will result in a decrease in the perceived size of the avatar. A series of static and dynamic camera treatments were recorded for each motion example to allow for future testing of the potential influence of camera location and angle.

Static Fixed Camera To present actor motion with no additional visual influence by the camera, a simple static camera view was used. Camera positioning was chosen so that entire actor motion sequences could be carried out without the actor leaving the camera's field of view. Static camera positions were chosen for each example starting in a similar location, positioned to the right side of the actor. In this position, if the actor were to move forward with no rotation, it would move in a parallel path across the camera's plane of view from left to right. For examples exhibiting actor rotation, the camera was positioned in such as way as to capture the entire motion sequence without the visible scale of the actor getting too small to be seen.

Static Follow Camera By modifying a simple static camera to allow for rotation along the horizontal and/or vertical axis, a fixed position camera that tracks the motion of a virtual actor is created. Depending on its coordinate placement, the perspective of view as presented by a static follow camera, can vary greatly. When used to track motion sequences with significant variance in horizontal or vertical rotation, the camera rotation can vary significantly, potentially emphasizing the actor's own rotation.

Moving Camera A camera can be "attached" to a moving actor, so that the location of the camera is moving through space alongside the actor. This camera view introduces the potential to track more complex actor motions. Moving cameras

can be set to follow the actor at all times, always keeping the actor in the visible frame, or can be fixed, always focusing in a single direction or angle.

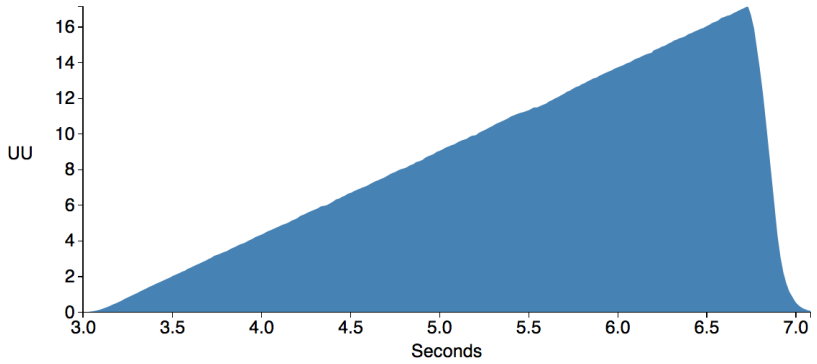
4.2.4 Acceleration

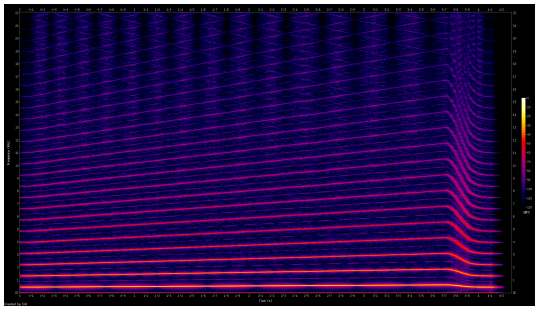
Avatar forward acceleration or an increase in speed was displayed by starting the avatar at a stationary position at the left of the screen and linearly increasing the speed of forward motion until coming to an abrupt stop at the far right side of the screen. Over a duration of 3500 ms, speed increased from 0 to 1000 UU per second.

Figure 4.2: Sequence showing accelerating forward motion with no turning/rotation.

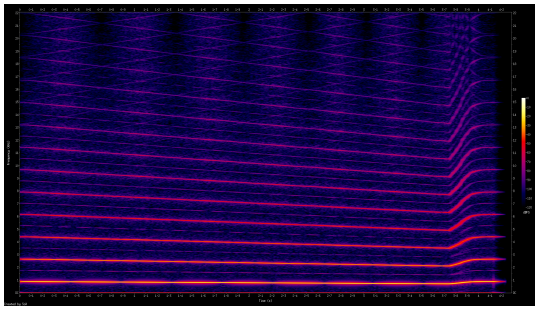
The OSCControl command sequence for this example (see Figure 4.2) was:

```
1  console freecameraon
2  cameramove z 240
3  cameramove y 2450
4  cameramove x 1300
5  cameramove pitch 10
6  cameramove yaw -90
7  playermove teleport 300 1300 10 0
8  wait 1000
9  console setstarton
10 wait 3000
11 console setstartoff
12 playermove speed 0 0
13 playermove setyaw 0 0
14 wait 3000
15 playermove x 1 0
16 playermove yaw 0 0 0
17 playermove speed 1000 3500 0
18 playermove stop 0
19 wait 2000
20 console setend
```

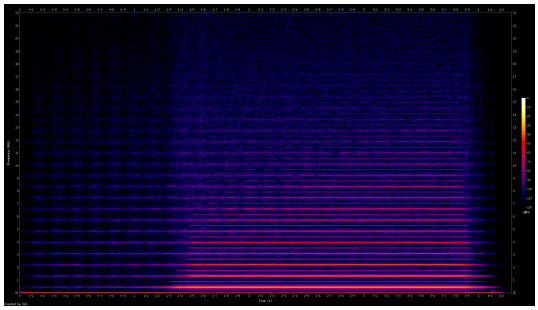




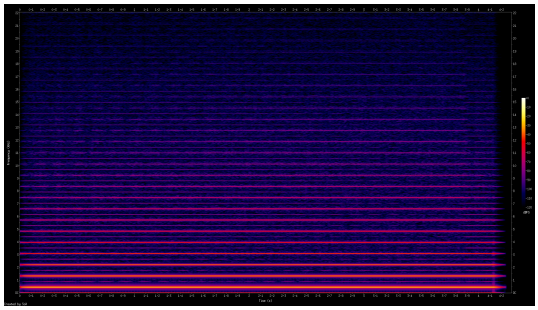
Speed to Frequency (direct)



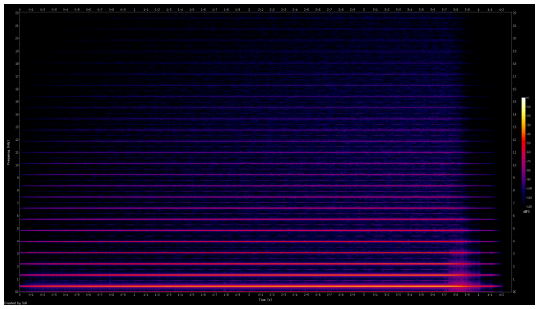
Speed to Frequency (inverse)



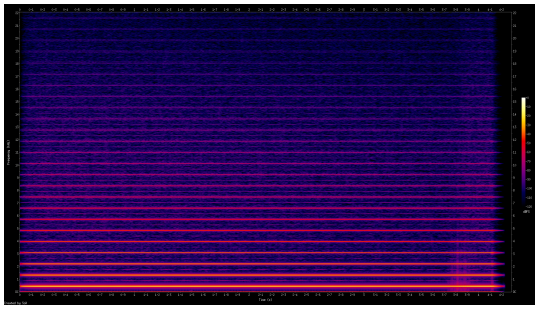
Speed to Breathpressure (direct)



Speed to Breathpressure (inverse)



Speed to Amplitude (direct)



Speed to Amplitude (inverse)

Figure 4.3: **Acceleration**: Raw OSC data and generated spectrograms for each crossmodal sonification. Y-axis represents Frequency: 0 - 22 kHz. Color represents decibels relative to full-scale: 0 (white) - 120 (black) dBFS.

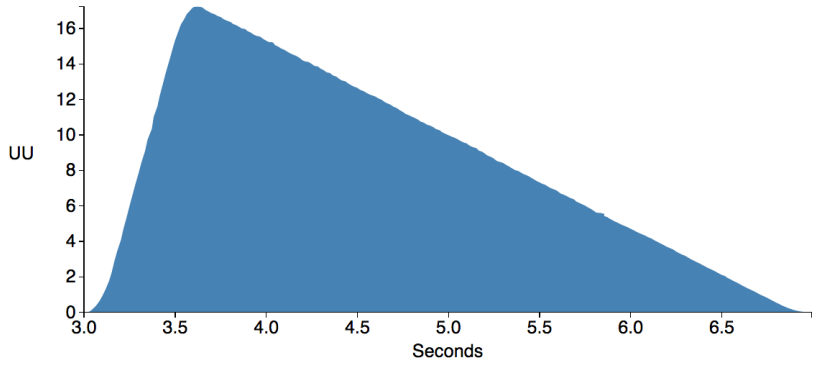
4.2.5 Deceleration

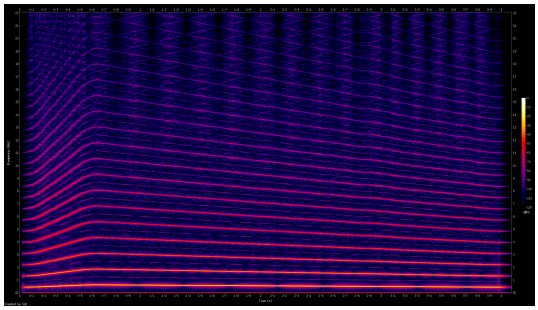
Avatar forward deceleration or a decrease in speed was displayed by starting the avatar at a stationary position at the left of the screen, initiating forward motion with a high speed value, then gradually decreasing the speed of forward motion until coming to an abrupt stop at the far right side of the screen. After 500 ms of forward speed of 1000, over a duration of 3100 ms speed decreased linearly from 1000 to 0 UU per second.

Figure 4.4: Sequence showing decelerating forward motion.

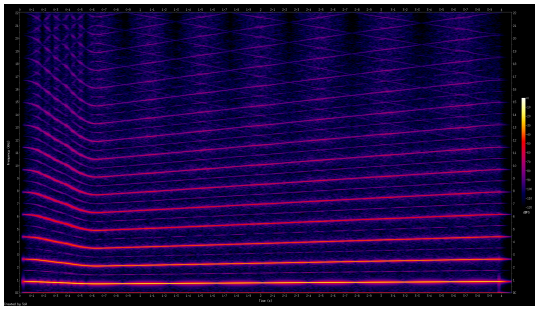
The OSCControl command sequence for this example (see Figure 4.4) was:

```
1  console freecameraon
2  cameramove z 240
3  cameramove y 2450
4  cameramove x 1300
5  cameramove pitch 10
6  cameramove yaw -90
7  playermove teleport 300 1300 10 0
8  wait 1000
9  console setstarton
10 wait 3000
11 console setstartoff
12 playermove speed 0 0
13 playermove setyaw 0 0
14 wait 3000
15 playermove x 1 0
16 playermove yaw 0 0 0
17 playermove speed 1000 0
18 wait 500
19 playermove speed -1000 3100 0
20 playermove stop 0
21 wait 2000
22 console setend
```

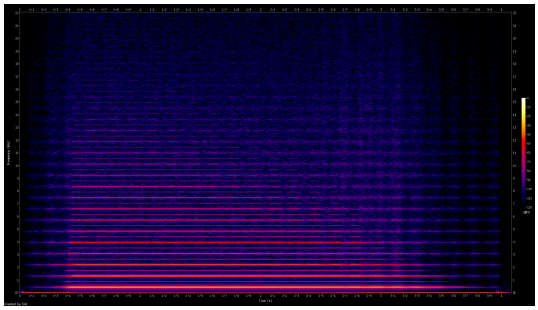




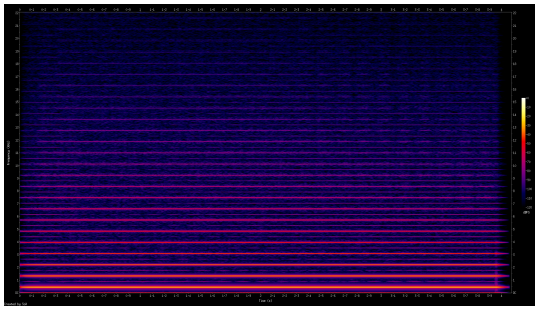
Speed to Frequency (direct)



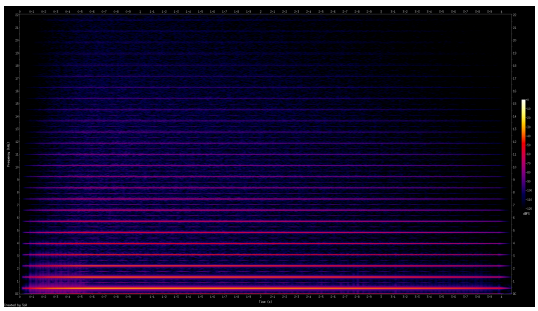
Speed to Frequency (inverse)



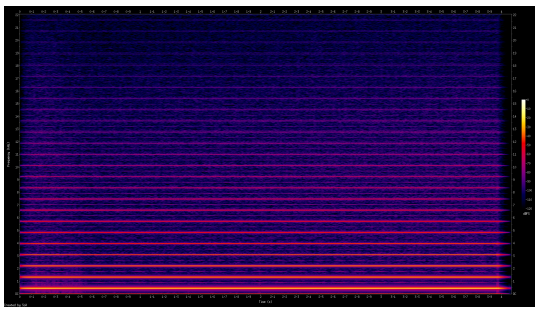
Speed to Breathpressure (direct)



Speed to Breathpressure (inverse)



Speed to Amplitude (direct)



Speed to Amplitude (inverse)

Figure 4.5: **Deceleration**: Raw OSC data and generated spectrograms for each crossmodal sonification. Y-axis represents Frequency: 0 - 22 kHz. Color represents decibels relative to full-scale: 0 (white) - 120 (black) dBFS.

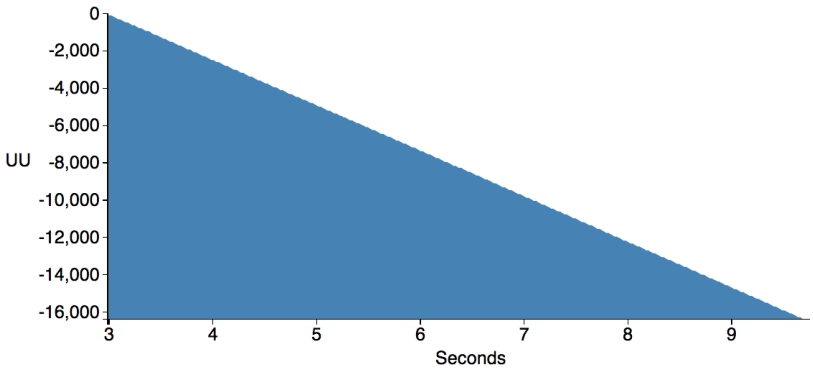
4.2.6 Continuous Rotation

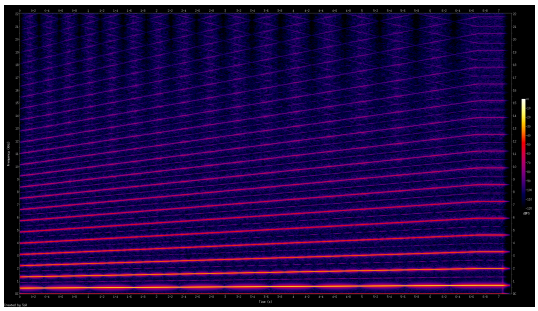
Forward avatar motion coupled with a linear increase in rotation was used to create a gradually curving path, with the avatar rotating 90 degrees over a period of 6300 ms. The direction of rotation turned the avatar away from the camera view, effectively moving the avatar into the visual scene's distance. For the purposes of this study, this direction was defined as an "increase" in rotation, in contrast to the negative direction of Unreal rotation values.

Figure 4.6: Sequence showing forward motion with continuous rotation.

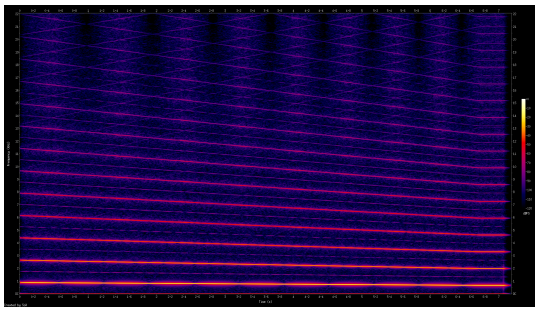
The OSCControl command sequence for this example (see Figure 4.6) was:

```
1  playermove setyaw 0 0
2  console freecameraon
3  cameramove z 240
4  cameramove y 2450
5  cameramove x 1300
6  cameramove pitch 10
7  cameramove yaw -90
8  playermove teleport 300 1300 10 0
9  wait 1000
10 console setstarton
11 wait 3000
12 console setstartoff
13 playermove speed 0 0
14 wait 3000
15 playermove x 1 0
16 playermove yaw 0 0 0
17 playermove speed 700 0
18 playermove yaw -90 6300 0
19 playermove stop 0
20 wait 2000
21 console setend
```

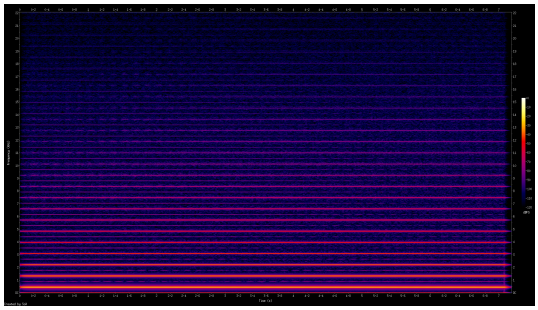




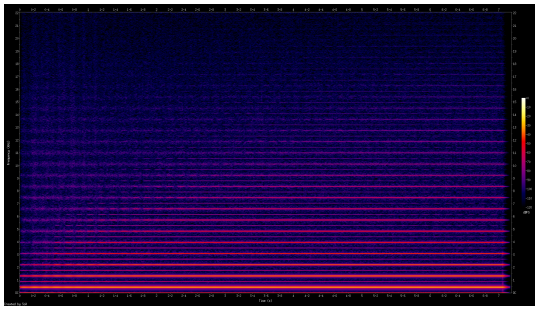
Rotation to Frequency (direct)



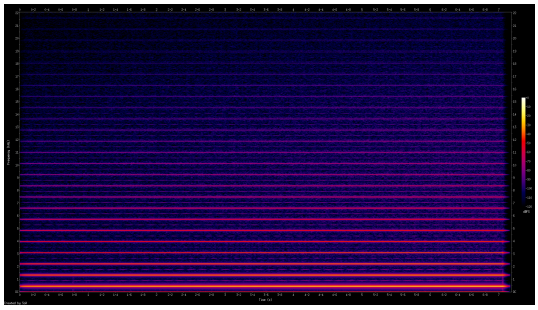
Rotation to Frequency (inverse)



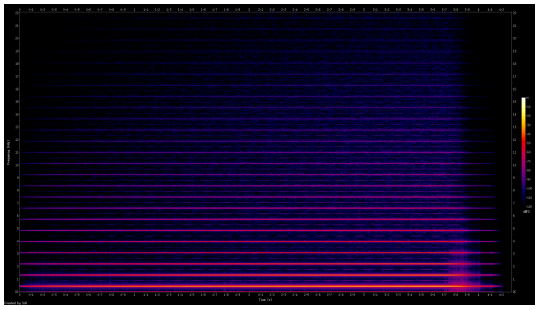
Rotation to Breathpressure (direct)



Rotation to Breathpressure (inverse)



Rotation to Amplitude (direct)



Rotation to Amplitude (inverse)

Figure 4.7: **Continuous Rotation**: Raw OSC data and generated spectrograms for each crossmodal sonification. Y-axis represents Frequency: 0 - 22 kHz. Color represents decibels relative to full-scale: 0 (white) - 120 (black) dBFS.

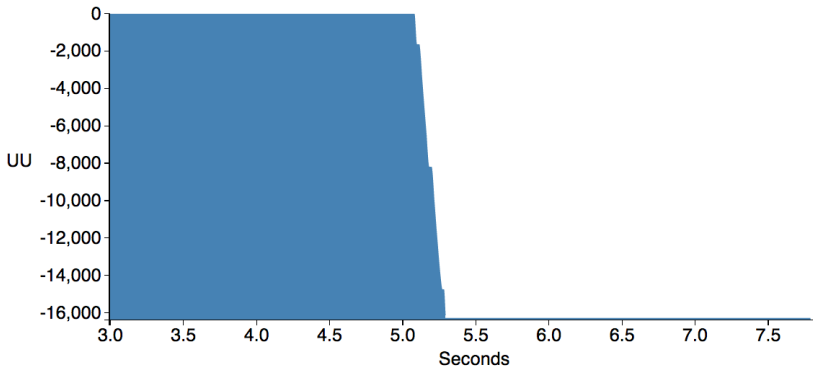
4.2.7 Discrete Rotation

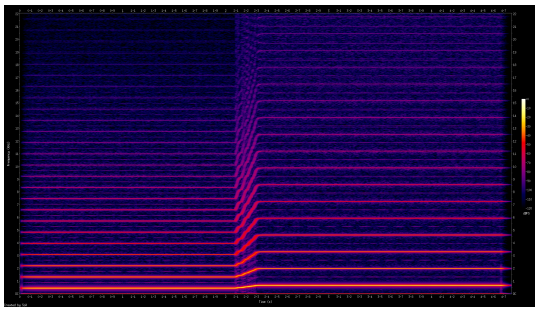
Forward avatar motion was coupled with a discrete increase in rotation to create a sharp 90 degree left turn in the avatar's path. In this example, rotation was quickly slewed over 200 ms to avoid potentially distracting visual artifacts while still presenting the directional change to participants as a discrete event rather than a continuous process. Like the examples of continuous rotation, the direction of rotation turned the avatar away from the camera view, effectively moving the avatar into the visual scene's distance.

Figure 4.8: Avatar forward motion with discrete turn event.

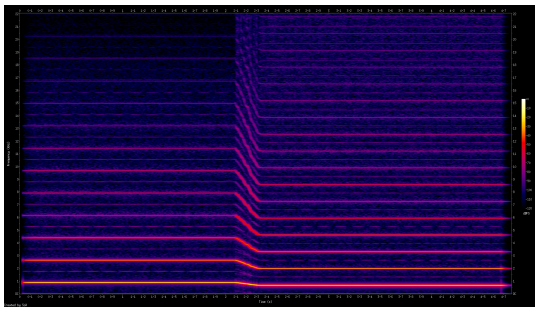
The OSCControl command sequence for this example (see Figure 4.8) was:

```
1  playermove setyaw 0 0
2  console freecameraon
3  cameramove z 240
4  cameramove y 2450
5  cameramove x 1300
6  cameramove pitch 10
7  cameramove yaw -90
8  playermove teleport 300 1300 10 0
9  wait 1000
10 console setstarton
11 wait 3000
12 console setstartoff
13 playermove speed 0 0
14 wait 3000
15 playermove x 1 0
16 playermove yaw 0 0 0
17 playermove speed 700 0
18 wait 2100
19 playermove yaw -90 200 0
20 wait 2000
21 playermove stop 0
22 wait 2000
23 console setend
```

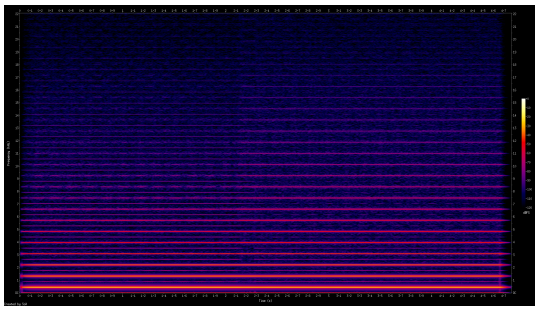




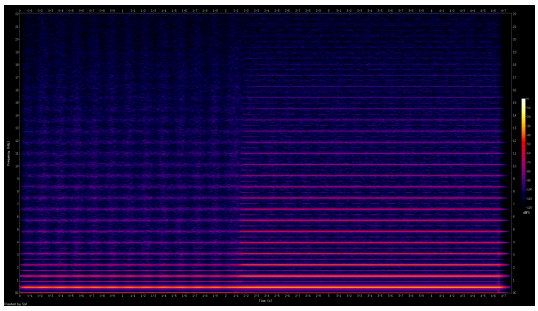
Rotation to Frequency (direct)



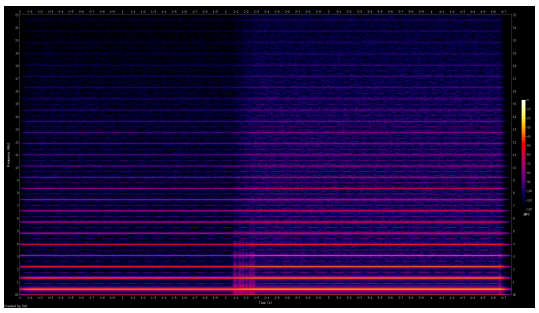
Rotation to Frequency (inverse)



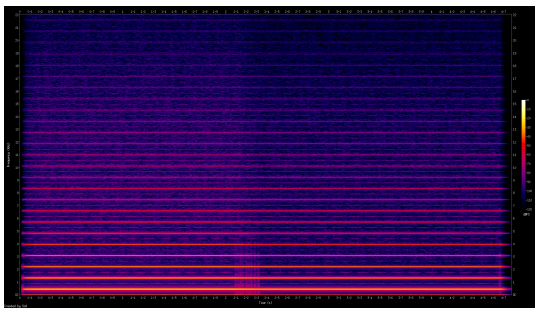
Rotation to Breathpressure (direct)



Rotation to Breathpressure (inverse)



Rotation to Amplitude (direct)



Rotation to Amplitude (inverse)

Figure 4.9: **Discrete Rotation**: Raw OSC data and generated spectrograms for each crossmodal sonification. Y-axis represents Frequency: 0 - 22 kHz. Color represents decibels relative to full-scale: 0 (white) - 120 (black) dBFS.

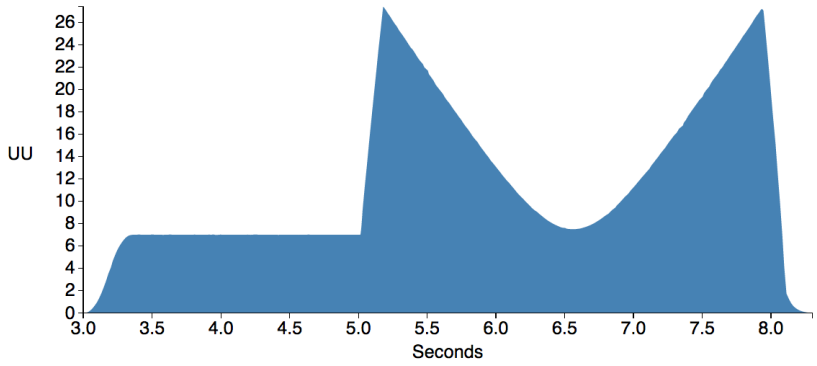
4.2.8 Jump Event

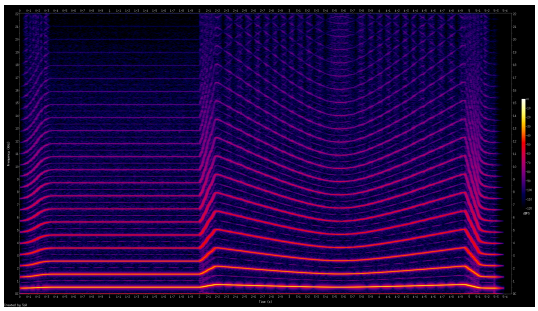
To explore vertical avatar motion, a jump event was used, creating an increase in the avatar's Z-coordinate followed by a matching decrease as the avatar landed.

Figure 4.10: Avatar forward motion with jump event.

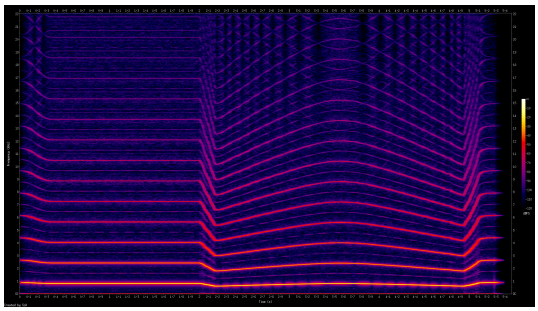
The OSCControl command sequence for this example (see Figure 4.10) was:

```
1  console freecameraon
2  cameramove z 240
3  cameramove y 2650
4  cameramove x 1300
5  cameramove pitch 14
6  cameramove yaw -88
7  playermove teleport 300 1400 10 0
8  wait 1000
9  console setstarton
10 wait 3000
11 console setstartoff
12 playermove speed 0 0
13 playermove setyaw 0 0
14 wait 3000
15 playermove x 1 0
16 playermove yaw 0 0 0
17 playermove speed 400 0
18 wait 2000
19 playermove jump 1600 0
20 wait 1000
21 playermove stop 0
22 wait 2000
23 console setend
```

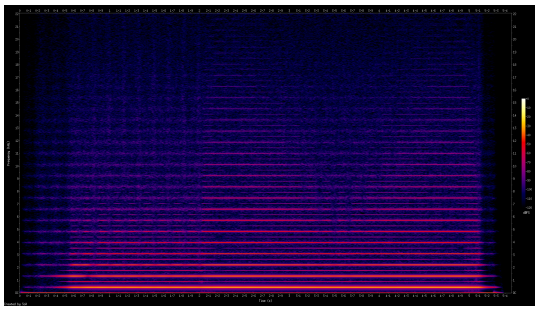




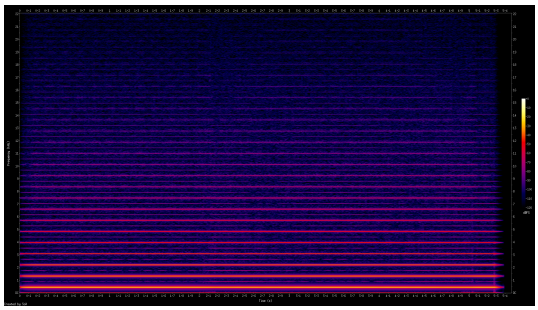
Speed to Frequency (direct)



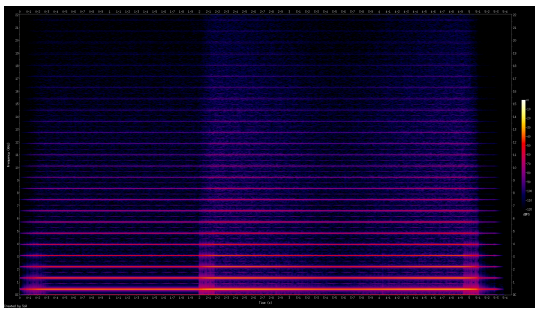
Speed to Frequency (inverse)



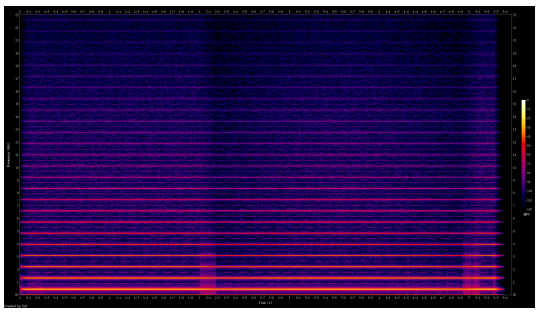
Speed to Breathpressure (direct)



Speed to Breathpressure (inverse)

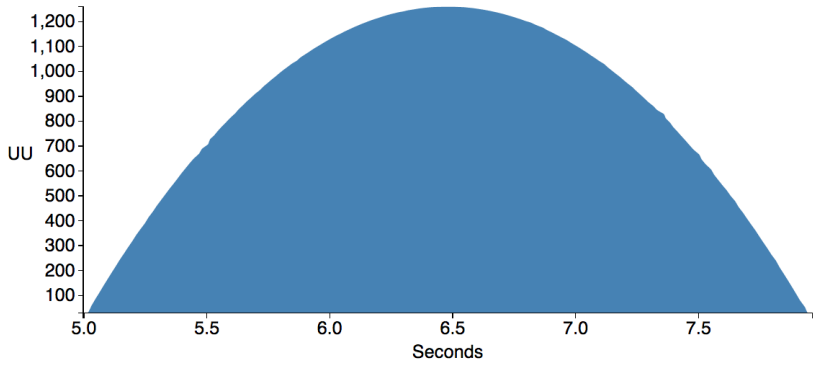


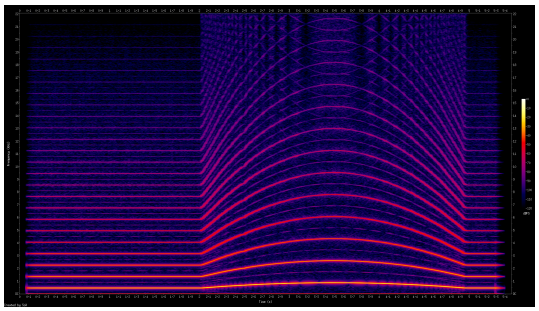
Speed to Amplitude (direct)



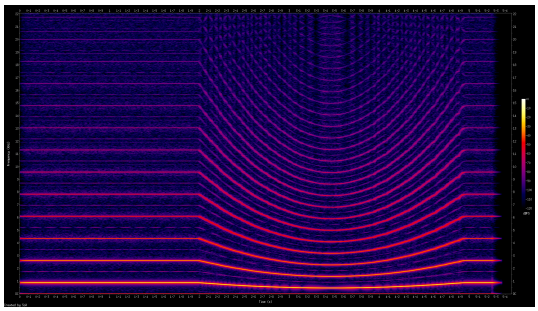
Speed to Amplitude (inverse)

Figure 4.11: **Jump Speed**: Raw OSC data and generated spectrograms for each crossmodal sonification. Y-axis represents Frequency: 0 - 22 kHz. Color represents decibels relative to full-scale: 0 (white) - 120 (black) dBFS.

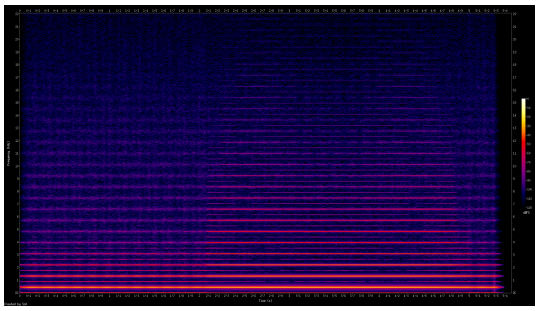




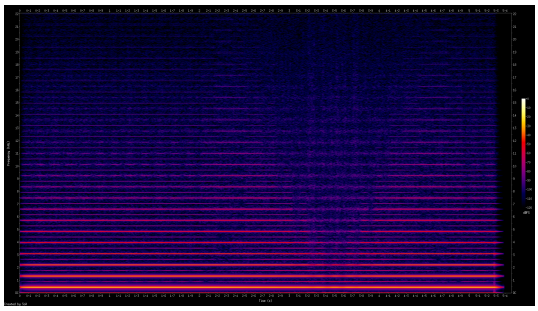
Height to Frequency (direct)



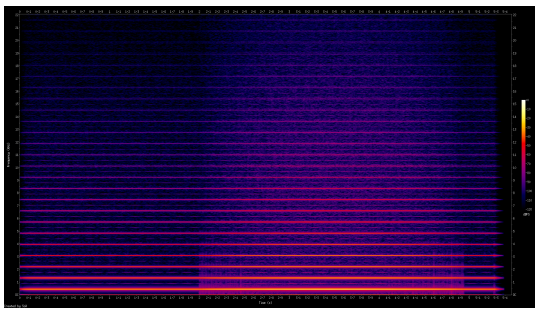
Height to Frequency (inverse)



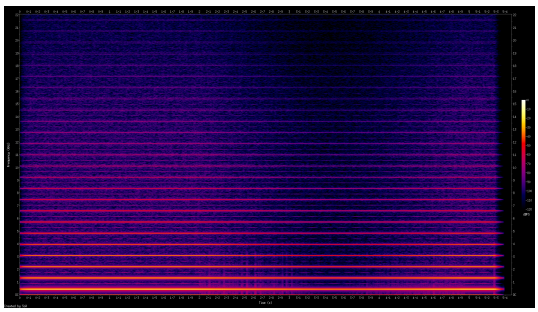
Height to Breathpressure (direct)



Height to Breathpressure (inverse)



Height to Amplitude (direct)



Height to Amplitude (inverse)

Figure 4.12: **Jump Height**: Raw OSC data and generated spectrograms for each crossmodal sonification. Y-axis represents Frequency: 0 - 22 kHz. Color represents decibels relative to full-scale: 0 (white) - 120 (black) dBFS.

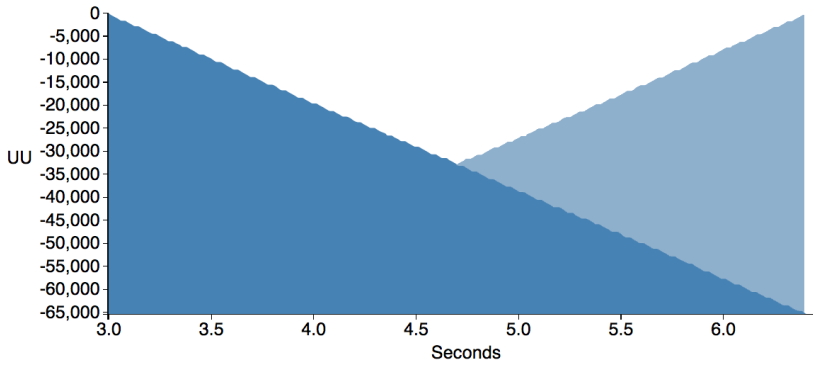
4.2.9 Circle

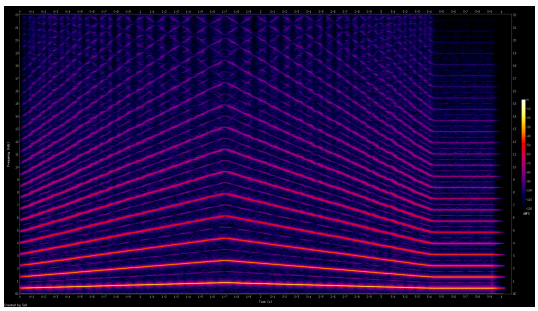
A complete 360 degree rotation was coupled with forward motion to create a circular path. In order to aid participants in viewing the circular shape in its entirety, the size of the circle and the duration of the circular path were kept small. Rotation for the circle example was tracked relative to the avatar's starting rotation meaning that the mapped output parameter was at its highest point when the avatar had rotated 180 degrees. As rotation increased from 180 to 360 degrees, the level of output decreased until it returned to its starting value. This can be seen as the top contour traced by the upper-bound of dark blue and light blue lines in Figure 4.14.

Figure 4.13: Avatar forward motion with full circle.

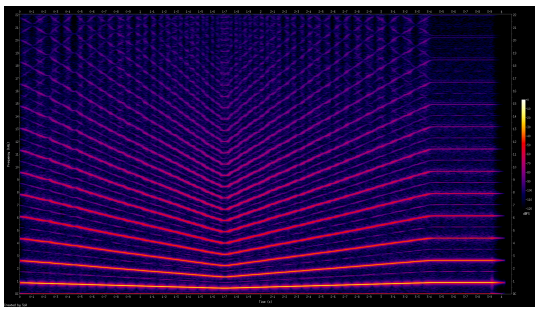
The OSCControl command used to generate this example (see Figure 4.13) was:

```
1  playermove setyaw 0 0
2  console freecameraon
3  cameramove z 240
4  cameramove y 2150
5  cameramove x 1300
6  cameramove pitch 10
7  cameramove yaw -110
8  playermove teleport 300 1400 10 0
9  wait 1000
10 console setstarton
11 wait 3000
12 console setstartoff
13 playermove speed 0 0
14 wait 3000
15 playermove x 1 0
16 playermove yaw 0 0 0
17 playermove speed 700 0
18 playermove yaw -360 3200 0
19 wait 142
20 playermove stop 0
21 wait 2000
22 console setend
```

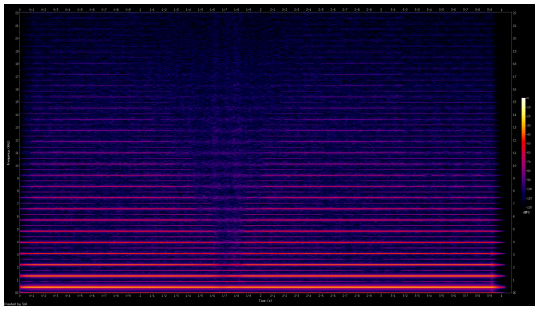




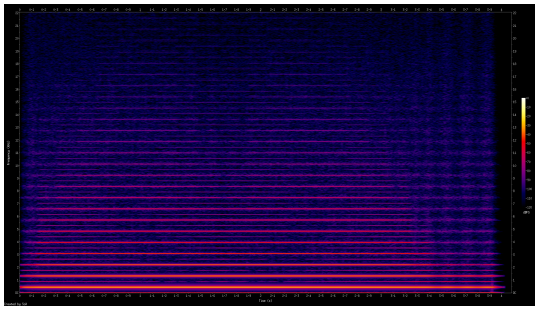
Rotation to Frequency (direct)



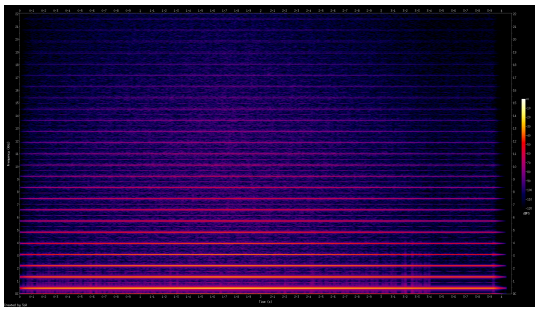
Rotation to Frequency (inverse)



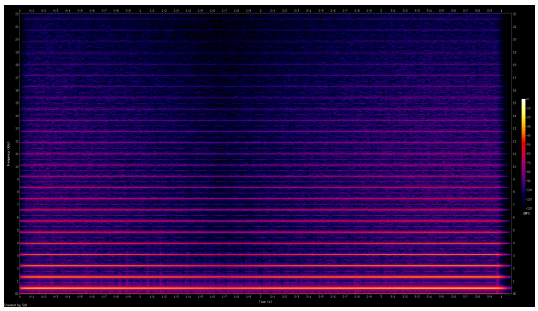
Rotation to Breathpressure (direct)



Rotation to Breathpressure (inverse)



Rotation to Amplitude (direct)



Rotation to Amplitude (inverse)

Figure 4.14: **Circle Rotation**: Raw OSC data and generated spectrograms for each crossmodal sonification. Y-axis represents Frequency: 0 - 22 kHz. Color represents decibels relative to full-scale: 0 (white) - 120 (black) dBFS.

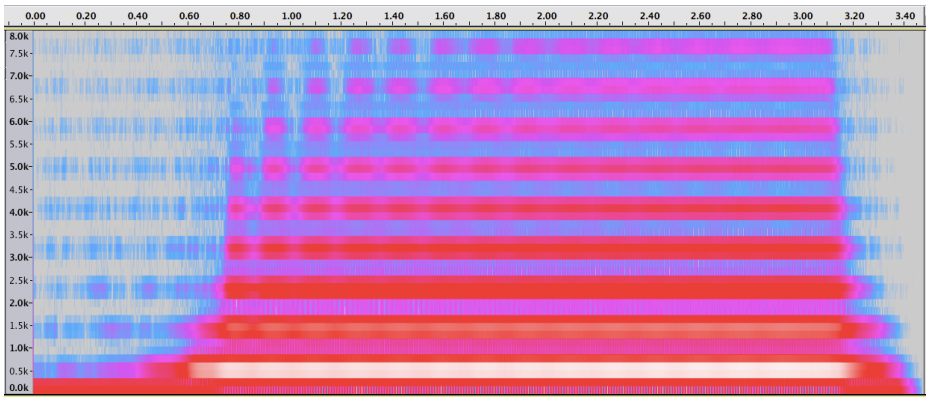


Figure 4.15: Reference spectrogram of the STK Clarinet model.

4.3 Audio Stimuli

Audio tracks for each video excerpt were generated by mapping individual parameters of motion to individual parameters of sound for a physically-modelled clarinet instrument. Each mapping schema was mapped both directly and inversely, that is to say that where a direct mapping would show a linear increase in the target parameter value for a given linear increase in the source parameter, an inverse mapping would show a linear decrease in the target parameter for the same linear increase in source.

4.3.1 Instrument: STK Clarinet

To create sonifications that exhibited musical characteristics alongside simple sonic characteristics, the musical sonifications used in this example set were generated using a physical model of a clarinet from the Synthesis Toolkit [27].

The use of a more complex software instrument for motion sonification allowed the exploration of various consistent parameter sets, such as breath pressure and vibrato, that are idiomatic to the sound source and still highly flexbile. The implementation of the instrument in Supercollider can be seen in Figure 4.16.

```
1 SynthDef("stkClarinet",{arg freq=440, reedstiffness=64, noisegain=10, vibfreq
      =64, vibgain=10,
2     breathpressure=64, gain=0.2, gate=1, bus=0, lag= 0.1, sloc=0, riset=0.2,
          decayt=0.2;
3     var z, env;
4     env=EnvGen.kr(Env.adsr(attackTime:riset, decayTime:0,sustainLevel:1,
5         releaseTime:0.1,peakLevel:1), gate:gate, doneAction:0);
6     z=StkClarinet.ar(freq:Lag.kr(freq, lag), reedstiffness:reedstiffness,
          noisegain:noisegain,
7        vibfreq:vibfreq, vibgain:vibgain, breathpressure:breathpressure, trig:
             gate);
8     Out.ar(bus, Pan2.ar(z,sloc)*env*gain);
9 }).load(s);
```

Figure 4.16: Supercollider instrument definition for the STK Clarinet instrument.

4.3.2 Parameter Mappings

The mapping of parameter data from each motion sequence to sound-generating synthesis process focused on parameters that exhibited change during the recorded motion event. The parameters of motion recorded for each motion example included:

- Speed
- Rotation
- Height

The parameters of sound driven by motion parameter mappings included:

- Frequency
- Amplitude
- Breath Pressure

4.3.3 Parameter Mapping Ranges

The mapping of motion parameter data to control data for the STK Clarinet was always performed linearly with no complex processing. Using the Supercollider linlin() function, incoming motion data within a given range was mapped linearly to

a pre-defined control range. The control ranges for each mapping were designed to demonstrate a significant variance in sound across the full range of motion input data. Table 4.1 shows mapping ranges used for the example sets that were subsequently used in user studies. Each range was applied to both direct and inverse mappings for each motion example in the set that exhibited a change in the given parameter.

Motion Parameter	Sound Parameter	Parameter Range (motion)	Parameter Range (sound)
Speed	Frequency	0.0 - 42.0	440.0 - 880.0
Speed	Amplitude	0.0 - 42.0	0.01 - 0.4
Speed	Breath Pressure	0.0 - 42.0	64 - 128
Rotation	Frequency	0 - 180.0	880.0 - 440.0
Rotation	Amplitude	0 - 180.0	0.5 - 0.1
Rotation	Breath Pressure	0 - 180.0	70 - 128
Height	Frequency	0.0 - 1300.0	440.0 - 880.0
Height	Amplitude	0.0 - 1300.0	0.1 - 0.5
Height	Breath Pressure	0.0 - 1300.0	70 - 128

Table 4.1: Parameter mapping ranges for the STK Clarinet instrument used in this user study. For each motion and sound pairing, both input (motion) and output (clarinet sound) parameters were linearly scaled to fit into an audible and clearly-discernable range.

4.4 Summary

This chapter has described a crossmodal dataset created by recording video examples of avatar motion in virtual space alongside the musical sonification of individual parameters tracked using the UDKOSC engine and Open Sound Control messages. Simple avatar motions were defined using the OSCControl scripting language including examples showing noticable changes in avatar speed (acceleration, deceleration), rotation (a discrete turn, a continuous curve and a circle event) and height (a jump event). The techniques used to create these example were discussed alongside the representation of each motion sequence and generated audio stimulus.

Chapter 5

Evaluating Perceived Coherence

Humans are natural viewers and listeners. As such we perceive countless sound-generating interactions each and every day. Our expectations for how any given interaction should sound as well as our abilities to perceive the causalities behind sonic events are shaped by a combination of our own embodied actions, our observations of the surrounding world and our understandings of the physics of our universe. Having acted and perceived interactions in the past we have learned to reasonably anticipate the audible result of not-yet perceived interactions between similar or previously understood objects and actions. Through our knowledge of how forces act upon physical bodies we can generalize and to a certain extent predict how different events and interactions will sound, at least within the confines of our known world. Put another way, we all know that we know that certain kinds of objects will create certain kinds of sounds when acted upon in certain ways.

Imagine a scene: two individuals raise their wine-glasses for a celebratory toast. With no additional information we can already build a mental model of a likely sonic event generated by the impending interaction. Given additional pieces of information, such as the level of wine in each glass or the speed of each hand moving toward the impending collision, we can alter our mental projections of the potential sonic event, mentally adjusting the frequency spectrum of the resultant "clink" or the likelihood of the glasses shattering upon impact. Taken one step further, the knowledge that one glass was made of rubber would fundamentally change our expectation for how this

interaction will sound. Each of these mental projections and expectations are made based on a combination of our understandings about how general types of objects and materials have interacted in the past (an experiential or causal understanding) as well as our understandings about how similarly characterized objects and materials act and interact based on the laws of physics in our known universe (a mechanistic understanding).

Now imagine the same scene while disregarding everything you know about hands, glass and wine, mass, inertia and force, sound and speed. Can you do it? Is it at all possible to put aside years of accumulated understanding and experience? Even if it were possible to block out the significant existing mental noise from such past experiences and understandings, we would be left without any basis upon which to predict this interaction's sonic outcome. Without experience or understanding to help shape our internal predictive models, we could not comprehend that an interaction of any sort was imminent. While not a likely scenario or need in the real-world, within the context of interactive and immersive audio-visual environments, users are routinely presented with novel virtual realities, complete with novel actions and interactions, each capable of generating visual, kinetic and auditory responses. And while designers and developers commonly base virtual experiences upon interaction paradigms from physical reality, users of such systems are often required to make important decisions based on events and interactions that would not be possible outside of a rendered virtual system.

When presented with sonic events in a rendered virtual environment - a potentially novel reality within which observers' *a priori* and *a posteriori* knowledge about sound, image and interactivity are no longer necessarily valid - observers must draw conclusions about perceived sonic interactions without full benefit of their own knowledge or past experiences. Their internal sonic lexicons, developed by everyday physical interactions during a lifetime of observation and experience, are intrinsically based in the laws of the familiar physical world. Within a generated environment where the rules governing interactions and "reality" itself are subject to the whim of a software designer or developer, observers are often forced to look outside of their own experiences when forming associations between visual and sonic action.

One increasingly common interaction model that exists outside of our physical reality is formed through the linking of action, motion and gesture with processes capable of generating sound that is musical in form and function. As cinema and interactive gaming experiences have grown more complex and integrated with technological processing, the interplays between visual action and musical sound have grown more pronounced and more tightly intertwined. Choreographies of camera angle and on-screen action are routinely synchronized to musical elements in background musical presentations within motion pictures and music videos. In video game development it has become increasingly common to design sonic events generated within gameplay to seamlessly blend with the game's musical score [113]. And for games based around musical paradigms themselves, gesture and motion in both virtual and real-world environments are routinely mapped to dynamic musical generating and modification processes [56, 57, 120, 54].

5.1 Study Overview

This research explores the perception of crossmodal relationships or correspondences between actions and gestures performed in virtual space and procedurally-generated sound processes. During the course of an exploratory user study, subjects were presented with a series of audio-visual stimuli in the form of short videos depicting humanoid avatar motion within a rendered three-dimensional environment. Musical sound, generated by mapping parameters of avatar motion to sound generating processes, is audible to subjects while viewing each video. Each stimulus consisted of video captures recorded alongside real-time data streams of avatar coordinate motion and state data. Each simple musical sonification was generated by mapping parameters from each example's multidimensional data stream to a set of parameters of a physically modeled instrument. Composite audio-visual examples were created by attaching and syncronizing the musical sonfications to each video example.

Subjects using the Mechanical Turk online tasking platform [92] were asked to watch short two-video example sets of these musically sonified avatar motions in a pairwise comparison task, choosing the example with the greatest perceived coherence

or 'fit' between visual and auditory events. During analysis, each visual and audio example were defined by a combination of motion and sound descriptors, allowing for the statistical analysis of correlated motion/sound pairs. For each of these modal pairs a weighted fit value was calculated and then used to calculate rankings for each example across the entire sample set, resulting in a measure of the perceived fit or coherence between individual component pairs across modalities. The perceived fit of examples exhibiting individual attributes of motion and sound were also calculated and ranked. Additional analyses were performed to gauge the influence of mapping direction and contour on perceived fit, as well as a separate analysis investigating the perceived similarity between examples.

5.2 Research Questions

In the context of this study, analysis was conducted with three primary goals or questions in mind.

1. Which examples exhibited the strongest fit across all participants?
2. Which attributes were the most significant predictors of fit?
3. Which crossmodal attribute pairs were the most significant predictors of fit?

At the same time, a series of secondary goals or questions were considered.

1. How does the directionality of mapping affect fit?
2. Do examples exhibiting matched contours across modalities affect fit?
3. Which attribute pairings exhibited the highest degree of perceived Similarity?

Additional questions were considered along the way, including previous results found by Eitan and Granot that supported assymetrical relationships between perceived parameters of audio stimuli and visual stimuli [32].

5.3 Study Procedures

Study participants were presented with a pairwise comparison task and asked to choose the audio-visual example which exhibited the strongest fit between elements in the visual modality and elements in the auditory modality. The examples were short video files showing humanoid avatar motion in a game-like rendered space. For each example, one attribute of avatar motion was mapped to one attribute of sound and used to procedurally generate an audio track. The sound for each example was generated by sending a given parameter of motion's value using Open Sound Control output from UDKOSC to a real-time synthesis process running in Supercollider. In total, 480 examples across 861 randomly-ordered pairings were presented, representing each unique combination of 3 attributes of motion, 3 attributes of sound, 2 directional mapping schemata and 1 instrument type. The three attributes of motion tracked were actor *speed*, *height* in coordinate space and degree of *rotation*. The three attributes of sound modulated were *frequency* or pitch, *amplitude* or volume and *breath pressure*, presenting as a timbral shift in tone color for the physically-modelled clarinet instrument used for each example.

5.3.1 Amazon Mechanical Turk

The study was carried out online, within the Amazon Mechanical Turk online service, in which participants are paid to carry out Human Intelligence Tasks (HITs) [83]. An html page was prepared using Amazon's web survey templates coupled with custom javascript tracking and video playback code. Each video example was rendered as both .mp4 and .webm compressed video files and stored on CCRMA's webserver. To build a sample set for each Turk experiment, a comma-separated .csv file of 861 video filename pairs was uploaded to the Amazon system with one pairing passed into the web-form for each session of the experiment. Response and session data - including not only subjects' selected form answer fields but also a time duration for the task and boolean tracking varibles showing playback counts for each video example - were retrieved using Amazon's Turk web toolkit. In total 6027 HITs were processed, with each of the 861 pairings viewed by seven different participants.

5.3.2 Participants

219 unique subjects participated in this study for which each were paid $0.09 for the successful and valid completion of each HIT. To present the study to a diverse group of participants while still retaining a high level of accuracy and validity, Workers were required to have achieved a previous HIT Approval Rate (or percentage of approved HITs) greater than or equal to 90% across all previously submitted tasks. To mitigate potential language issues, Workers were limited to those users whose locations (as verified through Amazon's billing and payments system) were determined to be in the United States. On average, participants completed approximately 28 HITs with a maximum per-participant HIT count of 392. The entire set of 6027 HITs was processed in less than eight hours with an average time of 1 minute 49 seconds for each assignment and an average pay rate of approximately $2.97 per hour.

5.3.3 Study Format

Participants who qualified for the study and who chose this study's HIT from the Mechanical Turk Available HIT listings were presented with an html page displaying two sets of video pairs for pairwise comparison (see Figure 5.1) and three forced-choice questions in the form of html radio-buttons. The videos were presented using a side-by-side layout, with their respective order of presentation randomly determined for each individual presentation. The first set of videos were the actual sonified motion examples. The second set of videos consisted of confound examples, designed to determine if participants were properly attending to each video watched or just clicking through the study as quickly as possible. Participants were instructed to watch and listen to each video and, using the radio buttons, select the video in which the visual and auditory content exhibited the best 'fit'. There also existed the option to choose "Same" if participants believed the fit of each video was approximately equal. One additional question accompanied the first set of videos asking participants to rank the similarity between excerpts on a scale from one to seven, where one represented no similarity and seven represented a high level of similarity.

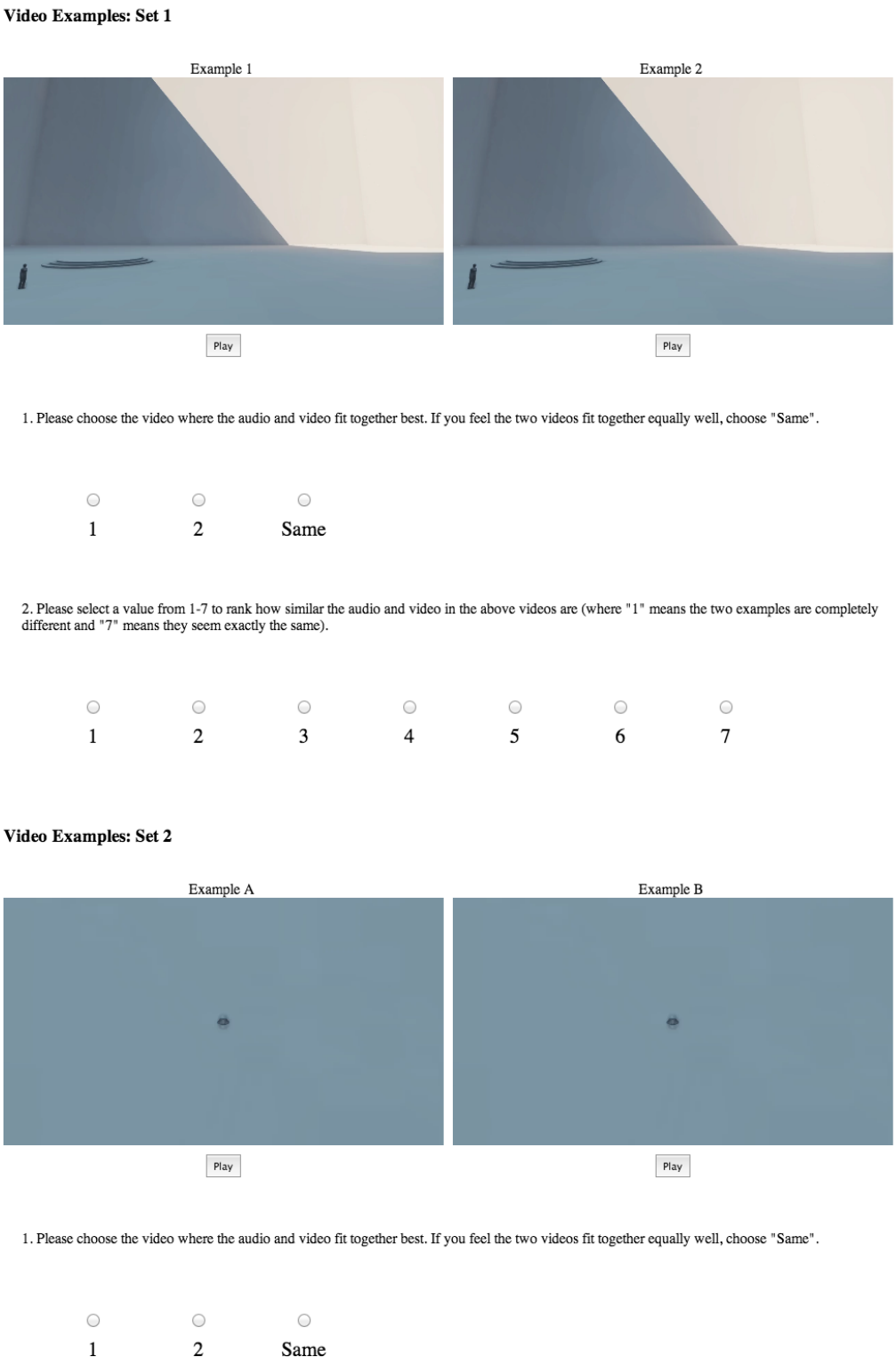


Figure 5.1: Excerpt from Turk study interface.

5.3.4 Data Validation

The Amazon Mechanical Turk is a for-pay service and as such there exists the possibility that participants taking part in any HIT might prioritize speed of completion over accuracy. To identify participants who may have not correctly completed the requested tasks, a series of validation techniques were used in this study. Result data from each HIT was audited by the study coordinator and any results which did not meet specific validation requirements were flagged and subsequently discarded.

Click counting The primary task of these HITs required the complete viewing of each short video example. If the videos were not watched, the HIT could not be considered valid. One potential technique would make use of a simple form validation which would disable the form's "Submit" button until each video had been viewed. However, since participants could work on multiple HITs, those participants trying to game the system would quickly learn how to click each video without necessarily attending to its content or engaging the required task. Instead, by simply tracking the number of times each video's "Play" button was clicked by users and passing that value to the Mechanical Turk engine, that play count was displayed in the results for the given task, allowing the study coordinator to flag any cases where each video had not been watched.

Task Duration If a minimum time threshold for the duration of the entire task was not exceeded, then the assumption was made that participants could not have watched the required videos in their entirety. Submitted HITs that failed to exceed the minimum time threshold had their results flagged and subsequently discarded. Across the entire 6057 HIT response set, the average work time duration was 109.32 seconds. A work time duration of 23 seconds or less was used to filter bad results from the subject set.

Confound Videos A set of confound videos was viewed by each participant as a second pairwise ranking task with the intention of determining whether or not participants were in fact attending to the video content or instead simply clicking

through the task as quickly as possible. One confound was designed to clearly exhibit a stronger fit than the other example so that users who selected the weaker fit choice could be flagged as potentially not paying attention. This was accomplished by delaying the sonification in the "poor" fit confound by approximately three seconds, creating one video that clearly exhibited a weaker fit between audio and visual modalities.

Each of the two confound examples consisted of the same visual recording of an avatar moving forward while rotating to the right, using a static camera angle positioned above and behind the avatar's location of origin. This motion example as well as this particular camera angle were not used for any example in the primary set. For the confound videos' sonification, the avatar's rotation was mapped directly to frequency.

Disqualified Trials During the manual validation stage, in which results submitted by users were examined by the study coordinator, 32 HITs were rejected based on null click count values for the two primary videos. As this stage of validation took place before approval and payment of each HIT, all of these 32 disqualified videos were re-introduced to the pool of Turk users and were all subsequently re-processed by new subjects.

357 HITs were flagged for not exceeding the minimum task duration time threshold of 23 seconds or less. These results were removed from the study after approval and payment had already been made and were subsequently not re-processed. Upon closer inspection, the majority of these 357 disqualified trials were submitted by a small group of repeat offenders who fit the profile of Turk users who were intentionally not properly carrying out each HIT.

The use of confound videos as automatic flags for the invalidation of a subject's results proved to be less clear than originally intended. Since the confound videos were exactly the same for each HIT (though their order was randomly generated), users who processed multiple HITs soon realized that they only needed to view one HIT to determine which was "correct". This conclusion was supported by comments submitted by subjects. Therefore missing confound click counts was not a factor used

to automatically disqualify any trials. Instead, confound results were used during manual review of potentially invalid results as an additional determining factor by the study coordinator.

5.3.5 Attribute Descriptors

At the heart of this study are the crossmodal relationships and measurable perceptual coherence between attributes of motion in virtual space and attributes of sound. Attributes from each modality mirror the parameters used when creating the crossmodal dataset described in Chapter 4. For this study, attributes were defined as simple directional binomial descriptors of each parameter of motion and each parameter of sound. Directional attributes marking the type and direction of a generalized trait (such as "increase in speed") were used rather than continuous values of attributes over time. Composite attribute descriptors, showing the coexistence of attributes from both modalities were used to search for trends related to the pairing of crossmodal parameters. During analysis, the impact of each individual attribute and crossmodal attribute pair on the aggregate perceived fit was determined across all examples in the set.

Primary Attributes The primary attributes of motion and the primary attributes of sound used in this study can be seen in Table 5.1.

Attributes of Motion	Attributes of Sound
Speed	Frequency
Rotation	Amplitude
Height	Breath Pressure

Table 5.1: Three primary attribute descriptors of both Motion and Sound were used in this experiment. Values for each were recorded as simple directional binomial descriptors.

Directional Attributes An analysis of the impact of whether a crossmodal example exhibits a single attribute of motion or sound is not a particularly useful one, as each low level attribute can be said to exist within every example. For instance,

all motions exhibit some level of speed, rotation and height at any given moment, as do all sonifications created using a physically-modeled clarinet exhibit some level of frequency, amplitude and breath pressure. Instead it is the directional delta or change of any one of these base attributes that can then be tracked across examples and across the entire study. The primary directional motion attributes used for this study were:

Directional Attributes of Motion	Directional Attributes of Sound
Increase in Speed	Increase in Frequency
Decrease in Speed	Decrease in Frequency
Increase in Rotation	Increase in Amplitude
Decrease in Rotation	Decrease in Amplitude
Increase in Height	Increase in Breath Pressure
Decrease in Height	Decrease in Breath Pressure

Table 5.2: Directionality as well as feature type were taken into account in the definition of each attribute, meaning for any given attribute like "Speed" there would be two attributes used, namely "Increase in Speed" and "Decrease in Speed".

Composite Attribute Pairings Of primary interest in this study was the furthering of our understanding about which *pairings* of crossmodal attributes significantly contribute to participants' perceived fit of crossmodal media examples. These pairings represent basic mapping schemata that themselves form the basis of more complex and artistic musical sonifications. Each video example in the sample set was tagged with composite attribute descriptors which allowed the ranking of example fits based upon not only the single attributes which were exhibited but also on which attribute pairings were being exhibited. For instance, video excerpts showing an increase in speed mapped directly to frequency would exhibit the composite attribute "Positive Speed, Positive Frequency" while an excerpt showing an increase in rotation mapped inversely to breath pressure would exhibit the composite attribute "Positive Rotation, Negative Breath Pressure". In this manner the directionality of each mapping can be examined both in the context of the direction of its original source motion as well as in the context of the direct or inverse mapping schema.

5.4 Data Analysis

By framing participants' subjective preference of crossmodal media examples as a discrete choice model, this study was designed to determine both the rank of preference for each example across the entire participant set as well as the relative effect of individual attributes and attribute pairs. Rank of preference and attribute contributions can be determined using a binomial choice model such at those proposed by Bradley and Terry [101, 10]. Data analysis was conducted using the R statistical programming language [63, 100] using the BradleyTerry2 package [116].

5.4.1 Bradley-Terry Model

The Bradley-Terry model (BTm) provides a method of extracting associative rankings from binomial choice datasets. Commonly used for the evaluation of multiple-participant binomial competitions such as baseball seasons or chess tournaments, Bradley-Terry has been used to model pairwise comparison tasks in fields ranging from genetics to marketing to election results [75]. By presenting examples to be compared as 'competitors' in a matched pairwise comparison task or 'contest', the Bradley-Terry model proposes a logit model for paired evaluations, capable of ranking examples based on their ability to 'win' a given comparison. One advantage of the Bradley-Terry model when compared to simpler averaging or mean comparisons is that the BTm factors the relative strength of competitors when calculating results, so a 'victory' in a pairwise comparison over a strong competitor counts more when calculating ranking scores than a victory over a weak competitor.

Essentially for any pairwise comparison or contest, the Bradley-Terry model assumes for any two paired 'players', i and j ($i, j \in \{1, \ldots, K\}$), the odds that player i beats player j can be represented as α_i/α_j, where α_i and α_j are positively-valued parameters representing 'ability'.

To express the Bradley-Terry model using a logit-linear form we can say

$$\text{logit}[\text{pr}(i \text{ beats } j)] = \lambda_i - \lambda_j, \tag{5.1}$$

where $\lambda_i = \log \alpha_i$ for all i. Therefore if we assume independence for all contests, maximum likelihood can estimate parameters $\{\lambda_i\}$[116].

The Bradley-Terry model can rank ability for explanatory variables or 'predictors' that can be found in each example, effectively allowing the algorithm to assess which component attributes of motion and sound in our dataset exhibit the most or least predictive power in participants' assessments of relative fit. And while there do exist extended techniques to factor 'ties' into the Bradley-Terry model [102], for this study ties were not allowed. Participants were allowed to choose 'Same' in their pairwise comparison task; these results were subsequently excluded from the Bradley-Terry calculation. In total, 1,487 results were marked as 'Same' and were not processed by the Bradley-Terry model.

5.5 Results

This section will detail some of the most noteworthy results derived from the Bradley-Terry analysis both in detail and in summary with primary attention given to the ranked perceived fit or coherence for the crossmodal video examples, core attributes and attribute pairings. Data collected on the role of parameter contour will also be presented, as will high-level results detailing perceived similarity between examples.

	Estimate	Std. Error	z value	Pr(>|z|)	
turn_rotation_frequency_inverse	0.6458	0.1940	3.33	0.0009	***
circle_rotation_breathpressure	0.5960	0.1935	3.08	0.0021	**
acceleration_speed_frequency_inverse	0.5321	0.1867	2.85	0.0044	**
jump_speed_frequency_inverse	0.5027	0.1909	2.63	0.0085	**
jump_speed_amplitude	0.4727	0.1930	2.45	0.0143	*

Table 5.3: Top five examples ordered by ranked fit.

5.5.1 Ranked Fit

42 unique crossmodal examples from the dataset described in Chapter 4 were used in this study. Table 5.3 displays the five examples exhibiting both the highest perceived fit as well as the most significant results from the Bradley-Terry model for pairwise comparison across the entire dataset. These five examples represent:

1. A discrete left turn with rotation inversely mapped to frequency, generating a sharp decrease in frequency corresponding to the left turn event.
2. A circular path with rotation mapped directly to breath pressure, increasing then subsequently decreasing pressure around the axis of 180 degrees.
3. An acceleration with speed inversely mapped to frequency for an acceleration, resulting in a decrease in frequency.
4. A jump event with speed inversely mapped to frequency, causing frequency to decrease then subsequently increase.
5. A jump event with speed directly mapped to amplitude, causing a decrease in amplitude, followed by an increase.

By contrast, table 5.4 displays five examples exhibiting the lowest fit across the entire dataset.

38. A discrete left turn with rotation mapped directly to amplitude.
39. A jump event with speed directly mapped to frequency.
40. A continuous curve with rotation mapped inversely to amplitude.
41. A circular path with rotation mapped inversely to breath pressure.
42. A discrete left turn with rotation inversely mapped to breath pressure.

	Estimate	Std. Error	z value	Pr(>|z|)
turn_rotation_amplitude	0.1429	0.1926	0.74	0.4581
jump_speed_frequency	0.1047	0.1883	0.56	0.5782
curve_rotation_amplitude_inverse	0.0850	0.1895	0.45	0.6537
circle_rotation_breathpressure_inverse	0.0739	0.1948	0.38	0.7042
turn_rotation_breathpressure_inverse	0	–	–	–

Table 5.4: Bottom five examples ordered by ranked fit.

A summary of these results include the following observations:

- Five of the top fifteen ranked examples were jump events
- Decreases in breath pressure resulting from mappings to increases or decreases speed performed approximately equally well regardless of mapping direction.
- Accelerations mapped inversely outperformed all direct mappings of acceleration regardless of which paired sound attribute was exhibited.
- The example exhibiting the lowest ranked fit a curve motion directly mapping rotation to frequency.
- Curve examples ranked poorly across all mappings.
- Intutive mappings such as height directly mapped to frequency during a jump event exhibited a low perceived fit ranking.

A complete table of results from the primary Bradley-Terry analysis can be found in Table 5.5.

i	Motion	Estimate	Std. Error	z value	Pr(>|z|)	Δinv	Δi	
1	turn_rotation_frequency_inverse	0.64579	0.19399	3.329	0.000872	0.42445	30	***
2	circle_rotation_breathpressure	0.59601	0.19354	3.079	0.002074	0.52208	39	**
3	acceleration_speed_frequency_inverse	0.53207	0.18672	2.85	0.004378	0.29232	24	**
4	jump_speed_frequency_inverse	0.50274	0.19095	2.633	0.008468	0.39802	27	**
5	jump_speed_amplitude	0.47273	0.19305	2.449	0.014334	0.07724	9	*
6	turn_rotation_breathpressure	0.46432	0.19178	2.421	0.015474	0.2315	23	*
7	acceleration_speed_breathpressure_inverse	0.46355	0.19417	2.387	0.016968	0.20287	18	*
8	deceleration_speed_breathpressure	0.45092	0.19677	2.292	0.021926	0.05637	7	*
9	circle_rotation_frequency_inverse	0.43556	0.19011	2.291	0.02196	0.03	3	*
10	circle_rotation_amplitude_inverse	0.40913	0.19307	2.119	0.034082	0.1305	14	*
11	jump_height_frequency_inverse	0.40895	0.19105	2.141	0.03231	0.21698	24	*
12	circle_rotation_frequency	0.40556	0.19085	2.125	0.033589	0.03	3	*
13	jump_speed_breathpressure_inverse	0.40042	0.18751	2.135	0.032727	0.24045	24	*
14	jump_speed_amplitude_inverse	0.39549	0.19149	2.065	0.038885	0.07724	9	*
15	deceleration_speed_breathpressure_inverse	0.39455	0.19375	2.036	0.041716	0.05637	7	*
16	acceleration_speed_amplitude_inverse	0.39134	0.19084	2.051	0.040307	0.02877	2	*
17	deceleration_speed_frequency_inverse	0.37316	0.18708	1.995	0.046075	0.13444	11	*
18	acceleration_speed_amplitude	0.36257	0.19695	1.841	0.065632	0.02877	2	.
19	deceleration_speed_amplitude	0.3482	0.19436	1.792	0.073209	0.11627	11	.
20	curve_rotation_breathpressure	0.33946	0.19013	1.785	0.074202	0.12245	12	.
21	jump_height_breathpressure	0.32299	0.19051	1.695	0.090009	0.15083	15	.
22	jump_height_amplitude	0.31485	0.19071	1.651	0.098759	0.06639	4	.
23	turn_rotation_amplitude_inverse	0.28314	0.19413	1.458	0.144708	0.14023	15	
24	circle_rotation_amplitude	0.27863	0.19074	1.461	0.144077	0.1305	14	
25	acceleration_speed_breathpressure	0.26068	0.18795	1.387	0.165453	0.20287	18	
26	jump_height_amplitude_inverse	0.24846	0.19226	1.292	0.19625	0.06639	4	
27	acceleration_speed_frequency	0.23975	0.18791	1.276	0.201997	0.29232	24	
28	deceleration_speed_frequency	0.23872	0.18816	1.269	0.204544	0.13444	11	
29	turn_rotation_breathpressure_inverse	0.23282	0.19058	1.222	0.221829	0.2315	23	
30	deceleration_speed_amplitude_inverse	0.23193	0.19346	1.199	0.230595	0.11627	11	
31	turn_rotation_frequency	0.22134	0.19207	1.152	0.249172	0.42445	30	
32	curve_rotation_breathpressure_inverse	0.21701	0.19239	1.128	0.259349	0.12245	12	
33	curve_rotation_frequency_inverse	0.20901	0.19185	1.089	0.275942	0.20901	9	
34	curve_rotation_amplitude	0.20352	0.19028	1.07	0.284811	0.11851	6	
35	jump_height_frequency	0.19197	0.1898	1.011	0.311819	0.21698	24	
36	jump_height_breathpressure_inverse	0.17216	0.18886	0.912	0.36199	0.15083	15	
37	jump_speed_breathpressure	0.15997	0.1937	0.826	0.40888	0.24045	24	
38	turn_rotation_amplitude	0.14291	0.19259	0.742	0.45807	0.14023	15	
39	jump_speed_frequency	0.10472	0.18832	0.556	0.57816	0.39802	27	
40	curve_rotation_amplitude_inverse	0.08501	0.1895	0.449	0.653716	0.11851	6	
41	circle_rotation_breathpressure_inverse	0.07393	0.19475	0.38	0.70423	0.52208	39	
42	curve_rotation_frequency	0	0	0	0	0.20901	9	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

	Estimate	Std. Error	z value	Pr(>|z|)
Std. Dev.	0.05218	0.04974	1.049	0.294

Table 5.5: Bradley-Terry model results showing perceived fit rankings (column ***i***) across the entire subject set in descending order of fit (column ***Estimate***).

5.5.2 Attributes as predictors of fit

When 'predictors' or explanatory variables are specified for each example being analyzed using a Bradley-Terry model, the estimated worth or predictive power of each variable can be determined. For this study, the predictors used are the directional attributes of motion and sound described in Table 5.1. Table 5.6 shows the estimated worth and significance of each directional attribute.

The results exhibiting the most significance and the strongest positive estimated worth are the directional motion attributes *frequency_decrease* and *speed_increase*, with estimates of 0.21220 and 0.11877 respectively. *height_increase* also shows a significant negative estimate of -0.26053. The relatively low estimates may suggest that single-modality attributes have a limited predictive ability without considering their paired crossmodal attribute counterparts. Results that contain estimates of 'NA' typically contain at least one parameter that has been set to zero.

	Estimate	Std. Error	z value	Pr(>|z|)	
frequency_decrease	0.21220	0.08963	2.367	0.0179	*
amplitude_decrease	0.12784	0.09039	1.414	0.1573	
breathpressure_increase	0.12253	0.09056	1.353	0.1761	
speed_increase	0.11877	0.07139	1.664	0.0962	.
breathpressure_decrease	0.09702	0.09082	1.068	0.2854	
amplitude_increase	0.09239	0.09114	1.014	0.3107	
speed_decrease	0.08322	0.07154	1.163	0.2447	
frequency_increase	0.02592	0.08944	0.290	0.7720	
height_increase	-0.26053	0.13625	-1.912	0.0559	.
height_decrease	*NA*	*NA*	*NA*	*NA*	
rotation_increase	*-0.08322*	*0.07154*	*-1.163*	*0.2447*	
rotation_decrease	*NA*	*NA*	*NA*	*NA*	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

	Estimate	Std. Error	z value	Pr(>|z|)
Std. Dev.	0.05218	0.04974	1.049	0.294

Table 5.6: Single-modality attributes and their respective predictive abilities from a Bradley-Terry analysis. Italicized values such as *rotation_decrease* or results of *NA* represent predictors that exhibit multicolinearity or singularities with other predictors in the set. Estimates have been provided where possible.

5.5.3 Directional attribute pairs as predictors of fit

Each crossmodal pairing represented by the examples used in this study can be described as a pairing of attributes from both the visual and auditory modality with an associated parameter-data direction. The directionality of parameter data from each modality, or whether a given parameter increases or decreases, gives us four paired states to consider, i.e. an increase in both attributes, a decrease in both attributes, or one increase paired with a decrease. By looking at these pairings as attributes themselves, we can plot the mean perceived fit for each state, for each crossmodal attribute pairing.

The interaction plots presented in this section display the relative variance of each crossmodal attribute pairing, displaying mean fit values (y-axis) against directional attribute pairings (x-axis). For each attribute pairing, "n" refers to a negative or "decreasing" parameter change-direction, while "p" refers to a positive parameter change-direction. Examples in which the increase in a motion parameter is in the same direction as the sound parameter can be said to have a correlated mapping direction. When attributes are not correlated, the mapping direction can be said to be inversely correlated.

5.5.3.1 Mean fit for Speed

Figure 5.2 shows the mean fit of each directional attribute pairing for examples driven by the motion parameter *speed*. We see a symmetrical relationship between correlated attribute pairs where speed is mapped to frequency on the green dotted line. Low mean values for the **n__n** and **p__p** mappings are displayed, contrasting with high mean values for the **p__n** and **n__p** mapping directions. The red line shows a relatively flat mid-range mean fit for all four examples when speed is mapped to breath pressure. Similarly, no strong pattern or mean values are seen for mappings between speed and amplitude.

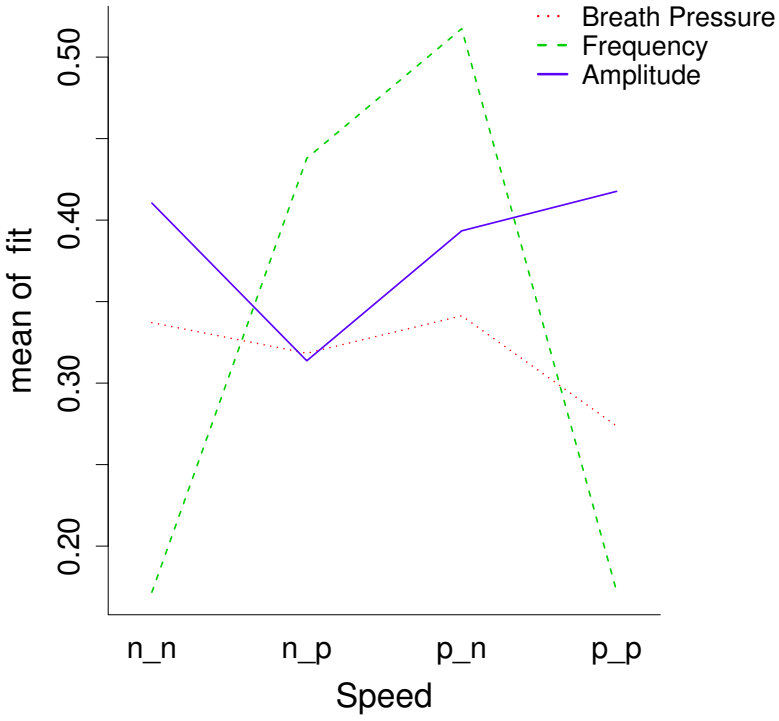


Figure 5.2: Interaction plots for speed mean of fit across sound parameter groupings.

5.5.3.2 Mean Fit for Rotation

Figure 5.3 shows the mean fit for each directional attribute pairing when rotation is mapped to each parameter of sound. Mappings with breath pressure show a strong symmetrical increase in mean fit for direct mappings and low mean fit values for each of the inversely correlated pairings. Relatively flat mean fit values can be seen for both frequency and amplitude mappings.

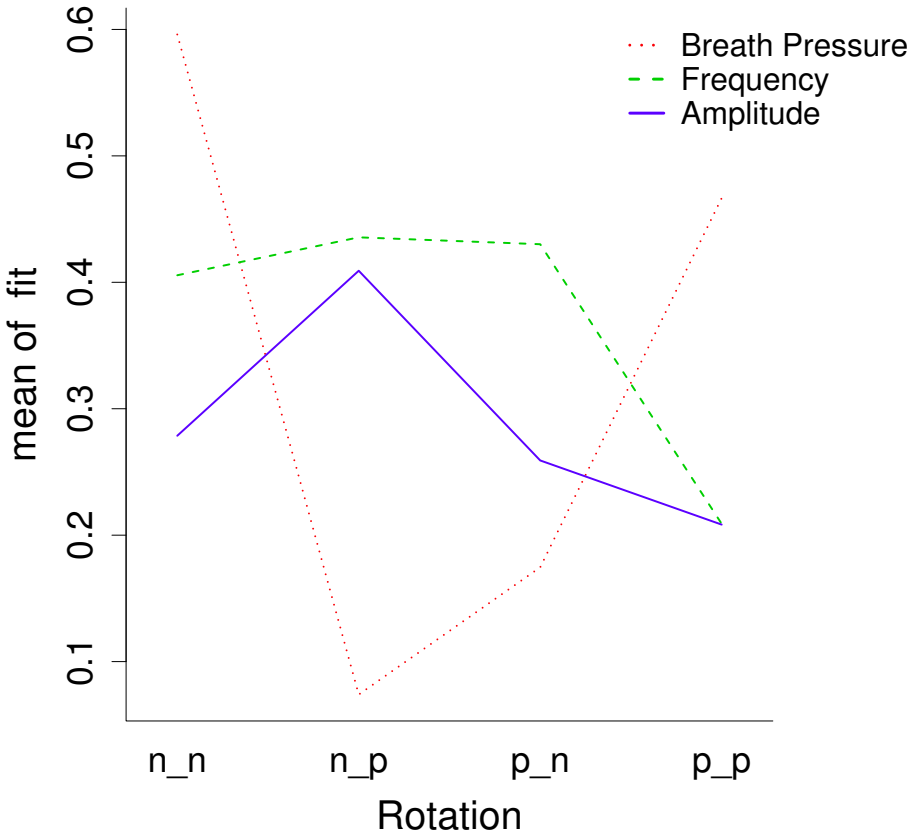


Figure 5.3: Interaction plot for rotation mean of fit across sound parameter groupings.

5.5.3.3 Mean Fit for Height

The symmetrical nature of the one height example used in this study can be seen in Figure 5.4, showing clearly symmetrical relationships for both correlated and uncorrelated mappings. Examples in which height was mapped to frequency show strong mean preferences for uncorrelated mappings, with both **n_p** and **p_n** mappings scoring twice as high as **n_n** and **p_p** mappings. For both breath pressure and amplitude mappings, the inverse case can be seen where correlated mappings are higher than uncorrelated mappings.

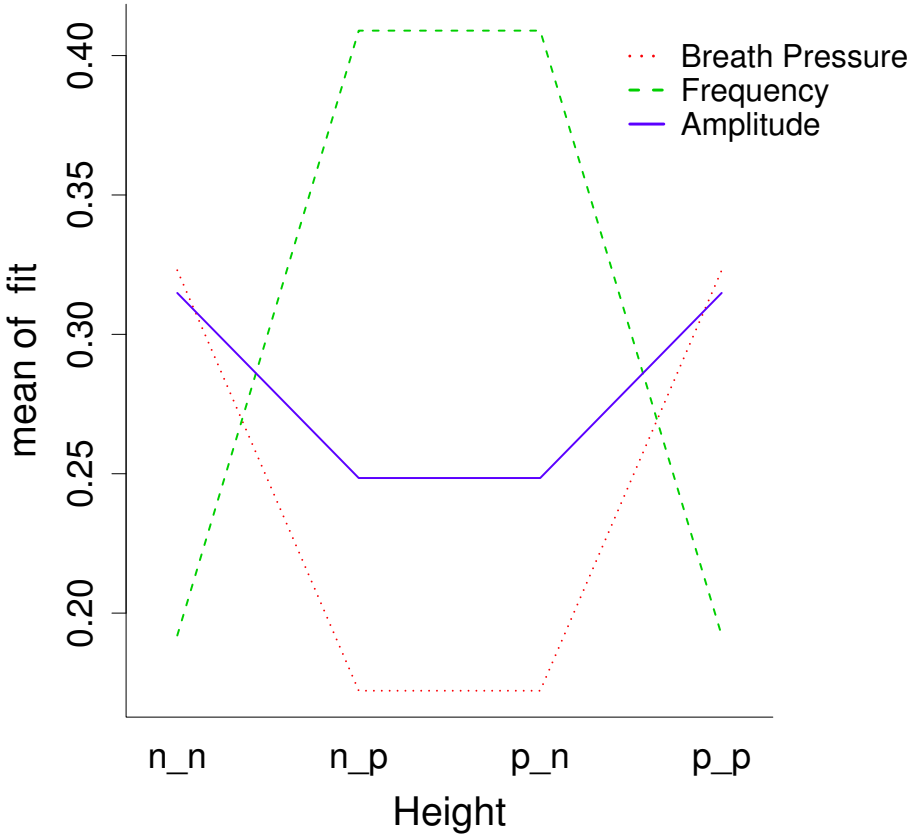


Figure 5.4: Interaction plot for height mean of fit across sound parameter groupings.

5.5.3.4 Mean Fit for Frequency

In Figure 5.5, we see the same interaction plots but this time comparing mean fit values for each motion example mapped to frequency. The results displayed for speed, rotation and height show similar contours across each mapping schema. Examples mapping attributes of motion to frequency show a strong preference for mappings in which mapping direction is inversely correlated between motion and sound. These mappings are strongly symmetrical for both speed and height mappings. Frequency mappings with rotation exhibit the same strong decrease for positively mapped rotation to frequency increase, while showing little change in mean fit for inversely correlated mappings.

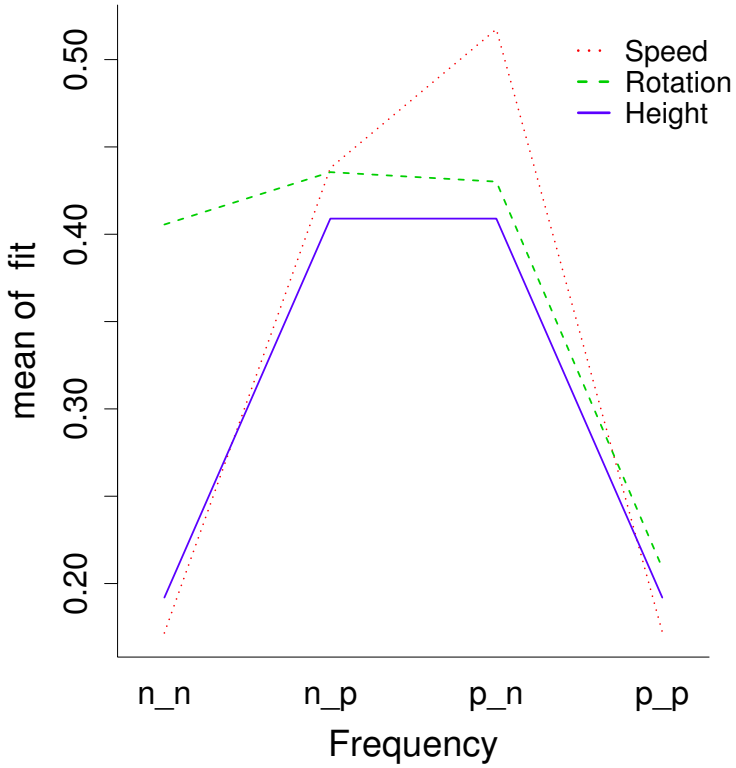


Figure 5.5: Interaction plot for frequency mean of fit across motion parameter pairings.

5.5.3.5 Mean Fit for Amplitude

In Figure 5.6 examples mapping attributes of motion to amplitude show the widest variance across all mappings, with few commonalities evident between mapping pairings with speed, rotation and height.

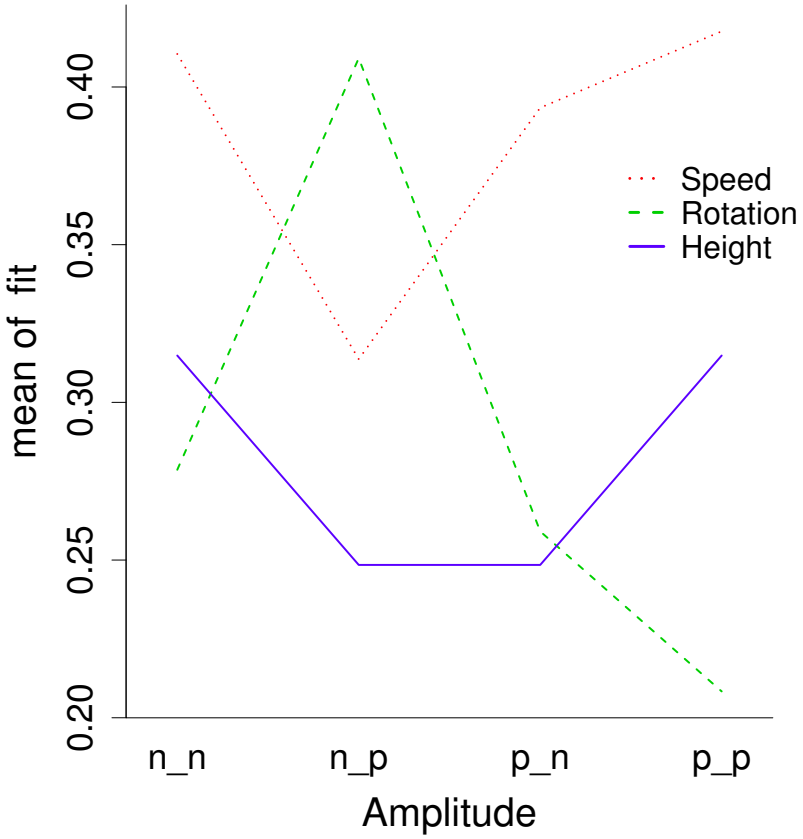


Figure 5.6: Interaction plot for amplitude mean of fit across motion parameter pairings.

5.5.3.6 Mean Fit for Breath Pressure

In Figure 5.7 examples mapping attributes of motion to breath pressure were again varied, with correlated mappings for rotation exhibiting the strongest overall fit. Both height and speed exhibited little variance regardless of whether mappings were correlated or not. As seen in previous examples, height and rotation exhibited strongly symmetrical results.

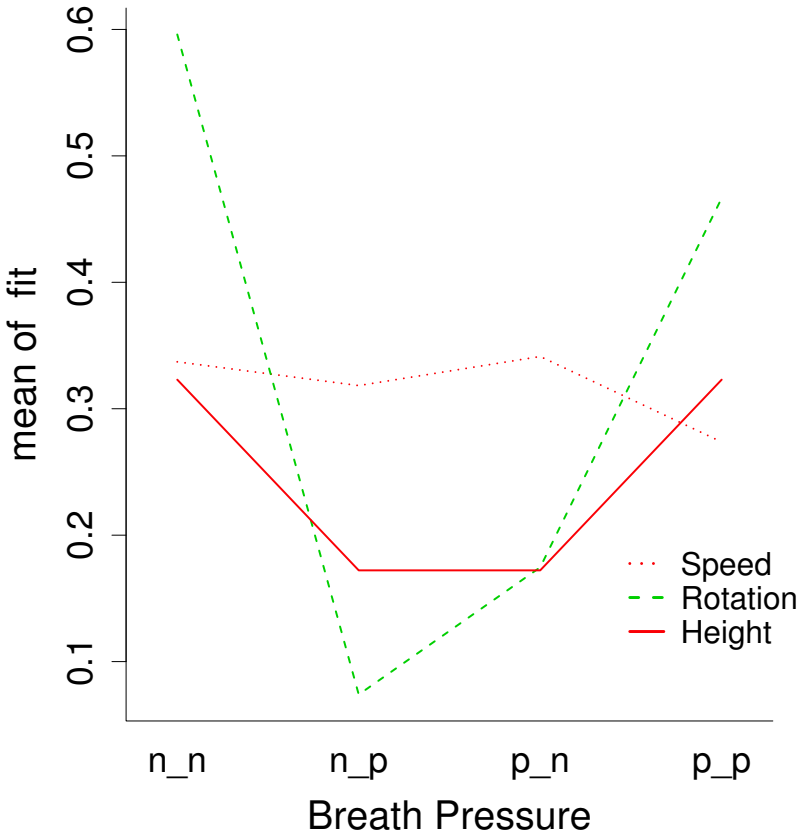


Figure 5.7: Interaction plot for breath pressure mean of fit across motion parameter pairings.

5.5.4 Inverse and Directional Pairings

For each directional mapping pair used in this study we can isolate the relationship of directionality, that is the use of direct or inverse mapping, by grouping and comparing perceived fit for each mapping pair. Table 5.7 orders each of the 21 grouping pairs by the absolute difference between their Estimate perceived fit seen here in column ***Δinv***. Individual rank for each example can be seen in column **i** while the difference in rank can be seen in column ***Δi***.

The effect of inverting mapping direction is strong in the first five groupings, which include the top four ranked individual examples:

1. A circlular motion mapping its rotation directly to breath pressure shows a high level of perceived fit (rank 2) while its inverse mapping shows an extremely low perceived fit (rank 41) with a resultant Δi of 39 and Δinv of 0.52208.

2. The example with highest perceived fit, a discrete turn mapping its rotation inversely to frequency, has a high Δi of 30 and Δinv of 0.42445, showing a fairly low level of perceived fit (rank 31) for its inverse mapping of a discrete turn directly mapping rotation to frequency.

3. A jump motion inversely mapping speed to frequency (rank 4) exhibits a Δi of 27 and Δinv of 0.39802 with its inverse pair (rank 39).

4. An acceleration example inversely mapping its speed to frequency (rank 3) has a Δi of 24 and Δinv of 0.29232 when paired with its inverse (rank 27).

5. A jump motion inversely mapping speed to breath pressure (rank 13) has a Δi of 24 and Δinv of 0.24045 when paired with its inverse (rank 37).

By contrast, looking at the relationship between the lowest-ranked grouped directional mapping pairs shows that some example pairs exhibit little difference based upon the directionality of mapping:

17. A jump motion directly mapping speed to amplitude (rank 5) exhibits a Δi of 9 and a Δinv of 0.07724 when paired with its inverse (rank 14).

18. A jump motion directly mapping height to amplitude (rank 22) exhibits a Δi of 4 and a Δinv of 0.06639 when paired with its inverse (rank 26).

19. A deceleration example directly mapping speed to breath pressure (rank 8) has a Δi of 7 and a Δinv of 0.05637 when paired with its inverse (rank 15).

20. A circular motion mapping rotation inversely to frequency (rank 9) has a Δi of 3 and a Δinv of 0.03 with its paired inverse (rank 12).

21. An acceleration example mapping speed inversely to amplitude (rank 16) has both the lowest Δi of 2 and the lowest Δinv of 0.02877 with its paired inverse (rank 18).

i	Motion	Estimate	Std. Error	z value	Pr(>|z|)	Δinv	Δi	
2	circle_rotation_breathpressure	0.59601	0.19354	3.079	0.002074	0.52208	39	**
41	circle_rotation_breathpressure_inverse	0.07393	0.19475	0.38	0.70423	0.52208	39	
1	turn_rotation_frequency_inverse	0.64579	0.19399	3.329	0.000872	0.42445	30	***
31	turn_rotation_frequency	0.22134	0.19207	1.152	0.249172	0.42445	30	
4	jump_speed_frequency_inverse	0.50274	0.19095	2.633	0.008468	0.39802	27	**
39	jump_speed_frequency	0.10472	0.18832	0.556	0.57816	0.39802	27	
3	acceleration_speed_frequency_inverse	0.53207	0.18672	2.85	0.004378	0.29232	24	**
27	acceleration_speed_frequency	0.23975	0.18791	1.276	0.201997	0.29232	24	
13	jump_speed_breathpressure_inverse	0.40042	0.18751	2.135	0.032727	0.24045	24	*
37	jump_speed_breathpressure	0.15997	0.1937	0.826	0.40888	0.24045	24	
6	turn_rotation_breathpressure	0.46432	0.19178	2.421	0.015474	0.2315	23	*
29	turn_rotation_breathpressure_inverse	0.23282	0.19058	1.222	0.221829	0.2315	23	
11	jump_height_frequency_inverse	0.40895	0.19105	2.141	0.03231	0.21698	24	*
35	jump_height_frequency	0.19197	0.1898	1.011	0.311819	0.21698	24	
33	curve_rotation_frequency_inverse	0.20901	0.19185	1.089	0.275942	0.20901	9	
42	curve_rotation_frequency	0	0	0	0	0.20901	9	
7	acceleration_speed_breathpressure_inverse	0.46355	0.19417	2.387	0.016968	0.20287	18	*
25	acceleration_speed_breathpressure	0.26068	0.18795	1.387	0.165453	0.20287	18	
21	jump_height_breathpressure	0.32299	0.19051	1.695	0.090009	0.15083	15	.
36	jump_height_breathpressure_inverse	0.17216	0.18886	0.912	0.36199	0.15083	15	
23	turn_rotation_amplitude_inverse	0.28314	0.19413	1.458	0.144708	0.14023	15	
38	turn_rotation_amplitude	0.14291	0.19259	0.742	0.45807	0.14023	15	
17	deceleration_speed_frequency_inverse	0.37316	0.18708	1.995	0.046075	0.13444	11	*
28	deceleration_speed_frequency	0.23872	0.18816	1.269	0.204544	0.13444	11	
10	circle_rotation_amplitude_inverse	0.40913	0.19307	2.119	0.034082	0.1305	14	*
24	circle_rotation_amplitude	0.27863	0.19074	1.461	0.144077	0.1305	14	
20	curve_rotation_breathpressure	0.33946	0.19013	1.785	0.074202	0.12245	12	.
32	curve_rotation_breathpressure_inverse	0.21701	0.19239	1.128	0.259349	0.12245	12	
34	curve_rotation_amplitude	0.20352	0.19028	1.07	0.284811	0.11851	6	
40	curve_rotation_amplitude_inverse	0.08501	0.1895	0.449	0.653716	0.11851	6	
19	deceleration_speed_amplitude	0.3482	0.19436	1.792	0.073209	0.11627	11	.
30	deceleration_speed_amplitude_inverse	0.23193	0.19346	1.199	0.230595	0.11627	11	
5	jump_speed_amplitude	0.47273	0.19305	2.449	0.014334	0.07724	9	*
14	jump_speed_amplitude_inverse	0.39549	0.19149	2.065	0.038885	0.07724	9	*
22	jump_height_amplitude	0.31485	0.19071	1.651	0.098759	0.06639	4	.
26	jump_height_amplitude_inverse	0.24846	0.19226	1.292	0.19625	0.06639	4	
8	deceleration_speed_breathpressure	0.45092	0.19677	2.292	0.021926	0.05637	7	*
15	deceleration_speed_breathpressure_inverse	0.39455	0.19375	2.036	0.041716	0.05637	7	*
9	circle_rotation_frequency_inverse	0.43556	0.19011	2.291	0.02196	0.03	3	*
12	circle_rotation_frequency	0.40556	0.19085	2.125	0.033589	0.03	3	*
16	acceleration_speed_amplitude_inverse	0.39134	0.19084	2.051	0.040307	0.02877	2	*
18	acceleration_speed_amplitude	0.36257	0.19695	1.841	0.065632	0.02877	2	.

Table 5.7: Bradley-Terry model results grouped into paired mappings highlighting Estimate difference and rank difference for direct and inverse mapping pairs.

5.5.5 Symmetrical Pairings

Following Eitan and Granot, symmetrical example pairings can be described as example pairs in which *both* the directional attributes of motion and the directional attributes of sound are inverted or diametrically opposed. Table 5.8 shows each possible symmetrical example pairing exhibiting changes for the motion attribute of speed. As this dataset contains motion examples exhibiting increasing and decreasing speed (acceleration and deceleration) symmetrical pairings can be examined for mappings to frequency, rotation and breath pressure. The difference in rank for each member of the pair can be seen in column ***Δi*** while the difference in perceived fit from column ***Estimate*** can be seen in column ***Δsym***

Two sets of pairings shown in Table 5.8 exhibit symmetrical tendencies, that is, their Δi and Δsym values are both extremely low. However the perceived fit (as seen in the **Estimate** column) is fairly low for both pairings with only one pairing showing significance. For these pairings to exhibit true symmetrical tendencies, not only should their perceived fits be approximately the same but they should also be fairly high.

i	Motion	Estimate	Std. Error	z value	Pr(>|z|)	Δsym	Δi	
27	acceleration_speed_frequency	0.23975	0.18791	1.276	0.201997	0.00103	1	
28	deceleration_speed_frequency	0.23872	0.18816	1.269	0.204544	0.00103	1	
18	acceleration_speed_amplitude	0.36257	0.19695	1.841	0.065632	0.01437	1	.
19	deceleration_speed_amplitude	0.3482	0.19436	1.792	0.073209	0.01437	1	.
7	acceleration_speed_breathpressure_inverse	0.46355	0.19417	2.387	0.016968	0.069	8	*
15	deceleration_speed_breathpressure_inverse	0.39455	0.19375	2.036	0.041716	0.069	8	*
3	acceleration_speed_frequency_inverse	0.53207	0.18672	2.85	0.004378	0.15891	14	**
17	deceleration_speed_frequency_inverse	0.37316	0.18708	1.995	0.046075	0.15891	14	*
16	acceleration_speed_amplitude_inverse	0.39134	0.19084	2.051	0.040307	0.15941	14	*
30	deceleration_speed_amplitude_inverse	0.23193	0.19346	1.199	0.230595	0.15941	14	
8	deceleration_speed_breathpressure	0.45092	0.19677	2.292	0.021926	0.19024	17	*
25	acceleration_speed_breathpressure	0.26068	0.18795	1.387	0.165453	0.19024	17	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

	Estimate	Std. Error	z value	Pr(>|z|)
Std. Dev.	0.05218	0.04974	1.049	0.294

Table 5.8: Bradley-Terry model results showing groups of paired symmetries for speed and each sound attribute. Column Δi represents the difference in index for each paired symmetry and Δsym shows the difference in Estimate for each member of the pair.

5.5.6 Contour Matching

One particularly interesting way of looking at the parameter data used in this study involves the reduction of each recorded motion attribute and generated sound attribute to simple parameter contours based upon a hypothesis that examples exhibiting matched contours between motion and sound would exhibit a greater perceived fit. Figure 5.8 shows nine simple contour shapes that are exhibited in the full dataset, with contours 2-8 exhibited in parameters sonified in this user study. For example, contour #2 shows a linearly increasing parameter value, such as would be exhibited by the speed parameter during a linear acceleration. During a direct mapping of acceleration to frequency, the contour exhibited by frequency would also be #2, while if an inverse mapping were to be used, the contour exhibited would instead be #3.

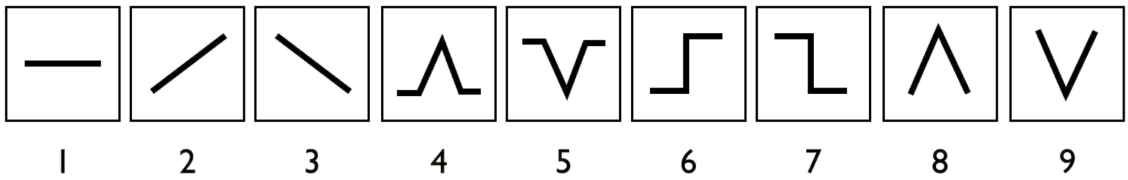


Figure 5.8: Individual parameter contours.

Figure 5.9 plots the mean perceived fit of examples exhibiting contours 2-9 for cases where the mapping exhibits a matched or unmatched contour.

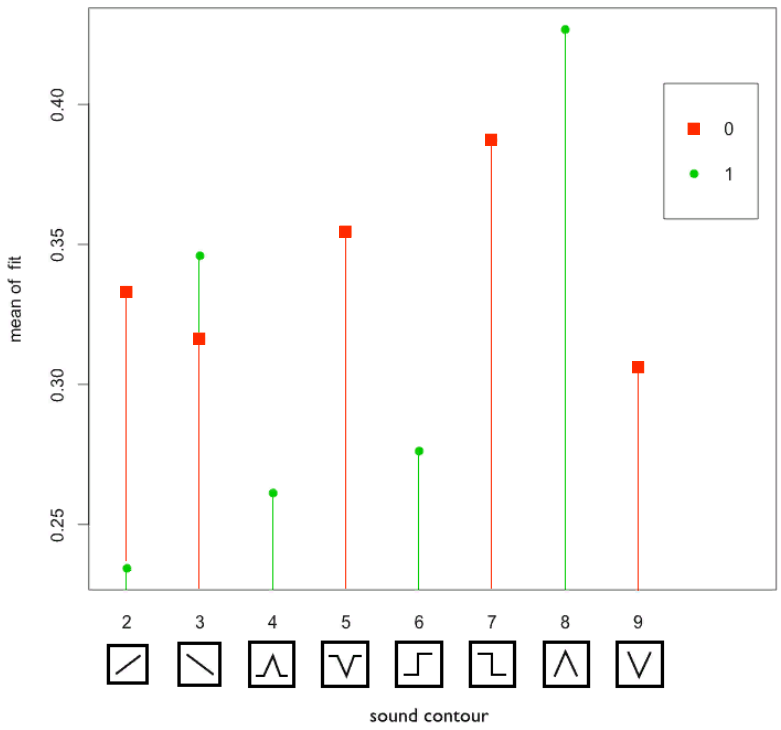


Figure 5.9: For examples exhibiting sound contours 2-9, the mean of fit for both matched (1/green circle) and unmatched (0/red square) contours are displayed.

We can see the following patterns of behavior:

- For contours 2 and 3, linear increase and decrease, we can see a discrepency between the mean fit for matched contour vs. unmatched contour. For linear increases, there is a much higher mean of fit for unmatched contours than for matched contours. For linear decreases, the mean fit values are approximately equal for matched and unmatched contours. Examples that exhibit contours 2 and 3 include direct and inverse mappings of speed for the acceleration and deceleration examples, and direct and mappings of rotation for the continuous curve example.

- Contours 4 and 5, a sharp parameter increase followed by a decrease, show a strong mean fit preference for the unmatched contour, which is exhibited for inverse mappings of the height parameter on the jump event motion example.

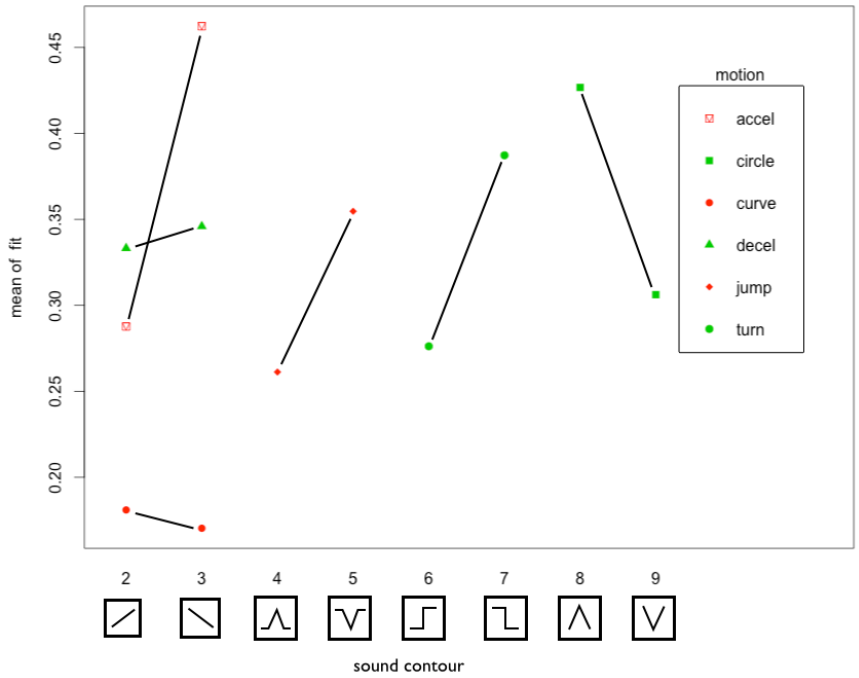


Figure 5.10: Mean of fit for each sound contour grouped by motion type is displayed.

Similarly, contours 6 and 7, found when mapping rotation on the discrete turn example, show a similar preference for the unmatched contour.

- Contours 8 and 9 show a mean fit preference for a continuous increase followed by a decrease, as exhibited when mapping rotation for a circle event. It should be noted that in these cases, rotation was judged to be "positive" when the actor turned left, or away from the camera location. If that choice had been reversed, then the pattern exhibited in contours 8 and 9 would match the same patterns evident when looking at contour pairs 4 and 5 or 6 and 7.

- Looking across contours 2 and 3, if the matched contour value for 2 is viewed as paired with the unmatched contour value in 3, we again see the same mean preference for unmatched contours as is evident in contours 4-9. In this case, the similar fit for unmatched contour 2 and matched contour 3 can then be seen as anomalous, in that there is no clear preference for unmatched contour.

Figure 5.10 displays sound contour mean of fit grouped by type of motion. At this level we can equate unmatched contour for each motion with an inverse mapping.

- In 2-3, acceleration shows a strong preference for an inverse mapping while deceleration and curve motion examples show little preference.
- Both jump (4-5) and discrete turn (6-7) events show a preference for inverse mapping.
- As seen previously, circle events (8-9) exhibit a preference for direct mapping, which could be explained by inverting the current subjective assignment of increased rotation.

5.5.7 Similarity

The perceived similarity between crossmodal examples presented to the study participants was recorded as a user-chosen integer value. Participants were presented with the following question: 'Please select a value from 1-7 to rank how similar the audio and video in the above videos are (where "1" means the two examples are completely different and "7" means they seem exactly the same).' Similarity ratings recorded using HTML radio buttons marked 1-7 were used to track each participant's perceived similarity for each presented pairing.

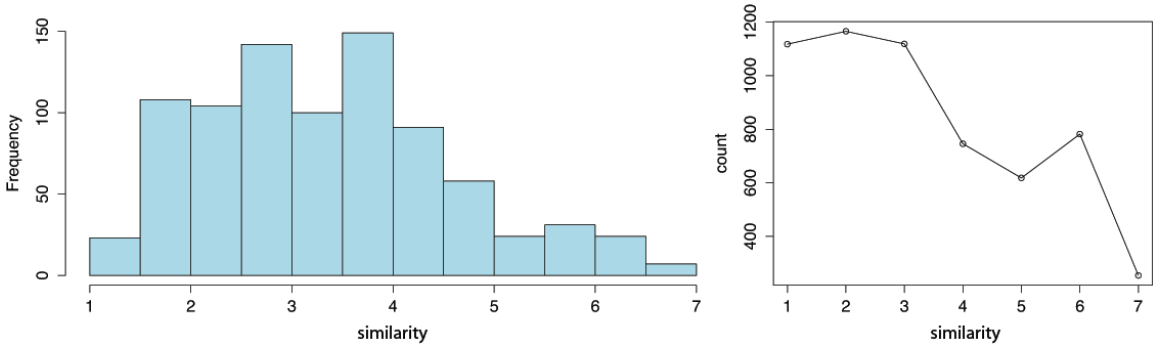


Figure 5.11: On the left, a similarity histogram displays the average score distribution across each of 861 unique example pairings. On the right, a plot showing total example counts at each similarity level

i			similarity
1	curve_rotation_breathpressure	curve_rotation_amplitude	6.857143
2	accel_speed_breathpressure_inverse	decel_speed_breathpressure_inverse	6.833333
3	decel_speed_breathpressure_inverse	decel_speed_amplitude_inverse	6.714286
4	decel_speed_breathpressure	decel_speed_breathpressure_inverse	6.571429
4	decel_speed_breathpressure	decel_speed_amplitude	6.571429
4	jump_speed_breathpressure	jump_height_breathpressure	6.571429
4	circle_rotation_breathpressure	circle_rotation_breathpressure_inverse	6.571429
8	jump_speed_breathpressure	jump_speed_amplitude	6.428571
8	jump_speed_amplitude_inverse	jump_height_amplitude_inverse	6.428571
8	turn_rotation_breathpressure	turn_rotation_amplitude_inverse	6.428571
8	turn_rotation_breathpressure	turn_rotation_amplitude	6.428571
8	circle_rotation_amplitude	circle_rotation_amplitude_inverse	6.428571
14	accel_speed_breathpressure_inverse	decel_speed_amplitude	6.285714
14	decel_speed_breathpressure_inverse	accel_speed_amplitude_inverse	6.285714
14	jump_speed_breathpressure_inverse	jump_height_amplitude_inverse	6.285714
14	accel_speed_amplitude_inverse	decel_speed_amplitude_inverse	6.285714
14	jump_speed_amplitude	jump_height_breathpressure	6.285714
14	circle_rotation_breathpressure	circle_rotation_amplitude	6.285714
21	accel_speed_breathpressure_inverse	decel_speed_amplitude_inverse	6.142857
21	accel_speed_breathpressure_inverse	accel_speed_amplitude	6.142857
21	jump_speed_breathpressure_inverse	jump_speed_amplitude_inverse	6.142857
21	jump_speed_breathpressure_inverse	jump_height_breathpressure_inverse	6.142857
21	jump_speed_breathpressure_inverse	jump_height_amplitude	6.142857
21	jump_speed_breathpressure	jump_speed_amplitude_inverse	6.142857
21	jump_speed_amplitude_inverse	jump_height_breathpressure_inverse	6.142857

Table 5.9: Top twenty-five most similar example pairings.

Figure 5.11 (on the right) shows the total count of all rated example pairs for each similarity level. The majority of rated pairs were judged to be relatively low scoring or not similar.

Table 5.9 shows the top twenty-five most similar pairings from the example set. From those results, the following key points can be seen:

- 20 of the top 25 example pairings ranked for similarity were *motion similar*, meaning they shared the same motion sequence with a different sonification or sonification mapping direction.

- 2 of the top 25 example pairings were exactly *sound similar*, meaning the generated sound result came from the same mapping contour and parameter range.

- 4 of the top 25 example pairings were inversely sound similar, meaning the examples were generated from inverse mappings of the same contour and parameter range.
- 8 of the top 25 were acceleration/deceleration pairings sharing similar or inverse contours.
- Breath pressure examples comprise 9 of the top 10 examples, while breath pressure paired with gain make up 29 of the top 50 examples.

To get an overall feel for the influence of individual attributes of both sound and motion on perceived similarity, Table 5.10 displays the average similarity rating for each attribute. Breath pressure and amplitude both show high relative similarity averages from the sound modality while attributes of motion acceleration and deceleration show high relative similarity. The lowest average similarity values can be seen from the circle motion and frequency sound attribute, both exhibiting a significantly lower average similarity rating than all other attributes.

Attribute	Avg. Similarity
Breath Pressure	3.568717354
Gain	3.56033717
Deceleration	3.470830762
Acceleration	3.455576169
Speed	3.392429793
Height	3.300762732
Turn	3.28860029
Curve	3.282622143
Jump	3.179633362
Rotation	3.167114366
Circle	2.971243035
Frequency	2.777136944

Table 5.10: Average similarity for each attribute of motion and sound.

5.6 Discussion

The analyses of subjects' perceived fit of image and sound components discussed in this chapter provide some possible insight into the types of crossmodal correlations we as humans tend to feel are more *coherent*. By making use of the Amazon Mechanical Turk service, the subject pool for this study was extremely large and avoided common limitations found in academic user studies that draw small subject counts from extremely homogenous environments. The speed and scale of the Mechanical Turk service also allowed for incredibly quick turn-around and validation of user data, affording the study coordinator the ability to rapidly iterate modifications to the formatting of the study presentation scripts and dataset. The use of the Bradley-Terry model for the statistical analysis of user results reframed the ranking of perceived fit from a scalar ranking issue to a more manageable pairwise comparison task. The Bradley-Terry model also presented a methodology for assessing the relative impact of individual and paired attributes of motion and sound on subjects' perceived fit across the entire study sample set.

A complete assessment of the reasons each member of this crossmodal dataset received their perceived rankings and attribute worth estimates is beyond the scope of this exploratory study. The data collected here will ideally foster the creation of new hypothesies and new mapping schemata linking virtual action and motion to procedurally-generated sound, moving this research closer to the perceptual ground truths that govern our cognitive understanding of the connections between motion, gesture, sound and music.

5.6.1 Assessing Perceived Coherence and Fit

The results answering the first primary goal of this exploratory study, namely "Which examples exhibited the strongest fit across all participants?" can be seen in Table 5.3 (the top five results) as well as ranked example results for the entire study set in Table 5.7. While these results themselves offer no explicit explanations for subjects' preferences, they do however suggest many interesting directions and new questions that can be addressed in subsequent more focused studies. Here let us consider some

of the potential approaches that can be considered.

For example, the study's top ranked result with a ranking of 0.64579 was a discrete left turn inversely mapping rotation to frequency, effectively causing a decrease in frequency during the turn event. Its inverse mapping - that is a discrete left turn directly mapping rotation to frequency - exhibits an increase in frequency and is ranked quite lowly (rank 31). Only one other turn event was ranked in the top 50% of examples (rank 6, a direct mapping of rotation to breath pressure) suggesting that the turn event itself wasn't a strong predictor for the high rank. Looking to the directional attributes described in Table 5.6, the decrease in frequency exhibited by this example does correspond with the relatively strong predictive ability of the *frequency_decrease* attribute.

5.6.2 The Role of Perceptual Invariance

As a starting point for this body of research, the computer environments and interaction schemata used have retained clear perceptual and conceptual ties to the physics-based world in which we live. Avatars used in each creative work as well as in the crossmodal dataset have taken relatively familiar human or animal forms. Each has been strongly based in the visual and kinematic modalities intimately understood as belonging to our own physics-based reality. Subtly reinforcing the connection between these virtual spaces and the "real-world" has been an implicit reliance on certain perceptual invariants such as gravity, force and even the behaviors of light and shadow. The impact of these perceptual invariants has likely had the role of contextualizing the behaviors and interactions experienced in such virtual environments in terms that are at the same time familiar and potentially restrictive.

J. J. Gibson attributed our ability to perceive and understand objects during states of motion and change to the principle of *invariance*, labeling perceptual invariants as properties of "non-change that persists during change" [45]. In the context of multimodal audio-visual environments, the existence (or inexistence) of such invariants could affect the perceived causation and therefore the perceived coherence between crossmodal attributes. For future work in this direction, the use of less-familiar avatar

forms, multiple camera viewpoints and more varied ecologies of multimodal interaction could provide insight into the role of invariants in our multimodal perception.

One hypothesis based in human experience and the perception of invariant physical phenomena such as Doppler shift and simple amplitude attenuation could suggest that as the 90 degree rotation directed the avatar to move away from the camera's point of view into the virtual "distance", subjects perceived the sound source as moving "away" and therefore a diminishing mapping schema was perceived as strongly coherent. If that were the case, it is interesting to note that the clearest example of such a diminishing mapping schema (i.e. "turn_rotation_amplitude_inverse", exhibiting a decrease in amplitude as virtual "distance" increases) was ranked as the twenty-third best fitting example. And the very similar "curve_rotation_frequency_inverse" example (a continuous curve inversely mapping rotation to frequency) which likewise exhibited a decrease in frequency as the avatar rotated and moved away from the camera location itself ranked quite poorly at rank 33. Looking to contour as a potential predictor of perceived fit, the mean fit contour results in Figure 5.9 indicate that our discrete left turn with decreasing frequency is described by sound contour 7 and motion contour 6, in line with contour 7's significant preference for unmatched crossmodal contours.

5.6.3 Symmetrical Mappings

The role of symmetry in the perception of crossmodal relationships, or more specifically the conclusion that "musical-spatial analogies are often *asymmetrical*, as a musical change in one direction evokes a significantly stronger spatial analogy than its opposite" was explored by Eitan and Granot [32]. Their study was based in analogy, with subjects visualizing and describing attributes of multi-dimensional motion when prompted by musical auditory stimuli.

In their initial hypothesis of "Symmetry of associative space", Eitan and Granot define crossmodal symmetry:

> Other things being equal, diametrically opposed musical processes <m,–m> would evoke diametrically opposed kinetic processes <k, –k>. In experimental terms: a listener who associates a musical stimulus m (e.g., a crescendo) with a kinetic quality k (e.g., a spatial ascent) would associate the inverse stimulus –m (e.g., diminuendo) with the opposite kinetic quality –k (e.g., descent).

Putting this hypothesis into terms that better relate to the experiment presented in this work, where one example exhibits a high-level of coherence or *fit* for a directional motion attribute (e.g. an increase in speed) when directly mapped to a directional sound attribute (e.g. an increase in frequency), the example exhibiting inverse directions for both motion and sound attributes (e.g. a decrease in speed mapped to a decrease in frequency) would also exhibit a high-level of coherence or fit. Results from the asymmetrical pairings of speed examples seen in Table 5.8 show only two example pairs exhibiting symmetrical tendencies for which the perceived coherence for both pairings is low. The other four example pairs demonstrate weak asymmetrical tendencies but the perceived fit difference or Δsym for each is not strong.

Looking at symmetrical relationships in a slightly different manner, the example groupings displayed in Table 5.7 do indicate that a number of the mapping schemata examined in this user study have exhibited strong inverse directional behaviors. Strongly perceived correspondences measured for one mapping schema of motion, sound and direction were often complemented by weakly perceived correspondences for the same motion-sound mapping's inverse direction. Out of 21 grouping pairs investigated here, 7 pairs (33% of the sample set) exhibited changes in index (Δi) of greater than 21 or 50% of the total index size. Additionally only 7 pairs exhibited relatively asymmetrical directional behaviors, with changes in Δi that were less than 10. So while the results from this study do not strongly support Eitan and Granot's interpretation of asymmetrical relationships, they do seem to show support for the regular existence of inverse directional asymmetrical relationships.

5.6.4 Potential Issues

With the approaches and methodologies detailed in this chapter there are a number of potential issues that should be mentioned and addressed in future work.

- In order to streamline the experiment to fit into a simple Mechanical Turk project template, there was no attempt to engage users in a minimum (or maximum) number of HITs, rather they were permitted to submit as many or as few as possible. This introduces the possibility of a small group of subjects who processed many HITs exerting more influence over the results than subjects who processed less. Due to the scale of unique combinations presented, a straightforward within-subjects design would not have been possible without inducing significant fatigue and likely carryover effects. Similar issues would complicate a standard between-subjects design. A middle-ground approach consisting of grouped stimulus pairs and limited group sizes could be investigated in the future to limit such individual impacts.

- While some attributes exhibited in the crossmodal dataset such as *speed* were addressed by multiple examples (acceleration, deceleration, jump event), others such as height were only exhibited by one example. Additional examples in the dataset should be created to gauge the influence of these attributes from multiple directions and sources. For *height*, simple examples showing an avatar walking up and down a slope or jumping up to a ledge would be useful additions.

- For the sake of consistency in the subject's viewpoint, each example in the dataset was created with avatars moving from screen-left to screen-right. The inverse direction showing motion from screen-right to screen-left should be added to take into account the perceived differences in general directional movement. Similarly all turn events showed the avatar turning left; examples showing turns to the right can also be added.

- Rotation examples were all generated using a polar mapping where parameter data increases linearly until rotation hits 180 degrees, then decreases until it reaches 360 or 0 degrees. While this mapping schema takes the human

understanding of “forwards” and “backwards” into consideration, it would be interesting to also explore a simple linear mapping for rotation.

- Rotation examples mapped turns to the left as increases in rotation. The mapping of a left turn to an “increase” in rotation was purely arbitrary and could have easily mapped a left turn to a decrease in rotation. In studies where multiple camera views are explored, one possible mapping of interest would cause rotations towards the camera to cause parameter increases, while rotations away from the camera would cause corresponding decreases.

- The use of a human-like avatar was intended to mimick similar avatars commonly used in commercial computer games. While the humanoid paradigm is indeed common, the motion of limbs and the inherent animation of the skeletal mesh could potentially be a distraction when tracking gross motion contours in the environment. One solution would be to create the same motion examples using generic block shapes without extraneous limb motion.

- As mentioned previously, the use of confound videos as markers signifying a subject’s attention to the task at hand proved to be less successful than intended. This was in part due to the limited set of two confound videos and the ease at which workers could select the “correct” HIT without even watching both videos. To make the confound videos more accurate predictors of user attention, a larger set of confound videos should be used to prevent this behavior.

- When subjects were permitted to choose ’Same’ in the primary example comparison task, 1,487 results were marked in this way and subsequently excluded from the Bradley-Terry model calculations. If users had been presented with a forced-choice between example 1 or example 2 these results would have contributed to the BTm results. Davidson did however propose an extension to the Bradley-Terry model that can accomodate the existence of ’tie’ results [30]. One future task will be to compare the current BTm results with results using the Davidson extension.

5.7 Summary

In this chapter, a user study designed to measure the perceived fit or coherence between auditory and visual stimuli found in short video examples from the dataset detailed in Chapter 4 was described. Within the Amazon Mechanical Turk online service, subjects were presented with HTML form showing video examples and were instructed to choose the example in which the visual and auditory stimuli exhibited the best *fit*. An analysis of subjects' results was carried out using the Bradley-Terry statistical model with each example pairing represented as a pairwise comparison. Attributes describing each motion and sound example were attached to each example, allowing the Bradley-Terry model to make an assessment of the predictive power of each attribute and of crossmodal attribute pairs. Results were also analyzed and discussed using the respective contours of tracked motion and mapped sound parameters, within the context of understanding the role that gross contours of data might play in perceived levels of crossmodal correlation. Users were also asked to rank the level of similarity perceived between each video example, results of which were subsequently reviewed and modeled.

Chapter 6

Conclusions and Future Directions

6.1 Review

The research and practice explored in this dissertation stems from one simple goal: the creation of multimodal experiences that tightly integrate motion and gesture in computer generated spaces with musical sound. The body of artistic works outlined in Chapter 3 explored the role of interactive sound and music in virtual worlds, driving the use of mapping schemata and crossmodal explorations with artistic intent. A series of software platforms designed to communicate between rendered interactive environments and dynamic music processing systems were described and detailed, with special attention given to mapping schemata and musical sonification techniques used in each multimodal musical work. The dataset of crossmodal media examples presented in Chapter 4 sonified parameters of avatar motion using a variety of mapping schemata to create a body of audio-visual examples to be used in the analysis of crossmodal correspondences. And to better understand the perceptual constraints and opportunities afforded by procedural music systems, the user study outlined in Chapter 5 used statistical analysis to investigate the perceived coherence and crossmodal correspondences between attributes of virtual motion and attributes of generated musical sound.

Towards this basic goal, these artistic and analytic pursuits are inherently complementary. By better understanding how humans perceive crossmodal audio-visual

stimuli, we begin to build a base set of assumptions, a cognitive toolkit of sorts, to guide artists, designers and programmers in the creation of intuitive mapping schemata. As technology continues to evolve, the lines between virtual and physical realities continue to blur. We are becoming more and more comfortable navigating virtual spaces using visual and auditory cues designed by programmers, not by eons of natural selection and evolutionary process. This new technology-fueled evolutionary process will itself design the virtual multimodal associations and experiences in which we will immerse ourselves. The better we understand our own crossmodal perceptual tendencies the better informed we can be when designing coherent interaction layers and standards that will power the next (virtual) stage in human exploration and experience.

6.2 Contributions

The contributions of this work include the following.

UDKOSC: A Multimodal Framework for Building Musically-Sonified virtual environments As a resource for creative musicians, game-developers and visual artists, the UDKOSC framework binds the Open Sound Control protocol to the Unreal Development Kit in a manner that is flexible and extensible. The codebase developed to drive novel gesture and motion-based musical interactions in artistic works like *Tele-harmonium* and *ECHO::Canyon* as detailed in this dissertation can be downloaded from the UDKOSC git repository and modified to fit new interaction paradigms and musical projects. More importantly, the techniques used to drive and shape musical interactions both in the spaces of virtual motion as well as musical sonification can be ported to new technologies and sensory modalities as they become available.

A Crossmodal Dataset of Musically-Sonified Motions in Virtual Space Each interaction and experience taking place in virtual space generates and interacts with a multitude of rich data streams. The dataset presented in this body of research

was designed to fuel further investigations into the perceived coherence and crossmodal correlations between actions and motions taking place in a visual computer-generated environment and procedurally-generated sound. Built using UDKOSC, this dataset and the techniques used to create each example in the dataset provide a starting point from which further experiments can be carried out and additional examples can be created.

A User Study of Perceived Coherence and Crossmodal Correspondence The user study presented in this dissertation measures the perceived fit or coherence of examples from the crossmodal audio-visual dataset across a large set of subjects. By framing the measurement of coherence as a pairwise comparison task using the Bradley-Terry statistical model, not only does this study succeed in ranking the perceived fit of each example from the crossmodal dataset but it also assesses the relative estimated worth of directional attributes describing stimuli in both the visual and auditory modalities. User assessments of similarity and the grouping of each motion and sound attribute delta into directional contours provide additional datapoints from which the predictive power of individual and paired attributes can be better understood.

A Toolkit for the Generation, Recording and Playback of OSC Messages To facillitate the scripting of avatar motion and the creation of crossmodal examples, a set of tools capable of generating, recording and playing back timed Open Sound Control streams was created. This OSC Toolkit is comprised of three lightweight Ruby applications: OSCControl, OSCRecorder and OSCPlayer. The motion and gesture scripts used to generate each example in the dataset (as well as used in live performances of *ECHO::Canyon*) were all created using OSCControl. The parameter output from each example of avatar motion was recorded using OSCRecorder and subsequently played back for sonification using OSCPlayer. Each of these tools is simple and lightweight and can be useful to artists or researchers in need of generating, recording or playing back timed OSC streams.

6.3 Future Work

There exist many exciting approaches and opportunities to expand the technological, creative and analytic methodologies surrounding the mapping and understanding of crossmodal and procedural music systems.

Application of Research Findings in the Creative Domain One of the most exciting aspects about investigating analytic and creative projects in the same domain is the ease at which findings and ideas can be shared between the two. Software tools and mapping techniques designed for UDKOSC and *ECHO::Canyon* were leveraged into the creation of the crossmodal dataset and musical sonifications used in the user study. In the same way, findings from the user study can serve as the basis for the design of new mapping schemata and creative interactions. In a work like *ECHO::Canyon*, mappings perceived as being highly coherent such as the inverse mapping of speed to frequency could be used to sonify musical motions ranging from speeding Valkordia to swarming projectiles. Circular motion mapping rotation to breath pressure could form the basis for a new kinematic Trumbruticus trunk instrument. As data and inspiration from each one of these conceptual modalities fuels innovation and exploration in the other, both research and creative practice are strengthened and pushed to the next level.

Virtual Instruments For the creative and analytic examples used in this body of work, mapping schemata have focused primarily on creating musical sonifications out of sequences of motion and incidental gesture, such as the posing states from *ECHO::Canyon.* One direction that is currently being investigated explores the creation of virtual instruments in computer-generated space, playable through avatar interaction or user motion and gesture. Through the use of newly commoditised immersive three-dimensional visual systems like the Oculus Rift headset [8, 104], motion tracking hardware like the Microsoft Kinect and more ubiquitous motion sensing devices like mobile phones and the Wiimote, there exists a rich space to create playable

novel virtual instruments. Networking technologies can already allow such instruments to be played by multiple performers across disparate spaces in real-time, allowing for the creation of new musical performance practices and interaction paradigms based firmly within the extra-physical realities offered by virtual environments.

Creative Work The research practices and analytic efforts carried out for this body of work have been shaped and informed by the ongoing application and modification of the mapping schemata, scripting techniques, control paradigms and musical sonifications designed and developed in the context of creative musical works. This iterative process sees the research and artistic practices as complementary, with gains in one practice immediately being applied into the other. Future work on new multimodal musical experiences is already underway with a series of new works under development using UDKOSC. As technologies shift and new display and control paradigms come online, the creative exploration of each new possibility only serves to strengthen this symbiotic relationship between artistic and creative practice.

Procedural Composition and Music Engines While this research has primarily focused on the creation of musical sounds and the control of low-level parameters of modeled instruments and processes, the control of higher-level compositional attributes such as structure, form, density and progress is an area of great interest. Control of generative or algorithmic music systems can be directly influenced in many ways using in-game data streams such as those currently being tracked. One particularly promising direction involves the analysis of pre-composed musical data, its subsequent modeling and re-generation scaled and directed by parameters of motion, gesture and environment all tracked in real-time. In this manner compositional elements can be created using traditional workflows, and can be subsequently modified and recontextualized to match events and progresses happening in the visual or narrative modalities.

An Expanded Dataset To augment the current crossmodal dataset, additional examples have already been generated. For each of the examples described in Chapter 4, there exist multiple views captured using dynamic cameras including first person views, static rotating cameras and moving rotating views. With these new viewpoints, new assessments of the role of the camera view can be made, including the use of attributes such as distance. New motion sequences such as those showing motion in directions such as screen-right to screen-left motion, up and down a rendered slope and rotation towards the camera view can be added to augment the current result set with additional examples for each of those attributes. The use of non-humanoid avatars without potentially distracting limb animations would be a useful addition as would new sonification examples generating rhythmic patterns based upon footstep and arm motion data. Sound examples exploring the use of additional mapping parameters of motion such as clarinet breath noise, vibrato rate and vibrato amplitude have also been scripted as have less complex sound sources such as a simple sine wave and a simple tuned noise band.

Predictive Model The results from the user study undertaken in this body of work show promise in isolating which specific attributes and pairs of attributes of motion and sound play the strongest roles in our perception of multimodal audio-visual events. Following this research path through to its logical completion, these results coupled with additional examples and more targeted analysis can start to define predictive models capable of determining how strongly coherent individual crossmodal mapping schemata are perceived by those attending to them. A predictive model can help guide the creative process for those building novel multimodal experiences, providing a base understanding of reality-agnostic human perception to designers and sound artists working in new realities.

6.4 Concluding Remarks

For those who have themselves grown up during the frenzied evolution of multimodal audio-visual systems, the connection between sight and sound, be it scripted and static in the context of film and video or dynamic and procedural in the context of computer gaming and interactive media, feels inherently natural. The technology-mediated future in no way needs to share the inherent limitations of physical reality, but in order to successfully blend virtual affordance with human perception, each novel experience presented by technology must be grounded within the constraints of our human perceptual systems. In more ways than one, the only limitation to this kind of technology-mediated future is truly our own imagination.

In this thesis, creative practice and analytical methods of research have been combined to investigate exciting opportunities for creating and understanding musical and visual experiences. It is the author's hope that the methodologies and models put forth here can contribute to the writing of the next chapters in this exciting and rich field.

...

An archive of related materials to the research presented in this dissertation is publically available at the following permanent Stanford Library URL:

http://purl.stanford.edu/yy758rc6782

Appendices

Appendix A

UDKOSC Installation Guide

The installation and setup of UDKOSC is outlined in this appendix. The details described below are based on the February 2014 Beta build of the Unreal Development Kit (UDK) [9], at the time of this writing the latest and potentially last publically-released Unreal 3 UDK[1]. A summary of the steps detailed below include:

1. Download and Install the UDK
2. Download UDKOSC codebase
3. Install UDKOSC
   (a) Install the OSC .dll
   (b) Install UDKOSC Class structure
   (c) Install UDKOSC Config files
   (d) Install Maps and resources
4. Configure and Build UDKOSC
5. Test OSC Output and Input

[1]The Unreal Engine 4 was released in early 2014 and replaces the Unreal Engine 3 as Epic Software's flagship gaming engine.

A.1 Download and Install the UDK

At the time of this writing, the latest version of Epic's Unreal Development Kit was the February 2014 build. The modifications made in UDKOSC have been tested for a number of recent UDK builds as well. The UDKInstall-2014-02.exe can be downloaded from www.unrealengine.com at: http://download.udk.com/UDKInstall-2014-02.exe. The Installer can be started simply by running the executable and choosing a location into which the UDK will be unpacked. For the purposes of this document, the install directory will be assumed to be C:/UDK/UDK-2014-02.

A.2 Download UDKOSC

The UDKOSC project is currently hosted on the author's github repository:

- http://github/robertkhamilton/udkosc

All files in this repository can be downloaded freely. There are two branches in this repository, the **master** branch and the **dev** branch. The latest code releases and bleeding-edge features are generally added to the **dev** branch, while a stable working version of UDKOSC can be found in the **master** branch.

To clone the UDKOSC repository use the link:

- https://github.com/robertkhamilton/udkosc.git

An archival .zip archive of the github repository from June 2014 has been included in the Stanford Library's archival repository for this body of research which can be accessed at the following permanent URL:

- http://purl.stanford.edu/yy758rc6782

A.3 Install UDKOSC

Installing UDKOSC requires the copying of certain directories and files from the downloaded UDKOSC repository into specific locations of a UDK installation. The UDKOSC project archive is made up of the following directories:

- /Classes
- /Config
- /docs
- /oscpack_1_0_2
- /Scripts

It is necessary to move files from the /Classes, /Config and /oscpack_1_0_2 directories into specific locations in the UDK. The /docs directory contains additional installation instructions for UDKOSC as well as some additional reference materials describing the project. The /Scripts directory contains the code and sample data files for OSCControl, OSCRecorder and OSCPlayer. Additional description of these three ruby scripts can be found in Appendix B of this document.

A.3.1 OSC .dll

All bi-directional Open Sound Control communication used in UDKOSC is controlled using a Windows dynamic-link library, or .dll file based on OSCPack. That file is named **oscpack_1_0_2.dll** and can be found in the UDKOSC project at udkosc/oscpack_1_0_2/oscpack_1_0_2.dll. The oscpack .dll must be copied into a specific UDK directory so that the UDKOSC custom class files can bind data structures and methods from the .dll into the project.

To install the OSC .dll into UDK, copy oscpack_1_0_2.dll into the following directory:

- C:/UDK/UDK-2013-02/Binaries/Win32/UserCode/

Note: if your UDK installation was made into a location other than C:/UDK/UDK-2014-02/ it will be necessary to substitute your file path for the one presented here.

A.3.2 UDKOSC Class structure

The modifications made to elements of game play as well as the hooks used to bind the oscpack_1_0_2.dll to UDK classes are all found in the udkosc/Classes folder. The UDK contains a directory in which development projects and their source code can be placed at C:/UDK/UDK-2014-02/Development/Src/.

To install the UDKOSC classes:

- Create a new directory entitled "ut3osc" at the following location: C:/UDK/UDK-2014-02/Development/Src/ut3osc/
- Copy the contents of the udkosc/Classes directory into the newly created C:/UDK/UDK-2014-02/Development/Src/ut3osc/ directory.

A.3.3 UDKOSC Config files

Data used to drive custom UDKOSC functionality is stored in a series of .ini files in the udkosc/Config directory. UDK configuration files are stored in the C:/UDK/UDK-2013-02/UDKGame/Config/ directory.

To install the UDKOSC config files:

- Copy the contents of udkosc/Config into the C:/UDK/UDK-2014-02/UDKGame/Config/ directory.
- Delete all files in the C:/UDK/UDK-2014-02/UDKGame/Config/ directory that start with the prefix "UDK". At runtime, the contents from each file with the prefix "Default" are used to create each "UDK" file (so the file named UDKEngine.ini is created at runtime from the original DefaultEngine.ini). However, files are only newly created if they currently *do not exist.* Therefore, if changes are made to any .ini files with the "Default" prefix, the corresponding file with a "UDK" prefix *must be deleted* for the new changes to take effect.

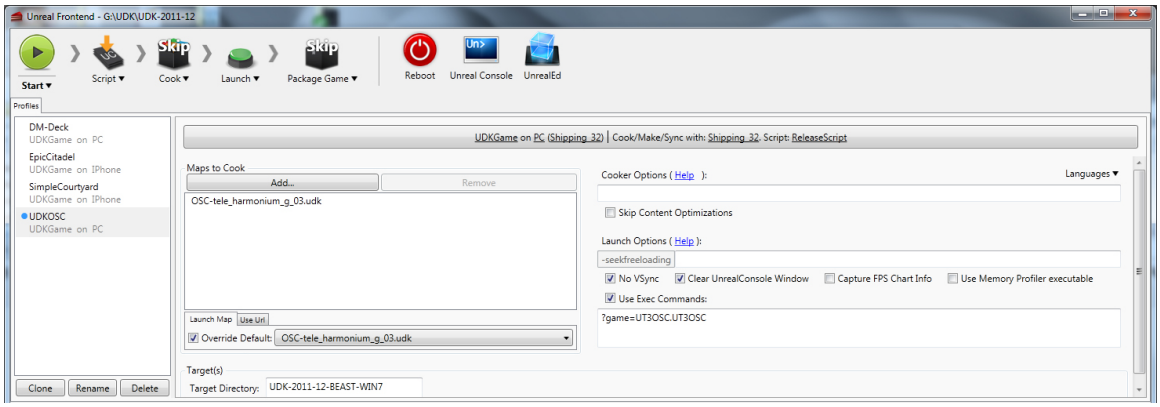


Figure A.1: The Unreal Frontend application can configure and run development code as well as compile projects for distribution. Of particular note, the UDK project profiles can be seen in the far left "Profiles" panel, the list of custom maps can be seen in the central "Maps to Cook" panel and the UDKOSC specific game type can be set in the "Use Exec Commands:" panel on the right bottom.

A.3.4 Maps and Resources

To use custom maps from within the UDKOSC project, each map name must first be modified with an "OSC-" prefix and then placed into a specific UDKOSC map directory. Custom resources to support each map must also be placed in a specific location.

To install custom maps:

- Create a new directory named "ut3osc" at the following location: C:/UDK/UDK-2014-02/UDKGame/Content/Maps/ut3osc/
- Prepend "OSC-" to each individual map filename. For example if your custom map is named "ccrma.udk", to use that map in UDKOSC it should be renamed "OSC-ccrma.udk".
- Copy each map into C:/UDK/UDK-2014-02/UDKGame/Content/Maps/ut3osc/
- Resource files to support custom maps must be installed into the C:/UDK/UDK-2014-02/UDKGame/Content/ directory.

A.4 Configure and Build UDKOSC Project

UDK projects can be compiled and run using the Unreal Frontend application included in the UDK. Unreal Frontend can be found at the following location: C:/UDK/UDK-2014-02/Binaries/UnrealFrontend.exe. A screen capture from Unreal Frontend can be seen in Figure A.1.

To configure and run a UDKOSC project from Unreal Frontend:

- Create a "UDKOSC" project profile by highlighting one of the existing profiles found in the "Profiles" window labeled "UDKGame on PC" such as "DM-Deck", pressing the "Clone" button at the bottom of the "Profiles" window, and renaming the newly cloned profile something like "UDKOSC".

- Add custom maps to the "Maps to Cook" panel by pressing the "Add..." button at the top of the panel and selecting the desired custom map. To set a particular map as the default map (meaning it will start automatically when Unreal Frontend launches the compiled project), click the "Override Default" button located at the bottom of the "Maps to Cook" panel, and select the desired map from the drop-down list-box.

- To set the custom game-type (thereby executing the custom UDKOSC codebase upon launch) for the current project, check the "Use Exec Commands" checkbox and add the following text into the text-box immediately below: *?game=UT3OSC.UT3OSC*

To get started building the fully-configured UDKOSC project:

- Delete all "UDK"-prefaced files from C:/UDK/UDK-2014-02/UDKGame/Config/ (this step only needs to be taken when .ini file changes have been made).
- Click on the "Script" button (second button from the left, on the top row) in Unreal Frontend and select "Full Recompile".
- Click on the "Cook" button (third button from the left) in Unreal Frontend and select "Clean and Full Recook".
- Click on the "Launch" button (fourth from the left) in Unreal Frontend.
- If the "Override Default" checkbox was not selected in the Unreal Frontend "Maps to Cook" panel, the UDK menu system will appear. At this point a "Game Mode" of "UT3OSC.UT3OSC" can be selected, any custom maps added can be selected, and the game can be launched.

Note: If there was an issue compiling the UDKOSC codebase, the project will display errors upon selecting "Full Recompile" in Unreal Frontend. It is outside of the scope of this document to address possible issues or solutions for compilation errors.

A.5 Test OSC Output and Input

After the UDKOSC project has launched successfully, OSC networking must be configured from within the running game. To acccess the UDK in-game console, press either the "~" key to enable a half-screen console" or the TAB key, to enable a single-line console at the bottom of the screen.

To configure and start OSC messaging:

- Start an OSC-receiving client application either on the localhost or on a machine that can be reached across the network. Make a note of that machine's IP address and the port configured in your OSC-receiving application.

- At the UDK console, set the target IP address and Port number by typing **setoschostname <hostname> <port>** where <hostname> is either "localhost" for processes running on the same machine or a valid IP address for networked machines such as **192.168.0.1**, and <port> is a valid and open listening port such as **7000.**

- To turn on OSC output from UDKOSC type **oscstartoutput**. If your OSC client is properly configured, data representing the avatar's position should be streaming across the network at this time.

- To turn on OSC input, allowing external OSC-generating software to send messages into UDKOSC, type **oscstartinput**. UDKOSC listens for incoming OSC input on port 7001.

Appendix B

OSC Toolkit

When working with Open Sound Control data in both artistic and research contexts, the ability to generate, record and playback timed OSC messages is essential. For creative works described in Chapter 3 such as *ECHO::Canyon,* OSC messages were used to control camera sequences and flocks of avatars during live concert performances. For the recording of sonified avatar motions that make up the crossmodal dataset described in Chapter 4, OSC streams from UDKOSC were captured and subsequently played back to create multiple sets of musical sonifications. For each of these purposes, a simple set of OSC tools was created using the Ruby programming language and the osc-ruby library [38]. OSCControl was designed specifically to generate real-time OSC output for controlling actor and camera trajectories as well as environment states for UDKOSC. OSCRecorder and OSCPlayer are simple Ruby scripts written by Tom Lieber that respectively record and playback OSC messages as binary packets using the YAML markup language [3].

B.1 OSCControl

OSCControl is a small ruby application that translates sets of human-readable control commands into OSC messages or bundles, complete with timing information. For UDKOSC, users can create scripts controlling both pawn and camera movements as well as turn on or off a series of environmental states and variables. OSCControl

scripts comprised of a series of timed commands are processed by OSCControl, first formatting them as valid timed OSC messages and subsequently sending them to the target IP and PORT of a running UDKOSC server.

Slewed parameter values over specific time intervals are easy to create, as are batched commands, sent as OSC bundles. OSCControl currently creates a time-ordered array of valid OSC messages and bundles and subsequently streams entire control scripts serially, meaning any timed commands written in the script will be sent in an ordered fashion. At each execution of OSCControl, the generated batch of OSC messages and bundles are streamed out to the desired target host and port in real-time.

OSCControl scripting commands make use of a simple a human-readable format that allows for single-line description of slewed and timed values. There are two primary classes of scripting commands, those that control indexed instances of the UDK OSCPawn class and those that control the default Camera class.

B.1.1 Setup

OSCControl uses both the "rubygems" and "osc-ruby" Ruby gems. To install, at a command-line type "gem install rubygems" and "gem install "osc-ruby". To use the included oscrecorder.rb and oscplayer.rb scripts, the YAML gem must be installed, using "gem install yaml".

B.1.2 Running OSCControl

OSCControl can be run either through a Ruby IDE such as RubyMine, or from a standard Terminal/Command-Line. A Target IP address and Port number can be set when launching from a command line using this syntax:

```
ruby osccontrol.rb <hostname> <port>
```

To target port 6666 on IP address 192.168.0.10, one would use:

```
ruby osccontrol.rb 192.168.0.10 6666
```

By default OSCControl will target port "7001" with hostname "localhost".

B.1.3 Actor Controls

In UDKOSC there are currently three types of Actors: human-controlled ”players”, script controlled ”pawns” and game AI-controlled ”bots”. ”Player” and ”Pawn” commands start respectively with ”playermove” or ”pawnmove”, and are followed by a series of parameters, their associated target values, an optional ”slew” time, and a userid number.

```
pawnmove <action> <target> <slew> <player-id>
```

For example, a simple command to immediately set a player's speed to 4000 (approximately 10x the default UDK pawn speed) would use the command:

```
playermove speed 4000.0 1
```

Here, ”4000.0” is the value of the player's speed parameter. As there is no slew time associated with this message, the speed change will happen instantly, resulting in an OSC message that looks like:

```
/udkosc/script/playermove/speed 4000.0 1
```

We can make this call more interesting by interpolating an actors speed from its current value (stored in OSCControl) to the target value (”2000.0” in this example) over a period of time, represented in milli-seconds using:

```
playermove speed 2000.0 200.0 1
```

It should be noted that while OSCControl's default temporal step-size is 20 ms, that value can be adjusted as necessary.

The generated OSC messages resulting from the slewed speed command are:

```
/udkosc/script/playermove/speed 200.000000 1
/udkosc/script/playermove/speed 400.000000 1
/udkosc/script/playermove/speed 600.000000 1
/udkosc/script/playermove/speed 800.000000 1
/udkosc/script/playermove/speed 1000.000000 1
/udkosc/script/playermove/speed 1200.000000 1
/udkosc/script/playermove/speed 1400.000000 1
/udkosc/script/playermove/speed 1600.000000 1
/udkosc/script/playermove/speed 1800.000000 1
/udkosc/script/playermove/speed 2000.000000 1
```

So over the slew period of 200.0 ms, we ramp the player's speed parameter from its current value (here a value of 0.0) to the target value of 2000.0.

Actors - either players or pawns - can be controlled using the following commands, some of which can be used with slew timings while others, which typically control single discrete events like "jump" or "crouch" cannot be used with slew timings. A description of each control follows:

```
pawnmove x <degree-value> <slew-time> <userid>
pawnmove y <degree-value> <slew-time> <userid>
pawnmove z <degree-value> <slew-time> <userid>
playermove pitch <degree-value> <slew-time> <userid>
playermove yaw <degree-value> <slew-time> <userid>
playermove roll <degree-value> <slew-time> <userid>
pawnmove speed <value> <slew-time> <userid>
playermove jump <height-value> <userid>
pawnmove teleport <x> <y> <z> <userid>
playermove stop <userid>
```

To move an actor in a specific direction, the "playermove" or "pawnmove" commands with X, Y, and Z coordinates are used to set the actor's direction vector with a

degree value relative to the world's absolute coordinate grid. If the user's speed is set to be non-0, setting the X and Y coordinates will start the user moving in the desired direction. As the Z coordinate represents the vertical plane, Z coordinate motion will only cause effect if the user is currently in a flying state.

Stop Control To stop an actor's motion, we use the "playermove stop <userid>" command. This will stop the actor moving. No value needs to be sent with a stop command and after a stop command is sent, the next playermove x, y, z, or jump command will toggle the stop state off, allowing the user to move freely.

Speed Control Actor speed can be set using "playermove speed <value> <ms-slew> <userid>". The UDK default speed is 300-400 in UnrealScript. A speed of "0" will not allow Actors to move in any direction. Speed values can be slewed, to create accelerations or decelerations.

Jump Control An Actor can be made to jump by sending a "playermove jump <height-value> <userid>" message. The height to which the user will jump is sent as the value for the jump message.

Teleport Control Actors can be moved to any coordinate location in the current environment instantly using the teleport command. This is useful in starting actor motions and actions from a specific location.

B.1.4 Camera Controls

The Camera associated with an actor can be controlled in- dependently using the following commands. Note that camera controls are formatted in much the same way as player controls except that there is currently no "cameraid" in use:

```
cameramove x <degree-value> <ms-slew>
cameramove y <degree-value> <ms-slew>
cameramove z <degree-value> <ms-slew>
cameramove pitch <degree-value> <ms-slew>
cameramove yaw <degree-value> <ms-slew>
cameramove roll <degree-value> <ms-slew>
```

X, Y, Z Coordinate controls Camera coordinate controls operate differently than Player coordinate controls. Where Player controls set the "direction" in which the player will move, Camera controls set the "location" to which the camera will move. So a "cameramove x 1000.0" control message will instantly move the camera to world-coordinate x=1000.0. The similar playermove call would turn the player to 1000 degrees and start them moving at the current speed.

B.1.5 Console Commands

Commands that toggle environmental states in UDKOSC including camera type, OSC functionality and specific tracking states used for the experiments described in this dissertation can be accessed using the “console” command:

```
console <command>
```

Table B.1 lists each of the console commands currently available in OSCControl.

B.1.6 Wait Command

OSCControl sends OSC messages as a stream of real-time output. We can use the "wait" command with a milli-second value to pad time before the next osc message is sent. So a message "wait 5000.0" would cause the output of messages to wait for 5 seconds before continuing on to the output of the next command.

OSCControl Console Commands	*Description*
oscstartoutput	Starts OSC output from UDKOSC
behindview	Toggles first-person camera view
oscmove	Mode toggle enabling avatar control with OSC
freecameraon	Turns on a freely-moving camera mode
freecameraoff	Turns off freely-moving camera mode
attachedcameraon	Attaches camera to avatar's location
attachedcameraoff	Turns off attached state
followcameraon	Static camera state tracking Pawn with rotation and height
followcameraoff	Turns off follow camera state
followlockcameraon	Static camera state with only rotation enabled
followlockcameraoff	Turns off follow lock camera state
followlockvertcameraon	Static camera tracking Pawn with height only
followlockvertcameraoff	Turns off follow lock vert camera
setstart	Send a "start" marker through the OSC output
setend	Send an "end" marker through the OSC output
setstarton	Send a "start" marker value through the OSC output
setstartoff	Send an "end" marker value through the OSC output
setendon	Send a "end on" marker value through the OSC output
setendoff	Send an "end off" marker value through the OSC output

Table B.1: OSCControl Console Commands.

B.1.7 Command Blocks

In addition to single-thread messages (messages or slews of messages that occur in series), OSCControl allows for the creation of "blocks" of messages, which will occur and slew concurrently. These messages will be sent as OSC bundles, which package a set of messages as a single bundle, arriving at the client at the same time.

Block control messages are formatted using square braces, marking the opening and closing of a given block:

```
[
  playermove speed 1000.0 5000.0 1
  playermove x 300.0 5000.0 1
  cameramove z 400.0
  cameramove x 1000.0 400.0
]
```

In this example, the playermove speed and x messages as well as the cameramove x message will all slew to their target values over the time specified in their slew parameter. The "cameramove z 400.0" message has no slew value so it will be output at the beginnning of the block. The entire block will run for 5000.0 ms, as that is the largest time value set in the block.

As this block executes, the camera will instantly move to a height (Z value) of 400.0, the playermove speed value and X value will slew from their current values to their respective target values of 300.00 of 1000.0 over 5 seconds, and at the same time, the camera x parameter will slew to a target of 1000.0 over 400.0 ms.

B.2 OSC Recording and Playback

Recording and playing-back Open Sound Control data streams can be accomplished using the simple Ruby helper scripts *oscrecorder.rb* and *oscplayer.rb*. To record and playback scripts without the (minimal) overhead of running OSCControl, two helper scripts, "oscrecorder.rb" and "oscplayer.rb" are included, which respectively encode OSC messages/bundles using YAML.

To use the oscrecorder.rb script to record an incoming OSC stream on port 7000, an example of the syntax would be:

```
$ ruby oscrecorder.rb 7000 > saved.yml
```

To use the oscplayer.rb script to playback the same recorded OSC stream to the localhost on port 7001, an example of the syntax would be:

```
$ ruby player.rb localhost 7001 false < saved.yml
```

In the latter case, a target IP address and target port are passed as arguments, followed by a boolean that toggles whether time before the first received packet at the beginning of the OSC recording should be ignored.

Bibliography

[1] Gtkradiant. URL http://www.q3radiant.com.

[2] Settembre Musica: Milano-Torino (MiTo) Music Festival. URL http://www.mitosettembremusica.it/en/festival/introduction.html.

[3] YAML Homepage. URL http://www.yaml.org.

[4] ioquake3. URL http://ioquake3.org/.

[5] Ffmpeg, 2000. URL http://ffmpeg.org.

[6] Lovebytes festival, 2006. URL (http://www.lovebytes.org.uk/.

[7] 2013 Sales, Demographic and Usage Data: Essential Facts About the Computer and Video Game Industry, 2013. URL http://www.theesa.com/facts/pdfs/esa_ef_2013.pdf.

[8] Oculus rift, 2014. URL http://www.oculusvr.com/.

[9] Unreal development kit, february 2014 beta, May accessed 2014. URL http://download.udk.com/UDKInstall-2014-02.exe.

[10] A. Agresti. *Categorical Data Analysis. 2nd edition.* John Wiley & Sons, San Francisco, 2002.

[11] A. Aristidou and J. Lasenby. Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver. Technical report, Cambridge University Engineering Department, Cambridge, UK, 2009.

[12] R. Bencina. oscpack. URL http://www.rossbencina.com/code/oscpack.

[13] I. H. Bernstein and B. A. Edelstein. Effects of some variations in auditory input upon visual choice reaction time. *Journal of Experimental Psychology*, 87:241–247, 1971.

[14] B. Blau. Gartner Says Worldwide Video Game Market to Total $93 Billion in 2013, 2013. URL http://www.gartner.com/newsroom/id/2614915.

[15] B. Bond and S. S. Stevens. Cross-modality matching of brightness to loudness by 5-year-olds. *Perception & Psychophysics*, 6:337–339, 1969.

[16] R. Boulanger and M. Mathews. The 1997 Mathews radio-baton and improvisation modes. In *Proceedings of the International Computer Music Conference*, pages 395–398, Thessaloniki, Greece, 1997.

[17] D. Bradshaw and K. Ng. Tracking Conductors Hand Movements Using Multiple Wiimotes. In *International Conference on Automated solutions for Cross Media Content and Multi-channel Distribution.*, pages 93–99, Nov 2008. doi: 10.1109/AXMEDIS.2008.40.

[18] A. Bregman. *Auditory Scene Analysis: The Perceptua! Organization of Sound.* MIT Press, Cambridge, MA., 1990.

[19] P. Brinkmann, C. McCormick, P. Kirn, M. Roth, R. Lawler, and H.-C. Steiner. Embedding pure data with libpd. In *Proceedings of the Fourth International Pure Data Convention*, pages 291–301, June 2011.

[20] J.-P. Cáceres and C. Chafe. JackTrip/SoundWIRE meets server farm. *Computer Music Journal*, 34(3):29–34, 2010.

[21] J.-P. Cáceres, R. Hamilton, D. Iyer, C. Chafe, and G. Wang. To the edge with china: Explorations in network performance. In *ARTECH 2008: Proceedings of the 4th International Conference on Digital Arts*, pages 61–66, Porto, Portugal, 2008. ISBN 978-989-95776-3-3.

[22] R. Cannon. *Meltdown*, page 44. Intellect Books, Bristol, UK, 2007.

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

[24] E. F. Clarke. Meaning and the specification of motion in music. *Musicae Scientiae*, 5:213–234, 2001.

[25] K. Collins. From Bits to Hits Video Games Music Changes its Tune. *Film International*, 12:4–19, January 2012.

[26] N. Cook. *Music, Imagination and Culture*. Clarendon Press/Oxford University Press, New York, NY, US, 1990.

[27] P. Cook and G. Scavone. The Synthesis ToolKit (STK). In *Proceedings of the International Computer Music Conference*, Beijing, China, 1999.

[28] S. Dahl, F. Bevilacqua, R. Bresin, M. Clayton, L. Leante, I. Poggi, and N. Rasamimanana. *Gestures In Performance*, pages 36–68. Routledge, New York, 2010.

[29] M. Danks. PD in video game Spore; forum communication. URL http://lists.puredata.info/pipermail/pd-list/2007-11/056307.html.

[30] R. R. Davidson. On Extending the Bradley-Terry Model to Accommodate Ties in Paired Comparison Experiments, March 1970. ISSN 0162-1459 (Print), 1537-274X (Online).

[31] T. Davis and O. Karamanlis. Gestural Control of Sonic Swarms: Composing with Grouped Sound Objects. In *Proceedings of the 4th Sound and Music Computing Conference*, Lefkada, Greece, July 2007.

[32] Z. Eitan and R. Y. Granot. How Music Moves: Mmusical Parameters and Listeners' Images of Motion. *Music Perception: An Interdisciplinary Journal*, 23(3):221–248, February 2006. doi: 10.1525/mp.2006.23.3.221. URL http://www.jstor.org/stable/10.1525/mp.2006.23.3.221.

[33] K. K. Evans and A. Treisman. Natural cross-modal mappings between visual and auditory features. *Journal of Vision*, 10(1)(6):1–12, 2010.

[34] A. Farnell. An Introduction to Procedural Audio and its Application in Computer Games. In *Audio Mostly Conference*, pages 1–31, Pitea, Sweden, 2007.

[35] A. Farnell. Keynote Lecture, Satellite Workshop at the 14th International Conference on Digital Audio Effects (DAFx), Paris, France, 2011. URL http://vimeo.com/35254559.

[36] C. Francois and D. Schön. Musical expertise boosts implicit learning of both musical and linguistic structures. *Cerebral Cortex*, 2011. doi: 10.1093/cercor/bhr022. URL http://cercor.oxfordjournals.org/content/early/2011/05/13/cercor.bhr022.abstract.

[37] J. Freeman, S. Ramakrishnan, K. Varnik, M. Neuhaus, P. Burk, and D. Birchfield. The Architecture of Auracle: A Voice-Controlled, Networked Sound Instrument. In *Proceedings of the International Computer Music Conference*, Barcelona, Spain, 2005.

[38] T. Funaba and C. Harris. osc-ruby, 2013. URL https://github.com/aberant/osc-ruby.

[39] K. Furukawa, M. Fujihata, and W. Muench, 2000. URL http://hosting.zkm.de/wmuench/small_fish.

[40] E. Games. Unreal Development Kit (UDK). URL https://www.unrealengine.com/products/udk/.

[41] E. Games. Unreal Unit conversion to real life conversion?, 2011. URL https://forums.epicgames.com/threads/872123-Unreal-Unit-conversion-to-real-life-conversion.

[42] J. Garcia. Samples Homogenization for Interactive Soundscapes. Msc thesis in sound and music computing, Universitat Pompeu Fabra, Spain, 2011.

[43] W. W. Gaver. What in the world do we hear? an ecological approach to auditory event perception. *Ecological Psychology*, 5:1–29, 1993.

[44] J. J. Gibson. *The Perception of the Visual World*. Houghton Mifflin, Boston, 1950.

[45] J. J. Gibson. *The Senses Considered as Perceptual Systems*. Houghton Mifflin, Boston, 1966.

[46] J. J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston, 1979. ISBN 0898599598.

[47] S. Gresham-Lancaster. The aesthetics and history of the Hub: The effect of changing technology on network computer music. *Leonardo Music Journal*, 8: 39–44, 1998.

[48] R. Hamilton. Maps and Legends: Designing FPS-based Interfaces for Multi-User Composition, Improvisation and Interactive Performance. *Lecture Notes in Computer Science: Computer Music Modeling and Retrieval*, 2008.

[49] R. Hamilton. q3osc: Or How I Learned To Stop Worrying And Love The Game. In *Proceedings of the International Computer Music Conference*, Belfast, Ireland, August 2008.

[50] R. Hamilton. Building Interactive Networked Musical Environments Using q3osc. In *AES 35th International Conference*, London, UK, February 2009.

[51] R. Hamilton. Udkosc An immersive musical environment. In *Proceedings of the International Computer Music Conference*, pages 717–720, Huddersfield, UK, August 2011.

[52] R. Hamilton. Sonifying game-space choreographies using udkosc. In *Proceedings of the New Interfaces for Musical Expression International Conference*, Daejeon, Korea, 2013.

[53] R. Hamilton, J.-P. Cáceres, C. Nanou, and C. Platz. Multi-modal musical environments for mixed-reality performance. *Journal on Multimodal User Interfaces*, 4:147–156, 2011. ISSN 1783-7677. doi: 10.1007/s12193-011-0069-1. URL http://dx.doi.org/10.1007/s12193-011-0069-1.

[54] R. Hamilton, J. Smith, and G. Wang. Social Composition: Musical Data Systems for Expressive Mobile Music. *Leonardo Music Journal*, 21, 2011.

[55] S. Handel. *Perceptual Coherence: Hearing and Seeing*. Oxford University Press, New York, 2006.

[56] Harmonix. Guitar hero, 2005. URL http://www.guitarhero.com.

[57] Harmonix. Rock band, 2007. URL http://www.rockband.com.

[58] C. Hendrix and W. Barfield. Presence within Virtual Environments as a Function of Visual Display Parameters. *Presence: Teleoperators and Virtual Environments*, 5(3):274–289, 1996.

[59] H. Hoffman, J. Prothero, M. Wells, and J. Groen. Virtual Chess: Meaning enhances users' sense of presence in virtual environments. *International Journal of Human-Computer Interaction*, 10(3):251–263, 1998.

[60] D. Horn, E. Cheslack-Postava, B. Mistree, T. Azim, J. Terrace, M. J. Freedman, and P. Levis. To Infinity and Not Beyond: Scaling Communication in Virtual Worlds with Meru. Technical report, 2010.

[61] G. R. I. Gestural Affordances of Musical Sound. pages 103–125, 2010.

[62] Id Software. Quake 3 arena. URL www.idsoftware.com/games/quake/quake3-arena.

[63] R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996. ISSN 1061-8600 (Print), 1537-2715 (Online). doi: 10.1080/10618600.1996.10474713.

[64] T. Iwai. *Electroplankton User Manual*. Nintendo of America, Inc., 2005.

[65] A. Jensenius, M. Wanderley, R. Godøy, and M. Leman. *Musical Gestures: Concepts and Methods of Research*, page 13. Routledge, New York, 2010.

[66] M. Johnson. *The Body in the Mind: The Bodily Basis of Meaning, Imagination, and Reason*. University of Chicago Press, Chicago, 1989.

[67] C. L. Krumhansl. *Cognitive foundations of musical pitch*. Oxford University Press, New York, 1990.

[68] M. Kubovy and V. V. D. Auditory and visual objects. *Cognition*, 80:97–126, 2001.

[69] M. Kubovy and M. Schutz. Audio-Visual Objects. *Review of Philosophy and Psychology*, 1(1):41–61, Jan. 2010. ISSN 1878-5158. doi: 10.1007/s13164-009-0004-5. URL http://link.springer.com/10.1007/s13164-009-0004-5.

[70] W. Köhler. *Gestalt psychology*. Liveright, New York, 1929.

[71] W. Köhler. *Gestalt psychology: An introduction to new concepts in modern psychology*. Liveright, New York, 1947.

[72] G. Lakoff and M. Johnson. *Metaphors We Live By*. University of Chicago Press, Chicago, 1980.

[73] G. Lakoff and M. Turner. *More Than Cool Reason: A Field Guide to Poetic Metaphor*. University of Chicago Press, Chicago, 1989.

[74] S. Lehar. Gestalt isomorphism and the primacy of subjective conscious experience: A gestalt bubble model. *Behavioral and Brain Sciences*, 26(4): 375–444, March 2004. URL http://cns-alumni.bu.edu/~slehar/webstuff/bubw3/bubw3.html.

[75] P. J. Loewen, D. Rubenson, and A. Spirling. Testing the power of arguments in referendums: A Bradley-Terry approach. *Electoral Studies*, 31(1):212–221, March 2012. ISSN 0261-3794.

[76] J. M. Loomis. Distal attribution and presence. *Presence: Teleoperators and Virtual Environments*, 1:113–119, 1992.

[77] D. Malham and A. Myatt. 3-D Sound Spatialization using Ambisonic Techniques. *Computer Music Journal*, 19(4):58–70, Winter 1995.

[78] F. Marinetti. The Founding and Manifesto of Futurism. *Le Figaro*, February 1909.

[79] J. Markoff. *What the Dormouse Said: How the Sixties Counterculture Shaped the Personal Computer Industry*. Penguin Group, New York, New York, 2005.

[80] L. E. Marks. On associations of light and sound: The mediation of brightness, pitch, and loudness. *The American Journal of Psychology*, 87:173–188, 1974.

[81] L. E. Marks. Cross-modal interactions in speeded classification. In C. S. G. A. Calvert and B. E. Stein, editors, *Handbook of multisensory processes*, pages 85–105. MIT Press, Cambridge, 2004.

[82] L. E. Marks, R. J. Hammeal, and M. H. Bornstein. Perceiving similarity and comprehending metaphor. *Monographs of the Society for Research in Child Development*, 52(1, Whole No. 215):1–102, 1987.

[83] W. Mason and S. Suri. Conducting behavioral research on Amazon?s Mechanical Turk. *Behavior Research Methods*, 44(1):1–23, 2011.

[84] M. Mathews. The radio drum as a synthesizer controller. In *Proceedings of the International Computer Music Conference*, pages 42–45, 1989.

[85] M. Mathews and R. Moore. GROOVE: A Program to Compose, Store, and Edit Function of Time. *Communication of the ACM*, 13(12), 1970.

[86] S. McAdams. *Spectral Fusion, Spectral Parsing and the Formation of Auditory Images*. PhD thesis, Stanford University, Stanford, California, 05/1984 1984. URL https://ccrma.stanford.edu/files/papers/stanm22.pdf.

[87] S. McAdams. Recognition of sound sources and events. *Thinking in sound: the cognitive psychology of human audition*, pages 146–198, 1993.

[88] J. McCartney. Rethinking the Computer Music Language: SuperCollider. *Computer Music Journal*, 26(4):61–68, Dec. 2002. ISSN 0148-9267. doi: 10.1162/014892602320991383. URL http://dx.doi.org/10.1162/014892602320991383.

[89] M. Minsky. Telepresence. *Omni*, June 1980. URL http://web.media.mit.edu/~minsky/papers/Telepresence.html.

[90] S. Mudd. Spatial stereotypes of four dimensions of pure tone. *Journal of Experimental Psychology*, 66:347–352, 1963.

[91] I. Nintendo. *Wii Operations Manual*. 2009.

[92] J. Oh and G. Wang. Evaluating crowdsourcing through amazon mechanical turk as a technique for conducting music perception experiments. In E. Cambouropoulos, C. Tsougras, P. Mavromatis, and K. Pastiadis, editors, *... Conference on Music Perception ...*, pages 738–743, Thessaloniki, Greece, 2012. URL http://icmpc-escom2012.web.auth.gr/sites/default/files/papers/738_Proc.pdf.

[93] J. Oliver. documentation of q3apd lovebytes06, 2006. URL https://www.youtube.com/watch?v=rlwGNtSv2g0.

[94] J. Oliver and S. Pickles. q3apd: Making music with bots, 2009. URL http://vimeo.com/1380686.

[95] L. J. Paul. Video Game Audio Prototyping with Half-Life 2. In R. Adams, S. Gibson, and S. Arisona, editors, *Transdisciplinary Digital Art. Sound, Vision and the New Screen*, volume 7 of *Communications in Computer and Information Science*, pages 187–198. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-79485-1. doi: 10.1007/978-3-540-79486-8_17. URL http://dx.doi.org/10.1007/978-3-540-79486-8_17.

[96] S. Pickles. fijuu2, 2007. URL http://sourceforge.net/projects/fijuu2/.

[97] C. C. Pratt. The spatial character of high and low tones. *Journal of Experimental Psychology*, 13:278–285, 1930.

[98] J. D. Prothero and H. D. Hoffman. Widening the field-of-view increases the sense of presence within immersive virtual environments: Human Interface Technology Laboratory Technical Report R-95-4. 1995.

[99] M. Puckette. Pure Data. In *Proceedings, International Computer Music Conference*, pages 269–272, San Francisco, 1996. International Computer Music Association.

[100] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. URL http://www.R-project.org.

[101] B. R.A. and T. M.E. Rank analysis of incomplete block designs i: The method of paired comparisons. *Biometrika*, 39:324–45, 1952.

[102] P. Rao and L. Kupper. Ties in Paired-Comparison Experiments: A Generalization of the Bradley-Terry Model. *Journal of the American Statistical Association*, 62:194–204, 1967. ISSN 0162-1459 (Print), 1537-274X (Online). doi: 10.1080/01621459.1967.10482901.

[103] S. K. Roffler and R. A. Butler. Factors that influence the localization of sound in the vertical plane. *The Journal of the Acoustical Society of America*, 43: 1255–1259, 1968.

[104] P. Rubin. The Inside Story of Oculus Rift and How Virtual Reality Became Reality. *Wired Magazine*, 22, June 2014. URL http://www.wired.com/2014/05/oculus-rift-4/.

[105] M. Schutz and M. Kubovy. Causality and cross-modal integration. *Journal of Experimental Psychology: Human Perception and Performance*, 35(6): 1791–1810, 2009.

[106] M. Schutz and S. Lipscomb. Hearing gestures, seeing music: Vision influences perceived tone duration. *Perception*, 36(6):888–897, 2007. ISSN 0301-0066. doi: 10.1068/p5635. URL http://www.perceptionweb.com/abstract.cgi?id=p5635.

[107] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, volume 3, 2011.

[108] M. Slater and A. Steed. A virtual presence counter. *Presence: Teleoperators and Virtual Environments*, 9(5):413–434, 2000.

[109] SoundWIRE Group. SoundWIRE research group at CCRMA, Stanford University, 2014. URL http://ccrma.stanford.edu/groups/soundwire/.

[110] C. Spence. Crossmodal correspondences: a tutorial review. *Attention, Perception & Psychophysics*, 73(4):971–995, 2011. doi: 10.3758/s13414-010-0073-7.

[111] C. Spence, D. Sanabria, and S. Soto-Faraco. *Intersensory Gestalten and Crossmodal Scene Perception*. Nihon University College of Humanities and Sciences, 2007.

[112] J. C. Stevens and L. E. Marks. Cross-modality matching of brightness and loudness. *Proceedings of the National Academy of Sciences*, 54:407–411, 1965.

[113] Thatgamecompany. Journey. [Digital Download], 2012.

[114] B. Tillmann. Music and language perception: Expectations, structural integration, and cognitive sequencing. *Topics in Cognitive Science*, pages 1–17, 2012. ISSN 1756-8757 print / 1756-8765 online.

[115] D. Trueman, P. R. Cook, S. Smallwood, and G. Wang. Plork: Princeton laptop orchestra, year 1. In *Proceedings of the International Computer Music Conference*, New Orleans, 2006.

[116] H. Turner and D. Firth. Bradley-Terry Models in R: The BradleyTerry2 Package. *Journal of Statistical Software*, 48(9):1–21, 2012. ISSN 1548-7660. URL http://www.jstatsoft.org/v48/i09/.

[117] M. Turner. Aspects of the Invariance Hypothesis. *Cognitive Linguistics*, 1(2): 254, 1990.

[118] C. Verron and G. Drettakis. Procedural audio modeling for particle-based environmental effects. In *Proceedings of the 133rd AES Convention*, San Francisco, 2012. Audio Engineering Society.

[119] G. Wang. *The ChucK Audio Programming Language: A Strongly-timed and On-the-fly Environmentality*. PhD thesis, Princeton University, Princeton, New Jersey, 2008.

[120] G. Wang. Designing Smule's iPhone Ocarina. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Pittsburgh, June 2009.

[121] G. Wang, R. Fiebrink, and P. R. Cook. Combining Analysis and Synthesis in the ChucK Programming Language. In *Proceedings of the International Computer Music Conference*, Copenhagen, 2007.

[122] G. Wang, N. Bryan, J. Oh, and R. Hamilton. Stanford Laptop Orchestra (SLORK). In *Proceedings of the International Computer Music Association Conference*, Montreal, Canada, 2009.

[123] F. W. Wicker. Mapping the intersensory regions of perceptual space. *The American Journal of Psychology*, 81:178–188, 1968.

[124] B. Witmer and M. Singer. Measuring immersion in virtual environments: Tech. Report No. 1014. Technical report, Alexandria, VA, 1994.

[125] B. Witmer and M. Singer. Measuring presence in virtual environments: A presence questionnaire. *Presence: Teleoperators and Virtual Environments*, 7 (3):225–240, 1998.

[126] M. Wright. Open sound control 1.0 specification, 2002. URL http://opensoundcontrol.org/spec-1_0.

[127] M. Wright. Open sound control: an enabling technology for musical networking. *Organised Sound*, 10:193–200, 2005.

[128] L. Zbikowski. Metaphor and Music Theory. *Music Theory Online*, 4, 1998. URL http://www.mtosmt.org/issues/mto.98.4.1/mto.98.4.1.zbikowski.html#FN1.

[129] L. Zbikowski. *Conceptualizing music: Cognitive structure, theory, and analysis*. Oxford University Press, New York, 2002.