RUNNING EXPRESSIONS:

PHYSIOLOGICAL MONITORS AND MOTION SENSORS MAPPED FOR MUSICAL PERFORMANCE

By JON P. BELLONA

A TERMINAL CREATIVE PROJECT

Presented to the University of Oregon School of Music and Department of Dance in partial fulfillment of the requirements for the degree of

Master of Music in Intermedia Music Technology

June 2011

Running Expressions is a terminal creative project prepared by Jon P. Bellona in partial fulfillment of the requirements for the Master of Music in Intermedia Music Technology degree in the School of Music and Department of Dance. This terminal creative project has been approved and accepted by:

Jeffrey Stolet, Chair of the Examining Committee

Date

Committee in Charge:

Jeffrey Stolet, Chair Molly Barth Robert Ponto © 2011 Jon Bellona

ACKNOWLEDGEMENTS

I would like to extend my gratitude to both my parents, Steve and Kristine Bellona, for their love and support, and to say thank you to my brother, David Bellona, for his creative insights, design critiques, and aberrant resources that have found their way into my work. I would also like to thank Professor Jeffrey Stolet, for without his knowledge and endearing support throughout my graduate studies at the University of Oregon, this project simply would not have been possible.

TABLE OF CONTENTS

| INTRODUCTION | 1 |
|---|----|
| PART I. UNDERLYING ARCHITECTURE (SIGNAL FLOW) | 3 |
| 1. Musical Hardware Connections | 3 |
| 2. Software Connections | 4 |
| 3. Video Connections | 5 |
| PART II. HARDWARE | 6 |
| 4. T-31 Coded [™] Polar Heart Rate Monitor Transmitter (HRM) | 6 |
| 5. Polar Heart Rate Monitor Interface (HRMI) | 6 |
| 6. Nintendo Wiimotes | 7 |
| 7. ADXL322 Dual-Axis Accelerometer | 7 |
| 8. JeeNode wireless Tx/Rx | 8 |
| PART III. SOFTWARE | 10 |
| 9. Processing | 10 |
| 10. OSCulator | 12 |
| 11. PacaConnect | 17 |
| 12. Max/MSP/Jitter | 18 |
| 12.a. Data Hub | 18 |
| 12.a.i. Heart Rate Monitor from Processing | 18 |
| 12.a.ii. Nintendo Wiimotes via OSC messages from OSCulator | 19 |
| 12.a.iii. JeeNode and Accelerometers from Serial Bus | 20 |
| 12.b. Musical Parameter Controller | 23 |
| 12.c. Video Projection Controller | 23 |
| 13. Kyma | 28 |

| PART IV. COMPOSITION AND PERFORMANCE STRUCTURE | |
|---|-----|
| 14. Section I: Exposition | 41 |
| 14.a. Heart Exposition | 41 |
| 14.b. Feet Exposition | 41 |
| 15. Section II: Development | 42 |
| 15.a. Running on Dillard (trombones, strings, piano) | 42 |
| 15.b. Running on Spencer's Butte (Climax) | 42 |
| 16. Section III: Recapitulation/Coda | 43 |
| APPENDIX | 44 |
| A.1. Controller_Kyma33_End4b.maxpat Figure Documentation | 44 |
| A.1.a. Exposition Sequencer | 47 |
| A.1.b. JeeNode Accelerometers | 50 |
| A.1.c. Heart Rate Monitor | 61 |
| A.1.d. Control Window | 64 |
| A.1.e. MIDI | 67 |
| A.1.f. Video Control | 68 |
| A.1.g. Wiimote Master | 78 |
| A.1.h. Wiimote 1 | 81 |
| A.1.i. Wiimote 2 | 86 |
| A.2. VPT_4.1b5_RunningExpressions.maxpat Figure Documentation | 96 |
| A.2.a. Videoplane Module: Running | 102 |
| A.2.b. Videoplane Module: Heart Rate | 106 |
| A.2.c. Videoplane Module: LCD | 109 |
| A.2.d. Preset Module | 109 |
| A.2.e. Movie Source Module: Running #1 | 114 |

| A.2.f. Movie Source Module: Running #2 | 121 |
|--|-----|
| A.2.g. Movie Source Module: Heart Rate LCD display | 123 |
| A.2.h. Movie Source Module: Heartbeat Movie | 124 |
| A.2.i. Movie Source Module: Distance LCD display | 126 |
| A.2.j. Cue List Mixer Module | 126 |
| A.2.k. Mixer Module: Running | 127 |
| A.2.1. Mixer Module: Heart | 128 |
| A.3. Master's Project Proposal | 129 |
| A.4. Graphical Icon Legend | 132 |
| A.5. Resource URLs | 135 |
| A.6. Included DVD Contents | 135 |

TABLE OF FIGURES

| Figure 1.1. Hardware Connections Flowchart | 3 |
|---|----|
| Figure 2.1. Software Connections Flowchart | 4 |
| Figure 3.1. Video Connections Flowchart | 5 |
| Figure 8.1. Accelerometer Spike Fluctuations | 9 |
| Figure 9.1. Processing optimization (before 100ms interval added) | 11 |
| Figure 9.2. Processing optimization (after 100ms interval added) | 11 |
| Figure 9.3. Additional Processing code, limits time between data requests | 12 |
| Figure 10.1. OSCulator Signal Flowchart | 13 |
| Figure 10.2. OSC messages from Max/MSP, routed to Kyma | 14 |
| Figure 10.3. OSC messages from Max/MSP, routed to Kyma | 15 |
| Wiimote messages, routed to Max/MSP | |
| Figure 10.4. OSC messages from Wiimotes, routed to Max/MSP | 16 |
| Figure 11.1. PacaConnect Signal Flowchart | 17 |
| Figure 12.a.1. MaxLink external object | 19 |
| Figure 12.a.2. OSC-route external object | 20 |
| Figure 12.a.3. JeeNode Tx and Accelerometer Pouch | 21 |
| Figure 12.a.4. Data stream table of the right leg accelerometer | 22 |
| Figure 12.c.1. Videoplane cleanup documentation | 24 |
| Original patch in Presentation mode. | |
| Figure 12.c.2. Videoplane cleanup documentation | 25 |
| Original patch in Editing mode. | |
| Figure 12.c.3. Videoplane cleanup documentation | 25 |
| Clean patch in Editing mode. | |
| Figure 12.c.4. Movie source cleanup documentation | 26 |
| Original patch in Presentation mode. | |
| Figure 12.c.5. Movie source cleanup documentation | 26 |
| Original patch in Editing mode. | |

| Figure | 12.c.6. Movie source cleanup documentation | 27 |
|--------|---|----|
| | Clean running movie source patch in Editing mode. | |
| Figure | 12.c.7. Movie source cleanup documentation | 28 |
| | Clean LCD movie source patch in Editing mode. | |
| Figure | 13.1. Kyma TL, with WaitUntil Sound track and MIDI note track highlighted | 29 |
| Figure | 13.2. WaitUntil Sound Object | 29 |
| Figure | 13.3. MIDI Output Pitch | 30 |
| Figure | 13.4. Heartbeat Sound | 30 |
| Figure | 13.5. Heartbeat Exposition Main Sound, Vocoder with Delays | 31 |
| Figure | 13.6. Heartbeat Exposition Sequencer | 32 |
| Figure | 13.7. Heartbeat Low Rumble | 32 |
| Figure | 13.8. Road Environment Ambient Sound | 33 |
| Figure | 13.9. Selectable Foot Sounds | 33 |
| Figure | 13.10. Feet Exposition Waltz | 34 |
| Figure | 13.11. Selectable Children Sounds | 34 |
| Figure | 13.12. Development Section for Trombones, Piano, and Strings | 35 |
| Figure | 13.13. Aorta Sound Transition | 35 |
| Figure | 13.14. Development Section Climax, part 1 | 36 |
| Figure | 13.15. Development Section Climax, part 2 | 36 |
| Figure | 13.16. Crashing Forests Sounds, Randomly Selected | 37 |
| Figure | 13.17. Final Piano Chord | 37 |
| Figure | 13.18. Wind Environment Sound | 38 |
| Figure | 13.19. Exposition Sequencer Revisited | 38 |
| Figure | 13.20. Exposition Heartbeat Vocoder Revisited | 39 |
| Figure | 13.21. Kyma TL, with Sections Labeled | 40 |
| Figure | A.1.1. Controller_Kyma33_End4b.maxpat Main Patch Window | 44 |
| Figure | A.1.2. Performance Setup Order Patch Window | 45 |
| Figure | A.1.3. Color Legend for Master Controller Max Patch | 46 |
| Figure | A.1.a.1. Exposition Sequencer Patch Window | 47 |

| Figure A.1.a.2. Sequencer Control Patcher | 48 |
|--|----|
| Figure A.1.a.3. Wiimote 1 Controls Presets Patcher | 48 |
| Figure A.1.a.4. Sequencer Tempo Control Patcher | 49 |
| Figure A.1.a.5. Tap Tempo Sequencer Control Patcher | 49 |
| Figure A.1.b.1. JeeNode Patch Window, in Presentation mode | 50 |
| Figure A.1.b.2. Serial Data Input Module | 51 |
| Figure A.1.b.3. Serial Port Formatting Menu Patcher | 51 |
| Figure A.1.b.4. Serial Port Formatting Message Patcher | 52 |
| Figure A.1.b.5. Serial Channel Data Display Module | 52 |
| Figure A.1.b.6. Accelerometer Threshold Counter Patcher | 53 |
| Figure A.1.b.7. Foot Distance Calculator Patcher | 54 |
| Figure A.1.b.8. Master Foot Distance Display Module 1 | 54 |
| Figure A.1.b.9. Master Feet Distance Calculator per Section Patcher, video control | 55 |
| Figure A.1.b.10. Master Accelerometer Control and Routing Module 2 | 55 |
| Figure A.1.b.11. Feet Accelerometer Tempo Control Patcher | 56 |
| Figure A.1.b.12. Master Feet Counter Calculator Patcher, controls video | 57 |
| Figure A.1.b.13. Master Feet Distance Calculator Patcher, lcd display | 57 |
| Figure A.1.b.14. Accelerometer Sends to Kyma Patcher part 1 | 58 |
| Figure A.1.b.15. Accelerometer Sends to Kyma Patcher part 2 | 58 |
| Figure A.1.b.16. Master Accelerometer Control Module 3 | 59 |
| Figure A.1.b.17. Master Feet Counter Control Patcher | 59 |
| Figure A.1.b.18. Master Feet Counter Test Patcher | 60 |
| Figure A.1.c.1. Heart Rate Routing Patch Window | 61 |
| Figure A.1.c.2. Heart Rate Controls Movie Playback Patcher | 62 |
| Figure A.1.c.3. Heart Rate Controls Heartbeat/Aorta Audio Playback Patcher | 62 |
| Figure A.1.c.4. Heart Rate Controls Switch Gate Patcher | 63 |
| Figure A.1.c.5. Master Heartbeat Audio Volume Control Patcher | 63 |
| Figure A.1.d.1. Performance Control Patch Window, in Presentation mode | 64 |
| Figure A.1.d.2. Performance Control Patch Window, in Patcher mode | 65 |
| | |

| Figure A.1.d.3. Performance Control Timer as Counter Patcher | 66 |
|---|----|
| Figure A.1.d.4. Performance Control Timer as Time Patcher | 66 |
| Figure A.1.e.1. MIDI Controls Patch Window | 67 |
| Figure A.1.f.1. Video Control Patch Window, overview of Window layout | 68 |
| Figure A.1.f.2. Video Control MIDI routing, part 1 | 68 |
| Figure A.1.f.3. Video Control MIDI routing, part 2 | 69 |
| Figure A.1.f.4. Video Section Command Descriptions Patcher | 70 |
| Figure A.1.f.5. QuickTime Movie 'qmetro' Toggle Module | 71 |
| Figure A.1.f.6. QuickTime Movie #1 'qmetro' Toggle Patcher | 71 |
| Figure A.1.f.7. QuickTime Movie #2 'qmetro' Toggle Patcher | 71 |
| Figure A.1.f.8. QuickTime Movie #4 'qmetro' Toggle Patcher | 72 |
| Figure A.1.f.9. QuickTime Movie #5 'qmetro' Toggle Patcher | 72 |
| Figure A.1.f.10. QuickTime Movie #6 'qmetro' Toggle Patcher | 72 |
| Figure A.1.f.11. QuickTime Frame Rate Multiplier Control Module | 73 |
| Figure A.1.f.12. QuickTime Movie #1 Frame Rate Multiplier Control Patcher | 73 |
| Figure A.1.f.13. QuickTime Movie #2 Frame Rate Multiplier Control Patcher | 73 |
| Figure A.1.f.14. Performance Control Window Comment Field Module | 73 |
| Figure A.1.f.15. QuickTime Movie Selection Module | 74 |
| Figure A.1.f.16. QuickTime Movie #1 Selection Patcher | 74 |
| Figure A.1.f.17. QuickTime Movie #2 Selection Patcher | 74 |
| Figure A.1.f.18. QuickTime Movie 'srcrect' Pixel Jitter Toggle Module | 75 |
| Figure A.1.f.19. QuickTime Movie Fade Control Module | 75 |
| Figure A.1.f.20. QuickTime Movie Main Mixer Fade Control Patcher | 75 |
| Figure A.1.f.21. QuickTime Movie Running Movie Fade Control Patcher | 76 |
| Figure A.1.f.22. QuickTime Movie Heartbeat Movie Mixer Fade Control Patcher | 76 |
| Figure A.1.f.23. QuickTime Movie Heartbeat Movie Fade Control Patcher | 77 |
| Figure A.1.f.24. LCD Display Fade Control Patcher | 77 |
| Figure A.1.f.25. Feet Accelerometer Section Distance Counter Reset Module | 77 |
| Figure A.1.f.26. Miscellaneous QuickTime Movie Control Module | 78 |

| Figure A.1.f.27. Master Video Control Switch Module | 78 |
|---|----|
| Figure A.1.g.1. Wiimote Master Control Patch Window | 78 |
| Figure A.1.g.2. All-Mute Wiimote Master Control Module | 79 |
| Figure A.1.g.3. Exposition Fade-Out Wiimote Master Control Module | 79 |
| Figure A.1.g.4. Final Piano Chord Wiimote Master Control Module | 80 |
| Figure A.1.g.5. Butte Pan Video Fade-In Patcher | 80 |
| Figure A.1.h.1. Wiimote 1 Control Patch Window, overview of Window layout | 81 |
| Figure A.1.h.2. Wiimote 1 Heart Rate Monitor Exposition Control Module | 81 |
| Figure A.1.h.3. Wiimote 1 Feet Exposition Control Module | 82 |
| Figure A.1.h.4. Environment Sound Select Patcher, in Feet Exposition | 82 |
| Figure A.1.h.5. Feet Sound Mute Patcher, in Feet Exposition | 83 |
| Figure A.1.h.6. Wiimote 1 Development Section Control Module, part 1 | 83 |
| Figure A.1.h.7. Wiimote 1 Development Section Control Module, part 2 | 84 |
| Figure A.1.h.8. Wiimote 1 Development Section Control Module, part 3 | 84 |
| Figure A.1.h.9. String Mute Patcher | 84 |
| Figure A.1.h.10. Panning of Trombones Control Patcher | 85 |
| Figure A.1.h.11. String Harmony Pitch Selection Patcher | 85 |
| Figure A.1.h.12. Wiimote 1 Development/Climax Section Control Module | 86 |
| Figure A.1.i.1. Wiimote 2 Control Patch Window, overview of Window layout | 86 |
| Figure A.1.i.2. Wiimote 2 Heart Rate Monitor Exposition Control Module | 87 |
| Figure A.1.i.3. Filter Control Presets Patcher | 87 |
| Figure A.1.i.4. Filter Preset Selection Patcher | 88 |
| Figure A.1.i.5. Interpolation Between Presets Patcher | 88 |
| Figure A.1.i.6. Time Constant Parameter Interpolation Patcher | 89 |
| Figure A.1.i.7. Side Level Parameter Interpolation Patcher | 89 |
| Figure A.1.i.8. Bandwidth Parameter Interpolation Patcher | 90 |
| Figure A.1.i.9. Wiimote 2 Feet Exposition Control Module | 90 |
| Figure A.1.i.10. Children Audio File Selection Patcher | 91 |
| Figure A.1.i.11. Road Ambience Sound Playback Rate Interpolation Patcher | 91 |

| Figure A.1.i.12. Wiimote 2 Development Section Control Module | 92 |
|--|-----|
| Figure A.1.i.13. Panning of Trombones Control Patcher | 92 |
| Figure A.1.i.14. Wiimote 2 Development/Climax Section Control Module, part 1 | 93 |
| Figure A.1.i.15. Wiimote 2 Development/Climax Section Control Module, part 2 | 93 |
| Figure A.1.i.16. Trumpet Time Index Selection Patcher | 94 |
| Figure A.1.i.17. Trumpet Time Index Interpolation Patcher | 94 |
| Figure A.1.i.18. Breath Rate Calculator Selection Patcher | 95 |
| Figure A.1.i.19. Breath Rate Interpolation Patcher | 95 |
| Figure A.2.1. VPT_4.1b5_RunningExpressions.maxpat Main Patch Window | 96 |
| Figure A.2.2. VPT Main Patch Window, in Patcher mode, part 1 | 97 |
| Figure A.2.3. VPT Main Patch Window, in Patcher mode, part 2 | 98 |
| Figure A.2.4. VPT Keyboard Shortcuts | 98 |
| Figure A.2.5. Main 'jit.window' and 'jit.gl.render' Control Patcher, part 1 | 99 |
| Figure A.2.6. Main 'jit.window' and 'jit.gl.render' Control Patcher, part 2 | 100 |
| Figure A.2.7. Custom Coordinates Control Patcher | 101 |
| Figure A.2.8. Custom Coordinates Input Patcher | 101 |
| Figure A.2.9. Conditional Statement Custom Coordinates Patcher | 102 |
| Figure A.2.a.1. Videoplane Running, in Presentation mode | 102 |
| Figure A.2.a.2. Videoplane Running Patch Window, overview of Window layout | 103 |
| Figure A.2.a.3. Videoplane Position Module | 103 |
| Figure A.2.a.4. Videoplane Color Swatch Module | 104 |
| Figure A.2.a.5. Videoplane Color Masks Patcher | 104 |
| Figure A.2.a.6. Videoplane 'jit.gl.render' Control Module | 105 |
| Figure A.2.a.7. Videoplane Positioning Control Module | 105 |
| Figure A.2.a.8. Videoplane Movie Masks Module | 106 |
| Figure A.2.b.1. Videoplane Heart Rate, in Presentation mode | 106 |
| Figure A.2.b.2. Videoplane Heart Rate, Position Control Module | 106 |
| Figure A.2.b.3. Videoplane Heart Rate, 'jit.gl.render' Control Module | 107 |
| Figure A.2.b.4. Videoplane 3D Positioning Control Module | 108 |

| Figure A.2.b.5. Videoplane Custom Corner Positioning Control Module | 108 |
|--|-----|
| Figure A.2.c.1. Videoplane LCD, in Presentation mode | 109 |
| Figure A.2.d.1. Preset Module, in Presentation mode | 109 |
| Figure A.2.d.2. Preset Module, in Patcher mode, part 1 | 110 |
| Figure A.2.d.3. Preset Module, in Patcher mode, part 2 | 111 |
| Figure A.2.d.4. Preset Module Controls Patcher | 112 |
| Figure A.2.d.5. Preset Module Recall Patcher | 113 |
| Figure A.2.d.6. Preset Module Data Confirmation Patcher | 113 |
| Figure A.2.e.1. Movie Source Running Patch Window, in Presentation mode | 114 |
| Figure A.2.e.2. Movie Source Running Patch Window, in Patcher mode | 114 |
| Figure A.2.e.3. Movie Source Select Module | 115 |
| Figure A.2.e.4. Movie Source External Select Control Module | 115 |
| Figure A.2.e.5. Movie Source Video Control Variables Module | 116 |
| Figure A.2.e.6. Movie Source Variables Assignment Patcher | 116 |
| Figure A.2.e.7. Movie Source Video Position Interpolation Calculator Patcher | 117 |
| Figure A.2.e.8. Movie Source Position Interpolation Timer Patcher | 118 |
| Figure A.2.e.9. Movie Control Module | 118 |
| Figure A.2.e.10. Interpolation for 'srcrect' X-Axis Jitter Patcher | 119 |
| Figure A.2.e.11. Interpolation for 'srcrect' Y-Axis Jitter Patcher | 120 |
| Figure A.2.f.1. Movie Source Running #2 Patch Window, in Patcher mode | 121 |
| Figure A.2.f.2. Wiimote Controls Panning Video Patcher | 122 |
| Figure A.2.f.3. Wiimote 1 Controls Kyma 8-channel Panning Patcher | 122 |
| Figure A.2.g.1. Movie Source Heart Rate LCD Display, in Patcher mode | 123 |
| Figure A.2.h.1. Movie Source Heartbeat Patch Window, in Presentation Mode | 124 |
| Figure A.2.h.2. Movie Source Heartbeat Patch Window, in Patcher Mode | 124 |
| Figure A.2.h.3. Movie Select Heartbeat Module | 125 |
| Figure A.2.h.4. Movie Heartbeat Video Control Variables Module | 125 |
| Figure A.2.h.5. Movie Control Heartbeat Module | 125 |
| Figure A.2.i.1. Movie Source LCD Patch Window | 126 |

| Figure A.2.j.1. Cue List Mixer Patch Window, in Presentation mode | 126 |
|---|-----|
| Figure A.2.j.2. Cue List Mixer Patch Window, in Patcher mode | 127 |
| Figure A.2.k.1. Heartbeat Movie Mixer Patch Window | 127 |
| Figure A.2.1.1. Running Movie Mixer Patch Window | 128 |
| Figure A.4.1. Hardware icons | 132 |
| Figure A.4.2. Connection Standards and Protocols icons | 133 |
| Figure A.4.3. Software icons | 134 |

INTRODUCTION

Running Expressions is a fusion of my two passions, electronic music and running. The result, a live electronic performance work, not only challenged my technical and compositional abilities, but also kindled my interests in human performance within electronic music. Running acted as the inspirational seed for both the music and the musical journey, and by taking biosignal, or physiological, information from the physical action of running, I facilitated the body in the creation and the control of music. Running then also served as a performance and a functional control over musical parameters.

Not only did I choose the human body as a way to generate data streams for the creation of music, but I necessitated the human performer inside an electronic work. By making the music rely on physiological data, the human became integral to the creation of the music. The music cannot exist without the human's input, and by so doing, I inject the human back into electronic music. I chose this dependent relationship for two reasons.

First, bringing the human performer back into electronic music helps shift electronic music closer to the music traditions of our past. Throughout the history of man, music has been created through the transference of acoustic energy enacted by humans. There is a direct relationship between sound and musical action. Because the energy for live electronic music is created through transductions recorded as digital data (0s and 1s), there is not always a direct relationship between sound and musical action. This indirect correlation between sound and action should not mean that the human performer's presence is lost inside the technology. For in the performance hall, I firmly believe there are benefits to having a human performance of electronic music, and can help engage the audience– even if these particular performer benefits stem directly from the perceptions of acoustic music and concert traditions.

Second, there are fewer works for live electronic music utilizing alternative controllers than electro-acoustic and fixed-media compositions.¹ Ever since the first musique concrète concerts of the 1950s and the introduction of computer music in the late 1950s, a tradition has evolved for fixed-media compositions and acoustic music with electronic accompaniment. Composers in the last sixty years have written fixed-media works, acousmatic music, and works for acoustic instruments with live electronics, but have largely ignored the genre of electronic music for real-time performance using recent technologies, due in part to the limitations of computer processing power. I chose to write a real-time electronic performance using alternative controllers because I felt, and still feel, that it is important to engage electronic works involving human performers, while, at the same time, to explore and perhaps help develop musical traditions for live electronic music.

Running Expressions uses three different alternative controllers– a heart-rate monitor, two Nintendo Wiimotes, and two dual-axis accelerometers. These three controllers are mapped to control musical parameters and to trigger sound events in real time. The work demands a human performer because the controllers require physiological data and motion for actualization. I explored the links of human motion and physiological data to sound and musical performance throughout the compositional process this past year.

Running Expressions also implements various software components and communication protocols, learned during my studies at the University of Oregon. In this documentation I will give a brief overview of the signal flow of all hardware and software components used in *Running Expressions*. Next, I will explain each component in detail, beginning with the various hardware components, followed by an in-depth review of each software component. Lastly, I will discuss the compositional and performance structure. The main topics (signal flow, hardware, software, composition) cohesively detail the development and the execution of *Running Expressions*. Detailed figures, including explanations, will appear throughout.

¹ For example, examining the works realized at CCRMA 1968–1992, there are 135 fixed-media compositions, 66 electro-acoustic compositions (acoustic instruments with tape), and 22 live-electronic works. These 22 works included compositions which incorporate either some type of electronic instrument or live-electronic manipulation of acoustically generated sounds. Of these twenty-two works, only seven compositions were written solely for live-electronics. The first of these seven compositions did not appear until 1988.

PART I. UNDERLYING ARCHITECTURE (SIGNAL FLOW) ²

1. Musical Hardware Connections

Running Expressions was written for one Polar Heart-Rate Monitor, two Nintendo Wiimote controllers, and two ADXL322 Dual-Axis accelerometers. The Heart-Rate Monitor attaches to the performer's chest, the Nintendo Wiis to the wrists, and each accelerometer to one leg, just below the knee. The Polar Heart-Rate Monitor sends information via a magnetic field to a Polar Heart Rate Monitor Interface, which sends its data to the computer via a standard USB cable. The Nintendo Wiimotes communicate to the computer via Bluetooth, and the dual-axis accelerometers via a high frequency radio signal using JeeNode Tx/Rx microcontrollers.



Figure 1.1. Hardware Connections Flowchart

 $^{^{2}}$ For a complete legend of all graphic icons used in the signal flow diagrams, please see the Appendix. Figures A.4.1 – A.4.3.

2. Software Connections

Using the computer, I poll each data stream with three different programs. Processing polls the heart-rate monitor interface for heart rate information, OSCulator polls the Nintendo Wiis for button and accelerometer information, and Max/MSP/Jitter polls the JeeNode Rx USB bub for accelerometer data of the X and Y axes. All sounds are generated by Kyma, but controlled in real time with MIDI and OSC messages sent from Max/MSP/Jitter. In this way, Max/MSP/Jitter serves as the master control software, and Kyma as the sound synthesis software/hardware engine. All other software components serve to bridge Kyma and Max/MSP/Jitter together, functioning as either direct communication links (PacaConnect, OSCulator) or data routers to/from Max/MSP and Kyma (OSCulator, Processing).



Figure 2.1. Software Connections Flowchart, includes connections to external devices.

3. Video Connections

Max/MSP/Jitter also controls the playback of video. Max/MSP/Jitter projects four different video planes at any given time, displaying video and LCD information in a 3D projection environment. Kyma sends a total of fourteen MIDI note messages to Max/MSP/Jitter to trigger the various video changes throughout the piece.



Figure 3.1. Video Connections Flowchart. Compiled Kyma Timeline on Paca(rana) sends MIDI messages via PacaConnect that serve as video controls within Max/MSP/Jitter.

PART II. HARDWARE ³

To actuate the work, "*Running Expressions*" uses several pieces of hardware. Each hardware device discussed below was selected after research in the fields of wireless network communication, physical sensors, and data protocols. There were different reasons for selecting each device, and I weigh the positive and negatives of each decision.

4. T-31 Coded[™] Polar Heart Rate Monitor Transmitter (HRM)

After failed attempts to find a working a solution using the ANT+ wireless protocol with Garmin heart rate monitor and foot pod products, I turned to the largest and oldest manufacturer of heart rate monitors, Polar. The T-31 Coded[™] Heart Rate Transmitter measures the electrocardiogram (ECG), which is the electrical signal produced by a heart in motion. Two electrodes must be wet and attached to the front part of the chest in order to transmit any signal, and the T-31 HRM uses a magnetic field to transmit data. I chose this particular heart rate monitor because I found a compatible computer interface. The transmitter and interface led me to a simple and stable solution after months of coding problems with the ANT+ protocol. The limitation of the HRM is the susceptibility to other electromagnetic signals. While the T-31 uses a Polar-coded signal in order to minimize interference, the physical range of the device must be limited in order to ensure a stable connection. This limited range was ultimately determined by the heart rate monitor interface.

5. Polar Heart Rate Monitor Interface (HRMI)

SparkFun, an online retailer of personal electronic projects, distributes an interface for the Polar T-31 Transmitter. Designed by DanJulioDesigns⁴, the Polar Heart Rate Monitor Interface converts ECG signals sent by the Polar Heart Rate Monitor into ASCII numbers (0-255). These ASCII numbers are separated by spaces, terminate with a carriage return, and sent serially, via

³ Pictures of representative hardware icons may be found in the Appendix. Figure A.4.1.

⁴ Dan Julio Designs, "Sparkfun HRMI," <u>http://danjuliodesigns.com/sparkfun/sparkfun.html</u> (accessed April 21, 2011).

USB, to the host computer. I use a Processing sketch to send commands to the HRMI and receive encoded heart rate information from the HRMI (Chapter 9). The main limitation of the HRMI device is the physical range of signal transfer between the HRM and the HRMI. Distances cannot exceed 80cm to 100cm (31.5 to 39.3 inches) before the signal begins to drop. Dropping of the ECG signal causes irregularity in the heartbeat information received by the HRMI, which I found impacts the composition control mappings, and due to these mappings, can cause audible changes in the music.

6. Nintendo Wiimotes

The Nintendo Wii Remote (or Wiimote) is a wireless game controller that features embedded accelerometers, gyroscope, infrared light, and button controls. The controller sends data via Bluetooth. Bluetooth is a wireless technology standard for exchanging data over short distances. I accessed the Wiimote data through OSCulator (Chapter 10), which incorporates a Wiimote Bluetooth setup panel as part of its software. Because of the Wiimote's wireless capabilities, amount of controls, ease of setup, and stable connection, I selected the Wiimote to serve as the composition's master controller, capable of triggering sound events, music section changes, and controlling musical parameters in real time. For *Running Expressions*, I only utilized the Wiimote's accelerometer and button controls. I did not use the infrared light, the Wii Motion Plus (a Tuning fork gyroscope that accents the accelerometer data), or any other Wiimote accessories, like the Wii Nunchuk.

7. ADXL322 Dual-Axis Accelerometer

Due to complications with the ANT+ wireless protocol, I was also unable to use the Garmin Foot Pod, a device for tracking a runner's cadence, speed and distance. Without this set of information, I would have been unable to capture and transfer the physical act of running into performance controls. Therefore, I needed a solution to track the running and walking motion of legs.

I found a solution using the ADXL322 Dual-Axis Accelerometer. The accelerometer measures dynamic acceleration resulting from motion, shock, or vibration and outputs voltage

signals. With an accelerometer, I would be able to generate data based upon the motion of the legs. In preparation of the final performance, I sought after my ideal performance situation– a wireless connection to the accelerometers attached to the legs. Cables connected to the legs would look bulky and potentially create an unwanted hazard to the performer and equipment. In order to minimize the hazards, I sought out another unique wireless data transfer method, radio signals.

8. JeeNode wireless Tx/Rx

JeeNode is a small wireless microcontroller board that communicates through a RFM12B radio module at either 433, 868, or 915 MHz. The JeeNode Tx/Rx boards served as the wireless solution to connecting the accelerometers to the body (one accelerometer is attached to each leg, just below the knee), and freed the performer of attached cables during performance. The JeeNode Tx (transmitter) collects information off the analog pins of the dual-axis accelerometer, before sending the information over a specific radio frequency.⁵ The JeeNode Rx (receiver) converts any received JeeNode Tx message into an 8-bit serial packet, which is then sent over a universal serial bus (USB) into the host computer⁶. The information is collected by Max/MSP/ Jitter for data mapping (Chapter 12).

I discovered both the ADXL322 Dual-Axis Accelerometer and the JeeNode Tx/Rx boards after taking a University of Oregon workshop with Brown Ph.D. candidate in electronic music, Kevin Patton. Because the hardware now belongs to the Intermedia Music Technology department, I was able to borrow the equipment for use in *Running Expressions*.⁷

The main limitation to the JeeNode Tx/Rx are the fluctuations in the incoming data streams. While I will discuss the data in further detail in Max/MSP/Jitter (Chapter 12), some of the data fluctuations should be noted here. Leaving the device on and alone for two hours connected to the computer resulted in forty seven spike occurrences in the incoming data stream

⁵ The JeeNodes used in *Running Expressions* implemented a 915MHz radio frequency.

⁶ The information sent to the computer travels at a 38400 baud rate.

⁷ Included in the DVD is my own generic Max/MSP template patch for use with the ADXL322 dual-axis accelerometers and JeeNode Tx/Rx devices.

(Figure 8.1).⁸ Further investigations revealed discrepancies in data between running and walking. The result were inconsistent triggers while running, which became apparent while mapping the accelerometer triggers. One example was the choppiness of video playback.

| stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 |
|-----------------------|-----------------------|-----------------------|-----------------------|
| stepRight: 118.000000 | stepRight: 119.000000 | stepRight: 87.000000 | stepRight: 119.000000 |
| stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 54.000000 | stepRight: 118.000000 |
| stepRight: 118.000000 | stepRight: 123.000000 | stepRight: 118.000000 | stepRight: 32.000000 |
| stepRight: 117.000000 | stepRight: 117.000000 | stepRight: 52.000000 | stepRight: 118.000000 |
| stepRight: 118.000000 | stepRight: 121.000000 | stepRight: 116.000000 | stepRight: 118.000000 |
| stepRight: 118.000000 | stepRight: 117.000000 | stepRight: 190.000000 | stepRight: 118.000000 |
| stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 54.000000 | stepRight: 118.000000 |
| stepRight: 118.000000 | stepRight: 119.000000 | stepRight: 118.000000 | stepRight: 118.000000 |
| stepRight: 118.000000 | stepRight: 120.000000 | stepRight: 117.000000 | stepRight: 118.000000 |
| stepRight: 118.000000 | stepRight: 246.000000 | stepRight: 118.000000 | stepRight: 118.000000 |
| stepRight: 117.000000 | stepRight: 62.000000 | stepRight: 81.000000 | stepRight: 118.000000 |
| stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 |
| stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 |
| stepRight: 117.000000 | stepRight: 102.000000 | stepRight: 119.000000 | stepRight: 118.000000 |
| stepRight: 117.000000 | stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 |
| stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 119.000000 |
| stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 |
| stepRight: 118.000000 | stepRight: 119.000000 | stepRight: 114.000000 | stepRight: 200.000000 |
| stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 |
| stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 114.000000 |
| stepRight: 118.000000 | stepRight: 126.000000 | stepRight: 118.000000 | stepRight: 116.000000 |
| stepRight: 118.000000 | stepRight: 114.000000 | stepRight: 119.000000 | stepRight: 117.000000 |
| stepRight: 118.000000 | stepRight: 126.000000 | stepRight: 118.000000 | stepRight: 118.000000 |
| stepRight: 119.000000 | stepRight: 116.000000 | stepRight: 119.000000 | stepRight: 118.000000 |
| stepRight: 117.000000 | stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 |
| stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 119.000000 |
| stepRight: 126.000000 | stepRight: 118.000000 | stepRight: 119.000000 | stepRight: 119.000000 |
| stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 117.000000 |
| stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 119.000000 |
| stepRight: 102.000000 | stepRight: 115.000000 | stepRight: 251.000000 | stepRight: 118.000000 |
| stepRight: 56.000000 | stepRight: 118.000000 | stepRight: 118.000000 | stepRight: 119.000000 |
| stepRight: 119.000000 | stepRight: 55.000000 | stepRight: 126.000000 | stepRight: 119.000000 |
| | | | |

Figure 8.1. Accelerometer Spike Fluctuations. JeeNode data packets received inside Max/MSP while the accelerometer was attached to the right leg. The accelerometer is in resting position as I am seated with no quick movements. Data acquired on February 9, 2011.

⁸ The normal data range of the JeeNode packets was between 0-255 with normal resting values incoming between a range of ten, usually 130-140, or 117-126. The forty-seven spikes in value were recorded for incoming values exceeding 200, a value spike of 60+ in value. Fluctuations less than ten were not recorded. Figure 8.1.

PART III. SOFTWARE 9

9. Processing

Processing acts as the master program that controls the HRMI microprocessor. Processing sends commands to the HRMI every 100ms to retrieve heart rate information stored in the microprocessor buffer. These data packets are sent in the form of ASCII values, and are comprised of a status byte followed by heart rate information and a carriage return. Once Processing receives any data packet, the program displays the current heart rate value in a small compiler window and simultaneously sends the heart rate value over to Max/MSP/Jitter. The cross-software communication is achieved through an external java library called MaxLink.¹⁰

During the compositional process, I discovered that my Processing sketch placed an inordinate load on the CPU of the computer. Normally, Processing sent commands to the HRMI upon every draw() function, a repetition occurring at the speed of the computer's processor. Because I didn't need a continuous update of the heart rate information, I placed a speed limit on the data transfer in order to maximize the computer's performance. After imposing a 100ms interval upon Processing's information request, the CPU load on the computer went from approximately 90% to 5% in computational load (Figures 9.1. and 9.2).

⁹ Pictures of representative hardware icons may be found in the Appendix. Figure A.4.3.

¹⁰ MaxLink, <u>http://jklabs.net/maxlink/</u> (accessed April 21, 2011).

| | | Activity Monitor | | | 1111 | |
|---------------------------|--|------------------|--------------|---------|----------|----------------|
| | () (| M | ly Processes | \$ | Q.+ F | |
| ss Inspect Sample Process | | | Show | | Filter | |
| rocess Name | | ▲ User | % CPU | Threads | Real Mem | Kind |
| 4~ | Activity Monitor | jpbellona | 1.2 | 2 | 26.3 MB | Intel (64 bit) |
| 0 | Address Book | jpbellona | 0.0 | 3 | 27.0 MB | Intel (64 bit) |
| | AirPort Base Station Agent | jpbellona | 0.0 | 3 | 5.8 MB | Intel (64 bit) |
| | AppleSpell.service | jpbellona | 0.0 | 2 | 9.2 MB | Intel (64 bit) |
| 8 | Cyberduck | jpbellona | 0.1 | 24 | 159.4 MB | Intel |
| | Dock | jpbellona | 0.0 | 3 | 17.5 MB | Intel (64 bit) |
| 4 | Finder | jpbellona | 0.0 | 8 | 67.5 MB | Intel (64 bit) |
| 3 | Firefox | jpbellona | 4.9 | 28 | 195.7 MB | Intel |
| | fontd | jpbellona | 0.0 | 3 | 4.9 MB | Intel (64 bit) |
| | fontworker | jpbellona | 0.0 | 3 | 3.2 MB | Intel (64 bit) |
| | heartbeatTemplate3optimize | jpbellona | 93.3 | 27 | 38.2 MB | Intel |
| | iTunesHelper | jpbellona | 0.0 | 3 | 2.9 MB | Intel (64 bit) |
| | launchd | jpbellona | 0.6 | 2 | 1.1 MB | Intel (64 bit) |
| | loginwindow | jpbellona | 0.0 | 2 | 8.2 MB | Intel (64 bit) |
| 6 | MaxMSP | jpbellona | 6.4 | 70 | 134.6 MB | Intel |
| | mdworker | jpbellona | 0.0 | 3 | 15.0 MB | Intel (64 bit) |
| | MIDIServer | jpbellona | 0.0 | 5 | 2.4 MB | Intel (64 bit) |
| | pboard | jpbellona | 0.0 | 1 | 856 KB | Intel (64 bit) |
| 44 | Preview | jpbellona | 0.0 | 2 | 43.3 MB | Intel (64 bit) |
| B | Processing | jpbellona | 4.3 | 35 | 94.1 MB | Intel |
| | Quick Look Helper | jpbellona | 0.0 | 6 | 6.7 MB | Intel (64 bit) |

Figure 9.1. Processing optimization (before 100ms interval added)

| | Act | ivity Monitor | - 12 | | | |
|----------------------------|---|--|--|---|---|---|
| <i>i</i> 😕 | | My Pro | ocesses | \$ | Q+ F | |
| ess Inspect Sample Process | | Show | | | Filter | |
| s Name | | ser | % CPU | Threads | Real Mem | Kind |
| Activity Monitor | jp | bellona | 1.2 | 2 | 26.4 MB | Intel (64 bit) |
| Address Book | jp | bellona | 0.0 | 2 | 27.0 MB | Intel (64 bit) |
| AirPort Base Station Agent | jp | bellona | 0.0 | 3 | 5.8 MB | Intel (64 bit) |
| AppleSpell.service | jr | bellona | 0.0 | 2 | 9.2 MB | Intel (64 bit) |
| Cyberduck | jp | bellona | 0.1 | 24 | 159.4 MB | Intel |
| Dock | jr | bellona | 0.0 | 4 | 17.7 MB | Intel (64 bit) |
| Finder | jr | bellona | 0.0 | 16 | 77.6 MB | Intel (64 bit) |
| Firefox | jr | bellona | 2.1 | 24 | 192.9 MB | Intel |
| fontd | jr | bellona | 0.0 | 4 | 5.0 MB | Intel (64 bit) |
| heartbeatTemplate3optimize | jp | bellona | 4.3 | 27 | 42.3 MB | Intel |
| iTunesHelper | jr | bellona | 0.0 | 3 | 2.9 MB | Intel (64 bit) |
| launchd | jr | bellona | 0.2 | 2 | 1.1 MB | Intel (64 bit) |
| loginwindow | jr | bellona | 0.0 | 2 | 8.2 MB | Intel (64 bit) |
| MaxMSP | jr | bellona | 7.9 | 70 | 134.6 MB | Intel |
| mdworker | jr | bellona | 0.0 | 3 | 13.7 MB | Intel (64 bit) |
| MIDIServer | jr | bellona | 0.0 | 5 | 2.4 MB | Intel (64 bit) |
| pboard | jp | bellona | 0.0 | 1 | 856 KB | Intel (64 bit) |
| Preview | jr | bellona | 0.0 | 2 | 43.3 MB | Intel (64 bit) |
| Processing | jp | bellona | 3.8 | 34 | 97.0 MB | Intel |
| QuickTime Player | jp | bellona | 0.0 | 13 | 55.7 MB | Intel (64 bit) |
| | Sample Process Sample Process so Name Activity Monitor Address Book AirPort Base Station Agent AdpleSpell.service Cyberduck Dock Finder Firefox fontd heartbeatTemplate3optimize ITunesHelper launchd loginwindow MaxMSP mdworker MIDIServer pboard Preview Processing QuickTime Player | Activity Monitor Jp Address Book Jp Address Book Jp Address Book Jp AirPort Base Station Agent Jp AppleSpell.service Jp Cyberduck Jp Dock Jp Finder Jp Finder Jp Finder Jp fortd Jp heartbeatTemplate3optimize Jp ItrunesHelper Jp Iaunchd Jp Ioginwindow Jp MaxMSP Jp Mdworker Jp MIDIServer Jp Processing Jp QuickTime Player Jp | Activity Monitor My Pro | Activity Monitor My Processes Image: Sample Process My Processes nspect Sample Process Show ss Name User % CPU Activity Monitor jpbellona 1.2 Address Book jpbellona 0.0 AirPort Base Station Agent jpbellona 0.0 AppleSpell.service jpbellona 0.0 Cyberduck jpbellona 0.0 Finder jpbellona 0.0 Firefox jpbellona 0.0 fortd jpbellona 0.0 heartbeatTemplate3optimize jpbellona 0.0 launchd jpbellona 0.0 Igipbellona 0.0 0.0 MaxMSP jpbellona 0.0 MIDIServer jpbellona 0.0 Proview jpbellona 0.0 Processing jpbellona 0.0 QuickTime Player jpbellona 3.8 | Activity Monitor My Processes Show ss Name My Processes Activity Monitor jpbellona 1.2 2 Address Book jpbellona 0.0 2 AirPort Base Station Agent jpbellona 0.0 2 AppleSpell.service jpbellona 0.0 2 Cyberduck jpbellona 0.0 2 Dock jpbellona 0.0 4 Finder jpbellona 0.0 4 Firefox jpbellona 0.0 4 fontd jpbellona 0.0 3 launchd jpbellona 0.0 3 launchd jpbellona 0.0 2 MiDServer jpbellona 0.0 3 MIDIServer jpbellona 0.0 3 pboard jpbellona 0.0 3 Processing jpbellona 0.0 1 Processing jpbellona 0.0 13 <td>Activity Monitor My Processes My Processes Show Fill Activity Monitor jpbellona 1.2 2 2.6.4 MB Address Book jpbellona 0.0 2 27.0 MB Address Book jpbellona 0.0 2 9.2 MB Address Estation Agent jpbellona 0.0 2 9.2 MB Address Estation Agent jpbellona 0.0 2 9.2 MB Address Estation Agent jpbellona 0.0 2 2 ME</td> | Activity Monitor My Processes My Processes Show Fill Activity Monitor jpbellona 1.2 2 2.6.4 MB Address Book jpbellona 0.0 2 27.0 MB Address Book jpbellona 0.0 2 9.2 MB Address Estation Agent jpbellona 0.0 2 9.2 MB Address Estation Agent jpbellona 0.0 2 9.2 MB Address Estation Agent jpbellona 0.0 2 2 ME |

Figure 9.2. Processing optimization (after 100ms interval added)



Figure 9.3. Additional Processing code, limits time between data requests

10. OSCulator

OSCulator is a software that connects many different devices and software together utilizing the Open Sound Control communication protocol. Open Sound Control (OSC) is a stable, 32 bit protocol used for interconnecting hardware controller devices to the computer, as well as software on one or more computers.¹¹ The protocol utilizes UDP/IP (User Datagram Protocol/Internet Protocol) packets, which are user-defined packets of information sent to/from computers and devices on the same local network. Because OSC offers reliable, programmable messages served on a local network, I chose OSC the communication protocol between the Wiimotes, Max/MSP, and Kyma. The OSCulator software displayed my individualized message packets, which eased the compositional process. OSCulator also provided a stable location where I could connect the Wiimotes to the computer and confirm data entry quickly and efficiently.

In *Running Expressions*, OSCulator serves three functions. First the software retrieves Wiimote data and translates the information into OSC messages. Second, OSCulator sends the translated Wiimote OSC messages to Max/MSP/Jitter. Third, OSCulator routes OSC data packets received from Max/MSP across to Kyma.

¹¹ Open Sound Control, <u>http://opensoundcontrol.org/spec-1_0</u> (accessed February 03, 2011).

In all cases, all messages received from Max/MSP were routed to Kyma as continuous controllers. Fifty-nine CC (continuous controller) connections were routed from Max/MSP to Kyma. All Wiimote buttons and motions were sent to Max/MSP except one. Wiimote 1 button 2 was routed directly to Kyma because this button serves a single function, triggering WaitUntil objects in the Kyma Timeline. The button enables the performer to trigger the beginning of the next section of music, which frees the performer from adhering to a particular time schedule.

OSCulator also defined MIDI channels for messages sent to Kyma; however, the Continuous Controller and MIDI channel information sent to Kyma use Symbolic Sound's MIDI-over-OSC protocol, which is why the OSC protocol icon is shown in Figure 10.1, and not the MIDI protocol icon.¹²



Figure 10.1. OSCulator Signal Flowchart

¹² Synthtopia. "Kyma gets OpenSoundControl (OSC) Support." <u>http://www.synthtopia.com/content/2010/03/05/</u> kyma-gets-open-sound-control-osc-support/ (accessed April 19, 2011).

| 0 | 🔵 📄 kym | aHRMTester | _newv | ersion3.oscd | | \bigcirc | |
|----------|-----------------------|------------|-------|----------------|--------------|------------|--|
| | 8000 D | efault ‡ | | | E Con | | |
| Pause | OSC Input Port | Presets | | Quick Look | Parameters W | iimote | |
| M | essage 🔺 | Event Type | | Value | Cha | n. ب | |
| | /allMute/ | Kyma CC | ÷ | 8 | ‡ 1 | ÷ 🕥 | |
| | /beat/ | Kyma CC | ÷ | 28 | ‡ 1 | ÷ | |
| | /boneAmplitude/ | Kyma CC | ÷ | 71 | ÷ 2 | ÷ | |
| | /boneAmplitude2/ | Kyma CC | ÷ | 73 | ÷ 2 | ÷ | |
| | /boneHarmony/ | Kyma CC | ÷ | 78 | ÷ 2 | Ŧ | |
| | /boneMelodyTrigger/ | Kyma CC | ÷ | 4 | ÷ 2 | Ŧ | |
| | /bonePan_Angle/ | Kyma CC | Ŧ | 74 | ÷ 2 | Ŧ | |
| | /bonePan_Angle2/ | Kyma CC | Ŧ | 75 | ÷ 2 | - | |
| | /bonePan_Radius/ | Kyma CC | Ŧ | 76 | ÷ 2 | | |
| | /bonePan_Radiusz/ | Kyma CC | * | // | ÷ 2 | | |
| | /breathAmplitude2/ | Kyma CC | * | 80 | ÷ 2 | - 1 | |
| | /breathkamp/ | Kyma CC | * | 109 | ÷ 1 | | |
| | /ButtemovieAngle/ | Kyma CC | * | 115 | ÷ 1 | IU | |
| | /childTrigger/ | Kyma CC | * | 12 - Attack | ÷ 1 | 1 | |
| | /China rigger/ | Kyma CC | | 12 - Bandwidth | + 1 | T | |
| | /Contraxboost/ | Kyma CC | * | 120 | ÷ 4 | T | |
| | /Couldboost/ | Kyma CC | * | 120 | ÷ 1 | * | |
| | /crashringger/ | Kyma CC | * | 10 | ÷ 1 | * • | |
| | /environSelect/ | Kyma CC | * | 11 – Attack | + 3 | * • | |
| | /ExpoBandwidth/ | Kyma CC | * | 97 | ÷ 1 | T A | |
| | /ExposideValue/ | Kyma CC | | 96 | ÷ 1 | T A | |
| | /Exposition_EadeOut/ | Kyma CC | * | 92 | ÷ 1 | * * | |
| | | Kyma CC | * | 94 | ÷ 1 | * | |
| | sitionVolRSequencer/ | Kyma CC | | 93 | ÷ 1 | | |
| | /ExpositionVolumeL/ | Kyma CC | 4 | 90 | ± 1 | ÷. | |
| | /ExpositionVolumeR/ | Kyma CC | 4 | 91 | ± 1 | ÷ | |
| | /ExpoTimeConstant/ | Kyma CC | 4 | 95 | ÷ 1 | ÷ | |
| | /feet totalCount/ | Kyma CC | 4 | 86 | \$ 1 | ÷ | |
| | totalCount_asTrigger/ | Kyma CC | 4 | 87 | \$ 1 | ÷ | |
| | /foot_BPM/ | Kyma CC | \$ | 85 | \$ 1 | \$ | |
| | /heartBoost/ | Kyma CC | \$ | 111 | \$ 1 | \$ | |
| | /heartrate/ | - | \$ | - | ÷ - | \$ | |
| | 0 | Kyma CC | \$ | 57 | \$ 1 | ÷ | |
| | 0⊁0 | Kyma CC | \$ | 57 | \$ 4 | ÷ | |
| | /key/keyNumber/ | Kyma CC | \$ | 47 - Gate | \$ 1 | ÷ | |
| V | /key/keyVelocity/ | Kyma CC | \$ | 3 | \$ 1 | ÷ | |
| | /keyNote/ | Kyma CC | \$ | 2 | \$ 1 | ÷ | |
| | /melodyAmp/ | Kyma CC | \$ | 79 | ‡ 2 | \$ | |
| V | /muteFeetMaster/ | Kyma CC | \$ | 12 - Bandwidth | ÷ 3 | ÷ 🔺 | |
| | /muteHeart/ | Kyma CC | \$ | 29 | <u>‡ 1</u> | ÷ . | |
| 🖯 Runr | \varTheta Running | | | | | | |

Figure 10.2. OSCulator, part 1. OSC messages received from Max/MSP/Jitter, routed to Kyma.

| 0 |) 🕙 📑 kym | aHRMTester | _newv | ersion3.oscd | | \bigcirc | |
|-------------------|-------------------------|-------------|-------|------------------|------------|------------|--|
| | 8000 D | efault ‡ | | t see | 2000 | | |
| Pause | OSC Input Port F | Presets | | Quick Look | Parameters | Wiimote | |
| | Message 🔺 | Event Type | | Value | Ch | ian. ච | |
| | /melodyAmp/ | Kyma CC | ÷ | 79 | ‡ 2 | \$ | |
| | /muteFeetMaster/ | Kyma CC | ÷ | 12 – Bandwidth | ‡ 3 | ÷ | |
| | /muteHeart/ | Kyma CC | ÷ | 29 | \$ 1 | ÷ | |
| | /note1/ | Kyma CC | ŧ | 101 | \$ 1 | ÷ | |
| | /note2/ | Kyma CC | ÷ | 102 | \$ 1 | ÷ | |
| | /note3/ | Kyma CC | ÷ | 103 | \$ 1 | ÷ | |
| | /note4/ | Kyma CC | ÷ | 104 | \$ 1 | ÷ | |
| | /note5/ | Kyma CC | ÷ | 105 | \$ 1 | ÷ | |
| | /note6/ | Kyma CC | ÷ | 106 | \$ 1 | ÷ | |
| | /note7/ | Kyma CC | ÷ | 107 | \$ 1 | ÷ | |
| | /panningLine/ | Kyma CC | ÷ | 6 | \$ 3 | ÷ | |
| | /pianoEvery/ | Kyma CC | ÷ | 110 | \$ 1 | ÷ | |
| | /pianoFinal/ | Kyma CC | ÷ | 112 | \$ 1 | ÷ | |
| | /rate/ | Kyma CC | ÷ | 27 | \$ 1 | ÷ | |
| | /rateAorta/ | Kyma CC | ÷ | 30 | \$ 1 | ÷ | |
| | /selectSound_feetStart/ | - | ÷ | - | ÷ - | ÷ | |
| | /selectSound_leftFoot/ | Kyma CC | ÷ | 81 | \$ 1 | ÷ | |
| | ound_leftFootTrigger/ | Kyma CC | ÷ | 82 | \$ 1 | ÷ | |
| | /selectSound_rightFoot/ | Kyma CC | ÷ | 83 | \$ 1 | ÷ | |
| | nd_rightFootTrigger/ | Kyma CC | ÷ | 84 | \$ 1 | ÷ | |
| | /timeindex/ | Kyma CC | ÷ | 70 | ‡ 2 | ÷ | |
| | /timeindex2/ | Kyma CC | ÷ | 72 | ‡ 2 | ÷ | |
| | /tptPreset/ | Kyma CC | ÷ | 108 | \$ 1 | ÷ | |
| | /wii/1/accel/pry | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | 0: pitch | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | 1: roll | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ 🕕 | |
| | 2: yaw | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | 3: accel | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | ▼/wii/1/accel/xyz | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | 0: x | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | 1: y | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | 2: z | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | /wii/1/button/1 | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | /wii/1/button/2 | Kyma Ext | ÷ | WiiButtonA | \$ 1 | ÷ | |
| | /wii/1/button/A | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | /wii/1/button/B | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | /wii/1/button/Down | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | /wii/1/button/Home | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | /wii/1/button/Left | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ | |
| | /wii/1/button/Minus | OSC Routing | ÷ | 1-localhost:9000 | ÷ - | ÷ 🔺 | |
| | /wii/1/button/Plus | OSC Routing | \$ | 1-localhost:9000 | ÷ - | ÷ L. | |
| \varTheta Running | | | | | | | |

Figure 10.3. OSCulator, part 2. OSC messages received from Max/MSP/Jitter, routed to Kyma. Wiimote messages received, routed to Max/MSP/Jitter.

| 0 | 0 | 📄 kymaHRMTester | _newversion3.oscd | \bigcirc | | | | |
|---------------------|----------------------------|-------------------|--|--------------------|--|--|--|--|
| | 8000 | Default ‡ | a const | () () | | | | |
| Paus | e OSC Input Port | Presets | Quick Look | Parameters Wiimote | | | | |
| | Message | Event Type | Value | ہے Chan | | | | |
| | /wii/1/button/L | eft OSC Routing | ‡ 1-localhost:9000 | | | | | |
| | /wii/1/button/w | linus OSC Routing | ‡ 1-localhost:9000 | | | | | |
| | /wii/1/button/P | iaht OSC Routing | + 1-localhost:9000 | • • • | | | | |
| | /wii/1/button/k | n OSC Routing | 1-localhost:9000 1-localhost:9000 | • • • | | | | |
| | /wii/1/motion/a | nales OSC Routing | 1-localhost:9000 1-localhost:9000 | ¥ • • • | | | | |
| | 0: pitch | OSC Routing | ‡ 1-localhost:9000 | ÷ | | | | |
| | 1: roll | OSC Routing | 1-localhost:9000 1-localhost:9000 | 1 - 1 | | | | |
| | 2: vaw | OSC Routing | 1-localhost:9000 | ± | | | | |
| | ▼/wii/1/motion/v | elo OSC Routing | \$ 1-localhost:9000 | ÷ = | | | | |
| | 0: pitch veloc | ity OSC Routing | \$ 1-localhost:9000 | ÷ = ÷ | | | | |
| | 1: roll velocity | y OSC Routing | \$ 1-localhost:9000 | ÷ = ÷ | | | | |
| ☑ 🗌 | 2: yaw velocit | y OSC Routing | ‡ 1-localhost:9000 | ‡ = \$ | | | | |
| ✓ | ▼/wii/2/accel/pry | OSC Routing | ‡ 1-localhost:9000 | \$ = \$ | | | | |
| V | 0: pitch | OSC Routing | ‡ 1-localhost:9000 | ‡ = \$ | | | | |
| ✓ | 1: roll | OSC Routing | ‡ 1-localhost:9000 | ¢ = 0 | | | | |
| ⊻ | 2: yaw | OSC Routing | ‡ 1-localhost:9000 | ‡ – | | | | |
| ✓ | 3: accel | OSC Routing | ‡ 1-localhost:9000 | ¢ = | | | | |
| ☑ | ▼/wii/2/accel/xyz | OSC Routing | ‡ 1-localhost:9000 | ‡ = - \$ | | | | |
| ✓ | 0: x | OSC Routing | ‡ 1-localhost:9000 | ‡ = | | | | |
| | 1: y | OSC Routing | ‡ 1-localhost:9000 | ÷ = ÷ | | | | |
| | 2: z | OSC Routing | ‡ 1-localhost:9000 | \$ = - \$ | | | | |
| | /wii/2/button/1 | OSC Routing | ‡ 1-localhost:9000 | ‡ = \$ | | | | |
| | /wii/2/button/2 | OSC Routing | ‡ 1-localhost:9000 | ‡ = ‡ | | | | |
| | /wii/2/button/A | OSC Routing | ‡ 1-localhost:9000 | ‡ = | | | | |
| | /wii/2/button/B | OSC Routing | ‡ 1-localhost:9000 | ÷ = - ÷ | | | | |
| | /wii/2/button/D | own OSC Routing | \$ 1-localhost:9000 | ÷ = ÷ | | | | |
| | /wii/2/button/H | ome OSC Routing | \$ 1-localhost:9000 | • - • | | | | |
| | /wii/2/button/L | eft OSC Routing | ‡ 1-localhost:9000 | | | | | |
| | /wii/2/button/M | linus OSC Routing | ‡ 1-localhost:9000 | - | | | | |
| | /wii/2/button/P | lus OSC Routing | 1-localhost:9000 1 localhost:0000 | | | | | |
| | /wii/2/button/k | ight OSC Routing | 1-localhost:9000 1 localhost:9000 | | | | | |
| | /wii/2/button/0 | p OSC Routing | + 1-localhost:9000 | · · · · | | | | |
| | vii/2/motion/a 0: pitch | OSC Routing | 1-localhost:9000 1 localhost:0000 | · · · · | | | | |
| | 0: pitch | OSC Routing | + 1-localhost:9000 | | | | | |
| | 2: vaw | OSC Routing | 1-localbost:0000 | | | | | |
| | ∑. yaw | elo OSC Routing | + 1-localhost:9000 | | | | | |
| | 0: pitch veloc | ity OSC Routing | 1-localhost:9000 1-localhost:9000 | | | | | |
| | 1: roll velocity | V OSC Routing | 1-localhost:9000 | - | | | | |
| | 2: yaw velocit | v OSC Routing | ‡ 1-localhost:9000 | 1 - 1 V | | | | |
| 0.0 | unning | , esenouting | , | • • • | | | | |
| e Running | | | | | | | | |

Figure 10.4. OSCulator, part 3. Messages received from Wiimotes, routed to Max/MSP/Jitter.

11. PacaConnect

PacaConnect is an OSX "user agent" program for the Mac that provides an advanced connectivity solution for Symbolic Sound's Paca(rana) device.¹³ The PacaConnect allows MIDI messages to be received and sent between Max and the Paca(rana) by serving as a virtual MIDI patchbay. The software was inexpensive and took care of potential hardware problems as the PacaConnect only requires one RS45 connector (no MIDI interface). While Figure 11.1 shows the full connectivity of the software, *Running Expressions* only utilizes the virtual MIDI patchbay via the App-to-App connection inside the Mac computer.



Figure 11.1. PacaConnect Signal Flowchart

¹³ PacaConnect, <u>http://www.delora.com/delora_products/pacaconnect/pacaconnect.html</u> (accessed April 21, 2011).

12. Max/MSP/Jitter

Max/MSP/Jitter is a visual programming environment for music, audio, and media. I chose to use Max/MSP/Jitter because of its flexibility in handling multiple tasks simultaneously, its ability to communicate between devices and software, and its ability to manipulate numbers, strings, and matrices. While many functions and protocols can be handled within the Max/MSP/Jitter software, I used Max/MSP/Jitter for three distinct purposes. Max/MSP/Jitter collects and modifies data received from the heart rate monitor, Nintendo Wiimotes, and accelerometers, controls musical parameters inside the Kyma environment, and lastly controls the video projections.

12.a. Data Hub

Max/MSP/Jitter collects data from the three musical controllers (heart rate monitor, two Nintendo Wiimotes, and two dual-axis accelerometers). Because previous sections discuss these three devices communication links as well as their associated software applications, I will focus instead on the direct communication links to/from Max/MSP/Jitter.

12.a.i. Heart Rate Monitor from Processing

I used an external java library called MaxLink, which enables communication between Max/MSP with Processing, to transmit the heart rate information to Max/MSP/Jitter. Max/MSP/ Jitter received heart rate information as integers using the external max object "mxj jk.link" (Figure 12.a.1).¹⁴

¹⁴ For more information about MaxLink, please visit <u>http://jklabs.net/maxlink/</u>



Figure 12.a.1. MaxLink external object inside of *Running Expressions* Max/MSP/Jitter patch.

12.a.ii. Nintendo Wiimotes via OSC messages from OSCulator

Stated before, Open Sound Control (OSC) is a stable, 32-bit protocol used for interconnecting hardware controller devices to the computer, as well as software on one or more computers. Max/MSP/Jitter collects Nintendo Wiimote information via OSC messages sent from OSCulator. I used an external Max object "OSC-route" created at the Center for New Music and Audio Technologies (CNMAT) to sort the OSC messages received from OSCulator (Figure 12.a. 2).¹⁵

¹⁵ For more information about the CNMAT downloads, please visit <u>http://cnmat.berkeley.edu/downloads</u>



Figure 12.a.2. OSC-route external object inside *Running Expressions* Max/MSP/Jitter patch.

12.a.iii. JeeNode and Accelerometers from Serial Bus

Max/MSP/Jitter collects accelerometer data directly via the serial ports located on the computer. JeeNode Tx messages are sent as 8-bit serial packets at a 38400 baud rate over a universal serial bus. Max/MSP/Jitter receives these packets as separate pin read-outs with values between 0-255. The normal resting values of the incoming JeeNode packets were between a range of ten, usually 130-140, or 117-126.

I used the leg's motion as triggers, creating a bang on the upswing of each leg, as the motion produced a predictable, although not completely reliable, pattern of numbers. I used the 'past' object in Max to trigger the bangs. Based upon tracking the data with both walking and running motions, the peaks of data were more consistent with the upswing of the knee, not on the down step of the foot, where I encountered inconsistent double peaks per footstep (Fig. 12.a.4.).

Even though I used a more consistent stream of numbers to trigger footsteps, the data continued to prove problematic.

Throughout the project, the data coming from the two accelerometers/JeeNode wireless microcontroller boards proved the most difficult to control. First, the physical location of the accelerometers attached on the legs could slightly change between performances. The variability of location was due less to the placement of the accelerometers on the legs than the constant motion of the performance. I helped minimize the physical location variability of the accelerometer with creating close-fit pouches for holding the JeeNodes in place.



Figure 12.a.3. JeeNode Tx and Accelerometer Pouch

I also found slight differences in incoming data whenever I changed the 9V batteries powering the JeeNode devices. The initial change of batteries processed more frequent 'spikes' in data. I define a spike as a sharp increase or decrease in number without any spontaneous motion of the accelerometer. For example, at rest, the accelerometer outputs data generally between 130-140, or 117-126. I encountered data spikes with values above 200 while the accelerometer was in rest position (Fig. 12.a.4.). These value spikes occurred more frequently with fresher batteries. As a reaction, I minimized these peaks by cutting out any incoming data above 200.

Third, I discovered that after each battery change, the 'past' object threshold had to be adjusted to stabilize the triggering function. It is possible that the physical shifts of the
accelerometer while placing the device inside the pouch could account for subtle differences in threshold values changing. However, the increase in the frequency of 'spikes' suggested that the variability of the device's power also caused a shift in the incoming data streams.

Due to the physical variables inherent in using these devices, I was unsuccessful in developing a stable platform with which to get consistent data streams. While I was successful in processing simple triggers with simple motions, the physical act of running and therefore, the increased tempo of trigger events, proved problematic to control. The instability effected the sounds in my Feet Exposition and Development sections. The playback of the video, which was directly linked to the motion of the legs developed an irregular or choppy playback. In addition, I could not rely on the feet to provide a stable bpm tempo with which to lock the music to. Therefore, I could not create a direct connection between sound modifications (like delay, echo, or tempo mapping) and the physical motion of the legs, limiting the number of direct performer-to-sound associations potentially perceived by the audience.



Figure 12.a.4. Data stream table of the right leg accelerometer. Maximum values show motion of the leg downward, and the minimum values show motion of the leg upward.

12.b. Musical Parameter Controller

The second purpose of Max/MSP/Jitter was to control musical parameters inside the Kyma environment. Max/MSP sent two types of control messages to Kyma, MIDI messages and OSC messages. All MIDI messages sent to Kyma were sent via the PacaConnect software. All other control messages specific to musical parameter controls (like panning, filter cut-off frequency shifts, and file playback rates) were sent via OSC messages using the OSCulator software.

The only external Max object not previously mentioned used in the creation of these controls was the "randdist" object. The "randdist" object is a random number generator created at CNMAT. I used this object for average foot distance displayed on the main video monitor as well as video jitter interpolations occurring when the video is paused. The video jitter simulates normal human eye scanning while in rest position. The perception of the video moving left to right with some vertical jitter uses a normal distribution of random numbers.

All data collection functions and musical controls were placed inside of the "Controller_Kyma33_End4b.maxpat" Max 5 patcher included on the DVD. Figures A.1.1 – A. 1.i.19 iconically represent the Max 5 patcher used for the final Master's recital performance.

12.c. Video Projection Controller

The third and final function of Max/MSP/Jitter was to control the video playback of several different movie files across several videoplanes. For controlling multiple videoplanes inside a 3D projection environment, I initially employed HC Gilje's Video Projection Tools.¹⁶ HC Gilje's VP Tools offered a flexible and direct way for me to project multiple videoplanes on a single, expandable screen. While his application was meant for generic use, I chose to modify his patch and subpatchers because I was not readily familiar with gl.videoplanes and gl.render objects insider Jitter. Having a pre-existing working template enabled me to get quick, working results while learning how to work with video inside Jitter. In one sense, I was reverse

¹⁶ HC Gilje, *Video Projection Tools*, <u>http://hcgilje.wordpress.com/resources/video-projection-tools/</u> (accessed November 2, 2010).

engineering his patch in order to work with videoplanes, scraping away the unneeded particulars for use in my Terminal Project. From a different viewpoint, I was learning how to build an optimized video system for use with multiple software/hardware components.

The work spent cleaning up these patches was worth the knowledge uncovered for not only learning video projection necessary for my Terminal Project, but also the work strengthened other skills integral to creating a computer music-system, including interface design, documentation, and system optimization. While I spent countless hours cleaning up HC Gilje's patches before I was able to modify them, the result was clean templates. These clean templates provided me with a strong starting point from which I created the video projection for *Running Expressions*.¹⁷

The cleanup documentation is shown with Figures 12.c.1 - 12.c.7. All other documentation of *Running Expressions* video projection Max/MSP/Jitter patches are contained in Figures A.2.1 – A.2.1.1, located in the Appendix.



Figure 12.c.1. Videoplane cleanup documentation. HC Gilje's patch in Presentation mode.

¹⁷ These templates are marked and included in the accompanying DVD.



Figure 12.c.2. Videoplane cleanup documentation. HC Gilje's patch in Editing mode.



Figure 12.c.3. Videoplane cleanup documentation. Cleaned-up patch in Editing mode.



Figure 12.c.4. Movie source cleanup documentation. HC Gilje's patch in Presentation mode.



Figure 12.c.5. Movie source cleanup documentation. HC Gilje's patch in Editing mode.



Figure 12.c.6. Movie source cleanup documentation. Cleaned-up running movie patch in Editing mode.



Figure 12.c.7. Movie source cleanup documentation. Cleaned-up LCD movie patch in Editing mode.

13. Kyma

Kyma is a graphical programming environment for live, interactive sound generation and manipulation. I used Kyma to not only help with composing sound material found throughout the work, but I used the system for the real-time control of audio for an eight-channel performance. Kyma also sent Max/MSP/Jitter fourteen distinct MIDI messages that were used to trigger various video controls.

I chose to use the Kyma Timeline for the performance. Inside the Timeline, I delineate sections using WaitUntil Sound objects, as I am able to control when the next section will begin, freeing the performer from adhering to a particular time schedule. The Timeline also facilitated the triggering of the fourteen MIDI notes (Figure 13.1). While there isn't enough space to adequately describe the various sounds, Figures 13.2 – 13.20 briefly showcase the sound material created inside Kyma.

| 0 | 0 | | | | | | runningExp | ressions_final.ktl | | | | | | | | |
|----------|---------|----------------------------|-----------------|----------|-------------|--------------------|--------------|--------------------|---------|-------------|----------------|-----------------|---------|-----------------|--------|----------|
| | • | •••• ••• •• | Time Code 🔻 | | 30 fps 🛛 🔻 | 1/4 frame 🔻 F | ree grid 🛛 🔫 | Time edit 🔻 Co | olors 🔻 | | | | | | | |
| Autom | ation 🔻 | Audio 👻 | | | | | | • | - | | | | | | | |
| | - | | | | | | | | | | | | | | | |
| Free ru | nning 🔻 | 00:00:00:00 | 00:00:00.00 | 0 | 0:00:40.00 | 00:01:20.0 | 0 | 00:02:00.00 | | 00:02:40.00 | | 00:03:20.00 | 1 1 | 00:04:00.00 | 00 | :04:4 |
| Trk 1 | [] | ℤ ⑲ ▦ы ᅇ | HrtPlstns125678 | Road At | mosphere | | Master_All | 5 | | | [| WindMlt_crssLtR | | | | 4 |
| Trk 2 | [] | ⊈ 🖲 🎹 🖬 🌑 | | | | 0 | | 0 0 0 | | 1 1 | < | Trk 2 | Wait | : Untils | | |
| Trk 3 | [] | ⊈ 🛯 🏛 🖬 🙆 | MixedHeartRate | Ft EQFIt | t | | Aorta | | | Crsh_p3 | Final pin chro | | | | | |
| Trk 4 | [] | ∡⊗⊞¤ © |] | | | SmplClds - 2 bns v | 2 | | | | | | | | | |
| Trk 5 | [] | ∡ 🖲 🏢 🖬 🌑 |] | | FtExpstn_fn | 12 | | | | | | | | | | |
| Trk 6 | [] | ⊈ № ⊞ 🛛 📱 | HBlow | | | | | | | | | | | | | |
| Trk 7 | [] | ⊈ 🖲 🏢 🖾 🙆 | SqncrOnl123478 | | Chldrn | | | | | | — . | | | | | |
| Trk 8 | [] | ∡⊚⊞¤ © | | | | | | | | <- | - Trk | 8. MIL | DInc | ote trigo | gers | |
| Trk 9 | [] | ∡ ⑲ ▦ ལ 🖡 | | | | | | | | | | | | | 0 | |
| Trk 10 | [] | ⊈ฃ⊞⊲Ѻ | | | | | | | | | | lite sequence | r | | | |
| Trk 11 | [] | ∡⊛⊞¤ 🖡 | | | | | | | | | | | Final I | Heart Vocoder2b | | ¥ |
| | | | • | + + + | ***** | ****** | | | | +++++ | | | | |) > >+ | ()- KE |
| Master C | ontrols | <u>*</u> | | | | | | | | | | | | | | ÷. |
| | | | | | | | | | | | | | | | | <u>+</u> |
| | | | | | | | | | | | | | | | | ÷ |
| • Co | ontrol | Source | - | | | | | | | | | | | | | |

Figure 13.1. Kyma TL, with WaitUntil Sound track and MIDI note track highlighted. The timeline duration does not matter because each section's duration is determined by triggering WaitUntil Sounds.



Figure 13.2. WaitUntil Sound Object. Wiimote 1 button 2 triggers each section, although mapped as !WiiButtonA inside Kyma.



Figure 13.3. MIDI Output Pitch, serves as video command trigger, which is mapped inside Max/MSP/Jitter.



Figure 13.4. Heartbeat Sound, first electronic sound heard in *RunningExpressions*.



Figure 13.5. Heartbeat Exposition Main Sound, Vocoder with Delays





Figure 13.7. Heartbeat Low Rumble



Figure 13.8. Road Environment Ambient Sound



Figure 13.9. Selectable Foot Sounds. Accelerometers serve as sound triggers.







Figure 13.11. Selectable Children Sounds. Eleven sounds triggered and panned by Wiimote 2.

4 (Text



cc74 one Sound with lows panned the same. (from 0-1) - pans around the front of the room.

cc75 one Sound with lows panned the same. (from 0--1) - pans around the back of the room.

Select a Sound - different bank of sample clouds - that play, so I'm not stuck with only Trombones. to help tie the music in with the outside world. (music = your brain, your emotions) It's about timing.



Figure 13.12. Development Section for Trombones, Piano, and Strings



Figure 13.13. Aorta Sound Transition



Figure 13.14. Development Section Climax, part 1



Figure 13.15. Development Section Climax, part 2



Figure 13.16. Crashing Forests Sounds, Randomly Selected



Figure 13.17. Final Piano Chord. Both Wiimotes' A buttons required to trigger sound.



Figure 13.18. Wind Environment Sound. Amplitude increases in wind sound cause sound to pan from Left to Right.



Figure 13.19. Exposition Sequencer Revisited



Figure 13.20. Exposition Heartbeat Vocoder Revisited

PART IV. COMPOSITION AND PERFORMANCE STRUCTURE

While *Running Expressions* is not traditionally notated, the composition becomes more solidified with every performance. I am developing towards a more codified version through performance because each performance helps to provide immediate feedback about the directions for the musical narrative. I am still working towards creating an objective performance notation so that a different player could perform this work.

My compositional methods for *Running Expressions* can be broken down into three parts. First, I acquired sound recordings based upon my ideas, and I expounded upon these sounds inside of Kyma. Second, I drew an annotated structural sketch of how I envisioned the music flowing, with my notes describing the sounds, the parametric controls, and the programming implementations. Third, I worked with my sounds and sketches to develop a working version, complete with the various software components, alternative hardware controllers, and sound controls mapped out. Tweaks and changes of all sections occurred throughout the compositional process.



Figure 13.21. Kyma TL, with Sections Labeled

14. Section I: Exposition

Because recent technologies lack performance conventions, the music can be, at times, difficult to access. Yet, the lack of conventions enables new ways to showcase a performance and gives freedom to mold the technology to the performance. Perceiving electronic performances in this way, using alternative hardware controllers necessitates an exposition of the device inside the composition. The device exposition helps establish a musical vocabulary with which the audience may gain access to the music. The first section of *Running Expressions* serves as an exposition of the various hardware devices used in the composition.

14.a. Heart Exposition

I chose to initially emphasize the heart rate monitor and its control over the playback and tempo of the music. The heart rate monitor allowed me to directly connect the body to the music, and I saw this as paramount to the introduction of *Running Expressions*. If the heartbeat equals 0, the music will not play or will fade away if present. First, the heartbeat rate controls the playback rate of a heartbeat audio analysis file, using the SumOfSines object in Kyma (Figure 13.4). The heart rate then is mapped to control the tempo of delays and sequencer material throughout the first section. The Wiimote acts as a music conductor, signaling the changes of the chords of the music, and facilitating timbre changes in the main rhythm through shifting EQ filter cut-off frequencies.

14.b. Feet Exposition

After introducing the heart rate monitor and the Nintendo Wiimotes, I focus attention on the accelerometers located on the performer's legs. There is an exposition of real environment sounds (ambient and footsteps), and the audience hears and sees that the audio and the video playback are directly tied to the accelerometers. After a brief introduction, the environmental sounds fade and a foot waltz begins. Because of the instability of accelerometer triggers, the subsequent sounds irregularly accent the other sounds in this section. The video immediately supplements the narrative by revealing a school playground and foreshadows later movements by showing Spencer's Butte in the background.

15. Section II: Development

15.a. Running on Dillard (trombones, strings, piano)

With the exposition of all three devices inside the composition, I move toward developing the music and the musical journey. Compositionally, I shift my material to augment the physical changes of a distance run. Climbing, internal dialogue, and moving from the presence of civilization can all occur within a distance run, and I wanted to have this shift also coincide with the intermedia elements. First, I shift the video away from the suburban setting of Eugene to the wooded views of a country road. Musically, I created a darker tone with the granulation of recorded trombone material, which helped the emphasis shift from external elements of ambient sounds to internal developments taken place inside the runner's mind.

This particular section is improvised by the performer. While there are eight static chords the strings may play and a small, randomized pitch set of piano notes, the performer is free to play this section how he/she wishes. The chord changes and the overall structure of the section I performed had become solidified during rehearsals, and I instead used piano note timings of randomly selected pitches to inform the phrasings of my improvised performance.

Because the Wiimote's roll effected the time index of the trombones, I had some control over the pitch material inside the granulations. I included a performance gesture that visibly cued a trombone pitch change, and the realized note from this gesture served as the section's harmonic dominant. Because of the continual shifts in granulation of the audio, the perception of a strong dominant became more solidifying than any chordal structure for the section. The trombone gesture and harmonic dominant enabled a closing section through alternating between the V (trombone gesture) and I (strings).

15.b. Running on Spencer's Butte (Climax)

In the final development section, I attempted to completely link the run to the music. The builds in tempo, instrumentation, amplitude, and rhythm of the music parallel the physical and virtual increase in running (video portrayal of running Spencer's Butte). Not only does the music serve as a literal translation to the run, but the figurative suggestions of the psychological impacts on the mind while running can also be found inside the music.

16. Section III: Recapitulation/Coda

Like in most runs, there is a return to home. Musically, I wanted to recapitulate the first theme in its entirety, much like a sonata form, but I instead chose to lightly reintroduce the sequencer and sounds found at the beginning of the work. The reuse of exposition materials serve not just the function of a musical return, but also suggest a physical return to home, as shown by the video's return to the suburban streets.

In addition, the reuse of materials suggest a changed emotional state, for in running, after accomplishing a goal, there is a level of joy achieved. The video manifests that joy through a switch from 1st person perspective to 3rd person perspective. The internal journey of the individual runner is an objectively shared journey by all those who run. The section's materials, the reuse of music found in the exposition, and the altered vantage point of the video function also as a coda, offering an additional insight to the musical journey and placing the listener inside a different space from where he/she began.

APPENDIX

A.1. Controller_Kyma33_End4b.maxpat Figure Documentation



Figure A.1.1. Controller_Kyma33_End4b.maxpat Main Patch Window



Figure A.1.2. Performance Setup Order Patch Window



Figure A.1.3. Color Legend for Master Controller Max Patch

A.1.a. Exposition Sequencer



Figure A.1.a.1. Exposition Sequencer Patch Window. The 'matrixctrl' object controls the on/off messages of fixed notes inside the sequencer. The sequencer was built inside Kyma.

0 0

[switches]

Sequencer Controls

Matrix grid controls the playback of a 16 step sequencer inside of Kyma. Using presets of various grids for the performance

| | Wii buttons | | night1 bute 1 0 moveP | (B) | rieft1 route 1 T | | temp | coun getco | nc2 rreset 0 inter 0 0 15 cycle through each sequencer column for note-ons |
|---|-------------|----------|-----------------------------|------|------------------------|---------|------|---------------|--|
| | | s | presetH | lorz | preset gri | d chang | je | 1 | output 'getcolumn' sent to the matrix grid |
| l | - | + | R | ٠ | | 3 (| | | |

Figure A.1.a.2. Sequencer Control Patcher

E

0 0

| Wiimote 1 moves through 16 presets |
|---|
| R Wii1 L Wii1 input Wii button |
| intro sequence r cmd21 value presetH value presetH if \$i1 <= 0 then 16 else \$i1 if \$i1 == 17 then 1 else \$i1 value presetH horizontal value of grid |
| output grid preset number |
| |

[movePresets]

Figure A.1.a.3. Wiimote 1 Controls Presets Patcher



Figure A.1.a.4. Sequencer Tempo Control Patcher



Figure A.1.a.5. Tap Tempo Sequencer Control Patcher

A.1.b. JeeNode Accelerometers



Figure A.1.b.1. JeeNode Patch Window, in Presentation mode



Figure A.1.b.2. Serial Data Input Module

| \varTheta 🔿 🔿 [formttingMenu] |
|-------------------------------|
| Serial Port Message Format |
| input serial port as string |
| route port |
| ts clear |
| zi ter 1 |
| prepend append |
| output with "append" |
| |

Figure A.1.b.3. Serial Port Formatting Menu Patcher

| \varTheta 🔿 🕙 [portMesage] |
|---|
| Change Serial object argument based upon umenu selection |
| input serial port number |
| sei 0 1 2 3 4 5 |
| a b c d e f |
| prepend port |
| output format "port alpha" |
| 🔺 🛨 🖶 🛒 🚺 🖼 🖽 🗰 |

Figure A.1.b.4. Serial Port Formatting Message Patcher



Figure A.1.b.5. Serial Channel Data Display Module

| 00 | [counter] |
|--|--|
| Takes Data of Le upon Threshold, | ft Foot, Generates a count based and outputs an Overall Step Count |
| input data | input threshold input reset bang |
| 0. scale 0. 255. 255. 0. if \$f1 <= 200. then \$f1 past 120 not | reverse scaling - cleaner data on the upswing of the knee. set \$1 0 rmal 130 |
| counter 0 1 10 | ounter number |
| | |
| 曲 + ↔ | |

Figure A.1.b.6. Accelerometer Threshold Counter Patcher

| 00 | [randomizedMeter] | |
|---|--|-------------------------------------|
| Takes Ste Length, ar | ps of Left Foot, Generates an nd outputs an Overall Distanc stepCounter | average Step e in Meters |
| if \$i1 > 0. the | en \$i1 iverage running stride is about 2 n listribution to account for discrepar uman stride. Stride = 2 steps | n. Using uniform acies in normal |
| randdist uniform unpack 1. 1.023474 | 0.85 1.15) meters | |
| s distLeft | send to section counter | |
| + 0. 1.023474 | distance taken in meters | |
| value leftFoo | otDistance | tfact |
| | | |

Figure A.1.b.7. Foot Distance Calculator Patcher



Figure A.1.b.8. Master Foot Distance Display Module 1



Takes Step Distance of Both Feet, generate an Overall Distance in Meters





| r counterLeft r counterRight | P totalDist | s |
|------------------------------|---------------|---|
| ▶ 0. | ▶0. | |
| s totalCount | s totalMeters | |
| stepCount | total meters | |
| p taptempo | BPM control | |
| | | |
| s opmilleet s n | nsreet | |
| bpm | ms | |
| p send | dsToKYMA | |

Figure A.1.b.10. Master Accelerometer Control and Routing Module 2



Figure A.1.b.11. Feet Accelerometer Tempo Control Patcher



Figure A.1.b.12. Master Feet Counter Calculator Patcher, controls video



Figure A.1.b.13. Master Feet Distance Calculator Patcher, lcd display
0 0

[sendsToKYMA]

Select-A-Sound Control

Foot Accelerometers controls sent to Kyma via OSCulator

| Foot Foot of file, reset trigger for KYMA select sound at random each bang from an array of 8 KYMA select sound at random each bang from array of 8 KYMA select sound at random each bang from array of 8 KYMA select sound at | Right Foot |
|--|--------------------------------|
| t b b del 100 counter 0 0 1 t - t - t - t - t - t - t - t - t - t - | at random each h array of 8 |
| OSCulator output /selectSound_leftFootTrigger/\$1 /selectSound_leftFoot/\$1 /selectSound_rightFootTrigger/\$1 /selectSound_ udpsend 127.0.0.1 8000 KYMA cc82 udpsend 127.0.0.1 8000 KYMA cc81 udpsend 127.0.0.1 8000 KYMA cc84 udpsend 127.0.0 | 1_rightFoot/ \$1 |
| | |

Figure A.1.b.14. Accelerometer Sends to Kyma Patcher part 1. When triggering sounds inside Kyma, Max/MSP must reset non-zero values back to zero in order to re-trigger a Kyma Sound object.

| r bpmFeet | r totalCount | |
|--|--|----------------------------------|
| cc85 - BPM of feet | cc86 - Total fe | pot count |
| Test Tap s testbpm b 0. b 0. | Master Test p masterTester s left s right | t b b del 10 counter 0 0 1 |
| /foot_BPM/ \$1 | /feet_totalCount/ \$1 | /feet_totalCount_asTrigger/ \$1 |
| udpsend 127.0.0.1 8000 KYMA cc85 | udpsend 127.0.0.1 8000 KYMA cc86 | udpsend 127.0.0.1 8000 KYMA cc87 |
| | | |

Figure A.1.b.15. Accelerometer Sends to Kyma Patcher part 2

| Reset | Foot gate | Feet test (w/o) gadgets |
|--|-------------------------------|-------------------------|
| esets step Counter and Distance reset won't trigger the sound in KYMA s resetCounter | counting on/off conOff | metro 300 |
| r feetTester Master Test P masterTester s left s right video | nster st Meters, per | s feetTester |

Figure A.1.b.16. Master Accelerometer Control Module 3

| 🖲 🔿 🚺 [startfeetCounter] |
|--|
| Foot Counter Gate |
| Trigger the Start of the Foot Counter. Absolute, only done once per performance, but need a way to stop if need be. |
| "s" "c" press "c" then key key "s" to trigger 29 29 29 sel 115 sel 99 I I pak 0 0 I unpack I I I ser \$\si1 + \$\si2\$ |
| 0 looking for a 2 |
| if \$i1 == 2 then bang else out2 \$i1 |
| bangs out ONLY if 's' is followed by a 'c'. To retrigger, must hit an 'c' followed by a 's'. |
| output bang |
| A = 🕂 🛒 🖬 🖻 🗉 🕮 |

Figure A.1.b.17. Master Feet Counter Control Patcher



Figure A.1.b.18. Master Feet Counter Test Patcher

A.1.c. Heart Rate Monitor



Figure A.1.c.1. Heart Rate Routing Patch Window. Communication shows information received from Processing, and routing to OSCulator.

| \varTheta 🕙 🕙 [heart | rateToPlaybackRate] | | |
|--|--|--|--|
| Convert HR to Movie Playback Rate (1 second clip) | | | |
| input heart | rate | | |
| expr 60./\$f1 | convert hr to seconds | | |
| - 1. | subtract 1 to inverse timescale | | |
| abs 0. | abs - can only use positive floats | | |
| scale 0. 0.5 1. 2. | scale seconds to playback rate 1 = normal speed 2 = double speed | | |
| output play | vback rate | | |
| 🔺 + 🕂 | 🕺 🚹 🗳 🗄 🏢 🇱 | | |

Figure A.1.c.2. Heart Rate Controls Movie Playback Patcher

| 😑 🔿 🔿 [scaleHeartBeat4] | | | | | |
|---|--|--|--|--|--|
| Heart rate scale to playback rate for Kyma ("AorticRegurtitation2 s256") non-linear playback rate - this patcher averages out the playback | | | | | |
| input heart rate | playback ranges between heartrates | 1/2 speed ranges between heart | trates different hrm range for ea | ich rate | |
| convert hrm to playback rates scale 59. 78. 1. 0.75 | split 78 96 scale 78. 96. 0.75 0.5 | split 96 130 scale 96. 130. 0.5 0.25 | split 130 158 scale 130. 158. 0.25 0.125 | split 158 192 scale 158. 192. 0.125 0.0625 | |
| p switchNumbers gate opens based upon HRM rate gate locks HRM goes 51 | eed if pelow | | | | |
| switch 5 | switch 5 | | | | |
| • 0. the final playback rate • 0. 64 bpm audio file in "AorticRegurtitation2 s256" is 2.455515d s long • 0. the Heartbeat rate at 0.75 speed = 78bpm. 0.5 = 96bpm 0.25 = 130bpm, 0.125 = 158bpm | | | | | |
| 🔺 + 🕂 🖌 i 🗳 🗄 | | | | | |

Figure A.1.c.3. Heart Rate Controls Heartbeat/Aorta Audio Playback Patcher

| | [switchNi | umbers1 | | |
|-------------------------------|------------------|--------------|-----------------------------|---|
| Heart rate opens of | orresponding gat | e depending | on rate | |
| input heart rate | | | | |
| (if \$i1 < 51 then 0) split 5 | 1 64 split 65 86 | split 87 128 | split 129 172 split 173 230 | |
| 1 | 2 | 3 | 4 5 | |
| output switch numb | er | | | |
| 🔺 🕂 🕂 🐖 i | | _ | | 1 |

Figure A.1.c.4. Heart Rate Controls Switch Gate Patcher



Figure A.1.c.5. Master Heartbeat Audio Volume Control Patcher

A.1.d. Control Window



Figure A.1.d.1. Performance Control Patch Window, in Presentation mode



Figure A.1.d.2. Performance Control Patch Window, in Performance mode

| \varTheta 🕙 🕙 [clockCounter] |
|---|
| Clock Counter |
| input 1 or 0 - start/reset |
| metro 1000 codmess 3600 max \$1 set 0 jam 0 |
| counter 0 3600 1 hour counter |
| • output time as integer |

Figure A.1.d.3. Performance Control Timer as Counter Patcher



Figure A.1.d.4. Performance Control Timer as Time Patcher

A.1.e. MIDI



Figure A.1.e.1. MIDI Controls Patch Window, all Make Note messages Sent to Kyma via PacaConnect.

A.1.f. Video Control



Figure A.1.f.1. Video Control Patch Window, overview of Window layout

| 00 | |
|---|--|
| Video Control via MIDI | Commands to Execute upon Receiving |
| MIDI IN from KYMA | |
| r videoReset 0 | p sectionCommands |
| route 1 2 3 4 5 6 7 8 9 10 11 12 13 14 21 eac s cmd1 s cmd2 s cmd3 s cmd4 | h command == bang to execute mainly video commar s cmd5 s cmd6 s cmd7 s cmd8 s cmd8 |

Figure A.1.f.2. Video Control MIDI routing, part 1

| [VideoControl] | |
|--|--|
| Receiving a MIDI message from KYMA | |
| All of these commands relate to controlling video/Jitter | |
| video commands (see below) nd8 s cmd9 s cmd10 s cmd11 s cmd12 | s cmd13 (s cmd14) (s cmd21) out of order intro command |

Figure A.1.f.3. Video Control MIDI routing, part 2



Video Commands to Execute upon Receiving a MIDI message from KYMA

1st wait until ensures correct speed and movie to moviesource be redundant in messages

1. gt # - on/off - mov1 on/ mov2 off qt multiplier - mov1 = 167.6, mov2 = 158 movie select -mov1= 2, mov2 = 0 jitter on/off - mov1 on/ mov2 off fade line - i.e. xfade - none reset section meters - mov1 = reset new section comment - EXPOSITION 2 xfade - running R to L, heart R to L post HeartExposition qt # - mov2 on movie select -mov2= 3, MixerRunningB = src2(#2) iitter on/off - mov2 on reset section meters - mov2 = reset FIRST before fade fade - x09cell off - 5sec xfade - running L to R - 1sec fade - x09cell on - 5sec new section comment - FEET EXPO before selectASound qt # - mov1 off at multiplier - mov1 = 154.55 movie select -mov1= 0 jitter on/off - mov1 off turns off movie two reset section meters - mov1 = reset Children Exposition at # - mov1 on movie select -mov1= 4 reset section meters - mov1 = reset jitter on/off - mov1 on xfade - R to L qt # - mov2 off after fade new section comment - CHILDREN Trombones at # - mov2 on qt multiplier - mov2 = ??? try 160 movie select -mov2= 5 (Dillard) reset section meters - mov2 = reset fade - x09cell off - 5sec xfade - running L to R - 1sec fade - x09cell on - 5sec qt # - mov1 off after fade new section comment - TROMBONES

7. Development Section qt # - mov1 on qt multiplier - mov1 = ??? try 160 movie select -mov1= 6 ("5_WoodsRunning_DevelopmentA") reset section meters - mov1 = reset fade - x09cell off - 10sec xfade - running R to L - 1sec fade - x09cell on - 10sec new section comment - DEVELOPMENT new breath value 0.555 (breath rate) tpt preset 1 8. Development Section B qt multiplier - mov2 = ??? try 160 movie select -mov2= 7 ("5_WoodsRunning_DevelopmentB") reset section meters - mov2 = reset xfade - running L to R - 1sec after xfade, select small moviefile to help framerate 9. Development Section C qt multiplier - mov1 = ??? try 160 movie select -mov1= 8 ("5_WoodsRunning_DevelopmentC") reset section meters - mov1 = reset xfade - running R to L - 1sec after xfade, select small moviefile to help framerate 10. Cut all off quick fade in once Fade out movie 1 both Wiis hit A Select mov 1 = 9 "6_Butte_View" Select mov 2 = 10 "6_ButtePan" Playback Rate of 2 to stop Feet Control, toggle this gate Playback Rate of 1 to stop Feet Control, toggle this gate Playback Loop Mode of mov 1 = 2. mov 1 = autostart Turn jitter interpolation off for both movies 11. XFade in Butte Pan from Butte View Loop mode back to 1 for mov 1 Xfade L to R = 500 ms Playback Rate of 2 to be controlled via Wii, toggle this switch Turn off meters and rate lcd displays in QT 12. Fade out Butte Pan Fade x09cell in 5 sec. XFade R to L = 5040 ms (arbitrary) 13. Final Movie load mov 1 = 11 the Final Movie begin playing mov 1 fade in mov 1 (x09cell) 8 sec fade



Figure A.1.f.4. Video Section Command Descriptions Patcher



Figure A.1.f.5. QuickTime Movie 'qmetro' Toggle Module

| 00 | 0 | | [Qtonoff1] | | |
|----|---------------|---------------|--|---------------|--------------------------------------|
| | output on/off | is jit.qt.mov | vie on/off) ut r cmd7 MUST KEI in order to Redundan 1st running | Ilize to save | AT ALL TIMES movies work thing |
| | F 🕂 | 🐖 i | | | |

Figure A.1.f.6. QuickTime Movie #1 'qmetro' Toggle Patcher. Controls Running Movie #1.



Figure A.1.f.7. QuickTime Movie #2 'qmetro' Toggle Patcher. Controls Running Movie #2.



Figure A.1.f.8. QuickTime Movie #4 'qmetro' Toggle Patcher, note Movie #3 does not exist. Movie heartbeat.

| 0 0 | [Qtonoff5] |
|-------------------------|---|
| QT on/off (turns jit.qt | movie on/off) utilize to save framerate |
| (r cmd1) (r cmd11) | |
| output on/off | LCD display heart rate (qt #5) |
| 🔺 + 🕂 🐖 | |

Figure A.1.f.9. QuickTime Movie #5 'qmetro' Toggle Patcher. LCD display heart rate.



Figure A.1.f.10. QuickTime Movie #6 'qmetro' Toggle Patcher. LCD display meters.



Figure A.1.f.11. QuickTime Frame Rate Multiplier Control Module



Figure A.1.f.12. QuickTime Movie #1 Frame Rate Multiplier Control Patcher

| 0 0 |) [framerateM | ultiplier2] | | | |
|--------------|--|---------------------------|--|--|--|
| QT f | QT framerate multiplier (how many frames will pass with each footstep) | | | | |
| (r cm 158 | d1) (rcmd6) (rcmd8) 3 180 200 | | | | |
| Q | output framerate | 2nd running movie (qt #2) | | | |
| | + 🖌 i 🖪 🗄 | | | | |

Figure A.1.f.13. QuickTime Movie #2 Frame Rate Multiplier Control Patcher

| Set Comment Field for Control Window designating each Section | | | | | | |
|---|---------------|---------------|---------------|---------------|---------------|--------------------|
| r cmd1 | r cmd3 | r cmd5 | r cmd6 | r cmd7 | r cmd11 | loadmess set OFF |
| s sectionName | s sectionName | s sectionName | s sectionName | s sectionName | s sectionName | I s sectionName |

Figure A.1.f.14. Performance Control Window Comment Field Module

| Movie Selection | | |
|-----------------------------|------------------------------|--|
| p movieSelectionQT1 | p movieSelectionQT2 | r cmd1 r cmd10 del 900 1 0 |
| s movieRunningSelect) qt #1 | s movieRunning2Select) qt #2 | s movieHeartSelect) qt#6 (Heart Movie) |
| r cmd1 r cmd3 4 2 | | |
| srun_movieMixerSelectB Mixe | er B | |

Figure A.1.f.15. QuickTime Movie Selection Module

| 0 0 | [movieSelectionQT1] |
|--|--|
| Movie Selection | |
| r cmd1 r cmd4 r cmd5 2 0 4 2 0 4 | r cmd7 r cmd8 r cmd6 r cmd9 r cmd10 r cmd13 del 3000 del 12000 del 900 6 0 8 9 11 ion int 1st rupping movie (qt #1) |
| | |

Figure A.1.f.16. QuickTime Movie #1 Selection Patcher

| 0 0 | [movieSelectionQT2] |
|-----------------|--|
| Movie Selection | |
| | r cmd5 r cmd6 r cmd8 r cmd10 r cmd13 del 4000 0 5 7 10 0 |
| | 2nd running movie (qt #2) |
| 🔺 🕂 🕂 🐖 | |

Figure A.1.f.17. QuickTime Movie #2 Selection Patcher

| Turn Interpolation on/off - screen motion when movie is paused | | | | |
|--|--|--|--|--|
| r cmd4 r cmd10 r cmd1 r cmd5 | r cmd1 r cmd10 r cmd13 r cmd5 r cmd11 | | | |
| 0 1 | 0 1 | | | |
| s itterRun s itterRun jitter on/off | s jtterRun2 s jtterRun11 jitter on/off | | | |

Figure A.1.f.18. QuickTime Movie 'srcrect' Pixel Jitter Toggle Module

| Fades (including xfades) | |
|--|-------------------|
| p faderMainMixer Fader top fader, use for fading between movies | p faderx09fade |
| p faderHRMmixer Fader bottom fader 0. s hrm_movieMixerFader heart movie mixer Fader bottom fader http://www.sec.en/ | p faderheartVideo |
| | p faderLCDdisplay |

Figure A.1.f.19. QuickTime Movie Fade Control Module

| 0 0 | [faderMainMixer] | | | |
|---|---|--|--|--|
| Fades (including xfades) main movie mixer Fader xfader, use for fading between movies | | | | |
| r cmd1 R to L qt #2 to #1 1. output mix fade (interpolated) | r cmd2 r cmd5 r cmd7 r cmd9 r cmd12 del 10000 del 40 del 5000 del 40 1., 0. 8000 1., 0. 4000 1., 0. 1000 0., 1. 1000 ine 0. | | | |
| 🔺 + 🕂 🐖 | | | | |

Figure A.1.f.20. QuickTime Movie Main Mixer Fade Control Patcher

| 00 | [faderx09fade] |
|-----------------------|--|
| Fades (including xfac | running movie alpha fader, use for fading between movies |
| r cmd1 out r cmd3 | rcmd6 rcmd14 rcmd7 rcmd10 rcmd12 del 800 del 1000 del 1000 r000 1., 0. 10000 1., 0. 5000 0., 1. 5000 0., 1. 5000 0., 1. 5000 0., 1. 8000 |
| 🔺 + 🕂 🛒 | |

Figure A.1.f.21. QuickTime Movie Running Movie Fade Control Patcher



Figure A.1.f.22. QuickTime Movie Heartbeat Movie Mixer Fade Control Patcher



Figure A.1.f.23. QuickTime Movie Heartbeat Movie Fade Control Patcher

| 0 0 | [faderLCDdisplay] | | | |
|------------------|---|--------------------------|--|--|
| Fades (including | xfades) | lcd display alpha fader | | |
| r cmd1 | r cmd10 delay 800 T 1., 0. 30 line 0. | | | |
| (interpolated) | | output bang when done | | |
| A + 🕂 🐬 | | | | |

Figure A.1.f.24. LCD Display Fade Control Patcher

| Reset Meters (necessary for footstep control of time scrub through rate) | | | | | |
|---|--|--|--|--|--|
| r cmd1 r cmd4 r cmd5 r cmd7 r cmd9 del 40 reset JeeNode Section Meters - used per video section for movie 1 s resetSectionCounter | rcmd3 rcmd6 rcmd8 del 4000 del 400 del 40 reset JeeNode Section Meters - used per video section for movie 2 s resetSectionCounter2 | | | | |

Figure A.1.f.25. Feet Accelerometer Section Distance Counter Reset Module

l

| Loop Mode, AutoStart, Rate | | | |
|---|----------------------------|--|--------------------|
| Loop Mode rcmd10 rcmd11 rcmd1 AutoStart | r cmd10 r cmd1 del 400 | r cmd10 r cmd1 del 200 | Rate rcmd10 rcmd1 |
| 2 1 s movieRunningLoop | 1 0 s movieRunningStart | autostart 1 autostart 0 s movieRunningAutoStart | s movieRunningRate |

Figure A.1.f.26. Miscellaneous QuickTime Movie Control Module

| Video Control Gates | |
|--------------------------------|----------------------------------|
| r cmd10 r cmd1 | Cremd11 (remd1) |
| MIDInn10 from KymaTimeline | Open and close Gates controlling |
| shuts off time control by Feet | the playback Rate of Movie 2 |
| s movieRunning_TimeGate | s movieRunningQT2_WiiRate |

Figure A.1.f.27. Master Video Control Switch Module

A.1.g. Wiimote Master



Figure A.1.g.1. Wiimote Master Control Patch Window



Figure A.1.g.2. All-Mute Wiimote Master Control Module



Figure A.1.g.3. Exposition Fade-Out Wiimote Master Control Module

| FINAL PIANO (| CHORD | Raise both Arms in order to assist the sound |
|-------------------------------------|-------------------------------------|---|
| A wii1 | + A wii2 | |
| pak 0 0 pak t unpack | he numbers so | no order is necessary |
| expr \$i1 + \$i2 Will o are p | only generate a pressed. Otherwi | 1 when both A buttons ise, always a 0. |
| if \$i1 == 2 then 1 else | e out2 \$i1 | Once a 1 is |
| | if \$i1 <= | = 1 then 0 release both buttons in order to switch |
| r cmd10 r cmd1 kyn 1 0 | na TL ds 0 | back to 0 (added as failsafe) |
| gate | Atop vide | o Butte (MIDI nn 10), open gate o fade in of Butte pan |
| r cmd9 | | p butteFadein |
| /pianoFinal/ \$1 KY | (MA CC112 | 0. s x09cellFade |
| | | |

Figure A.1.g.4. Final Piano Chord Wiimote Master Control Module



Figure A.1.g.5. Butte Pan Video Fade-In Patcher

A.1.h. Wiimote 1

| 00 | | [wiiMote1] | | |
|----------------------|---|--|---|---|
| WiiM | ote 1 Controls associated with the Wii Remote 1 (left hand | d) | | |
| (| EXPOSITION CONTROL MIDI channel 01 | | | |
| | roll Volume L Pulsations scale 0.1.0.5.0.95 (R chopper) scale 0.1.0.3.0.56 Wilrel controls volume | (B) + (roget) (doit) Sequencer presets Wi controls associated | (B) + (home2) Exposition fade out Fades out main | |
| | Wi roli controls volume for Sequencer (left min, for Heart Exposition (left right max) min, right max) | with the sequencer | Exposition Master Mixer Fader | |
| OSCulator outputs | ExpositionVolumeL/ \$1 KYMA CC90 /ExpositionVolRSequencer/ \$1 KYMA CC93 | (note1/ - /note7/ KYMA CC 101-107 Sent by Sequencer. Found in "p ExpositionSequencer" | | = |
| l | | | | |
| | FEET CONTROL - Transition MIDI channel 03 | | | |
| | Teach regit Environment nght Select #1-3 for Selectable Sound in pervironmesselection Mute Feet 1 Current Enviro selection ponoffinterpolation 1 Current Enviro selection 0.0 | Counter 0 0 2 repeat loop twice point 0 2 repeat loop twice point 0 2 0-1 in varying times 0, 1, 200 1, 0, 4200 me 1.20 1, 0, 4200 | rs lin rs | |
| OSCulator outputs | /environSelect/ \$1 KYMA CC11 /muteFeetMaster/ \$1 KYMA CC12 udpsend 127.0.1 6000 udpsend 127.0.1 6000 | /panningLine/\$1 KYMA CC06 udpsend 127.0.0.1 8000 | | |
| | | | | |
| | TROMBONE CONTROL - 2nd Section MIDI channel 02 | | | |
| < | Control Control <t< th=""><th>plume pDrawPan</th><th>keep at 1. do not change</th><th>e</th></t<> | plume pDrawPan | keep at 1. do not change | e |

Figure A.1.h.1. Wiimote 1 Control Patch Window, overview of Window layout

| EXPOSITION CON | NTROL MID | I channel 01 | | |
|---|--|---|---|---|
| roll Vo scale 0. 1. 0.5 0.95 Wir min, | Dlume L ulsations chopper) rol controls volume reart Exposition (left righ max) | Volume R Sequencer Wii roll controls volume for Sequencer (left min, rigth max) | (B) + (right) (left) Sequencer presets Wii controls associated with the sequencer | (B) + (home2) Exposition fade out Fades out main Exposition Master Mixer Fader |
| OSCulator outputs / ExpositionVolumeL/ \$1 | KYMA CC90 /ExpositionVolRSeque udpsend 127.0.0.1 8000 | encer/ \$1 KYMA CC93 | /note1/ - /note7/ KYMA CC 101-107 Sent by Sequencer. Found in "p ExpositionSequencer" | |

Figure A.1.h.2. Wiimote 1 Heart Rate Monitor Exposition Control Module

| (| FEET CONTROL - Transition | MIDI channel 03 | | |
|----------------------|--|--|---|--|
| | (left) left route 1 p environOneSelect c environSelect t c environSelect c environSel | fhome if \$i1 == 1. then bang else out2 bang p onOffInterpolation 0. | Counter 0 0 2 repeat loop twice split 0 1 0-1 in varying times 0, 1. 2000 ine 1.20. 1, 0. 4200 ine 1.20. | |
| OSCulator outputs | /environSelect/ \$1 KYMA CC11 L udpsend 127.0.0.1 8000 | /muteFeetMaster/ \$1 KYMA CC12 depsend 127.0.0.1 8000 | /panningLine/ \$1 KYMA CC06 udpsend 127.0.0.18000 | |
| | | | | |

Figure A.1.h.3. Wiimote 1 Feet Exposition Control Module

| 🖲 🔿 🧑 [environOneSelect] |
|--|
| Wlimote 1 select btw. 1-3, ambient sounds |
| input Wii left 2 input Wii right |
| value envrionOne -1 - +1 if \$i1 <= 0 then 3 else \$i1 |
| T if \$i1 == 4 then 1 else \$i1 |
| value envrionOne variable |
| output new sound selection |
| 🔺 + 🕂 🕅 🖬 🖬 🗰 |

Figure A.1.h.4. Environment Sound Select Patcher, in Feet Exposition



Figure A.1.h.5. Feet Sound Mute Patcher, in Feet Exposition



Figure A.1.h.6. Wiimote 1 Development Section Control Module, part 1

| raccel_r (raccel_yaw) roll yaw p DrawPan | Panning left Panning keep at 1. do not change | rieft1 right right right right Time Index (Pitch) Selection Current Pitch selection | | |
|---|--|---|--|--|
| /bonePan_Angle/ \$1 KYMA CC74 udpsend 127.0.0.1 8000 Wii in Physical Space to control Panning | /bonePan_Radius/ \$1 KYMA CC76 udpsend 127.0.0.1 8000 of Trombone SampleClouds | /boneHarmony/ \$1 KYMA CC78 Ludpsend 127.0.0.1 8000 | | |

Figure A.1.h.7. Wiimote 1 Development Section Control Module, part 2

| | (minus) |
|---|--|
| Trigger/Mute Strings | Breaths Mute |
| if \$i1 == 1 then bang else out2 bang p interpolateMuteStrings | if \$i1 == 1 then bang else out2 bang 0., 0.45 1000 0.45, 0 1000 line 1. 20. line 1. 20. |
| /melodyAmp/ \$1 KYMA CC79 | /breathAmplitude2/ \$1 KYMA CC80 |
| | Wii A to control Amplitude of Breaths (push/hold to turn on) |

Figure A.1.h.8. Wiimote 1 Development Section Control Module, part 3

| 🖲 🔿 🔵 [interpolateMuteStrings] | | | | |
|--|--|--|--|--|
| Interpolate Strings M | ute - Fade in/out | | | |
| input bang ON | input bang OFF | | | |
| 0., 0.25 350 line 1. 20. 2 s stringSw switch 1 2 | 0.25, 0 1200 line 1. 20. 1 s stringSw | | | |
| output interpolated v | alue | | | |
| 🔺 🕂 🕂 🕅 i | | | | |

Figure A.1.h.9. String Mute Patcher

0 0 [DrawPan] Panning based upon the Wii's position - drawing within a 2D space Pan Angle is opposite of Wii Two. So same motions up/down will cause Wlis to be in opposite places. This is input roll 1 input yaw input pitch 3 due to the control of volume on the verical pitch axis. expr \$f1 + \$f2 / 2 cc74/76 - Panning I **Trombone One** speedlim 30 T scale 0.05 1. 0. 127. scale 0.02 1.5 0. 127. Т scale 0. 127. 0. 0.969 if \$f1 < 63. then -1. else 1. ▶0.

loadmess 1.

Bone Radius

2 KYMA CC76

| Figure A.1.h.10. Panning of Trombones Control Patcher | |
|---|--|
|---|--|

* 1.

▶0.

Δ

!Angle

Bone Pan Angle
KYMA CC74

Angle Left to Right dependent upon front and back of room

Ŕ

| 0 0 | [pitchesOne | Select] |
|--|--|------------------|
| Wii select St | ring chords (1 | 1-8) |
| input W | vii left 🚺 in | put Wii right |
| value pitchesOr -1 if \$i1 <= 0 th if \$i1 == 9 th loadr value pitche | ne value pitch + 1 nen 8 else \$i1 nen 1 else \$i1 ness 0 sele e pitchesOne varia | resOne action |
| output | t new selection | |
| A + + | 🕺 i | |

Figure A.1.h.11. String Harmony Pitch Selection Patcher

Ö.



Figure A.1.h.12. Wiimote 1 Development/Climax Section Control Module

A.1.i. Wiimote 2



Figure A.1.i.1. Wiimote 2 Control Patch Window, overview of Window layout

| | EXPOSITION CONTROL | MIDI channel | 01 | | | |
|----------------------|--|---|--------------------------------------|--|---|---|
| | roll Volume R Pulsations (L chopper) | raccel_2 roll Volume L Sequencer | toggles sequencer on/off (MAX) | p filterControlPreset | sKYMA | |
| | scale 0. 1. 0.5 0.95 | scale 0. 1. 0.3 0.56 | 0.001 (1.00) | Time Constant | Side Value | Bandwidth 0.04325 |
| OSCulator outputs | /ExpositionVolumeR/ \$1 KYMA CC91 | /ExpositionVolLSequencer/ \$1 udpsend 127.0.0.1 8000 | KYMA CC94 | /ExpoTimeC KYMA CC95 onstant/ \$1 udpsend 127.0.0.1 8000 | /ExpoSide KYMA CC96 Value/ \$1 udpsend 127.0.0.1 8000 | /ExpoBand KYMA CC97 width/ \$1 udpsend 127.0.0.1 8000 |
| | | | To Control Window | s filterExpoValue1 s | filterExpoValue2 | s filterExpoValue3 |

Figure A.1.i.2. Wiimote 2 Heart Rate Monitor Exposition Control Module

0 0 [filterControlPresetsKYMA] Cycle through Preset 1-6, output interpolation for three filters input R Wii2 input B Wii2 input L Wii2 r cmd2 3 2 ш route 1 route 1 sel 0 del 5 L. p selectFilterPre) O gate 6 Heartbeat Vocoder Levels T I Time Constant minimum 0.1403 maximum 2.000 Side Level minimum 0.5421 maximum 0.7188 p interpolateFilterValues Bandwidth minimum 0.04325 maximum 0.125 (use only with Time Constant = 2.0) s filterNumber output output output 2 1 Time Side Band to Control Window Constant width Level 4 8 Δ

Figure A.1.i.3. Filter Control Presets Patcher

| ● ○ ○ [selectFilterPre] | | | | |
|---|--|--|--|--|
| Filter presets 1-6, store as variable | | | | |
| input bang 2 input bang | | | | |
| value expoFiltersPreset value expoFiltersPreset +1 if \$i1 <= 0 then 6 else \$i1 if \$i1 >= 7 then 1 else \$i1 codmess 0 value expoFiltersPreset value expoFiltersPreset | | | | |
| | | | | |

Figure A.1.i.4. Filter Preset Selection Patcher

| 😁 😳 👘 [interpolateFilterValues] | | | | | | |
|---|--|--|---|--|--|--|
| Interpolate the three filter values of the Exposition with six presets all inputs are bangs | | | | | | |
| Preset 1 | Preset 2 | 🛐 Preset 3 🚺 Preset 4 | 5 Preset 5 6 Preset 6 | | | |
| 0.1403 0.5421 0.04325 s tcPre s siPre s bwPre | 2. 0.5421 0.04325 s tcPre s slPre s bwPre | 2. 0.5421 0.125 2. 0.7188 0.125 s tcPre s slPre s bwPre s tcPre s slPre s bwP | 0.1403 0.5421 0.125 0.1403 0.7188 0.04325 re stcPre siPre swPre stcPre siPre swPre | | | |
| p onOffInterpolationTC | | SIde Level ponOffInterpolationSideLevel | rbwPre Bandwidth p onOffInterpolationBandwidth | | | |
| address 0 1402 | | Master Filter Interpolate Time Value expositionFilterInterpolationTime1 | | | | |
| 0.1403 | | 0.5421 | 0.04325 | | | |
| output Time Constant | | output Side Level | 3 output Bandwidth | | | |
| 🔺 + 🕂 🐖 i | | | 1 | | | |

Figure A.1.i.5. Interpolation Between Presets Patcher

[onOffInterpolationTC] 0 0

Interpolate between Time Constant values

| _ | input float | stored variable is banged out first, then stores new value for next interpolation |
|---|--|---|
| | tfb bang stored value and then new valu value timeConstant | e time constant variable time interpolation value (r expositionFilterInterpolationTime1) |
| | bondo ensures proper packing | 1000. |
| | sprintf %.4f %.4f %ld | construct line message as string |
| | t b l 2., 2. 1000 line 0. 0. value timeCons | set string as message and then bang (line won't take a string) |
| | output interpolated values | output bang when done |
| Δ | + 🕂 🐖 i | |

Figure A.1.i.6. Time Constant Parameter Interpolation Patcher

| ● ○ ○ [onOffInterpolationSideLevel] | | | | | |
|--|---|--|--|--|--|
| Interpolate between Side Level values | | | | | |
| input float stored variable is banged out first, then stores new value for next interpolation | _ | | | | |
| bondo 2 bondo 2 bondo 2 bondo 2 bondo 2 bondo ensures proper packing t b l 0.5421, 0.5421 1000 line 0. 0. value sideLevel | | | | | |
| output interpolated values output bang when done | _ | | | | |
| | | | | | |

Figure A.1.i.7. Side Level Parameter Interpolation Patcher

| Interpolate between Bandwidth values | | | | | |
|---|---|--|--|--|--|
| input float | Stored variable is banged out first, then stores new value for next interpolation | | | | |
| tfb bang store and then n value bandwic | id value iew value dth bandwidth variable | | | | |
| bondo 2 bondo ensures proper packing sprintf %.5f %.5f %ld t b l 0.04325, 0.125 1000 line 0. 0. value band | set string as message and then bang (line object won't accept a string) | | | | |
| output interpolate | d values output bang when done | | | | |
| 🔺 🕂 🕂 🕅 | | | | | |

Figure A.1.i.8. Bandwidth Parameter Interpolation Patcher

| | FEET TRANSITION | MIDI channel 01 | |
|----------------------|--|--|--|
| | Children Childrenselect Childrenselect Children Sound Selection Current Child selection Current Child selection | Kyma eeo eeo Children Sound TL5 test KYMA OSC before performance | B2 (racel/2 pitch Children and road packground gate playback speeds Value roadAmbienceRate 0. playback rate |
| OSCulator outputs | /childSelect/ \$1 KYMA CC11 L udpsend 127.0.0.1 8000 | /childTrigger/ \$1 KYMA CC12 | /environRate/ \$1 KYMA CC10 |

Figure A.1.i.9. Wiimote 2 Feet Exposition Control Module

| \varTheta 🔿 🕐 [ChildrenSelect] | | | | | |
|---|--|--|--|--|--|
| Wiimote 2 select children sounds (1-11) via bangs | | | | | |
| input Wii left 2 input Wii right | | | | | |
| value childOne value childOne + 1 | | | | | |
| if \$i1 <= 0 then 11 else \$i1 T if \$i1 == 12 then 1 else \$i1 | | | | | |
| value childOne selection value childOne | | | | | |
| output new selection | | | | | |
| 🔺 🛨 🗭 🛒 🖬 🖻 🕒 , | | | | | |

Figure A.1.i.10. Children Audio File Selection Patcher

| 😝 🔿 😁 [roadValueInterpolation] | | | | | |
|---|--|--|--|--|--|
| Return to normal playback rate with Wii A | | | | | |
| input Wii A to output stored value | | | | | |
| value roadAmbienceRate stored variable is banged out from Wii A, Wii B serves as gate | | | | | |
| sprintt %.2N, 1. 1818 1818ms = 33 rpm t b l r 0.66, 1. 1818 (line won't take a string) | | | | | |
| line 0. Value roadAmbienceRate 0. store new value for next interpolation | | | | | |
| output interpolated values 2 output bang when done | | | | | |
| | | | | | |

Figure A.1.i.11. Road Ambience Sound Playback Rate Interpolation Patcher

| [| TROMBONE CONTROL - 2nd Section MIDI channel 02 | | | | | |
|-----------|--|--|------------------------|---|--|--|
| | (accel_r2) pitch | (82) | (B2) (raccel_p2) pitch | (raccel_r2) roll (raccel_yaw2) yaw (raccel_p2) pitch | | |
| | Time Index | | Volume | Panning | | |
| | | 0.45 0.1000 | | p DrawPan | | |
| | scale 1. 01. 1.1 | Line 1. 20. | gate | | | |
| OSCulator | /timeindex2/ \$1 KYMA CC72 | /boneAmplitude2/ \$1 KYMA CC73 | /boneAmplitude2/ \$1 | /bonePan_Angle2/ \$1 KYMA CC75 | | |
| outputs | udpsend 127.0.0.1 8000 | udpsend 127.0.0.1 8000 | udpsend 127.0.0.1 8000 | udpsend 127.0.0.1 8000 | | |
| | Wii Roll to control TimeIndex of Trombone SampleClouds push/hold to turn off | Wii B to control Amplitude of Trombone SampleClouds | | Wii in Physical Space to control Panning of Trombone SampleClouds | | |

Figure A.1.i.12. Wiimote 2 Development Section Control Module



Figure A.1.i.13. Panning of Trombones Control Patcher

| | DEVELOPMENT CONTROL | | MIDI channel 01 | | |
|---------------------|--------------------------------|---|--|---|-------------|
| | left route 1 right Turbelect | B triggers sel 1 interpolation | Presets minus | rminus2 rplus2 oute 1 plus route 1 p BreathRamp | Breath Ramp |
| | s tptPresetn to Control Window | expr 0.1112 * \$i2 p interpolateBtwPresets | 1 second interpolation to Control Window | expr 0.555 + (\$f1 * 0.03) 0 p interpolateBtwPresets s breathRamp | r cmd7 |
| OSCulato outputs | r | /tptPreset/ \$1 KYMA C | C108 | /breathRamp/ \$1 KYMA udpsend 127.0.0.1 8000 | CC109 |
| | | | | | |

Figure A.1.i.14. Wiimote 2 Development/Climax Section Control Module, part 1



Figure A.1.i.15. Wiimote 2 Development/Climax Section Control Module, part 2
| \varTheta 🔿 🔿 🛛 [Т | ptSelect] |
|---|---|
| Wiimote 2 select T | pt Time Index 0-9 |
| input Wii left | input Wii right |
| value tptPreset -1 if \$i1 <= -1 then 9 e if \$i1 == 10 then 0 e | value tptPreset +1 else \$i1 else \$i1 |
| value tptPreset | tpt preset variable |
| output tpt pres | set |
| 🔺 🕂 🕂 🛠 | |

Figure A.1.i.16. Trumpet Time Index Selection Patcher

| 0 0 | [interpolateBtwPresets] | | |
|---|---|--|--|
| Interpolate between Trumpet Time Index | | | |
| input float | stored variable is banged out first, then stores new value for next interpolation | | |
| bondo 2 bondo ensures proper packing sprintf %.4f %.4f t b l 1.0008, 1.0008 100 line 0. | tored value en new value cmd 7 sets up initial tpt value from Kyma Timeline etTpt value kld construct line message as string set string as message and then bang (line won't take a string) | | |
| output interpo | lated values output bang when done | | |
| A + 🕂 🕅 | | | |

Figure A.1.i.17. Trumpet Time Index Interpolation Patcher

| 😝 🔿 🚫 [BreathRamp] | |
|--|---|
| Wiimote 2 select breath ramp 0-1 | 14 |
| 1 input Will left 2 input Willing | nt |
| value breathRmp value breathRmp | |
| if \$i1 <= -1 then 14 else \$i1 T if \$i1 == 15 then 0 else \$i1 | |
| 0 breath ramp variable | r cmd7 loadmess 0 value breathRmp |
| output selection | |
| 🔺 🕂 🕂 🐺 🖬 🖼 | - |

Figure A.1.i.18. Breath Rate Calculator Selection Patcher

[interpolateBtwPresets]

0 0

| Interpolate between rate of Breaths stored variable is banged out first, then | | | |
|---|--|--|--|
| stores new value for next interpolation | | | |
| t f b bang stored value and then new value r cmd7 cmd7 sets up initial value from Kyma Timeline value presetBreath bondo 2 | | | |
| bondo ensures proper packing 1000. time interpolation value | | | |
| sprintf %.4f %.4f %ld construct line message as string | | | |
| t b l 0.855, 0.885 1000 (line won't take a string) | | | |
| | | | |
| Value presetBreath | | | |
| output interpolated values output bang when done | | | |
| a + + 🛠 🕺 i 🗈 🗉 🕮 | | | |

Figure A.1.i.19. Breath Rate Interpolation Patcher



A.2. VPT_4.1b5_RunningExpressions.maxpat Figure Documentation

Figure A.2.1. VPT_4.1b5_RunningExpressions.maxpat Main Patch Window, in Presentation mode



Figure A.2.2. VPT Main Patch Window, in Patcher mode, part 1



Figure A.2.3. VPT Main Patch Window, in Patcher mode, part 2



Figure A.2.4. VPT Keyboard Shortcuts



Figure A.2.5. Main 'jit.window' and 'jit.gl.render' Control Patcher, part 1



.....ŝ

jt.gl.render mofo

2

output frames per second

Figure A.2.6. Main 'jit.window' and 'jit.gl.render' Control Patcher, part 2

t]

rlb

0.

T

[screencornerpin]

Custom Coordinates of Cornerpins for reposition of jit.gl.render object



Figure A.2.7. Custom Coordinates Control Patcher

| 🖲 😜 💭 👘 | [cursortocorner] | | | |
|--|-----------------------------------|--|--|--|
| Custom Coordinates for mapping jit.gl.render objects | | | | |
| l l | r ocoords | | | |
| p conditionals s condition | conditions open the right gate | | | |
| (| unpack 0. 0. | | | |
| gate 4 | qate 4 | | | |
| | | | | |
| ≙ + ⊕ | | | | |

Figure A.2.8. Custom Coordinates Input Patcher

| 00 | [conditionals] |
|---------------------------------|---|
| Conditional Statements r | un against custom coordinates |
| input coordinate floats | |
| | |
| if \$f1<0. && \$f2>0. then 1 if | \$f1<0. && \$f2<0. then 2 [if \$f1>0. && \$f2<0. then 3] [if \$f1>0. && \$f2>0. then 4 |
| | |
| | |
| | |
| output true integers | |
| | |
| Figure A.2.9. Condit | ional Statement Custom Coordinates Patcher |

A.2.a. Videoplane Module: Running



Figure A.2.a.1. Videoplane Running, in Presentation mode



Figure A.2.a.2. Videoplane Running Patch Window, overview of Window layout



Figure A.2.a.3. Videoplane Position Module



Figure A.2.a.4. Videoplane Color Swatch Module



Figure A.2.a.5. Videoplane Color Masks Patcher



Figure A.2.a.6. Videoplane 'jit.gl.render' Control Module



Figure A.2.a.7. Videoplane Positioning Control Module

| movie masks | |
|---------------------|---|
| gate povar uisat | rui_kill gate pvar uisource pattr source |

Figure A.2.a.8. Videoplane Movie Masks Module

A.2.b. Videoplane Module: Heart Rate



Figure A.2.b.1. Videoplane Heart Rate, in Presentation mode



Figure A.2.b.2. Videoplane Heart Rate, Positioning Control Module, with 'pictslider' object



Figure A.2.b.3. Videoplane Heart Rate, 'jit.gl.render' Control Module



Figure A.2.b.4. Videoplane 3D Positioning Control Module



Figure A.2.b.5. Videoplane Custom Corner Positioning Control Module

A.2.c. Videoplane Module: LCD



Figure A.2.c.1. Videoplane LCD, in Presentation mode

A.2.d. Preset Module



Figure A.2.d.1. Preset Module, in Presentation mode



Figure A.2.d.2. Preset Module, in Patcher mode, part 1



Figure A.2.d.3. Preset Module, in Patcher mode, part 2



Figure A.2.d.4. Preset Module Controls Patcher



Figure A.2.d.5. Preset Module Recall Patcher

| 0 | 0 | [st | tore] | | |
|---------------|----------------|-------------|-----------|-----------|-------------|
| Store | es prese | et of pat | ch par | ameters | 5 |
| 1 | input ban | 2 | input st | ore int | |
| t b pack s | tore 0 | | | | |
| tbbl | | Т | \square | | |
| delay 3 | 300 | | | | |
| 255 0 | 0 2 | 255 255 0 | | | |
| bgcold | or \$1 \$2 \$3 | 3 255 | | | |
| þ | output co | lor confirm | 2 | output pr | reset store |
| | • • | 7 | | | |

Figure A.2.d.6. Preset Module Data Confirmation Patcher

A.2.e. Movie Source Module: Running #1







Figure A.2.e.2. Movie Source Running Patch Window, in Patcher mode

| r folderupdate | Movie Select |
|---|--------------|
| sprintf symout %s/video/ | |
| t b s | Т |
| "Macintosh HD:/Users/jpbellona/Documents/ mmer 2010/Thesis/MaxMSP_Thesis/VF _LCD//video/" | Su PT |
| prepend prefix | |
| 6 | st \$1 |
| r movSelect qt #1 | |
| autopopulate 1 | |
| (rmovieRunningSelect) | |
| 0_HRvid320x240letterbox_Red.mov | |
| value movieRunning preper | td read |
| s bangum s mov | ieFileName |

Figure A.2.e.3. Movie Source Select Module



Figure A.2.e.4. Movie Source External Select Control Module

| Video Control VARIABLES P VideoControl | | | |
|---|--|--|--|
| r value_time rvalue_total 0. 600.60058 pipe 0. 10 value total_t value time | _time rvalue_timescale | value_framecount value_fps | |
| (r totalCount) | rsectionMeters | rsectionQTmulitplier | |
| ▶ <u>0.</u> | ▶0. Total Meters | 0. | |
| p interpolateNextMovieTime | expr \$f1 * \$f2 | | |
| 0. | 0. | will change per section | |
| stime | s value_time | | |
| the expression will change b dividing the distance by the actualize the synchronous p for "SidewalkRunTest.mov" of | ased upon the movie t frames to come up with layback speed use 170.420319 | hat is playing because you are a step count that will | |

Figure A.2.e.5. Movie Source Video Control Variables Module

[VideoControl]



Figure A.2.e.6. Movie Source Variables Assignment Patcher

0 0

l

Interpolate Time Values Based Upon Foot Step Interval

| del 5 value time last time value 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. | <pre>terpolate time value string</pre> |
|--|--|
| output interpolated time | value |

Figure A.2.e.7. Movie Source Video Position Interpolation Calculator Patcher



Figure A.2.e.8. Movie Source Position Interpolation Timer Patcher



Figure A.2.e.9. Movie Control Module



Interpolate between constantly changing 'srcrect' pixel X-axis values



Figure A.2.e.10. Interpolation for 'srcrect' X-Axis Jitter Patcher

Interpolate between constantly changing 'srcrect' pixel Y-axis values



Figure A.2.e.11. Interpolation for 'srcrect' Y-Axis Jitter Patcher

A.2.f. Movie Source Module: Running #2



Figure A.2.f.1. Movie Source Running #2 Patch Window, in Patcher mode



Figure A.2.f.2. Wiimote 1 Controls Panning Video Patcher



Figure A.2.f.3. Wiimote 1 Controls Kyma 8-channel Panning Patcher

A.2.g. Movie Source Module: Heart Rate LCD display



Figure A.2.g.1. Movie Source Heart Rate LCD Display, in Patcher mode

A.2.h. Movie Source Module: Heartbeat Movie



Figure A.2.h.1. Movie Source Heartbeat Patch Window, in Presentation Mode



Figure A.2.h.2. Movie Source Heartbeat Patch Window, in Patcher Mode

| r folderupdate L sprintf symout %s/video/ | Movie Select |
|--|-----------------|
| tbs | |
| T | T |
| "Macintosh HD://Jsers/jpbellona/Documer mmer 2010/Thesis/MaxMSP_Thesis _LCD//video/" | nts/Su ;/VPT |
| r movdest | |
| r movSelect | list \$1 |
| autonopulato 1 | 1 |
| | qt #1 |
| r movieHeartSelect | |
| _white.mov | |
| PR PR | epend read |
| s movHeartBU | movHeartName |

Figure A.2.h.3. Movie Select Heartbeat Module



Figure A.2.h.4. Movie Heartbeat Video Control Variables Module



Figure A.2.h.5. Movie Control Heartbeat Module

A.2.i. Movie Source Module: Distance LCD display

| ● ○ ○ | o moviesource_distance | | | |
|--|--|--|--|--|
| VPT Distance LCD Display LCD display to be used with Video Projection Tools as a moviesource. | | | | |
| (cTonoffi (thispatcher) (ctonoffi (atr movon) (atr mov | Takes distance generated by feet accelerometers, and displays on a videoplane as a jit.lcd i b font information, bg color and text placement inside matix write 35 font "Andale Mono" 25, textface bold, brgb 0 0 0, color 0, clear, moveto 10 28 Andale Mono pak font genera 14 prepend textface | | | |

Figure A.2.i.1. Movie Source LCD Patch Window

A.2.j. Cue List Mixer Module



Figure A.2.j.1. Cue List Mixer Patch Window, in Presentation mode

| 00 | o cuelistmi | xer_run |
|-----------------|---------------|-----------------------------------|
| VPT Main M | lovie Mixer | Controls Xfades between movies |
| Click to write: | Click to read | Ŀ |
| t b b | t b read | thispatcher mode outmode 1 |
| coll mycoll | | A B |
| | | A B |
| | | m m |
| | | |

Figure A.2.j.2. Cue List Mixer Patch Window, in Patcher mode

A.2.k. Mixer Module: Running

| 0 0 | o mixer_module_heart |
|-----------------------------------|--|
| Mixer Heart Mov | Mixer Xfader for Heartbeat Movie |
| rhrm_movieMixerFader xfade \$1 | pattr A A prepend set prepend set prepend set prepend set prepend set prepend set |
| jit.xfade | |
| 🔺 🕂 🕂 🗍 | |

Figure A.2.k.1. Heartbeat Movie Mixer Patch Window

A.2.1. Mixer Module: Heart

| n 🧿 🔴 🔴 | mixer_module_run |
|----------------------|---|
| Mixer Run Movie | Mixer Xfader for Running Movies |
| rrun_movieMixerFader | A pattr B pattr out out out prepend set prepend set prepend set |
| jit.xfade | |
| 🔺 + 🕂 🏹 i 🗳 | |

Figure A.2.1.1. Running Movie Mixer Patch Window

A.3. Master's Project Proposal

Terminal Creative Project Proposal M.Mus. in IMT, University of Oregon Jon Bellona

Project Objective:

For my terminal creative project in Intermedia Music Technology, I will compose a musical work translating the physical and physiological experience of running into musical performance. The piece will explore the creative potentials afforded by data gathered from physiological monitors and digital motion sensors and mapped to control audio files and musical parameters in real time.

Introduction: Running and Art

Endurance running as realized in sport and the aesthetic principles of Western art and music both have roots in ancient Greece. Greek writing influenced the Western concepts of intervals and views on musical affectation.¹⁸ Styles of European sculpture and architecture copied Greek forms. Ancient Greek scripture helped lay the foundation for Western literature.¹⁹

The origins of modern endurance running also began in Greece. The legend of Pheidippides, who ran 26 miles²⁰ to Athens to announce the Greek victory at Marathon, inspired the modern marathon race. Today there are thousands of 26.2 mile marathons across the world with millions of participants. Modern endurance running has spawned its own culture, complete with its own language, literature, and aesthetic. With running and Western music principles sharing historical roots I proposed the question, "How can running shape the music we create?"

Digital technologies provide a vehicle for answering this question. Advances in music technology have lead to the development of new electronic instruments, new compositional tools, and new styles. Composers using digital technology have at their disposal tools that both facilitate and inform creative decisions in their pursuit towards art. Integration of new technologies has changed how composers both think about and compose music. Through digital data selection, acquisition, modification, and mapping to create and control music, composers now shape streams of data into musical journeys.

The application of technology in running, specifically digital monitoring systems used in research on the human body, reveal that the body produces constant data. Heart rate, body temperature, oxygen levels, neurotransmitters, brainwaves: all of these internal processes can be recorded and stored. The body's physical movement may also be captured using sensors measuring distance, time, and acceleration. Both external movement and internal body data may

¹⁸ J. Peter Burkholder, Donald Grout, Claude Palisca, eds., *A History of Western Music*, 7th ed. (New York, NY: W.W.Norton, 2006).

¹⁹ H. James, Sarah Lawall, Lee Patterson, eds., *The Norton Anthology of Western Literature, Volume 1* (New York, NY: W.W.Norton, 2006).

²⁰ Primary evidence of historian Herodotus suggests that Pheidippides actually ran 145 miles to Sparta and back requesting troops for the famous 490 B.C.E. battle against the Persians.
be mapped to control non-physiological variables, like musical parameters. The experience of running can thus be recorded, modified, and digitally mapped to create music.

Through technology, running and music have an opportunity to merge. Yet, this time, the relationship between music and running will not be its historical influence, but rather will be a modern alliance, creating music from data emitted by the human body. The exploration into the musical possibilities of physiological monitors and digital motion sensors provide an excellent avenue with which to compose new electronic music.

Outline of Project Proposal

I will use physiological monitors and digital motion sensors to translate the human activity of running into a musical performance. Building upon histories of gestural performance and parametric musical relationships, I will collect data of the physiological status of a runner in real time and map this data to create an original composition to be performed live. The compositional process will involve several distinct steps.

First, I will acquire data from the human body in real time. The word 'acquire' here means to track the internal and external components of the body as streams of data and transfer this information into the computer. The streams of 'human body' data I am specifically interested in acquiring relate to running: heart rate, arm swing acceleration, foot cadence, pace, and relative distance. To this end, I will investigate various physiological monitors, digital motion sensors, and preexisting digital controllers as they relate to the human body in motion. These monitors and controllers will include heart rate monitors, foot pods, oximeters, FSR sensors, and Nintendo Wii controllers.

Next, I will research and implement various transmission protocols in order to transmit the data into the computer for musical mapping. These transmission protocols will include, but are not limited to, the ANT+ wireless protocol, RF transmission, RS-232 serial transmission, Bluetooth and the OSC protocols. My research on transmission protocols will influence the computer software I will adopt for polling the data.

Once the streams of human body data are inside the computer, I will explore the various ways I can modify the data in programming environments learned during the course of my studies at the University of Oregon. Several software/hardware systems I will integrate for the composition and performance include Max/MSP/Jitter, Processing, and Kyma. Implementing and combining these programming environments will be important in the final execution of the piece. Max/MSP/Jitter will be the central data hub, modifying the data in various ways, and routing the information to and from Processing, to and from Kyma. Processing will draw supporting visuals, making the performance a multimedia experience. Kyma will map the data to control various parameters of the digital sound processing of real-time audio. Kyma will also serve as the audio generator and mixer, outputting the sounds of the composition for an 8-channel loudspeaker performance.

My creative terminal project will permit me to harness new, specific data streams in order to explore their creative use, and will also allow me to explore the expositions of gestural performance through a preexisting language: the perceptual and cultural language of running. Because I hope to articulate the journey of a run through the use of physiological monitors and digital motion sensors, I will study the design trends of new digital musical instruments in order to learn more about digital mapping strategies and performance practices.

My creative terminal project will attempt to creatively combine my passions of running and technology and will be the culmination of my coursework and studies here at the University of Oregon. The project will employ digital sensors and protocols I first uncovered during my graduate studies. The composition and performance will utilize programming languages and graphic environments that I learned during the program's coursework. My terminal project would not have been possible before coming to Oregon, and it is the hope that the work will display the breadth of my technical, creative, and performance skills polished through the Intermedia Music Technology program.

In addition to the creation of a final composition and performance, I will document the compositional process, recording research on physiological monitors, transmission protocols, mapping strategies, and compositional methods used and explored. I will compile the various processes of my project into a small portfolio. This portfolio will include descriptions of mapping strategies used, an annotated list of hardware equipment, software, and data protocols considered, and layouts of original programs created with Max/MSP/Jitter and Processing. I will also capture audio and video recordings of the final performance. This documentation will supplement my creative terminal project and will serve as a resource for anyone wishing to explore the creative applications afforded by these tools. The creative terminal project also lays a working foundation, as after the completion of this project and my Masters degree, I hope to continue composing using these tools, strategies, and techniques.

Hardware





Rate Monitor

Heart Rate Monitor Interface (HRMI)





Nintendo Wiimote

Dual-Axis Accelerometer + JeeNode Tx

JeeNode Rx USB bub



MacBook Pro



-61

Paca(rana)



Figure A.4.1. Hardware icons used throughout the documentation

Connections/Protocols

😵 Bluetooth

Bluetooth



Universal Serial Bus (USB)



Firewire 800



Ethernet



VGA



Open Sound Control (OSC)



Musical Instrument Digial Interface (MIDI)

Figure A.4.2. Connection Standards and Protocols icons

Software



Max/MSP/Jitter



Kyma



OSCulator



Processing



PacaConnect

Figure A.4.3. Software icons

A.5. Resource URLs

ANT+ wireless: http://www.thisisant.com/pages/technology/what-is-ant

CNMAT (OSC-route, randdist) Max objects: http://cnmat.berkeley.edu/downloads

HC Gilje's Video Projection Tools: http://hcgilje.wordpress.com/resources/video-

projection-tools/

JeeNode v.4: http://jeelabs.net/projects/hardware/wiki/JN4

Max/MSP/Jitter: http://cycling74.com/products/maxmspjitter/

MaxLink: http://jklabs.net/maxlink/

OSCulator: http://www.osculator.net/

OSC: http://opensoundcontrol.org/introduction-osc

PacaConnect: http://www.delora.com/delora_products/pacaconnect/pacaconnect.html

Polar product - HRM: <u>http://www.polarusa.com/us-en/support/downloads?</u>

product=&category=User+manuals&document=/gip/PEUS1kb-public.nsf/web_cat/

85256F470048B0BC8525747300610169

Processing: http://processing.org/

SparkFun - ADXL322 dual-axis accelerometer: http://www.sparkfun.com/products/849

SparkFun - HRMI: http://www.sparkfun.com/products/8661

Symbolic Sound (Kyma): http://www.symbolicsound.com/

A.6. Included DVD Contents

- a. Running Expressions .pdf Documentation
- e. External Libraries
 - i. CNMAT objects (OSC-route, randdist)
 - ii. MaxLink (version 0.36)
- c. Video documentation: Studio 74 performance, April 15, 2011
- d. Stereo and Eight-channel Audio documentation: Studio 74 performance, April 15, 2011
- b. Performance files
 - i. Kyma Files (version 6.79)

ii. Max/MSP/Jitter Patches (version 5)

iii. OSCulator File (version 2.10.6.2)

iv. Processing sketch (version 1.1)

- f. Template Max Patches
 - i. Max/MSP/Jitter template patch for JeeNode & Accelerometers

ii. Max/MSP/Jitter template patches for Video Projection Tools