# 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

# 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

# TABLE OF FIGURES

# 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 bio-signal, 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– the performer serves as an accessible gateway to the 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.

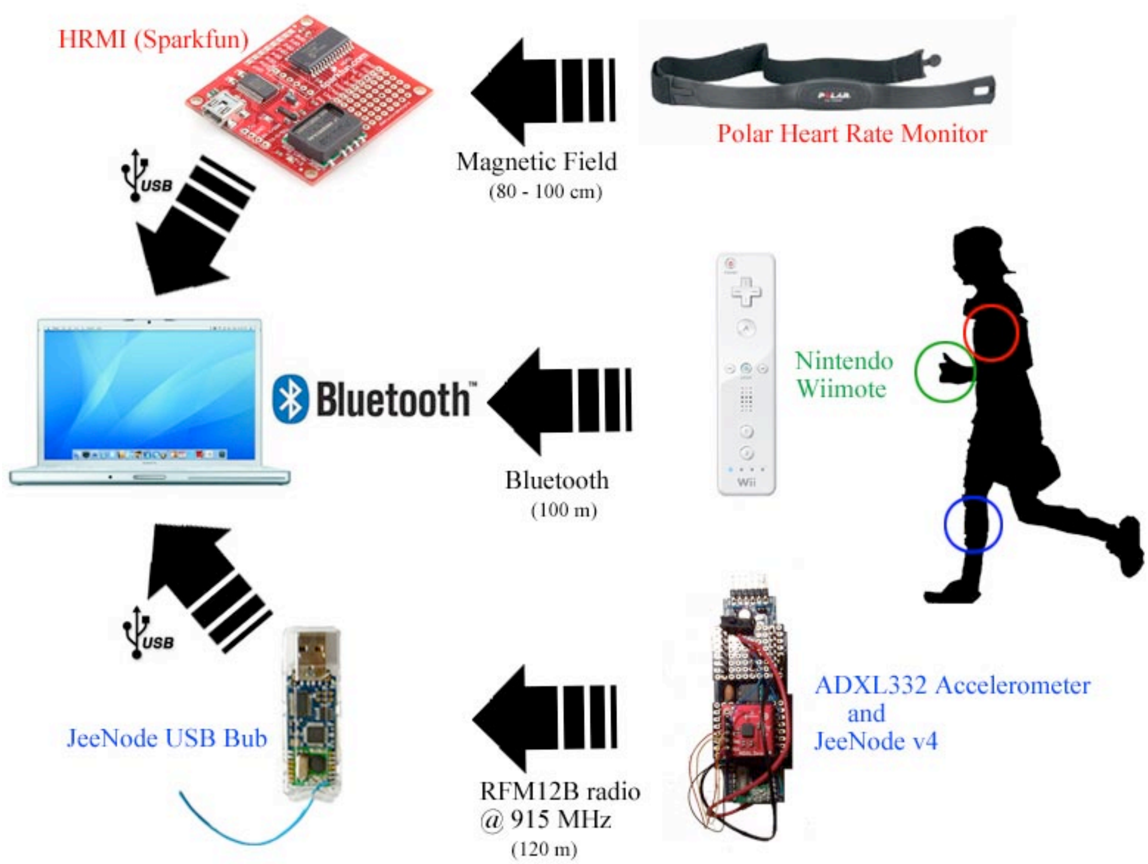



Figure 1.1. Hardware Connections Flowchart

[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).

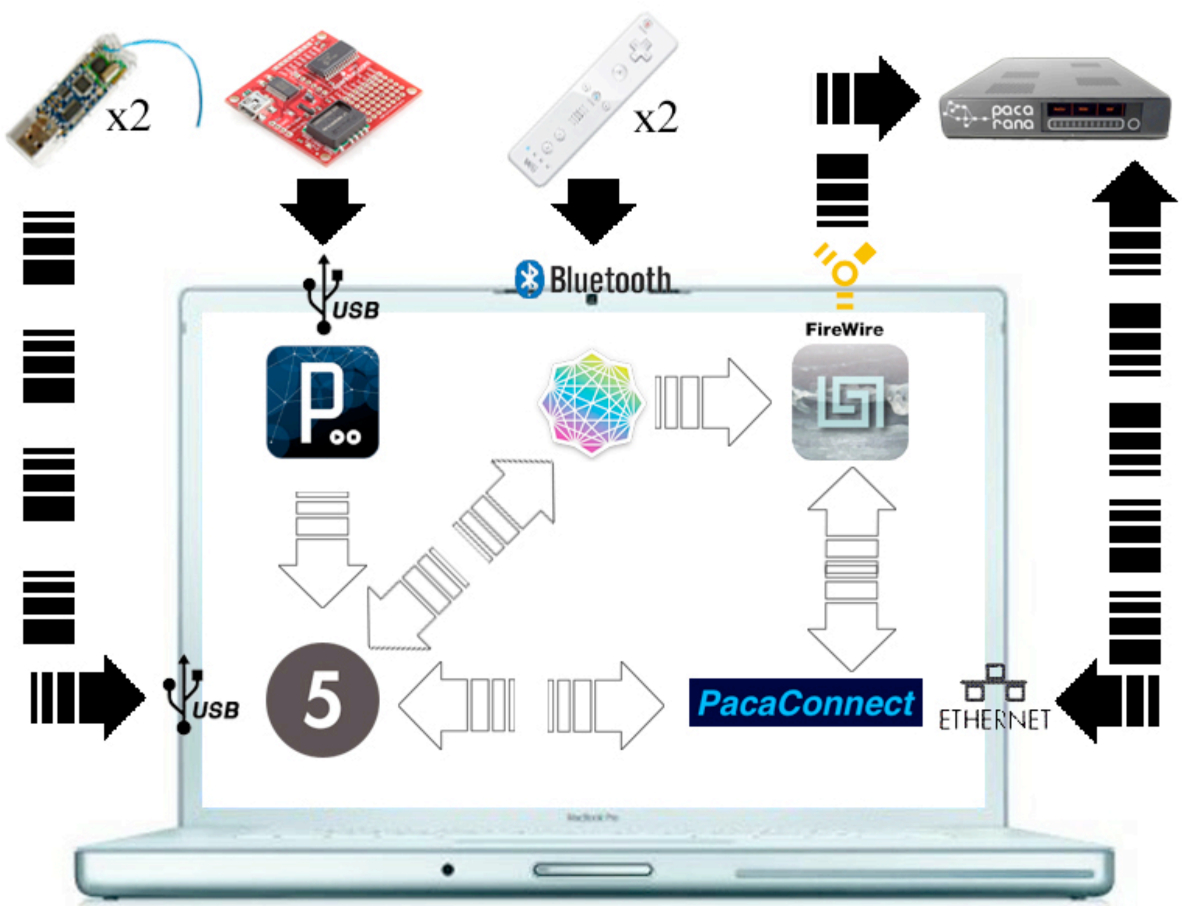



Figure 2.1. Software Connections Flowchart, includes connections to external devices.

## 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.

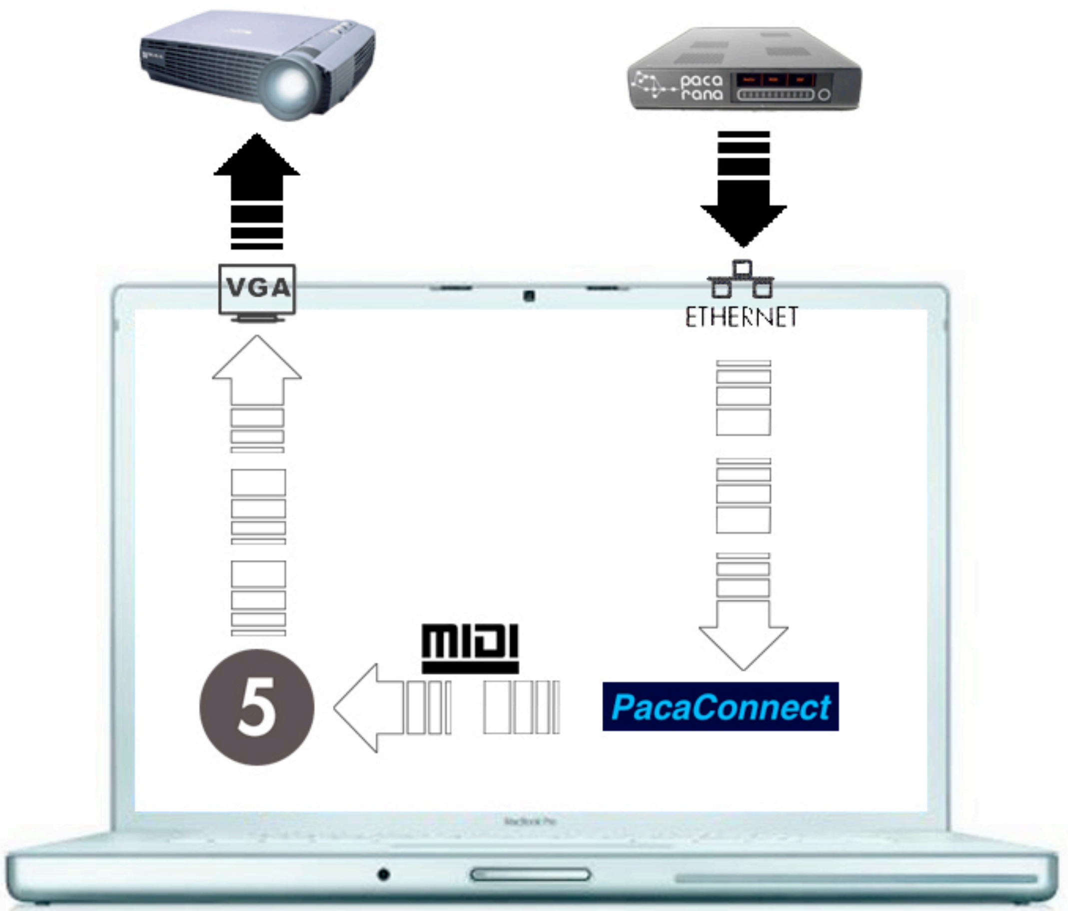



Figure 3.1. Video Connections Flowchart. Compiled Kyma Timeline on Paca(rana) sends MIDI messages via PacaConnect that serve as video controls within Max/MSP/Jitter.

# 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 DanJulioDesigns[4], 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," http://danjuliodesigns.com/sparkfun/sparkfun.html (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 computer[6]. 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

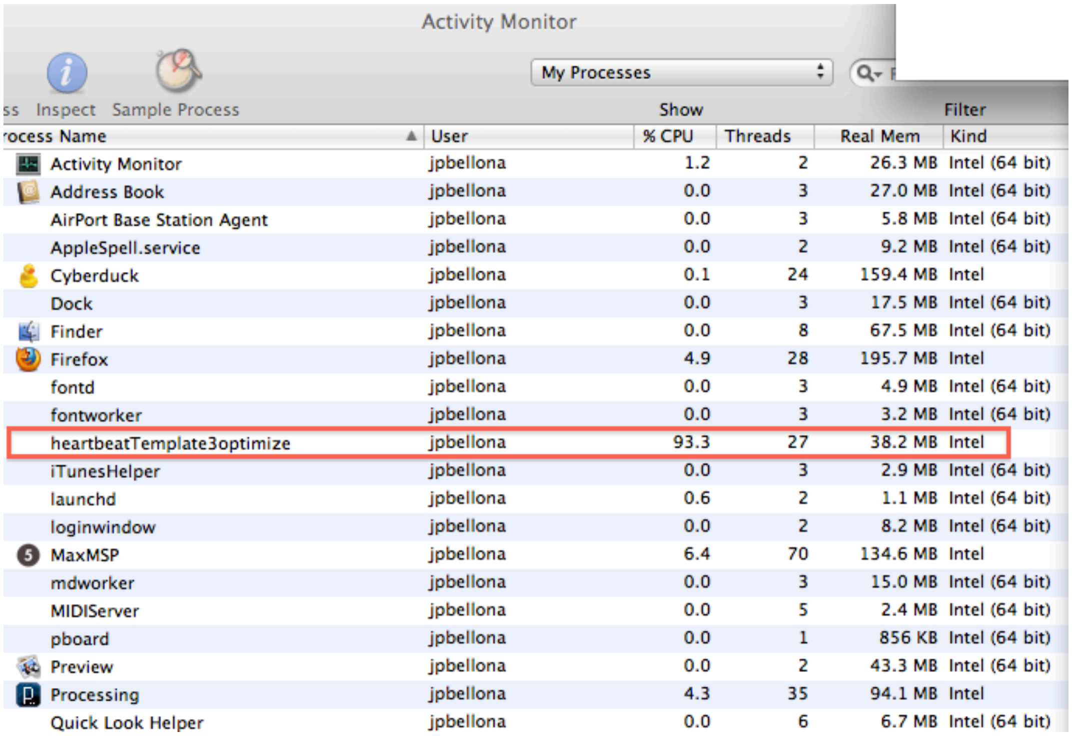
Processing acts as the master program that controls the HRMI microprocessor. Processing sends commands to the HRMI every 100ms to retrieve heart rate information stored in the microprocessor buffer. These data packets are sent in the form of ASCII values, and are comprised of a status byte followed by 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.

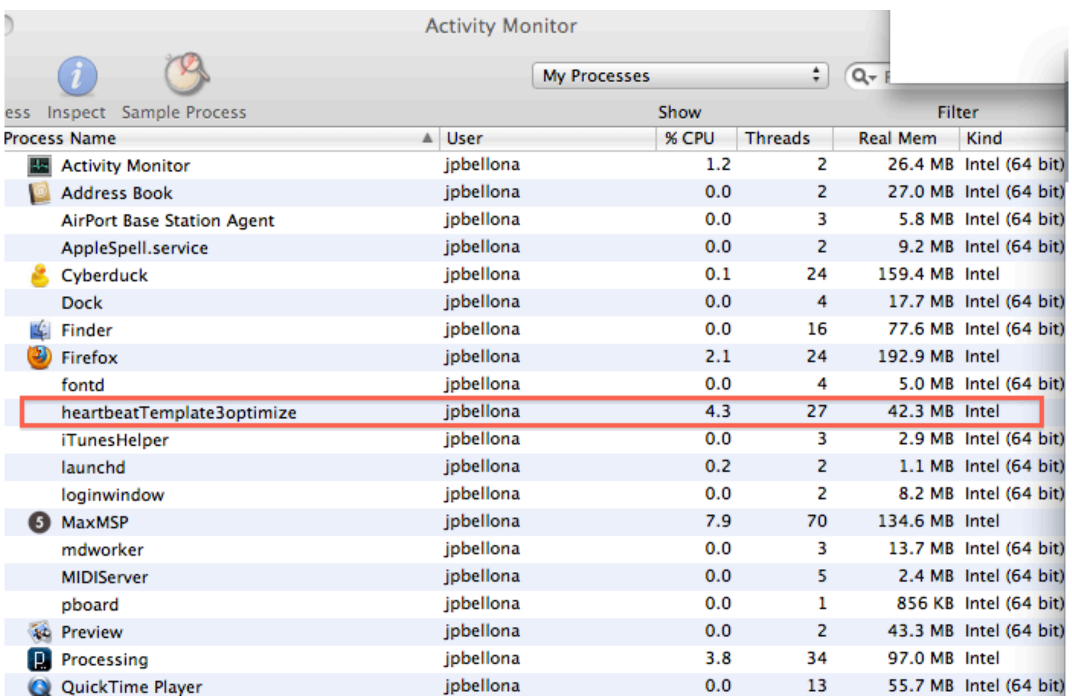
[10] MaxLink, http://jklabs.net/maxlink/ (accessed April 21, 2011).



Activity Monitor

| Process Name | User | % CPU | Threads | Real Mem | Kind |
|---|---|---|---|---|---|
| Activity Monitor | jpbellona | 1.2 | 2 | 26.3 MB | Intel (64 bit) |
| 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) |
| Cyberduck | jpbellona | 0.1 | 24 | 159.4 MB | Intel |
| Dock | jpbellona | 0.0 | 3 | 17.5 MB | Intel (64 bit) |
| Finder | jpbellona | 0.0 | 8 | 67.5 MB | Intel (64 bit) |
| 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) |
| 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) |
| Preview | jpbellona | 0.0 | 2 | 43.3 MB | Intel (64 bit) |
| 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)



Activity Monitor

| Process Name | User | % CPU | Threads | Real Mem | Kind |
|---|---|---|---|---|---|
| Activity Monitor | jpbellona | 1.2 | 2 | 26.4 MB | Intel (64 bit) |
| Address Book | jpbellona | 0.0 | 2 | 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) |
| Cyberduck | jpbellona | 0.1 | 24 | 159.4 MB | Intel |
| Dock | jpbellona | 0.0 | 4 | 17.7 MB | Intel (64 bit) |
| Finder | jpbellona | 0.0 | 16 | 77.6 MB | Intel (64 bit) |
| Firefox | jpbellona | 2.1 | 24 | 192.9 MB | Intel |
| fontd | jpbellona | 0.0 | 4 | 5.0 MB | Intel (64 bit) |
| heartbeatTemplate3optimize | jpbellona | 4.3 | 27 | 42.3 MB | Intel |
| iTunesHelper | jpbellona | 0.0 | 3 | 2.9 MB | Intel (64 bit) |
| launchd | jpbellona | 0.2 | 2 | 1.1 MB | Intel (64 bit) |
| loginwindow | jpbellona | 0.0 | 2 | 8.2 MB | Intel (64 bit) |
| MaxMSP | jpbellona | 7.9 | 70 | 134.6 MB | Intel |
| mdworker | jpbellona | 0.0 | 3 | 13.7 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) |
| Preview | jpbellona | 0.0 | 2 | 43.3 MB | Intel (64 bit) |
| Processing | jpbellona | 3.8 | 34 | 97.0 MB | Intel |
| QuickTime Player | jpbellona | 0.0 | 13 | 55.7 MB | Intel (64 bit) |

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

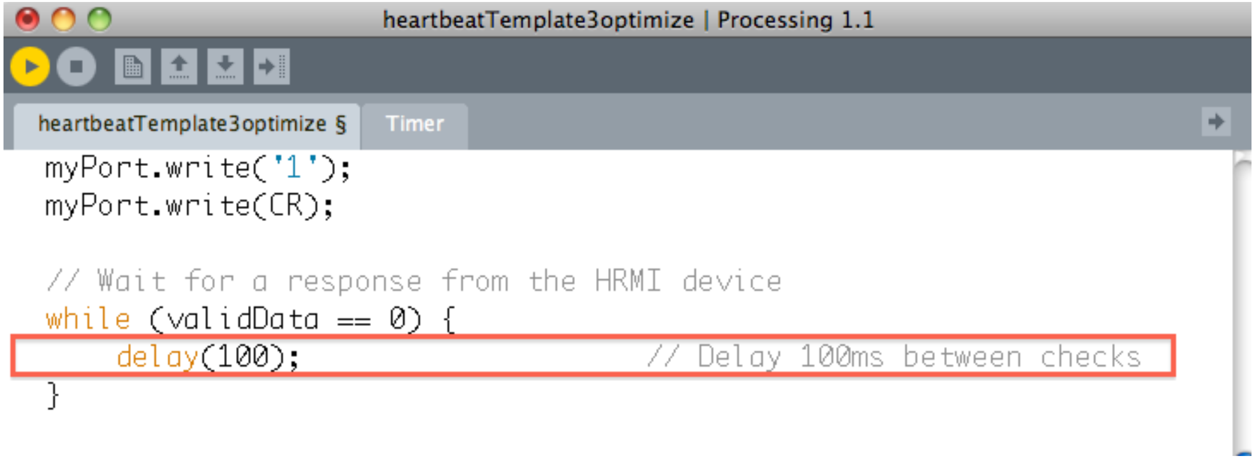



Figure 9.3. Additional Processing code, limits time between data requests

## 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, http://opensoundcontrol.org/spec-1_0 (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]

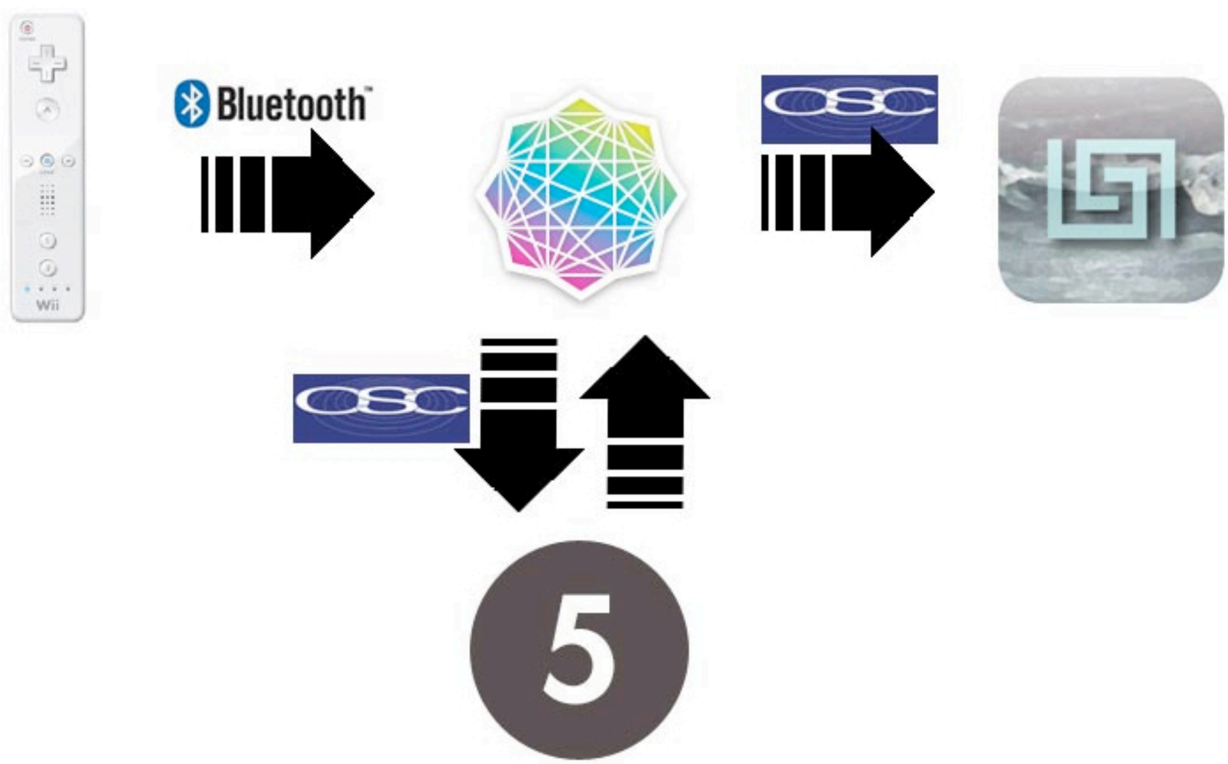
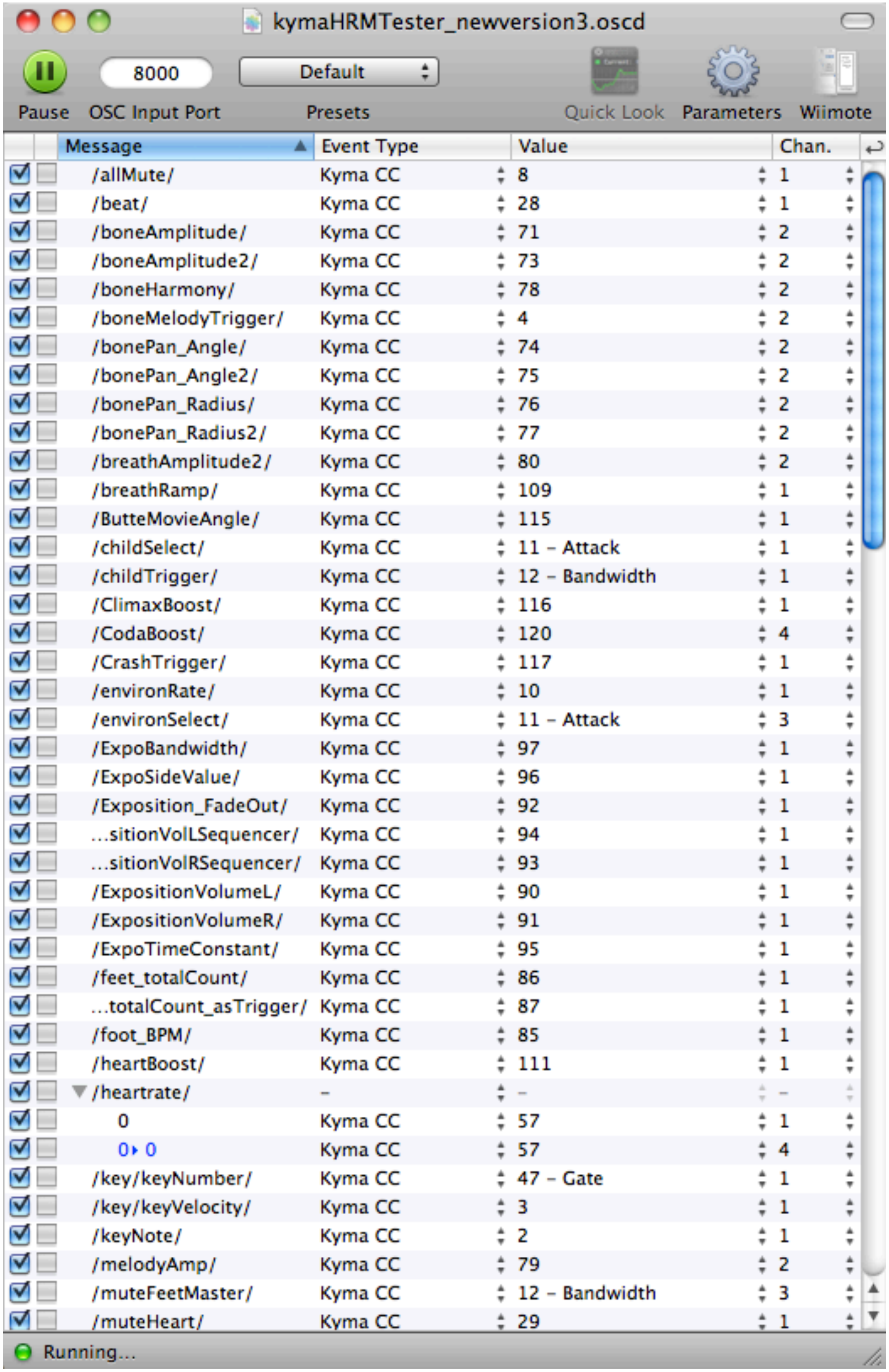



Figure 10.1. OSCulator Signal Flowchart

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




| Message | Event Type | Value | Chan. |
|---|---|---|---|
| /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 |
| /boneMelodyTrigger/ | Kyma CC | 4 | 2 |
| /bonePan_Angle/ | Kyma CC | 74 | 2 |
| /bonePan_Angle2/ | Kyma CC | 75 | 2 |
| /bonePan_Radius/ | Kyma CC | 76 | 2 |
| /bonePan_Radius2/ | Kyma CC | 77 | 2 |
| /breathAmplitude2/ | Kyma CC | 80 | 2 |
| /breathRamp/ | Kyma CC | 109 | 1 |
| /ButteMovieAngle/ | Kyma CC | 115 | 1 |
| /childSelect/ | Kyma CC | 11 – Attack | 1 |
| /childTrigger/ | Kyma CC | 12 – Bandwidth | 1 |
| /ClimaxBoost/ | Kyma CC | 116 | 1 |
| /CodaBoost/ | Kyma CC | 120 | 4 |
| /CrashTrigger/ | Kyma CC | 117 | 1 |
| /environRate/ | Kyma CC | 10 | 1 |
| /environSelect/ | Kyma CC | 11 – Attack | 3 |
| /ExpoBandwidth/ | Kyma CC | 97 | 1 |
| /ExpoSideValue/ | Kyma CC | 96 | 1 |
| /Exposition_FadeOut/ | Kyma CC | 92 | 1 |
| ...sitionVolLSequencer/ | Kyma CC | 94 | 1 |
| ...sitionVolRSequencer/ | Kyma CC | 93 | 1 |
| /ExpositionVolumeL/ | Kyma CC | 90 | 1 |
| /ExpositionVolumeR/ | Kyma CC | 91 | 1 |
| /ExpoTimeConstant/ | Kyma CC | 95 | 1 |
| /feet_totalCount/ | Kyma CC | 86 | 1 |
| ...totalCount_asTrigger/ | Kyma CC | 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 |
| /key/keyVelocity/ | Kyma CC | 3 | 1 |
| /keyNote/ | Kyma CC | 2 | 1 |
| /melodyAmp/ | Kyma CC | 79 | 2 |
| /muteFeetMaster/ | Kyma CC | 12 – Bandwidth | 3 |
| /muteHeart/ | Kyma CC | 29 | 1 |

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

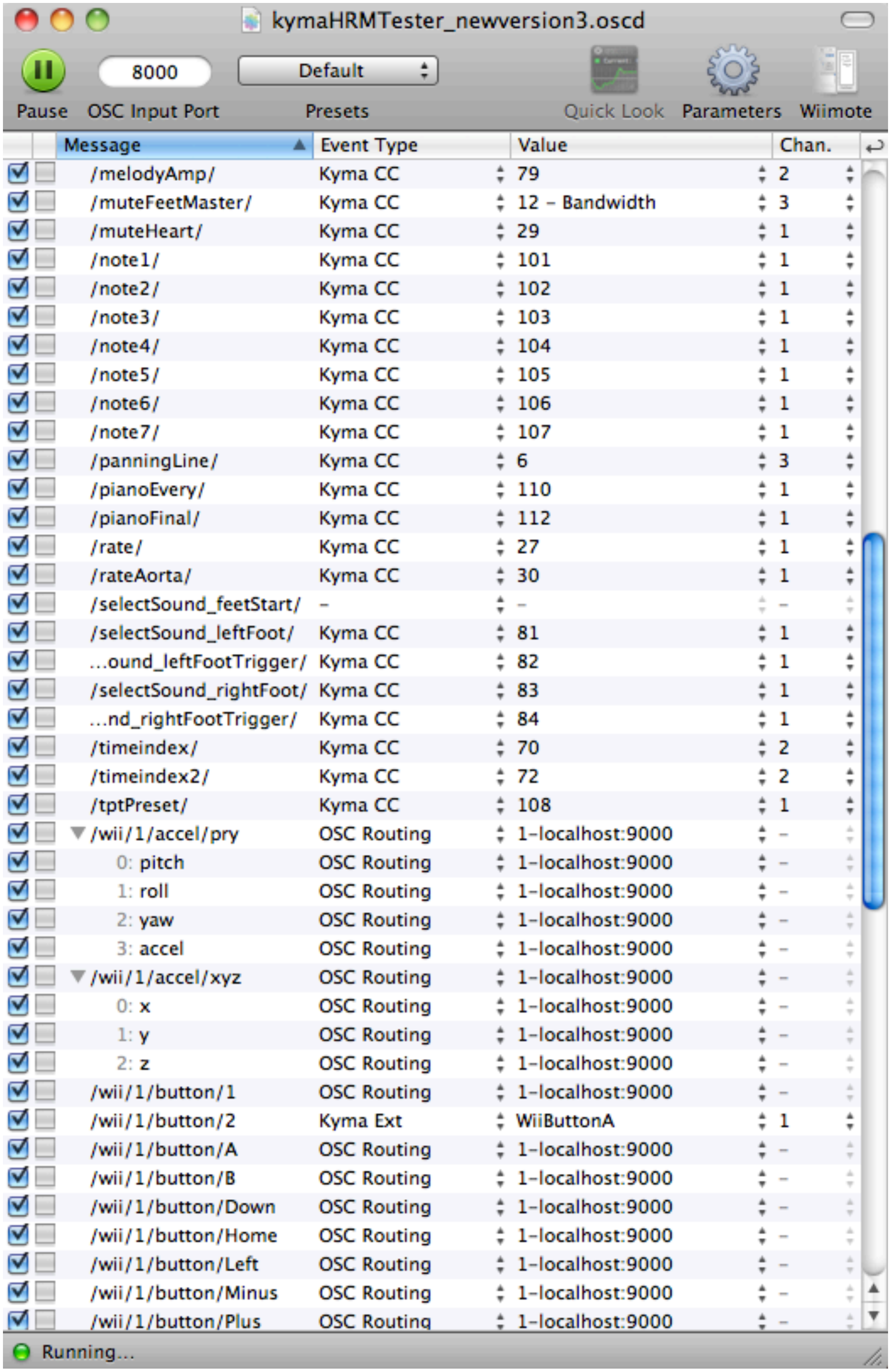



Figure 10.3. OSCulator, part 2. OSC messages received from Max/MSP/Jitter, routed to Kyma. Wiimote messages received, routed to Max/MSP/Jitter.

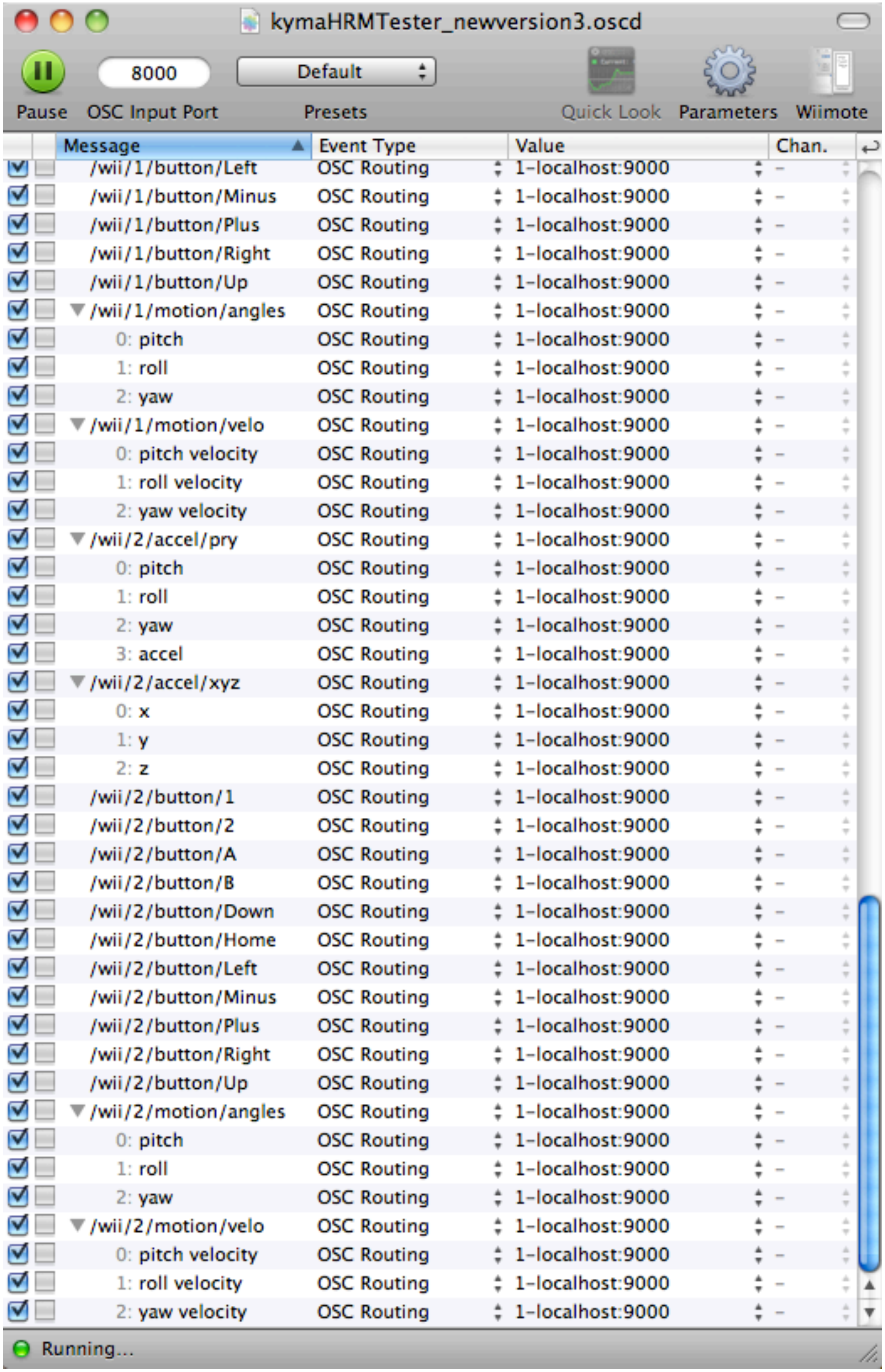



Figure 10.4. OSCulator, part 3. Messages received from Wiimotes, routed to Max/MSP/Jitter.

## 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.

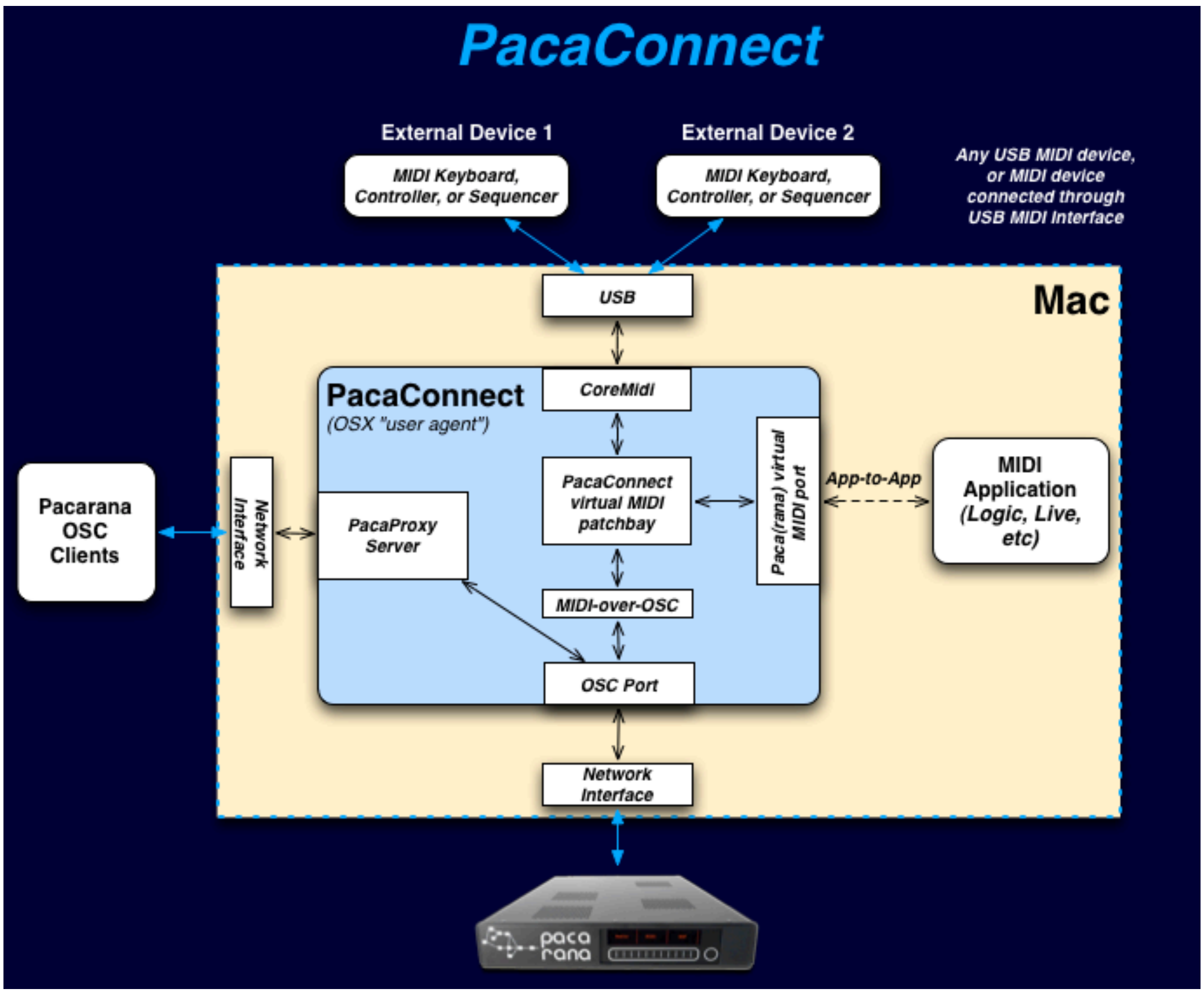



Figure 11.1. PacaConnect Signal Flowchart

---

[13] PacaConnect, http://www.delora.com/delora_products/pacaconnect/pacaconnect.html (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 http://jklabs.net/maxlink/

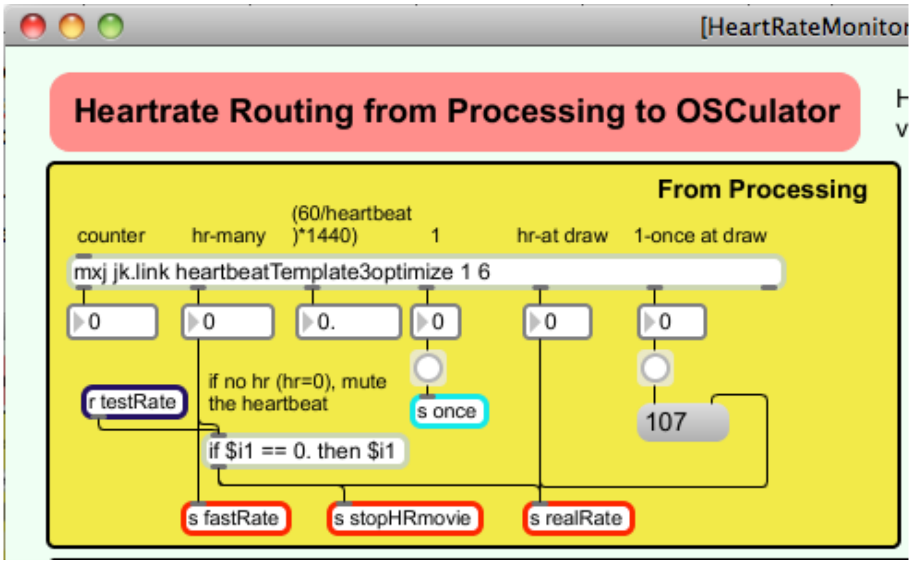



Figure 12.a.1. MaxLink external object inside of *Running Expressions* Max/MSP/Jitter patch.

## 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 http://cnmat.berkeley.edu/downloads

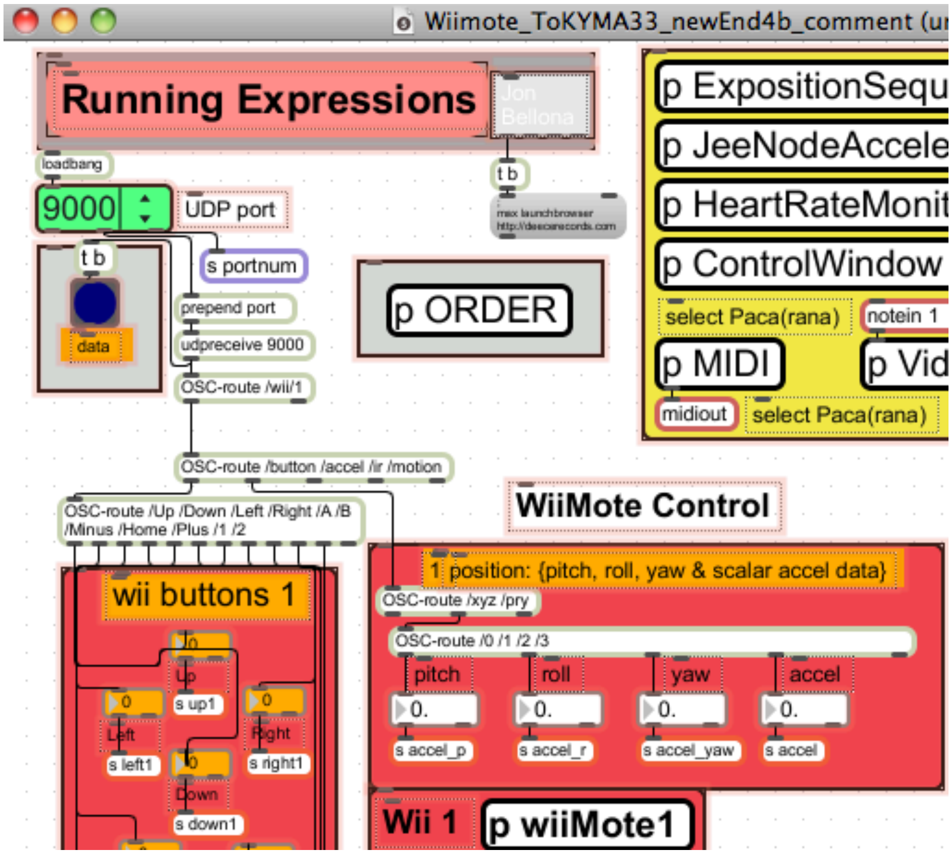



Figure 12.a.2. OSC-route external object inside *Running Expressions* Max/MSP/Jitter patch.

## 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.

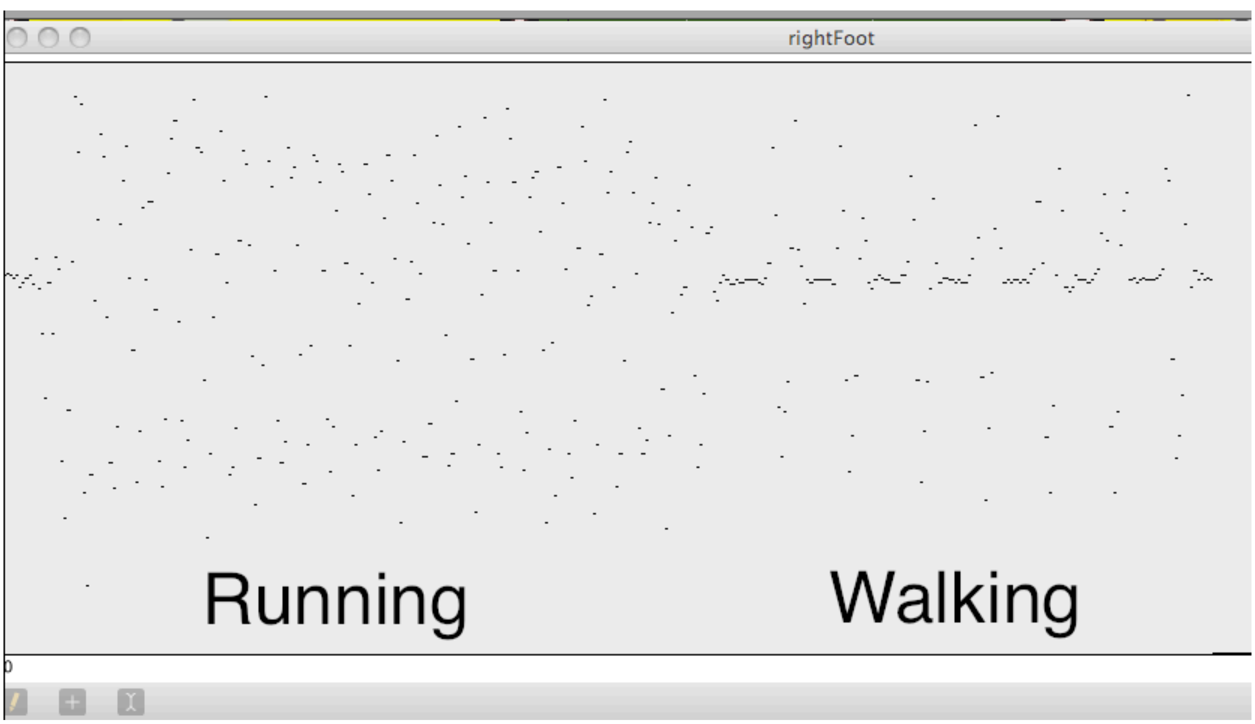



Figure 12.a.4. Data stream table of the right leg accelerometer. Maximum values show motion of the leg downward, and the minimum values show motion of the leg upward.

## 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*, http://hcgilje.wordpress.com/resources/video-projection-tools/ (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.l.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.

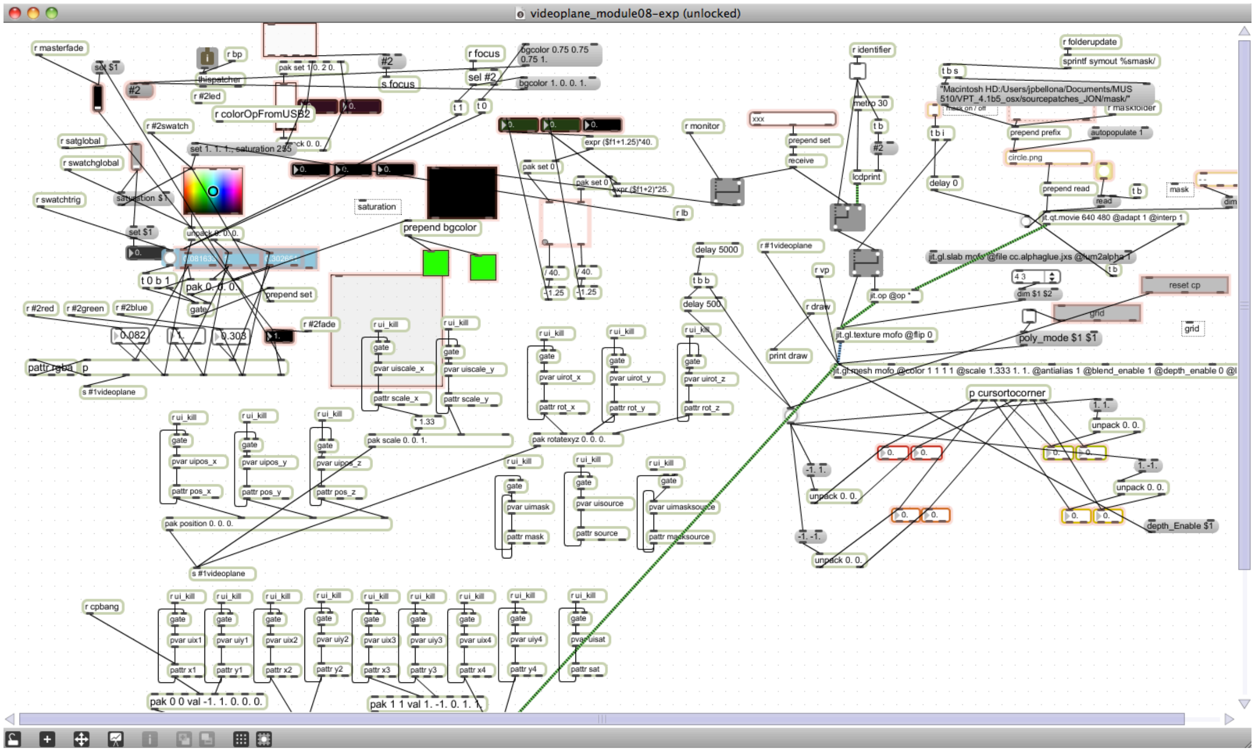



Figure 12.c.2. Videoplane cleanup documentation. HC Gilje's patch in Editing mode.

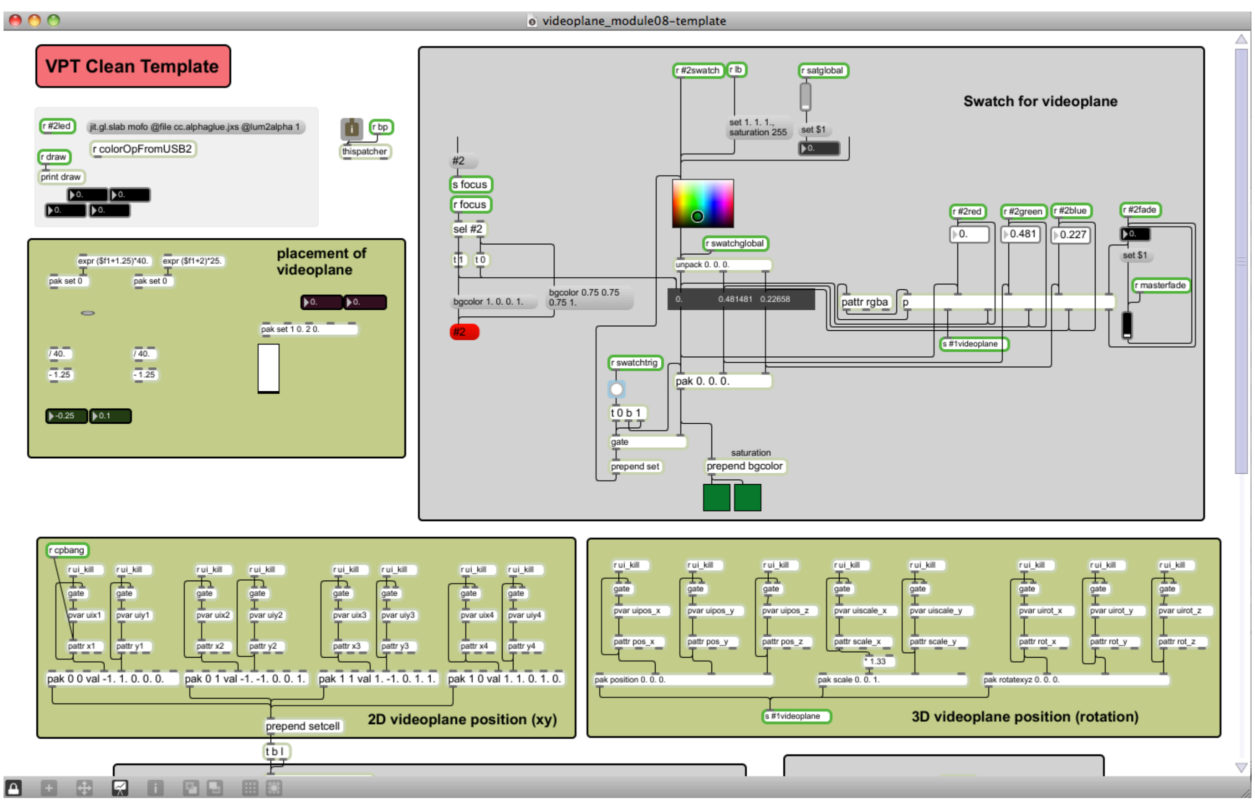



Figure 12.c.3. Videoplane cleanup documentation. Cleaned-up patch in Editing mode.


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

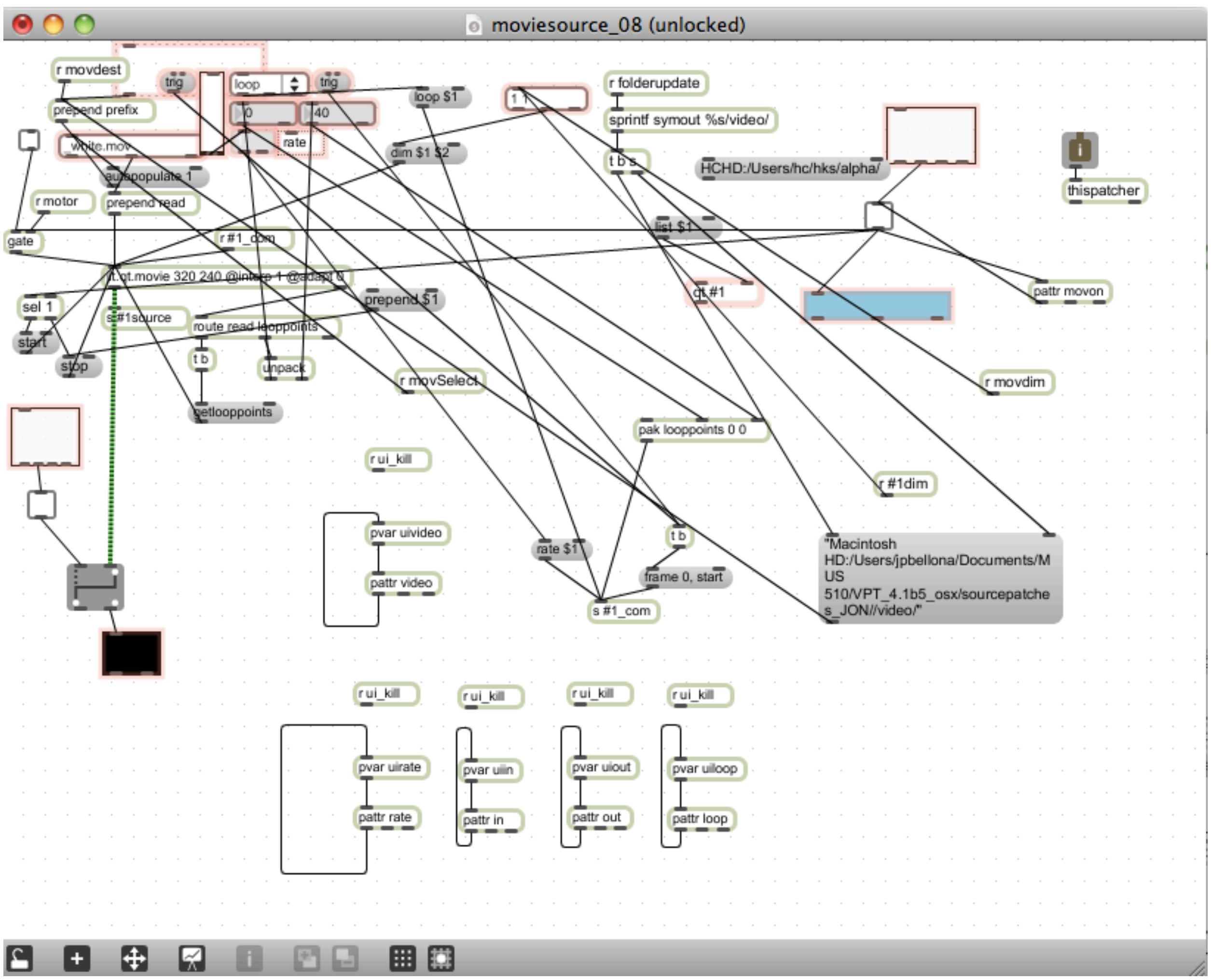



Figure 12.c.5. Movie source cleanup documentation. HC Gilje's patch in Editing mode.

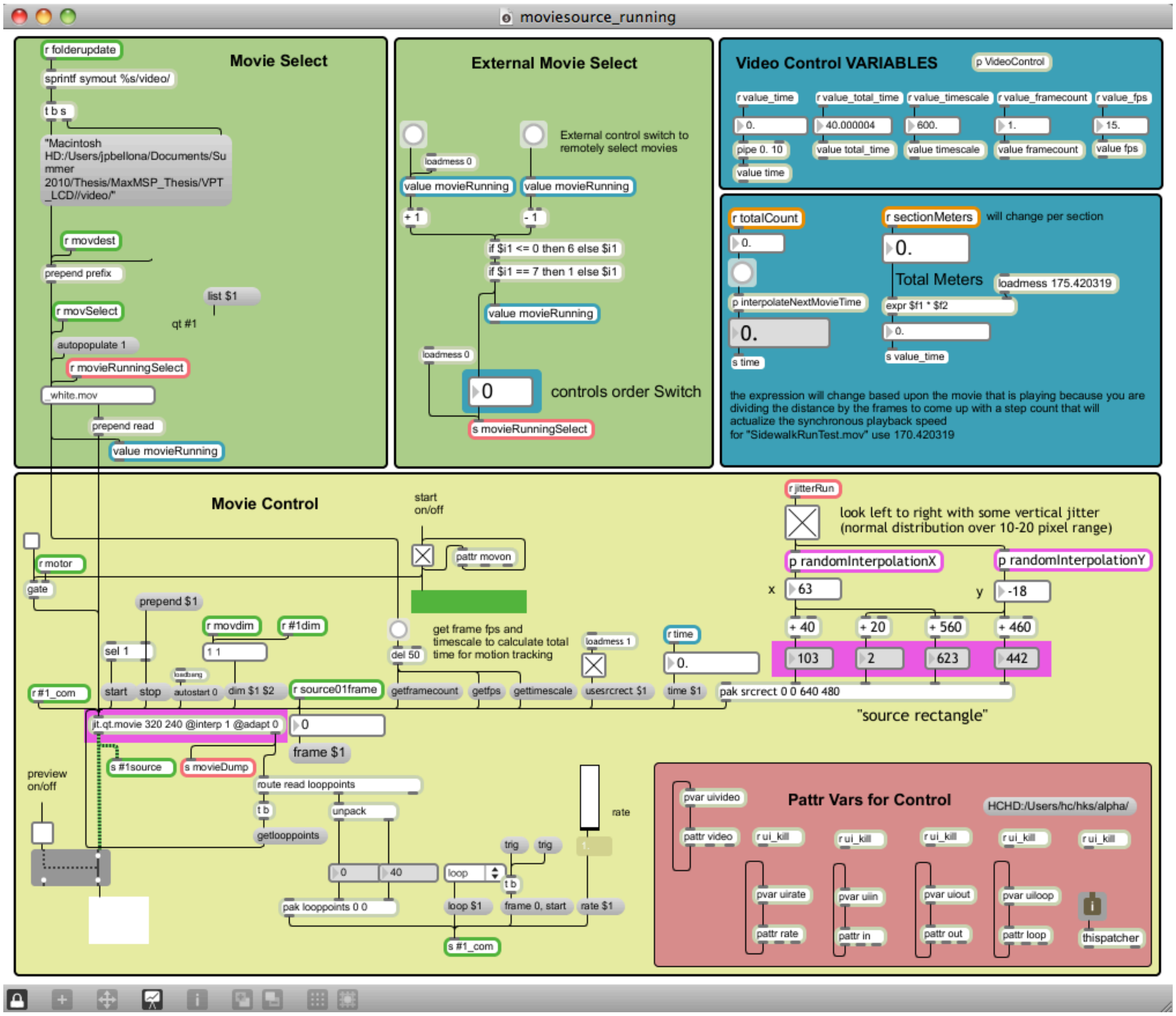



Figure 12.c.6. Movie source cleanup documentation. Cleaned-up running movie patch in Editing mode.

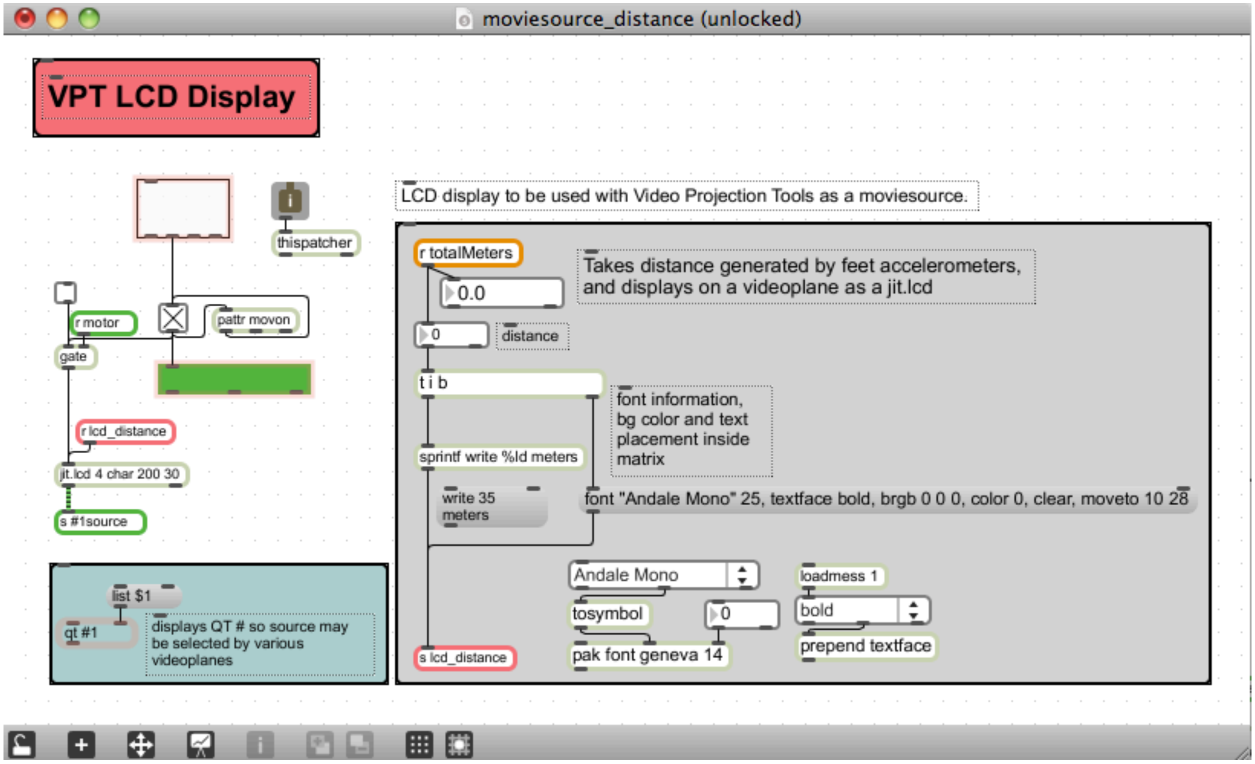



Figure 12.c.7. Movie source cleanup documentation. Cleaned-up LCD movie patch in Editing mode.

## 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.

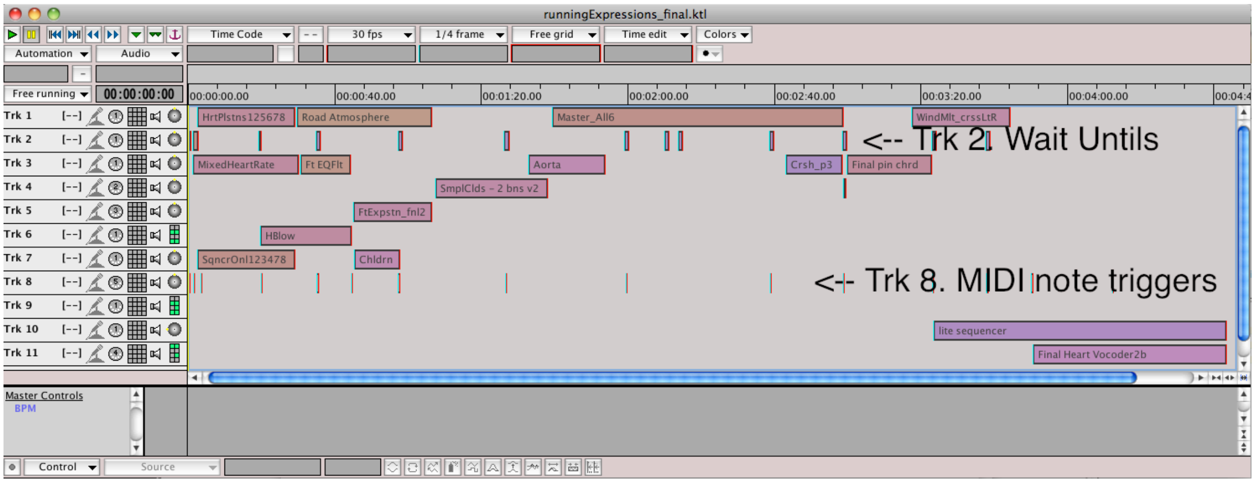



Figure 13.1. Kyma TL, with WaitUntil Sound track and MIDI note track highlighted. The timeline duration does not matter because each section's duration is determined by triggering WaitUntil Sounds.

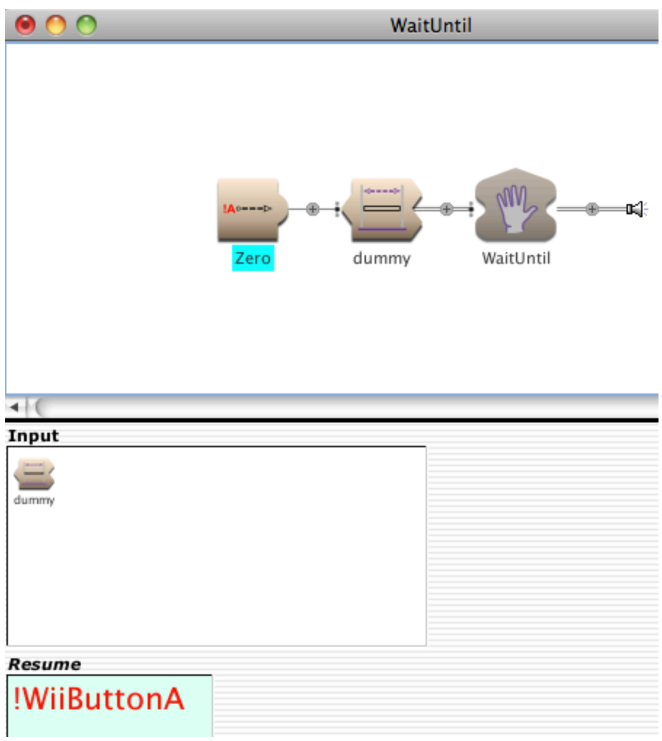



Figure 13.2. WaitUntil Sound Object. Wiimote 1 button 2 triggers each section, although mapped as !WiiButtonA inside Kyma.

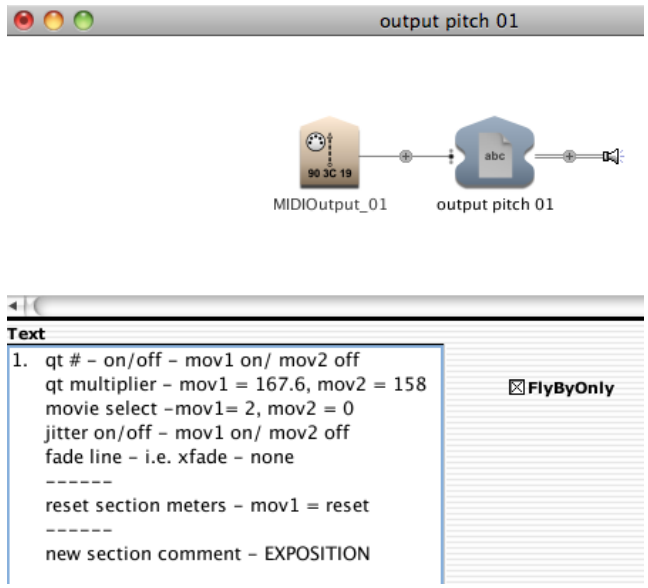



Figure 13.3. MIDI Output Pitch, serves as video command trigger, which is mapped inside Max/MSP/Jitter.

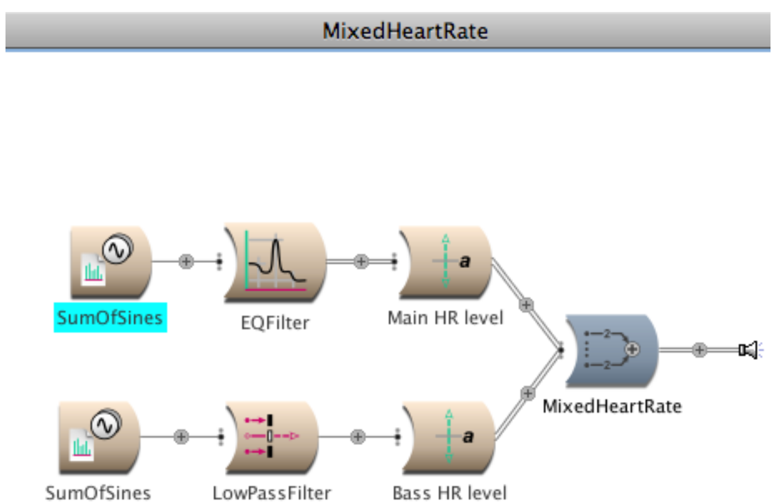



Figure 13.4. Heartbeat Sound, first electronic sound heard in *RunningExpressions.*

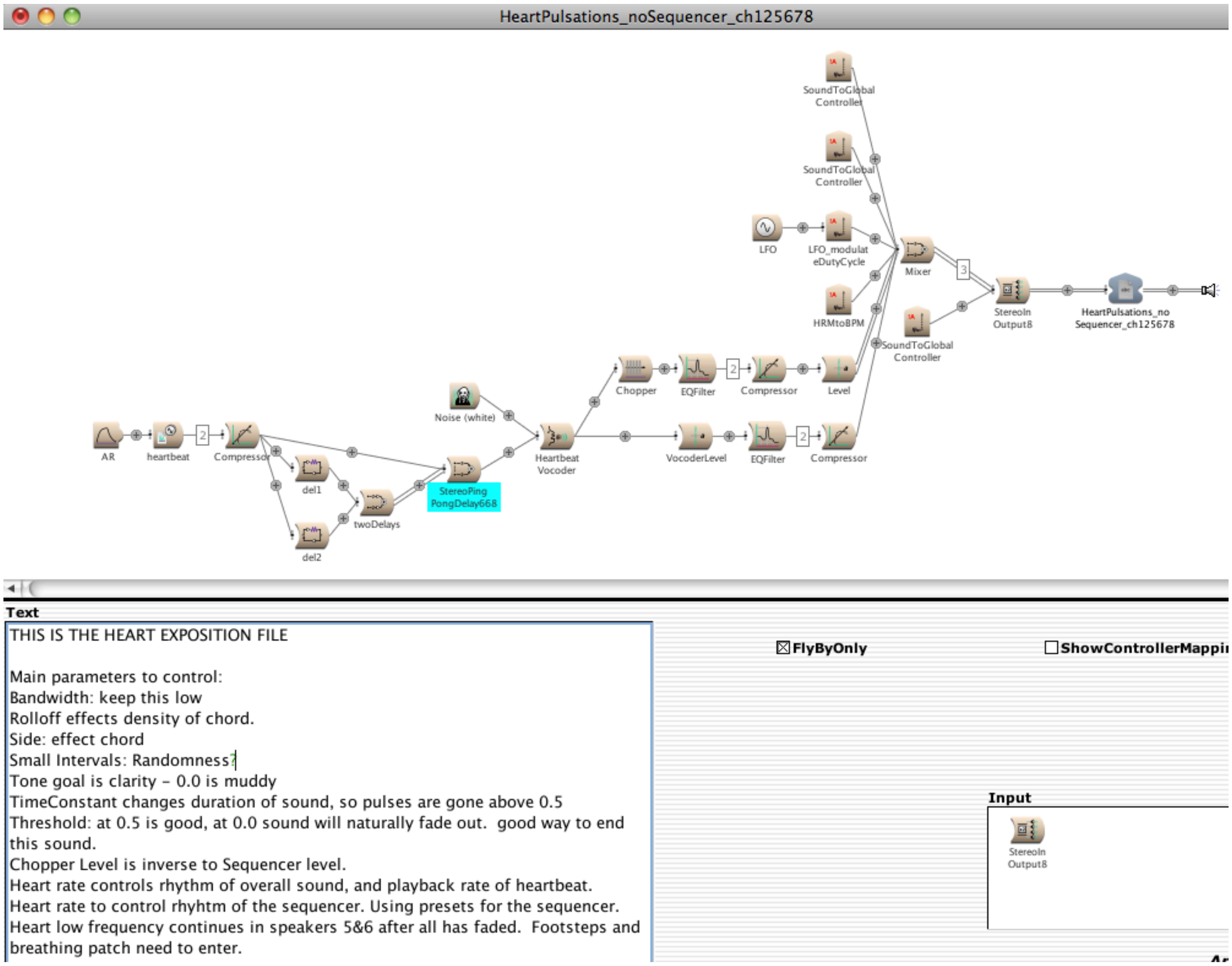



Figure 13.5. Heartbeat Exposition Main Sound, Vocoder with Delays

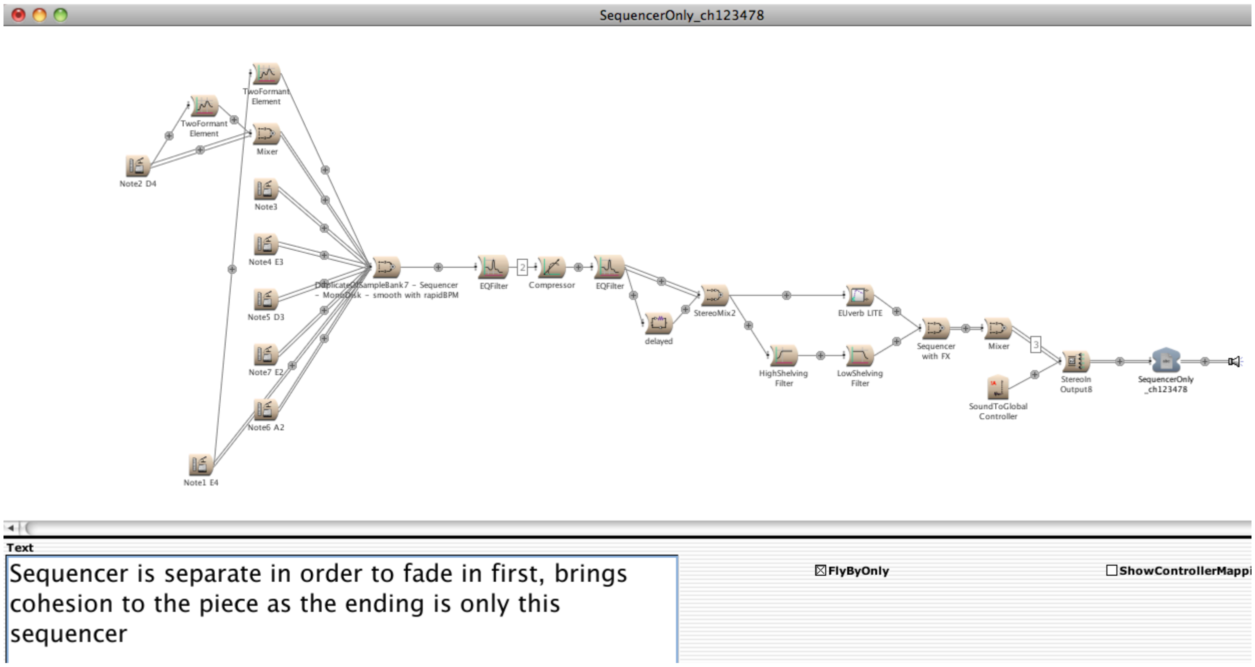



Figure 13.6. Heartbeat Exposition Sequencer

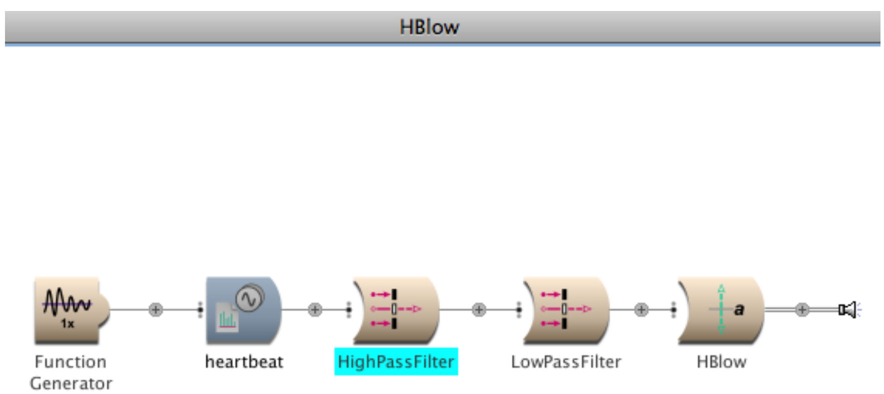



Figure 13.7. Heartbeat Low Rumble

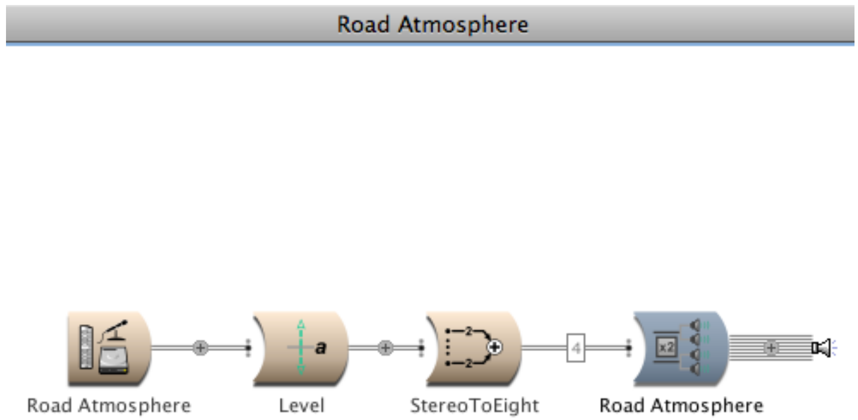



Figure 13.8. Road Environment Ambient Sound

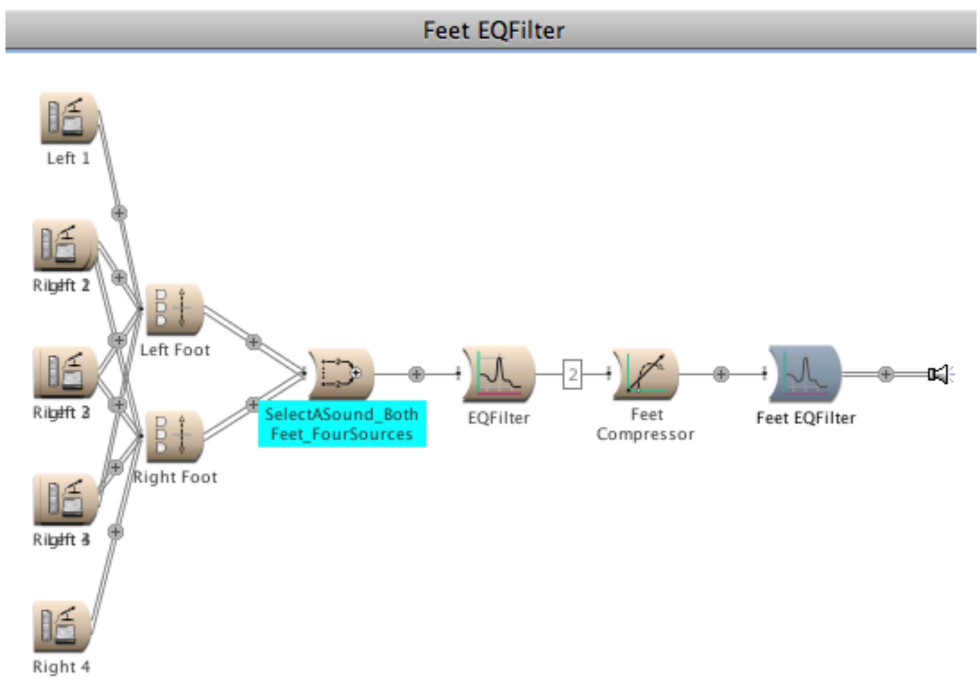



Figure 13.9. Selectable Foot Sounds. Accelerometers serve as sound triggers.

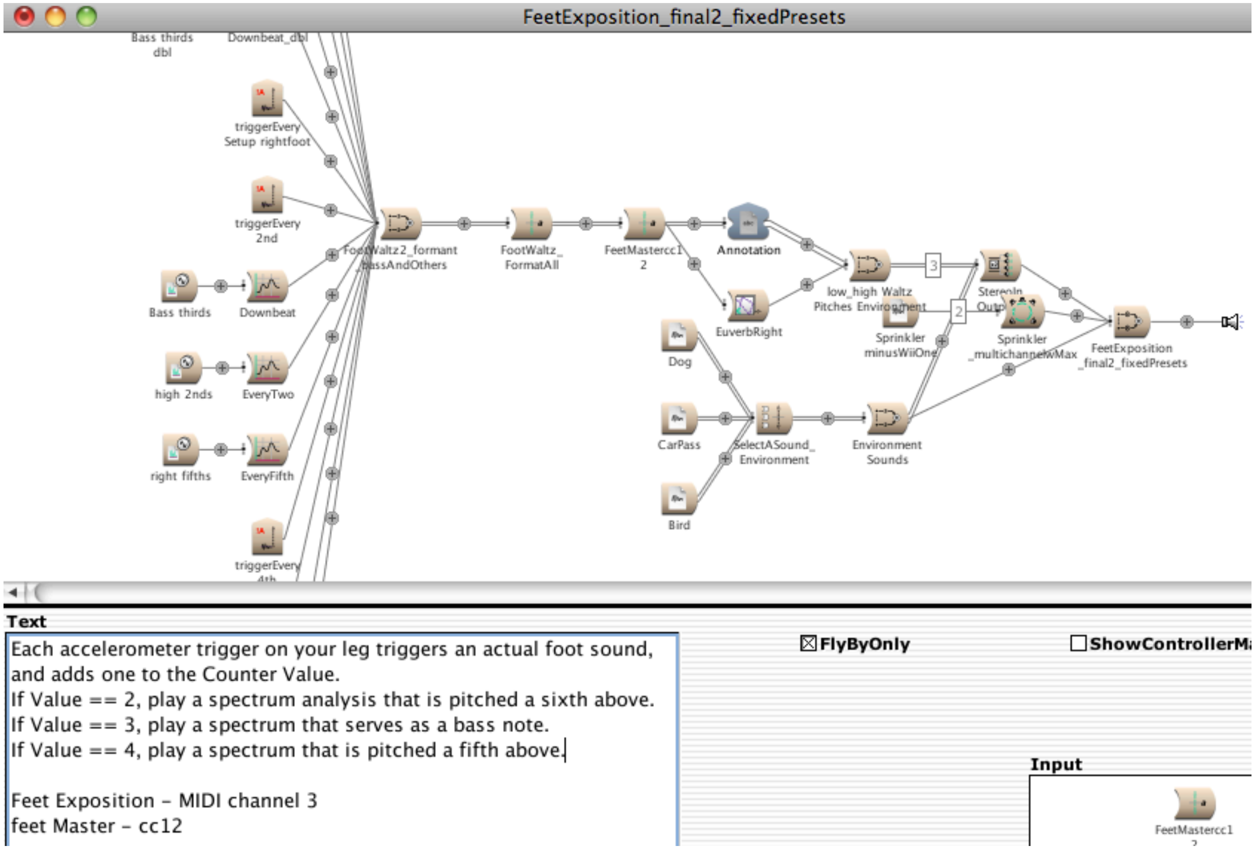



Figure 13.10. Feet Exposition Waltz

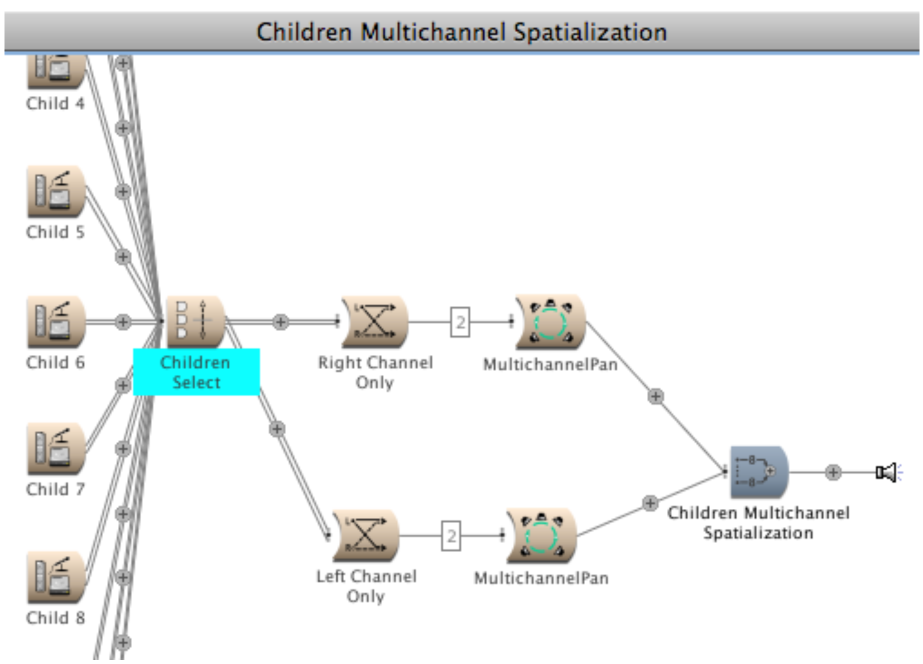



Figure 13.11. Selectable Children Sounds. Eleven sounds triggered and panned by Wiimote 2.

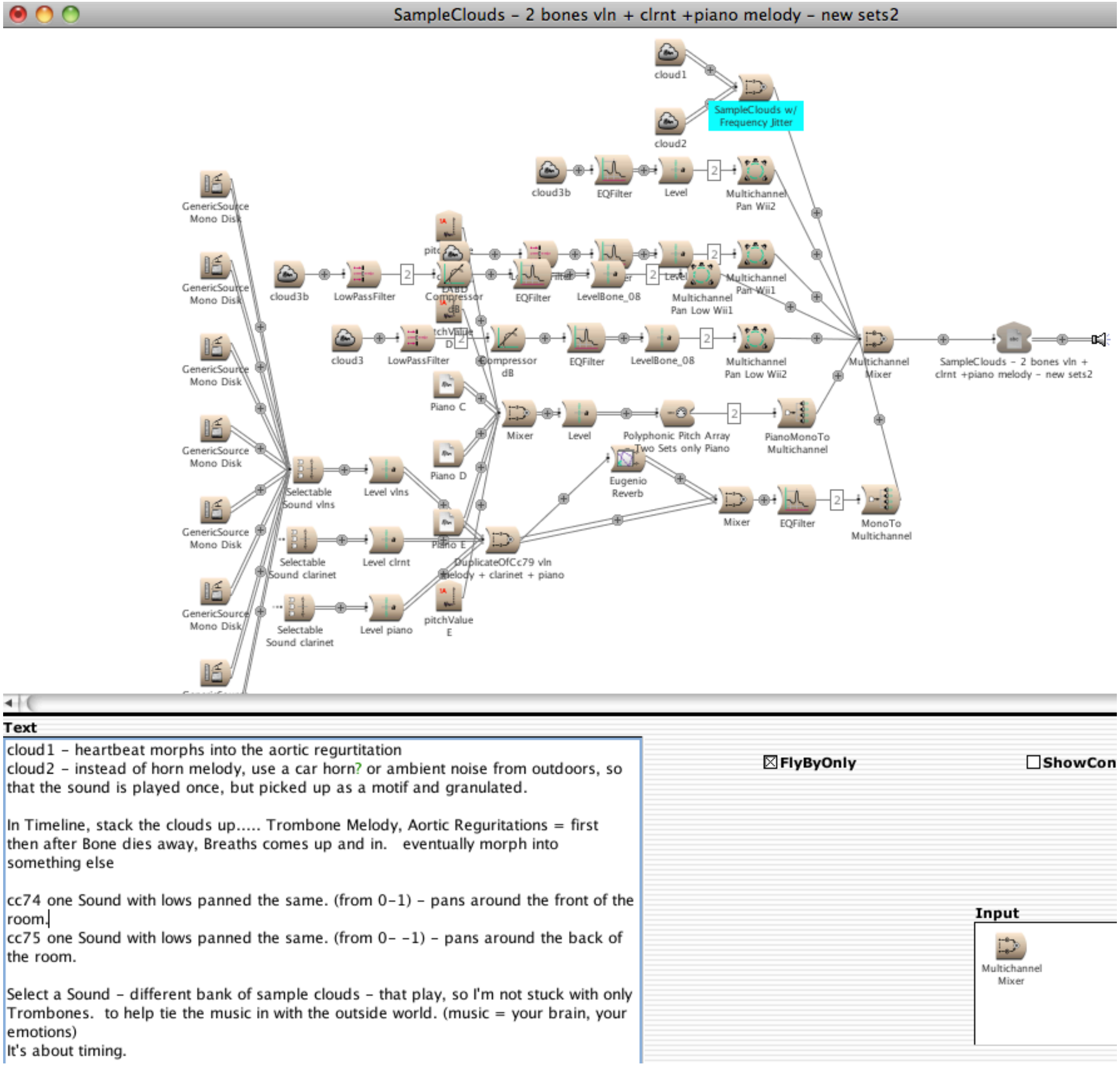



Figure 13.12. Development Section for Trombones, Piano, and Strings

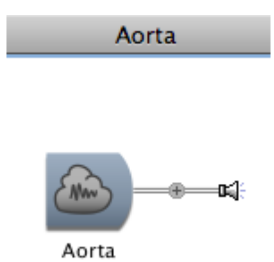



Figure 13.13. Aorta Sound Transition

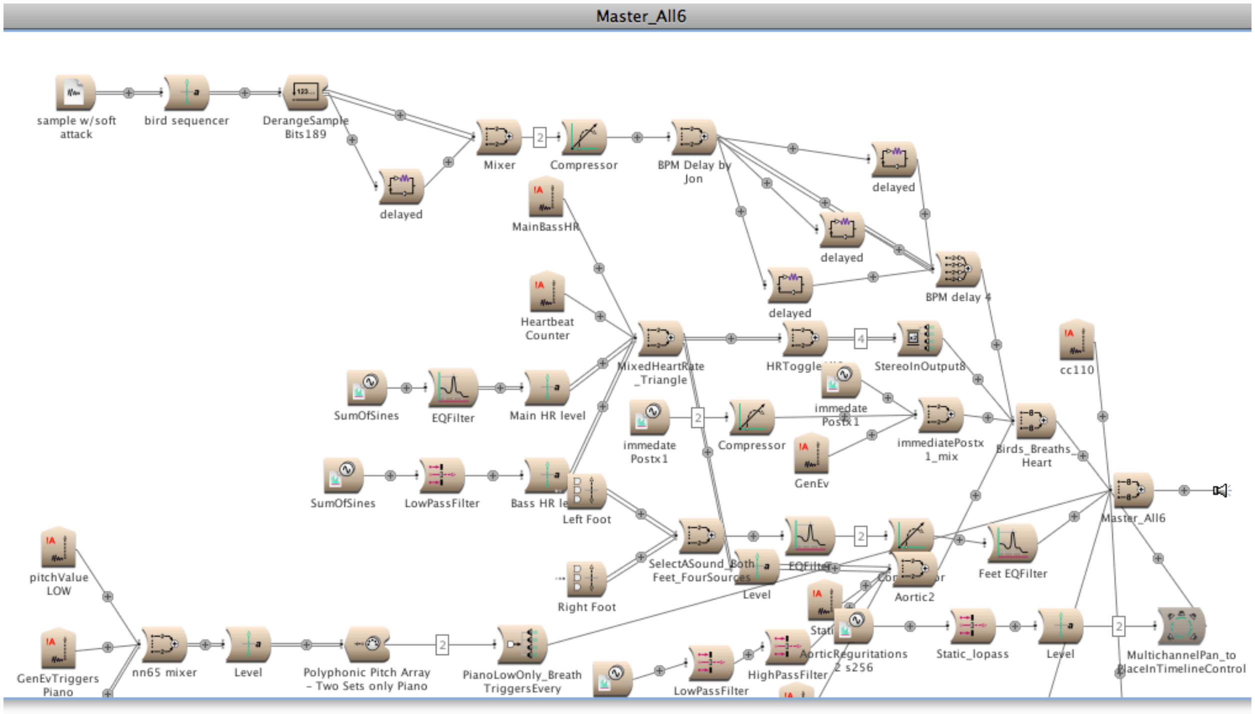



Figure 13.14. Development Section Climax, part 1

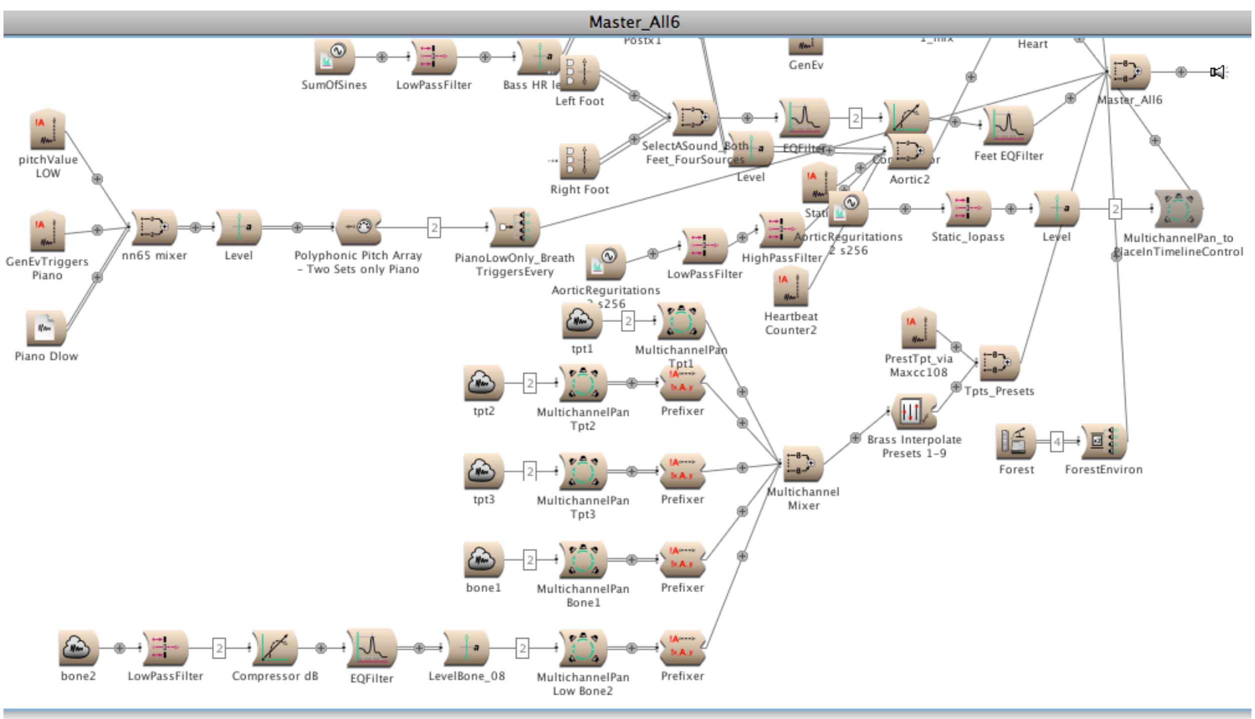



Figure 13.15. Development Section Climax, part 2

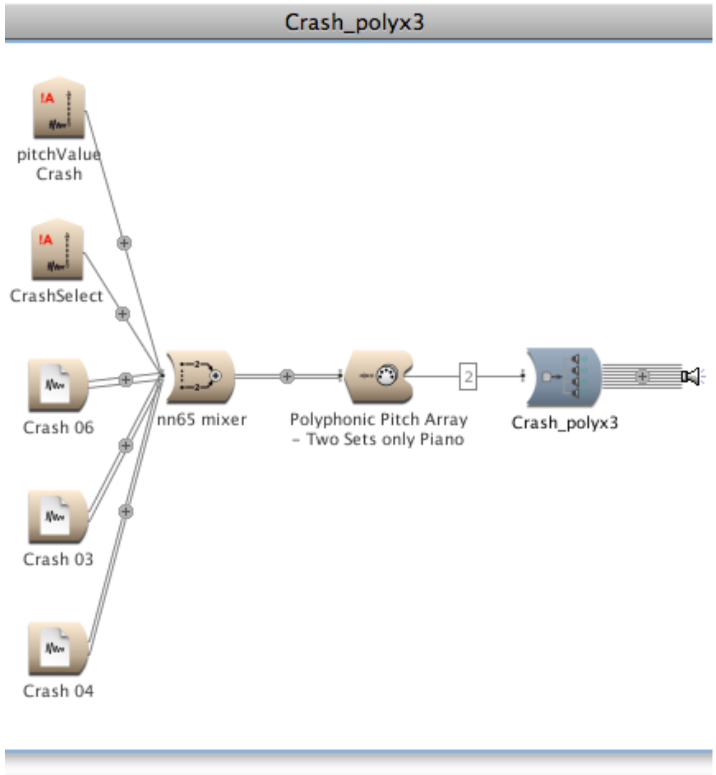



Figure 13.16. Crashing Forests Sounds, Randomly Selected

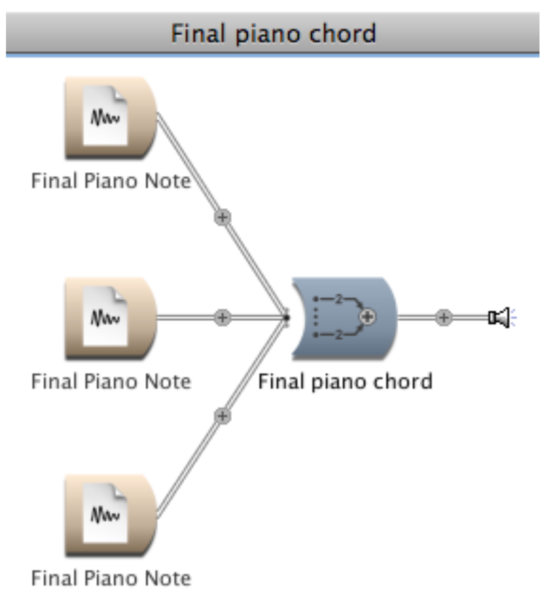



Figure 13.17. Final Piano Chord. Both Wiimotes' A buttons required to trigger sound.

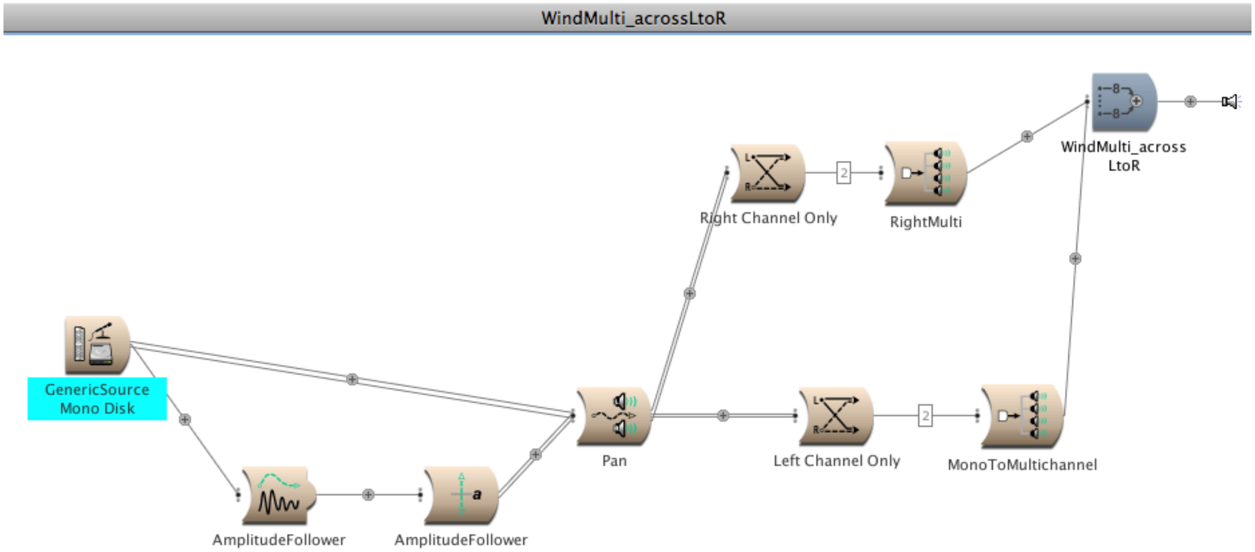



Figure 13.18. Wind Environment Sound. Amplitude increases in wind sound cause sound to pan from Left to Right.

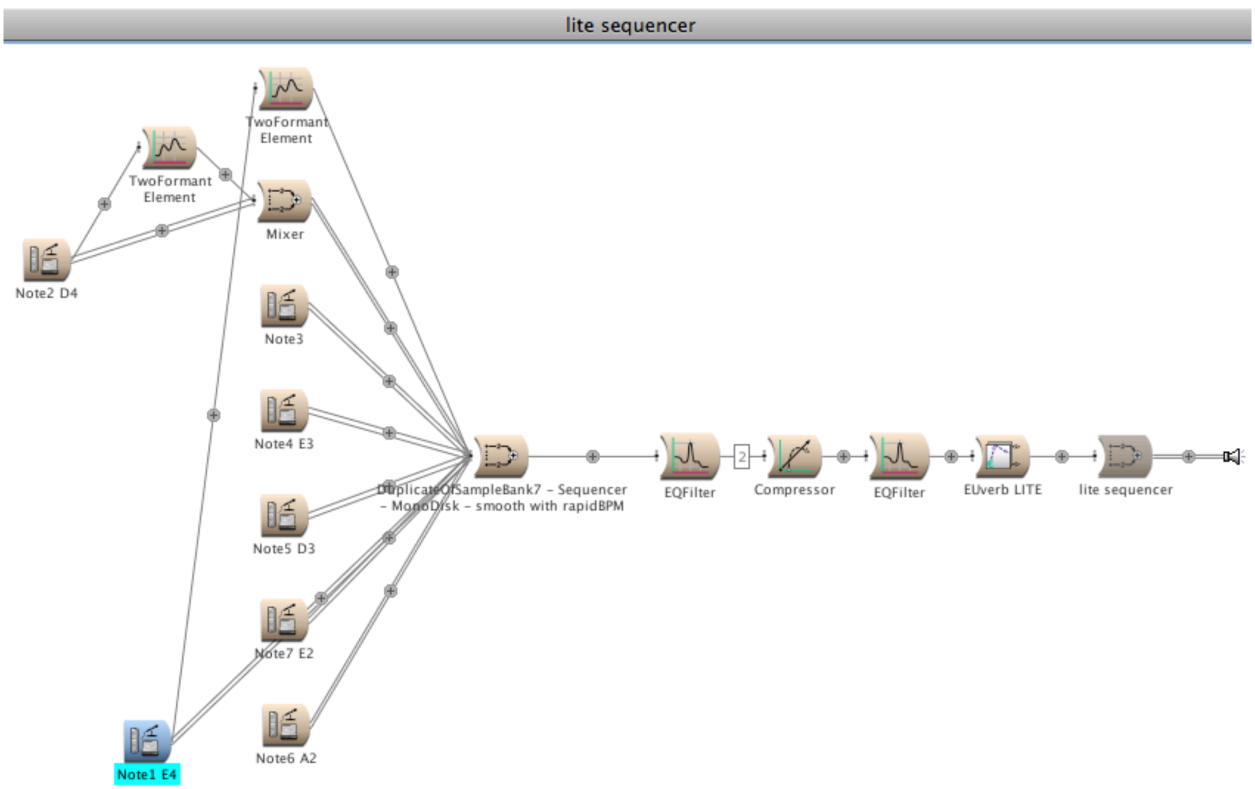



Figure 13.19. Exposition Sequencer Revisited

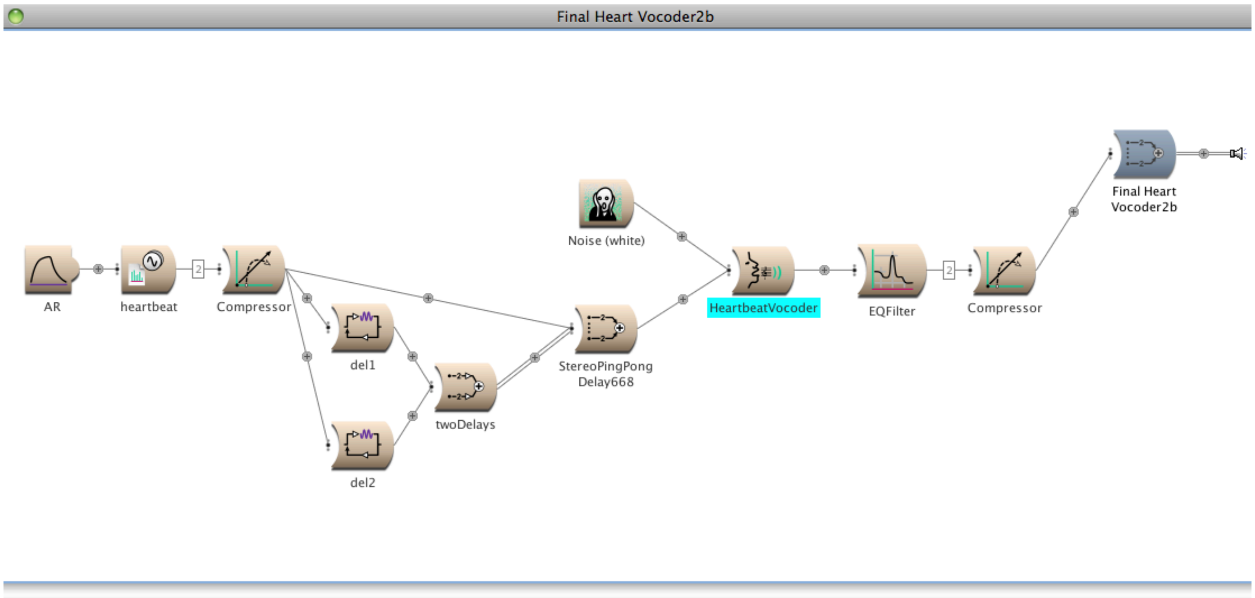



Figure 13.20. Exposition Heartbeat Vocoder Revisited

# 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.

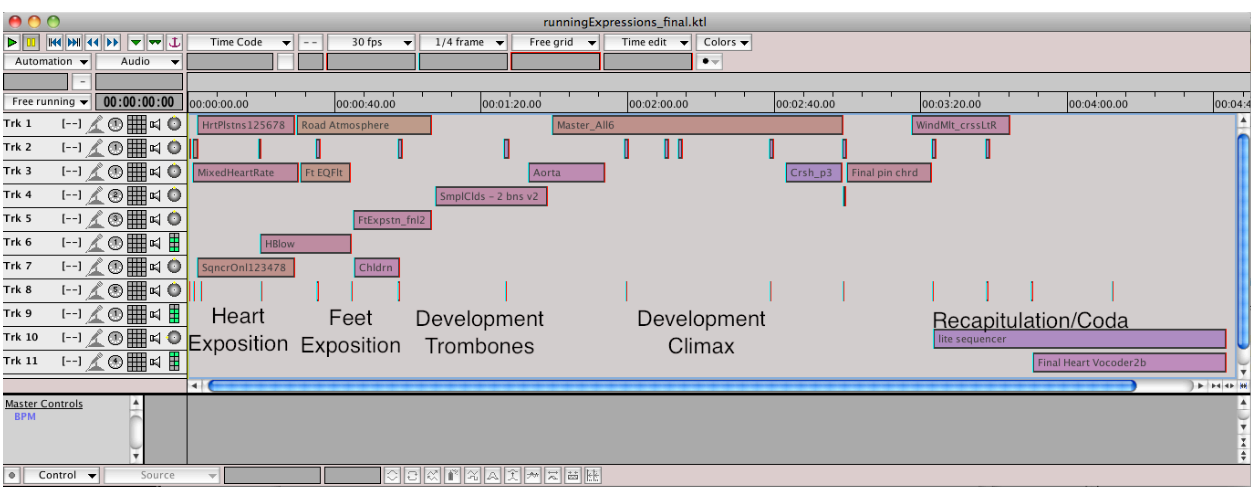



Figure 13.21. Kyma TL, with Sections Labeled

## 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

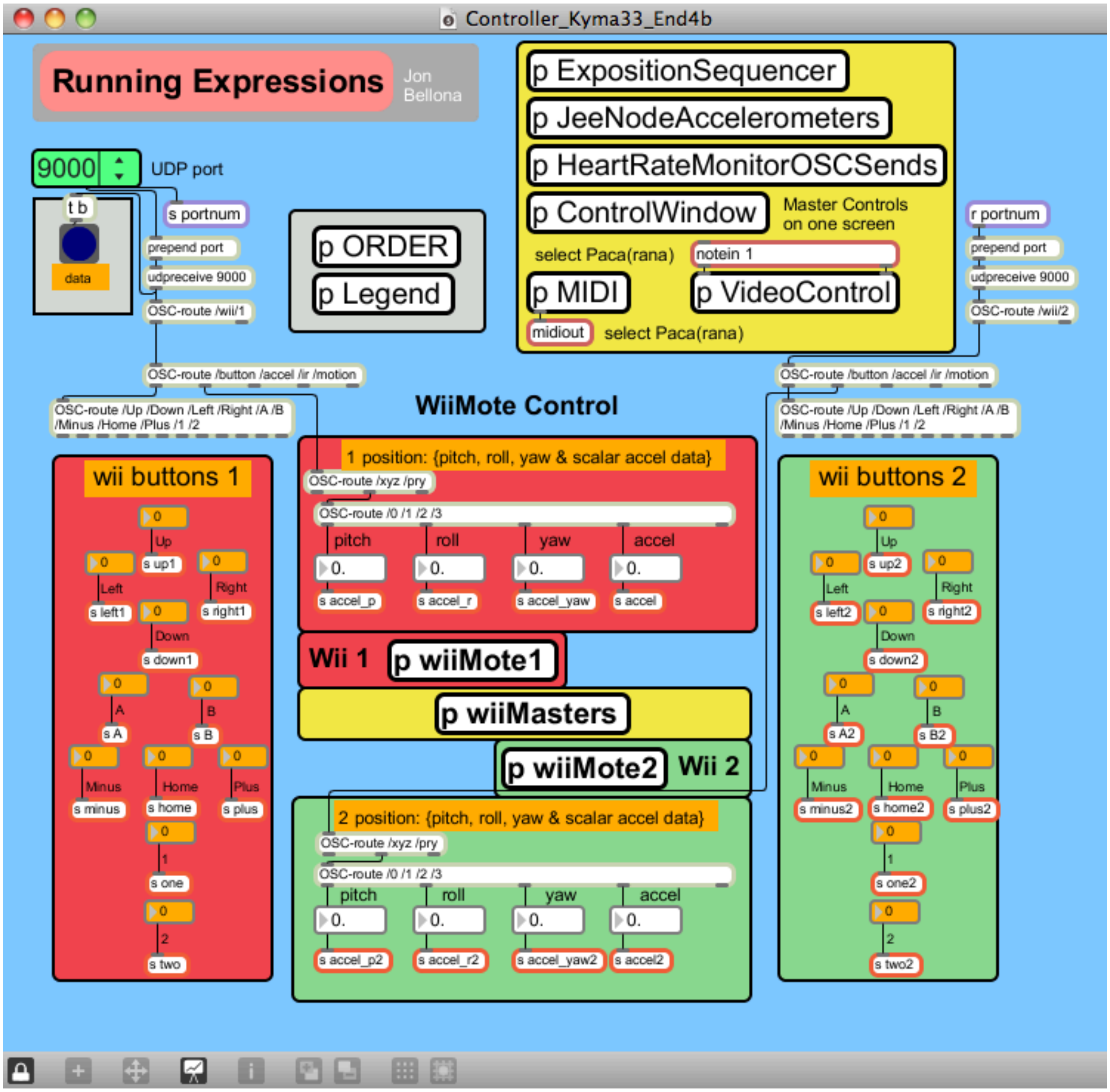


Figure A.1.1. Controller_Kyma33_End4b.maxpat Main Patch Window

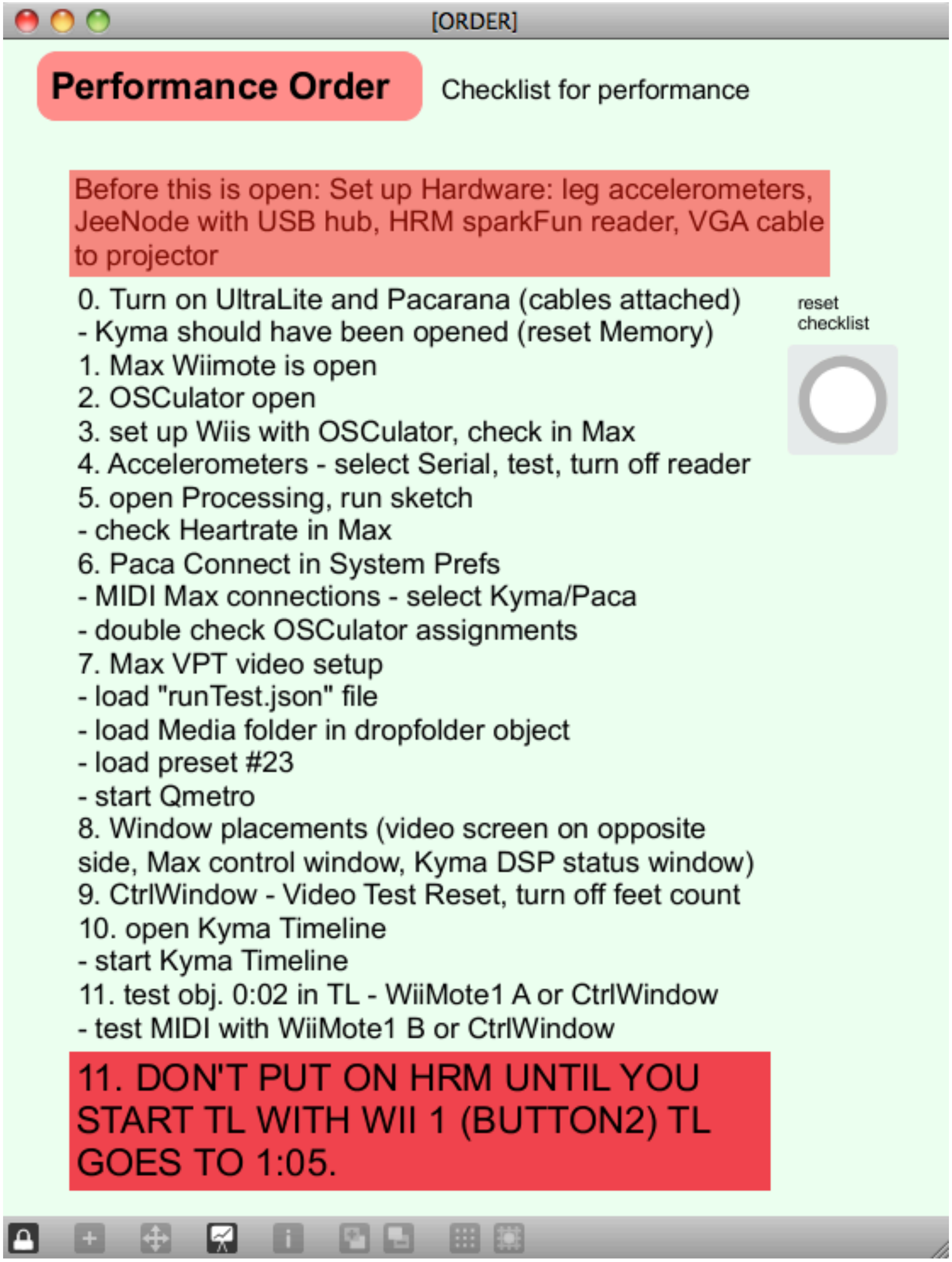



Figure A.1.2. Performance Setup Order Patch Window

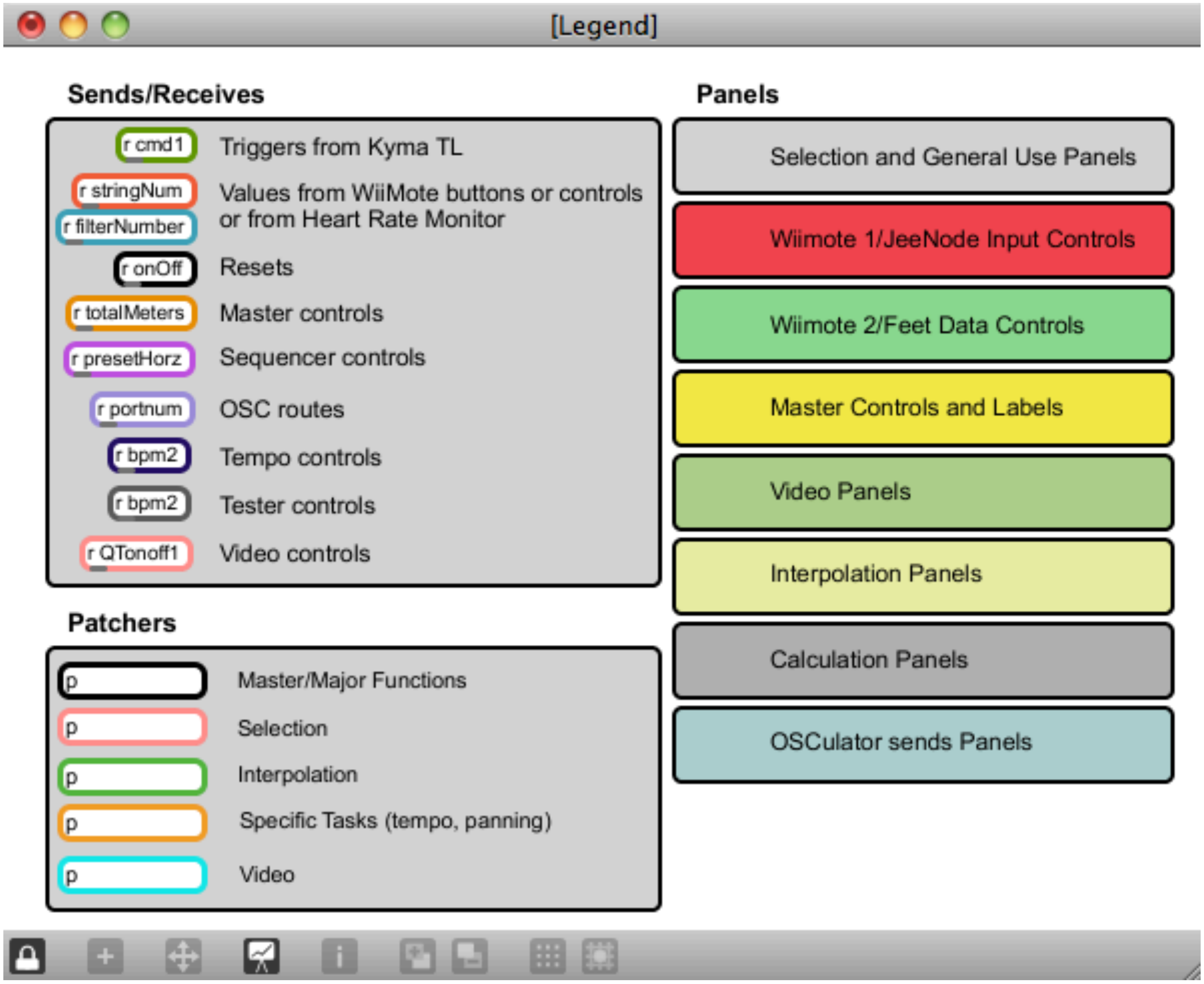



Figure A.1.3. Color Legend for Master Controller Max Patch

A.1.a. Exposition Sequencer

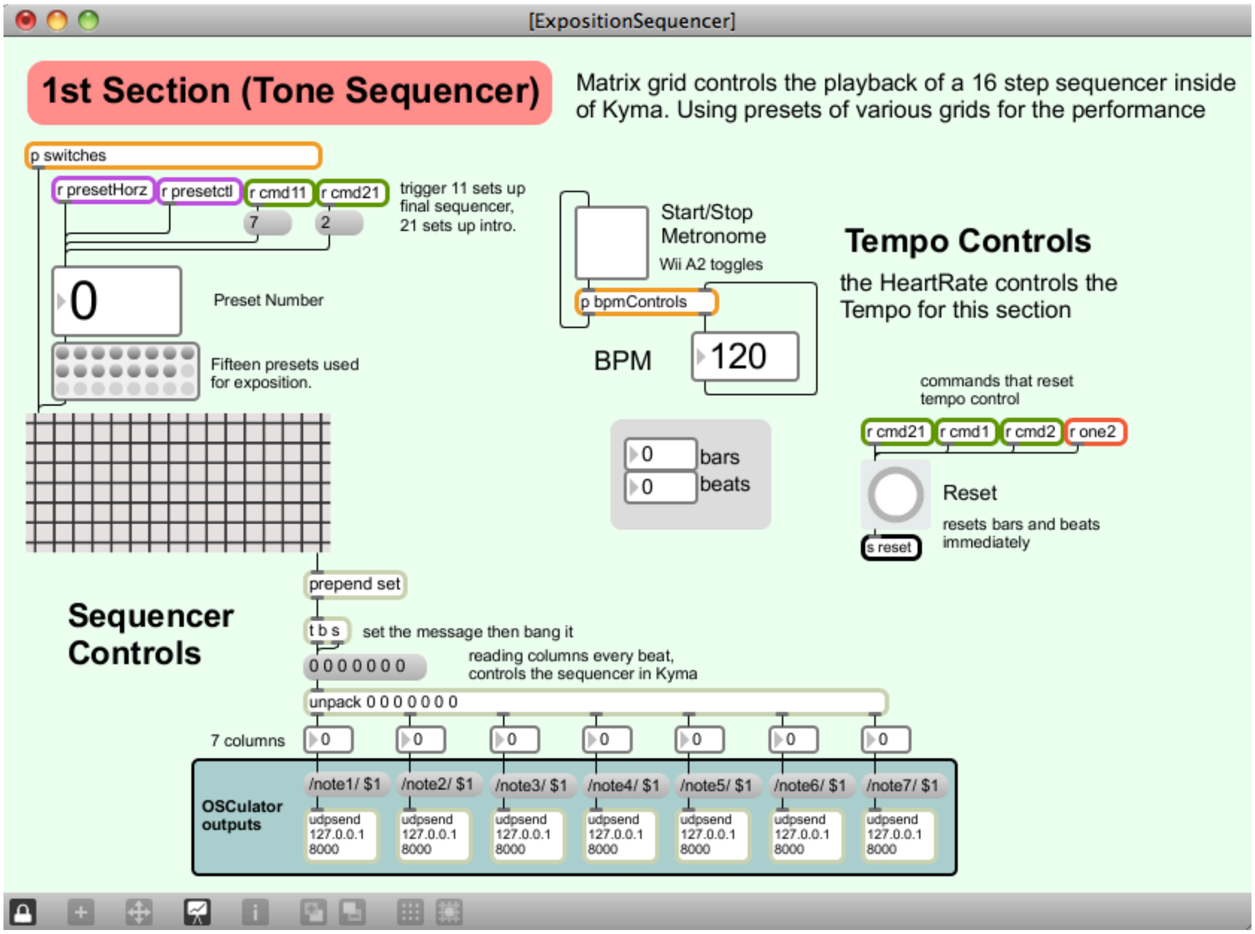



Figure A.1.a.1. Exposition Sequencer Patch Window. The 'matrixctrl' object controls the on/off messages of fixed notes inside the sequencer. The sequencer was built inside Kyma.

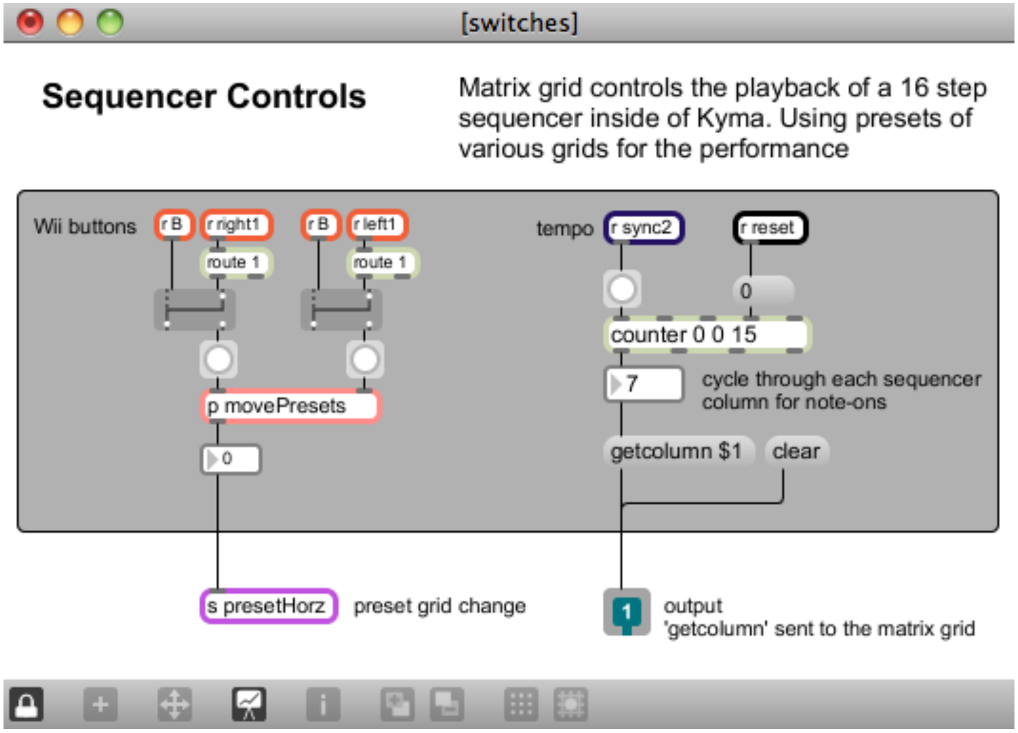



Figure A.1.a.2. Sequencer Control Patcher

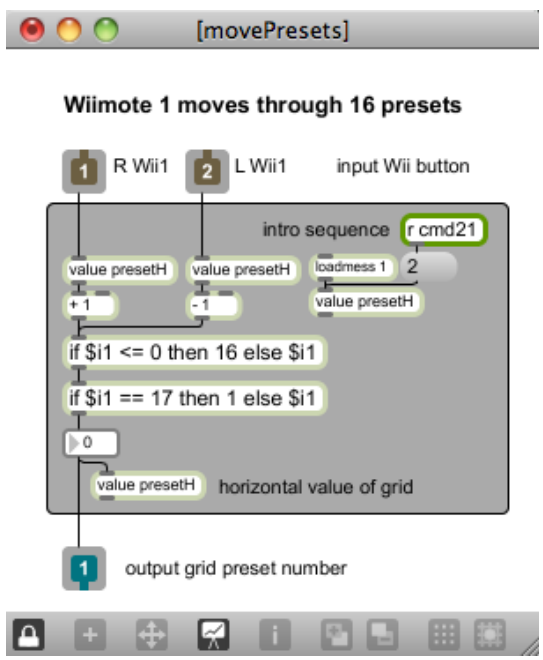



Figure A.1.a.3. Wiimote 1 Controls Presets Patcher

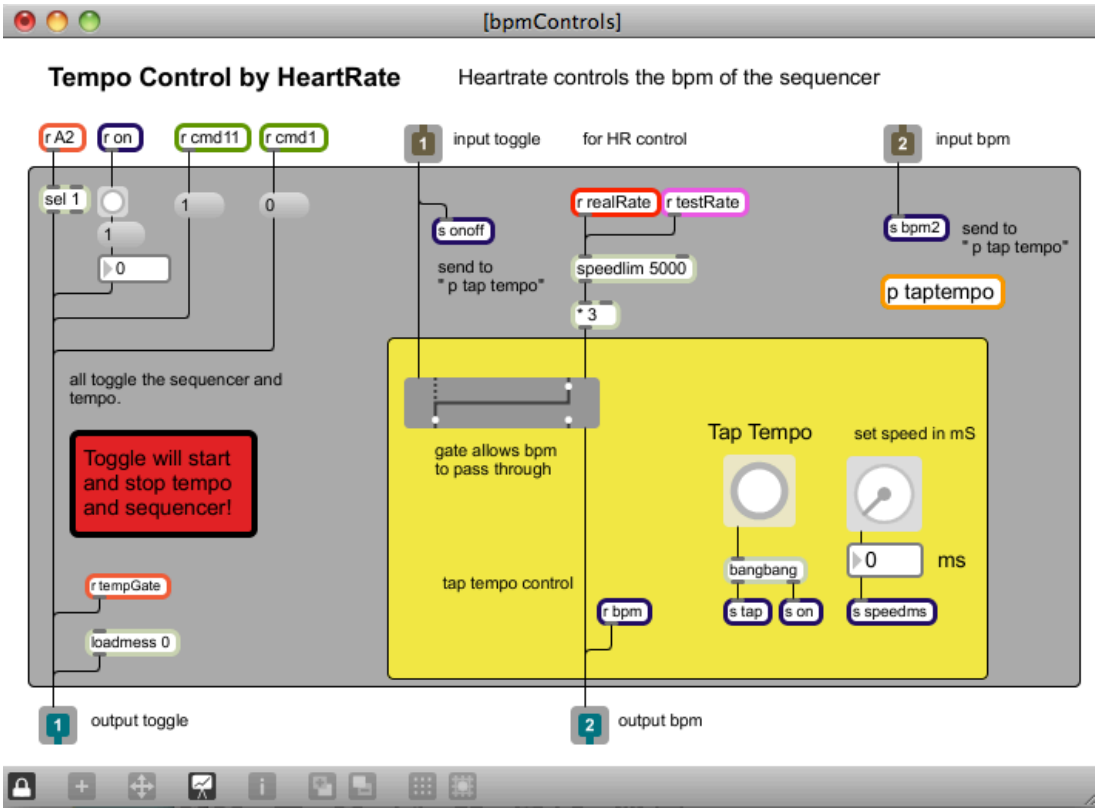



Figure A.1.a.4. Sequencer Tempo Control Patcher

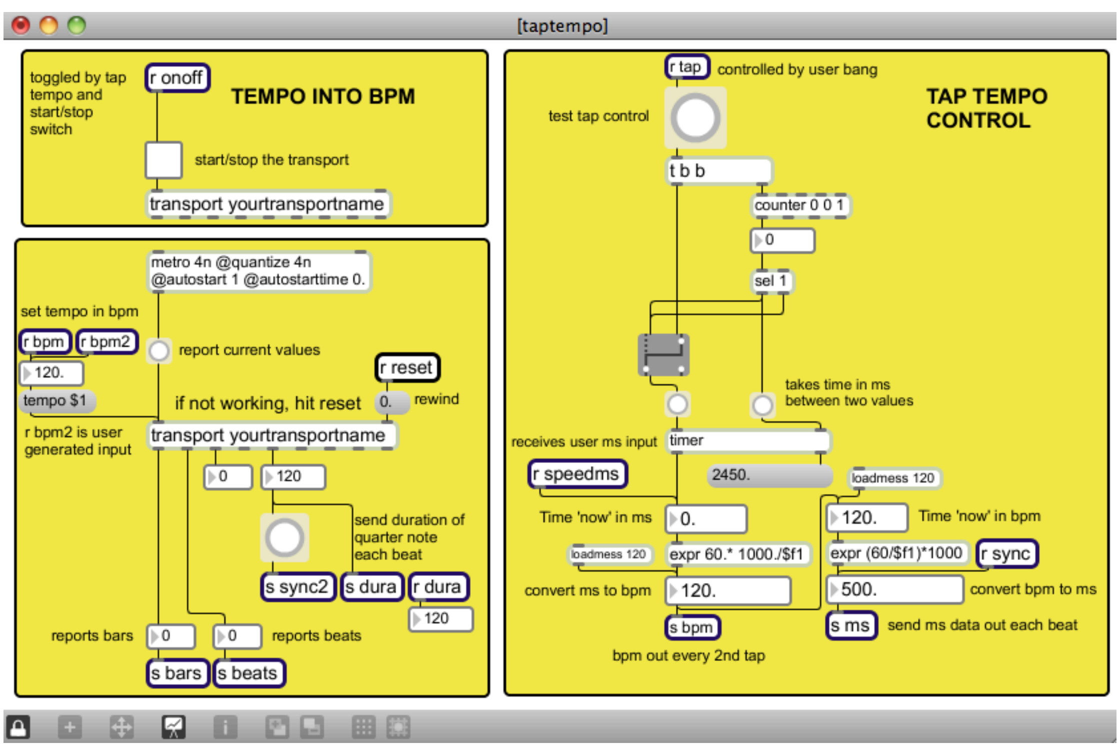



Figure A.1.a.5. Tap Tempo Sequencer Control Patcher

A.1.b. JeeNode Accelerometers

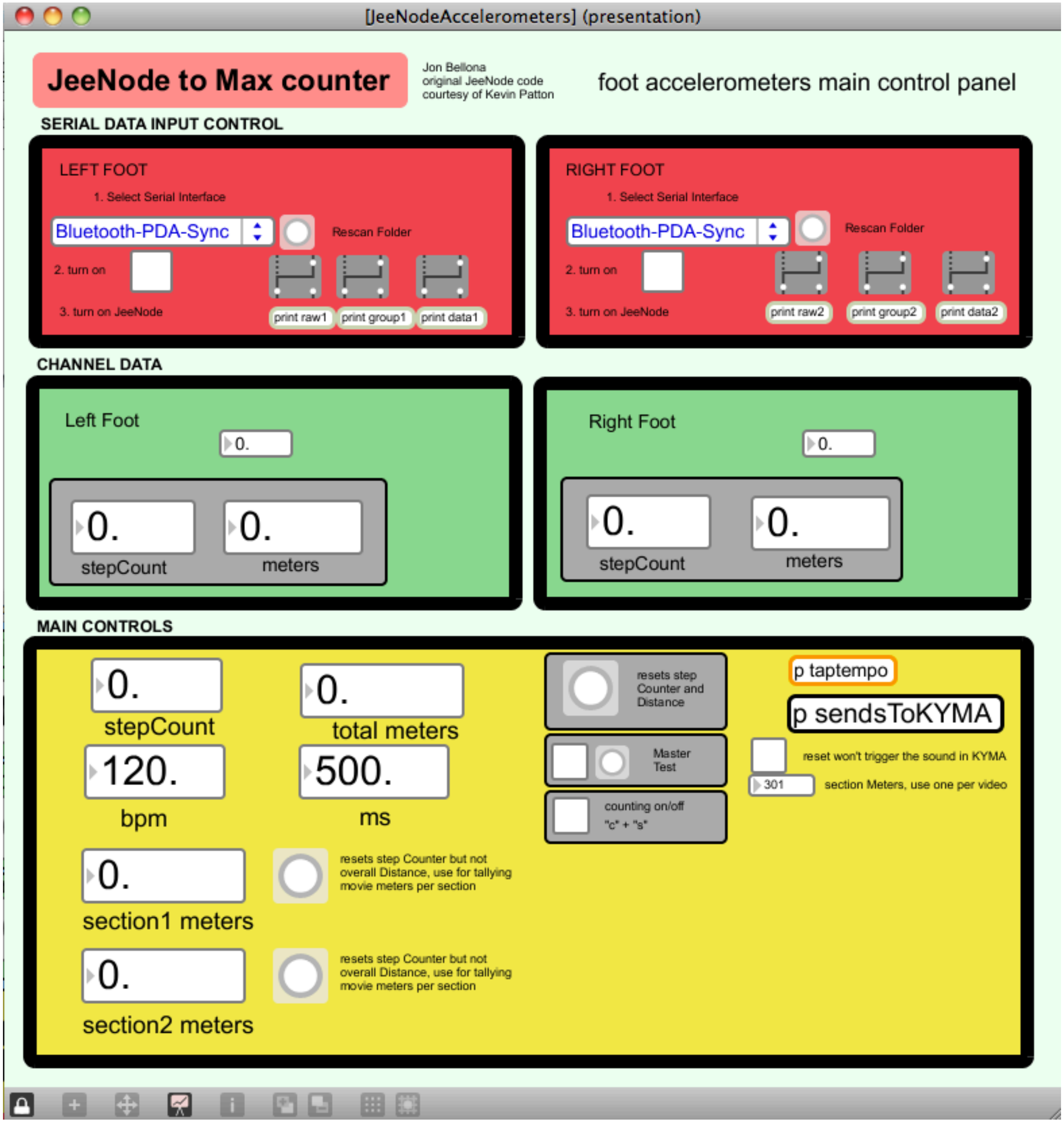



Figure A.1.b.1. JeeNode Patch Window, in Presentation mode

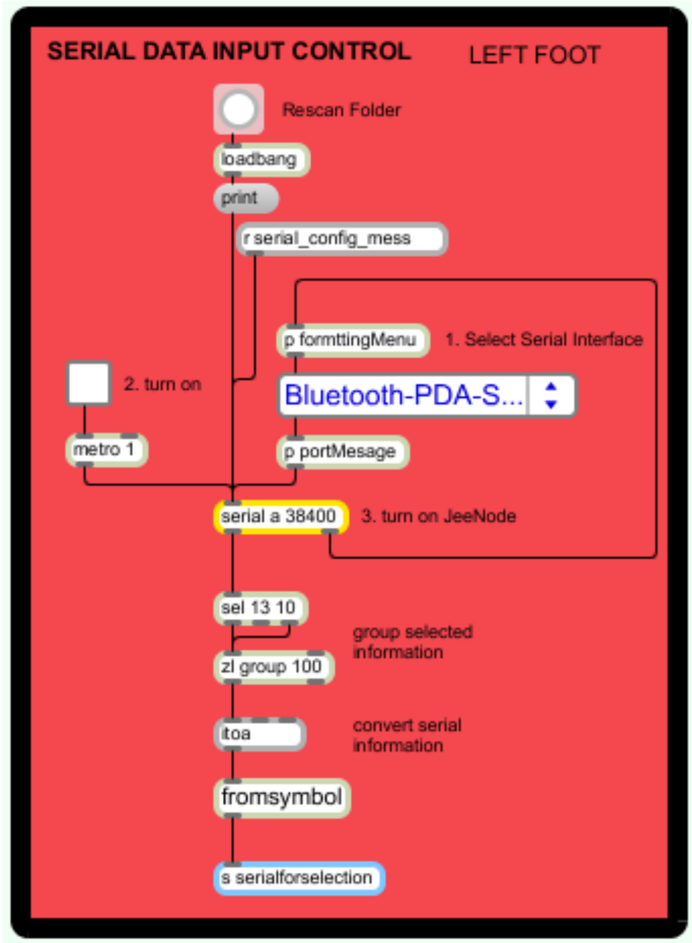



Figure A.1.b.2. Serial Data Input Module

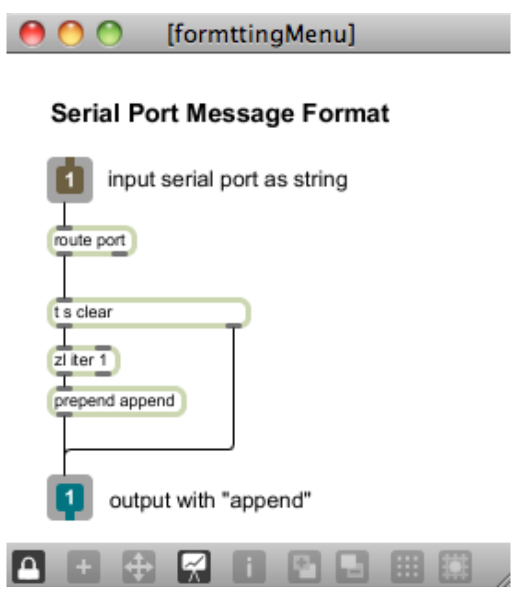



Figure A.1.b.3. Serial Port Formatting Menu Patcher

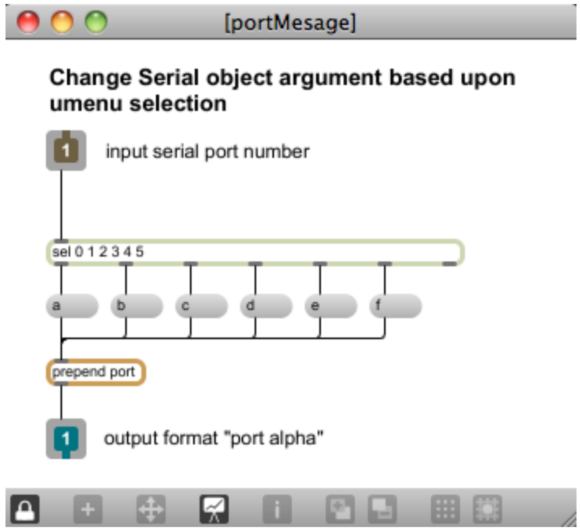



Figure A.1.b.4. Serial Port Formatting Message Patcher

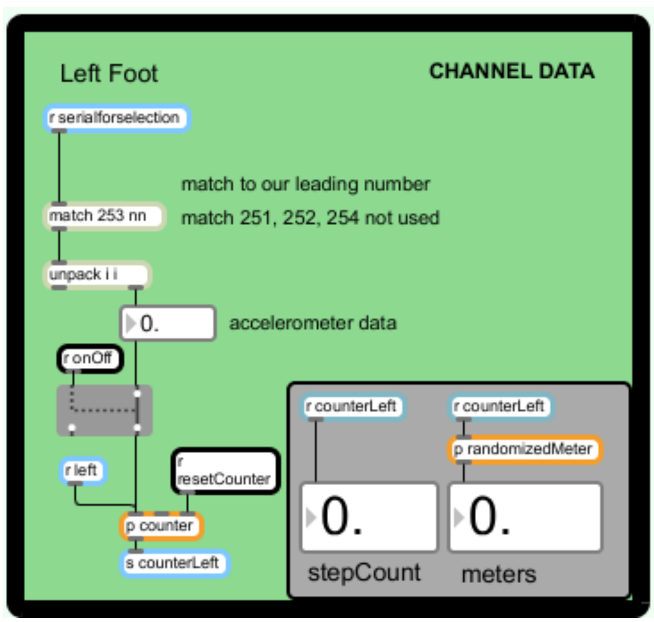



Figure A.1.b.5. Serial Channel Data Display Module

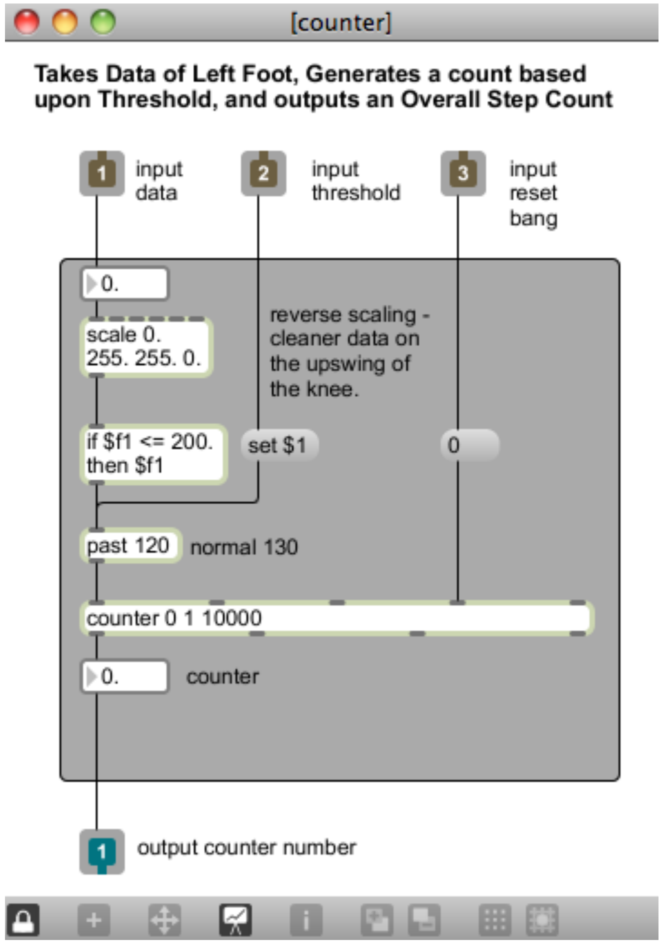



Figure A.1.b.6. Accelerometer Threshold Counter Patcher

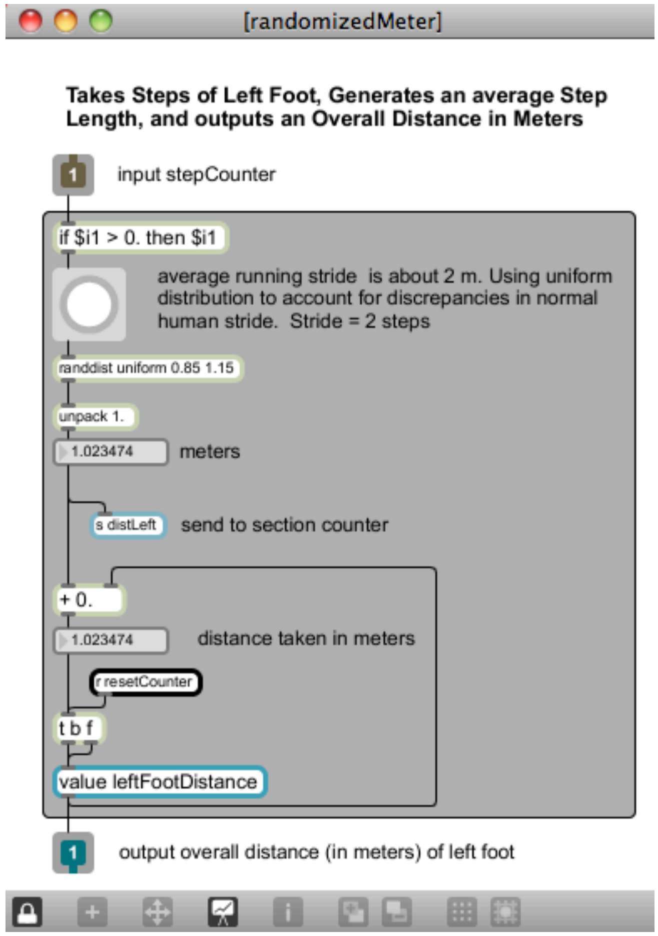



Figure A.1.b.7. Foot Distance Calculator Patcher

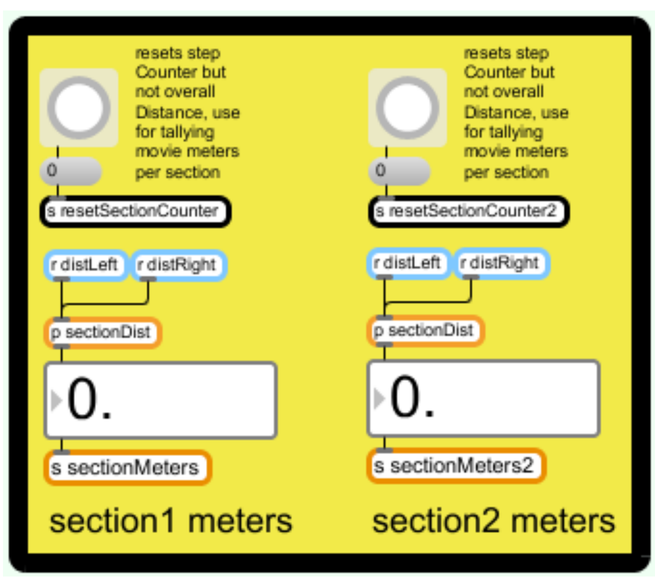



Figure A.1.b.8. Master Foot Distance Display Module 1

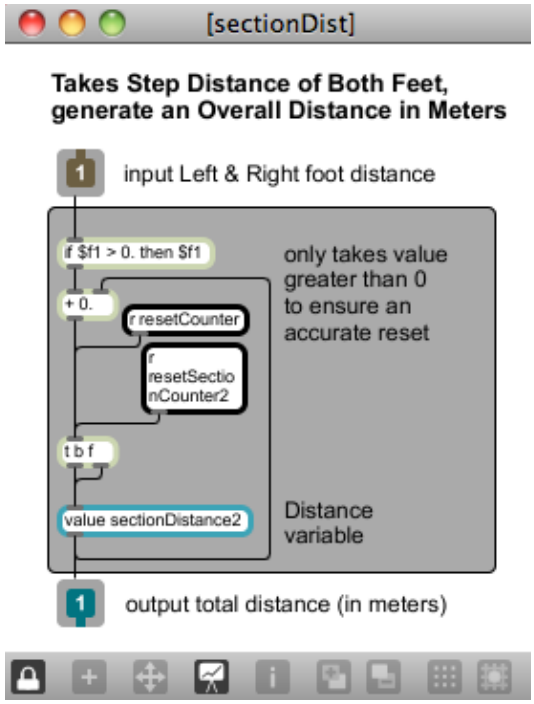



Figure A.1.b.9. Master Feet Distance Calculator per Section Patcher, video control

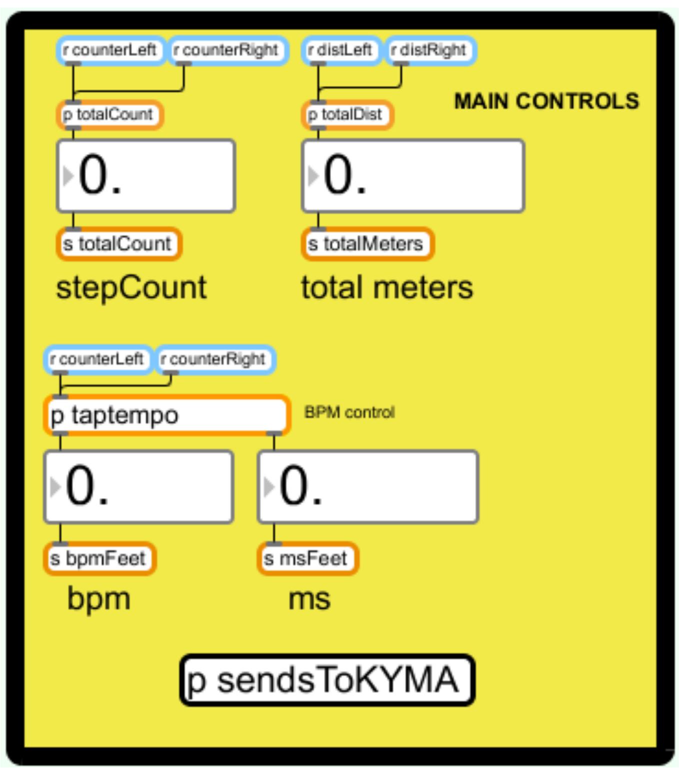



Figure A.1.b.10. Master Accelerometer Control and Routing Module 2

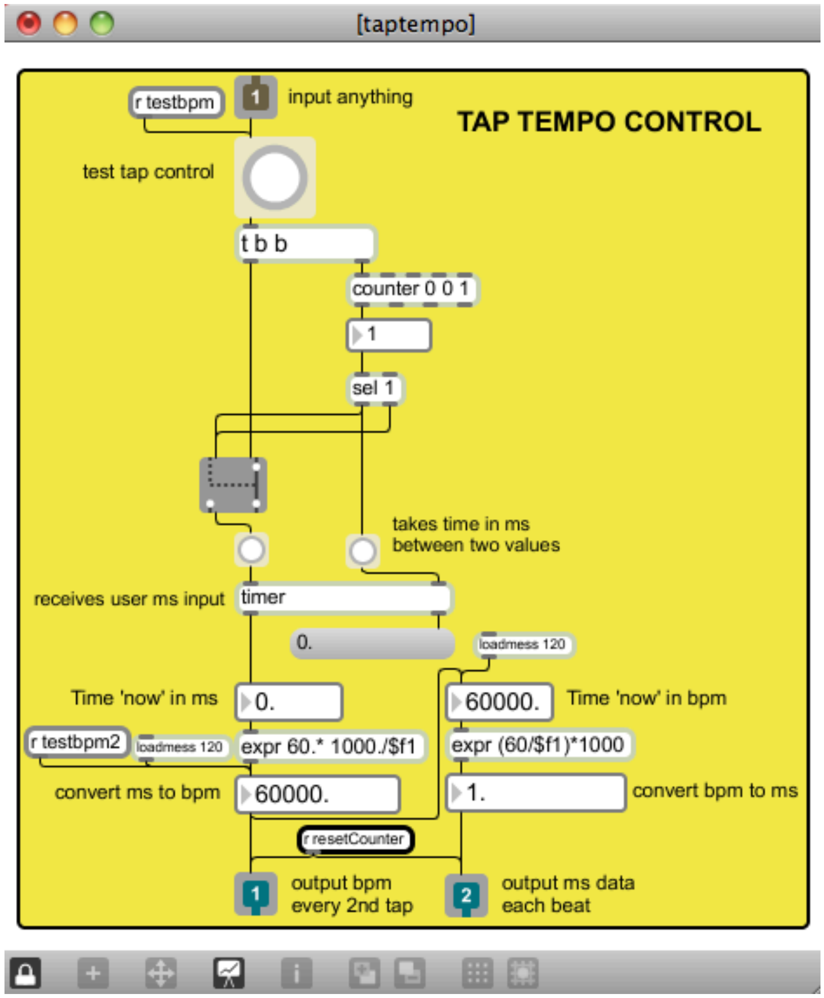



Figure A.1.b.11. Feet Accelerometer Tempo Control Patcher

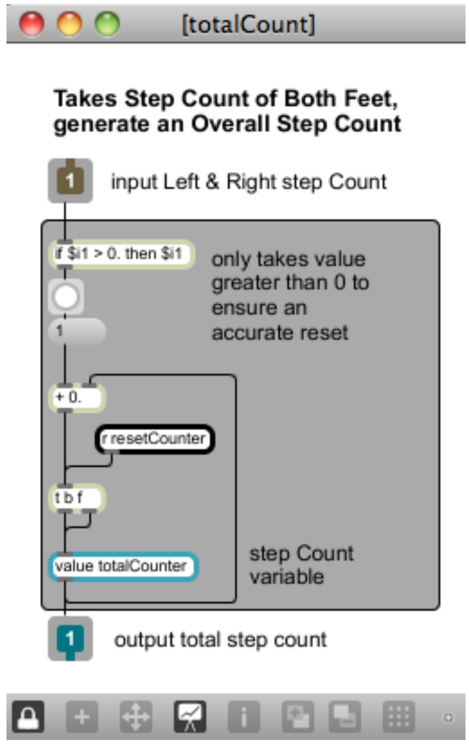



Figure A.1.b.12. Master Feet Counter Calculator Patcher, controls video

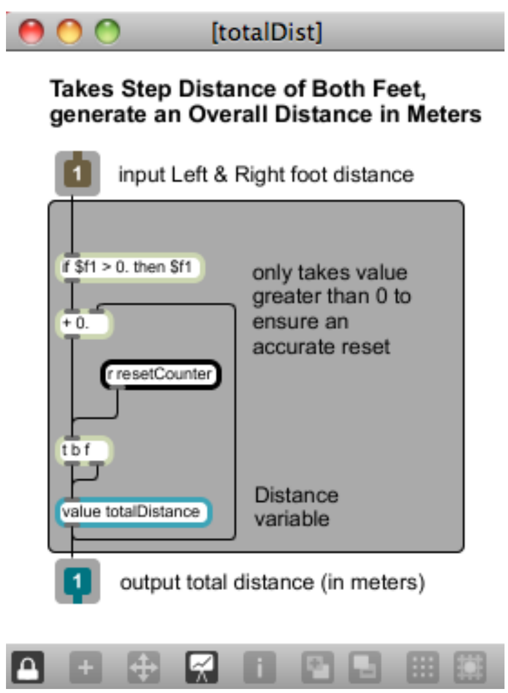



Figure A.1.b.13. Master Feet Distance Calculator Patcher, lcd display

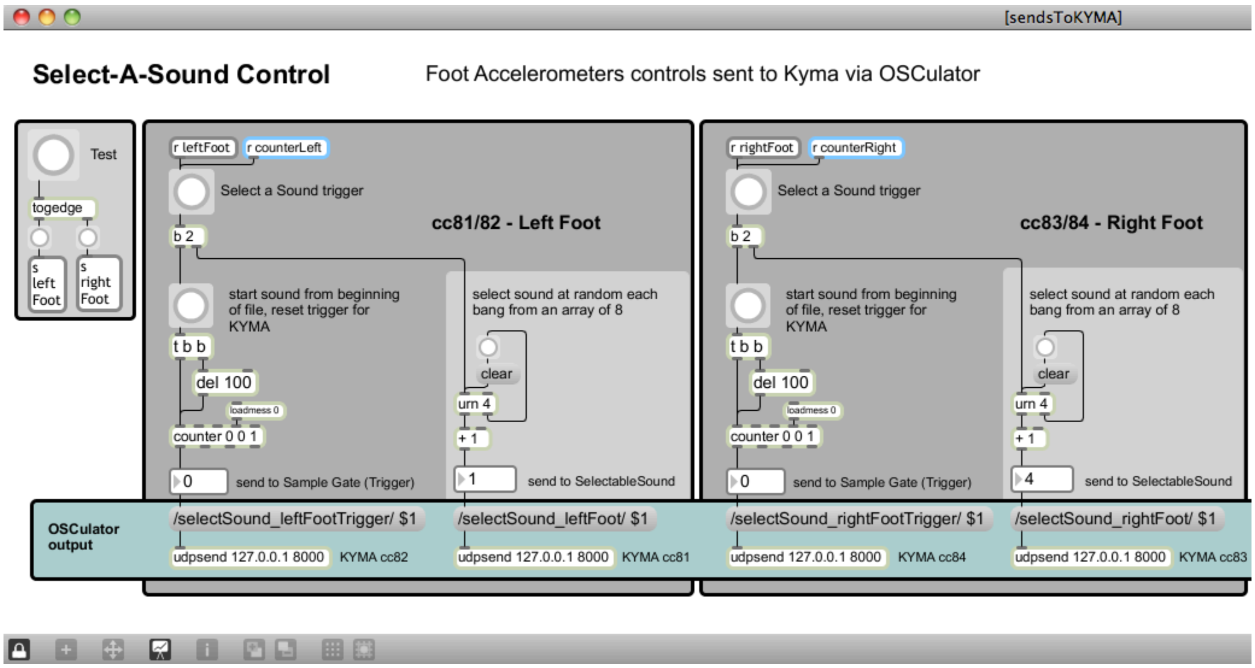



Figure A.1.b.14. Accelerometer Sends to Kyma Patcher part 1. When triggering sounds inside Kyma, Max/MSP must reset non-zero values back to zero in order to re-trigger a Kyma Sound object.

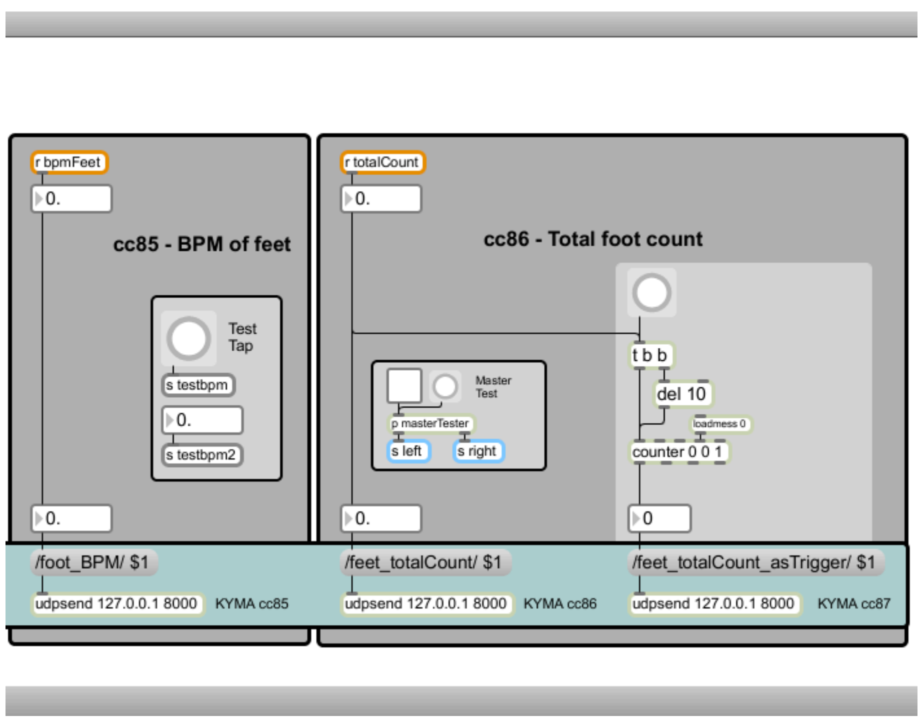



Figure A.1.b.15. Accelerometer Sends to Kyma Patcher part 2

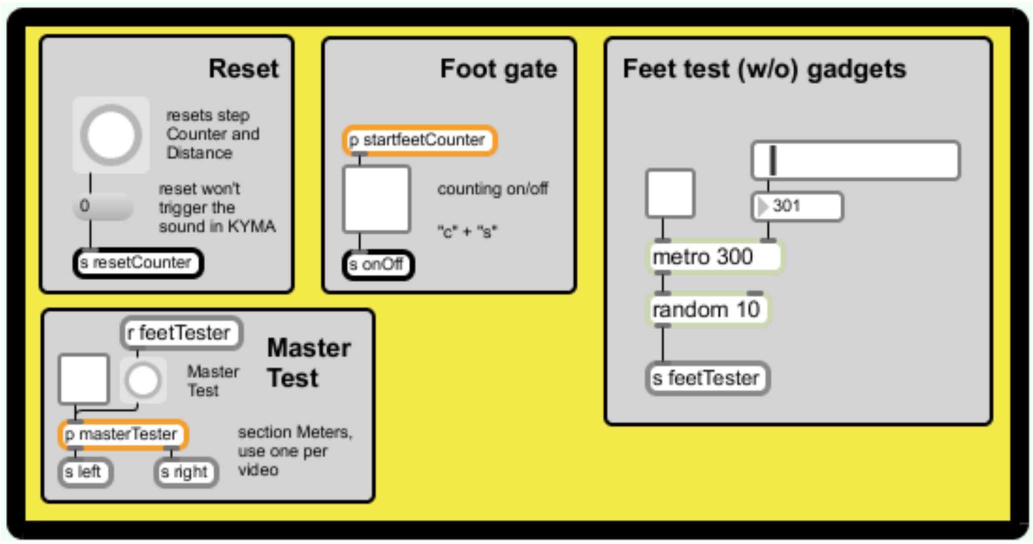



Figure A.1.b.16. Master Accelerometer Control Module 3

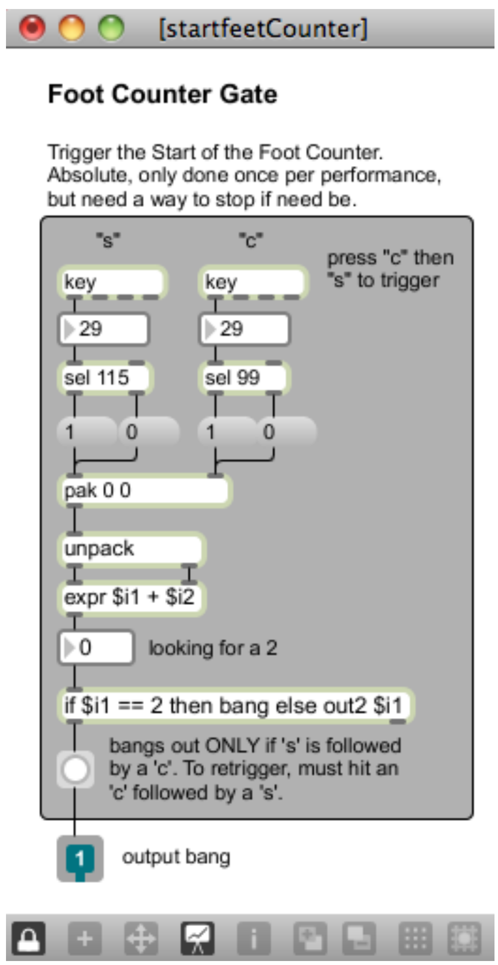



Figure A.1.b.17. Master Feet Counter Control Patcher

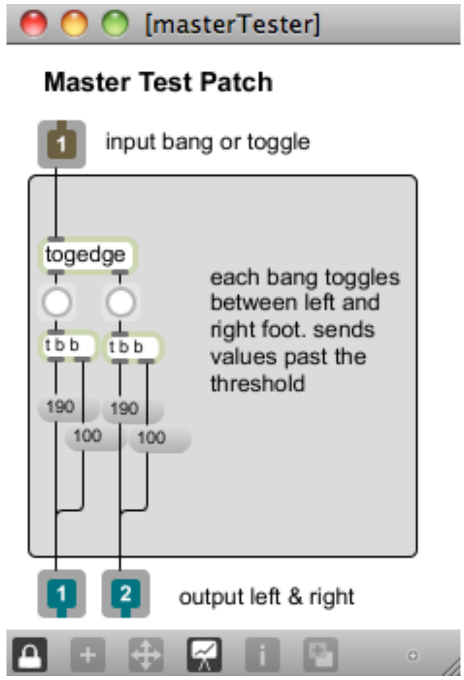



Figure A.1.b.18. Master Feet Counter Test Patcher

A.1.c. Heart Rate Monitor

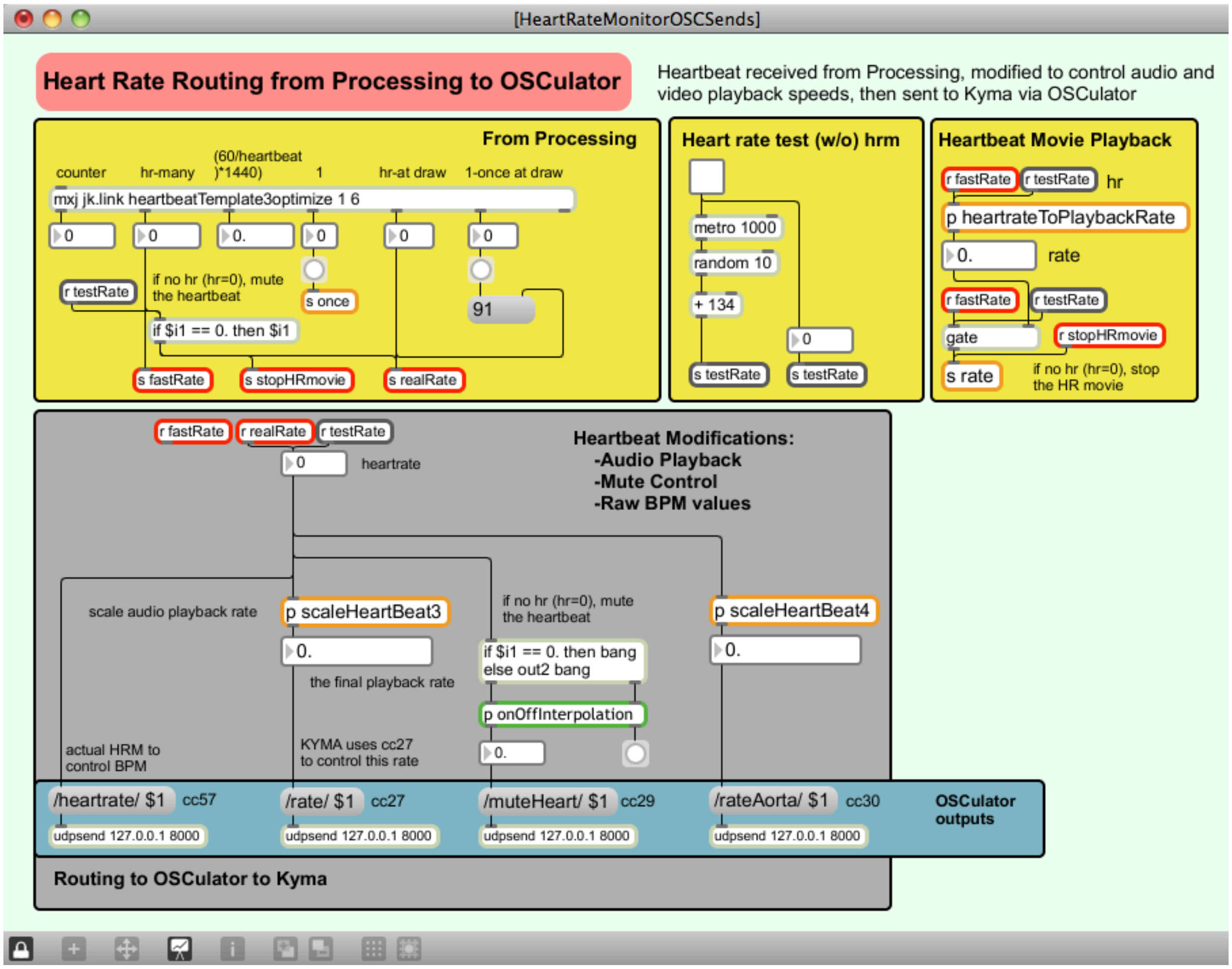



Figure A.1.c.1. Heart Rate Routing Patch Window. Communication shows information received from Processing, and routing to OSCulator.

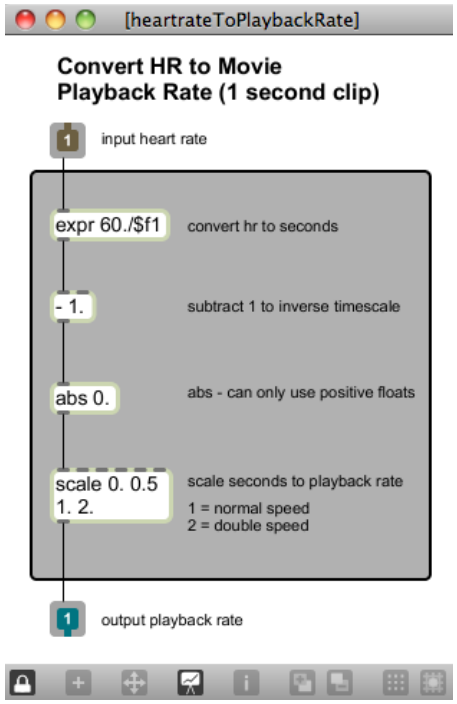



Figure A.1.c.2. Heart Rate Controls
Movie Playback Patcher

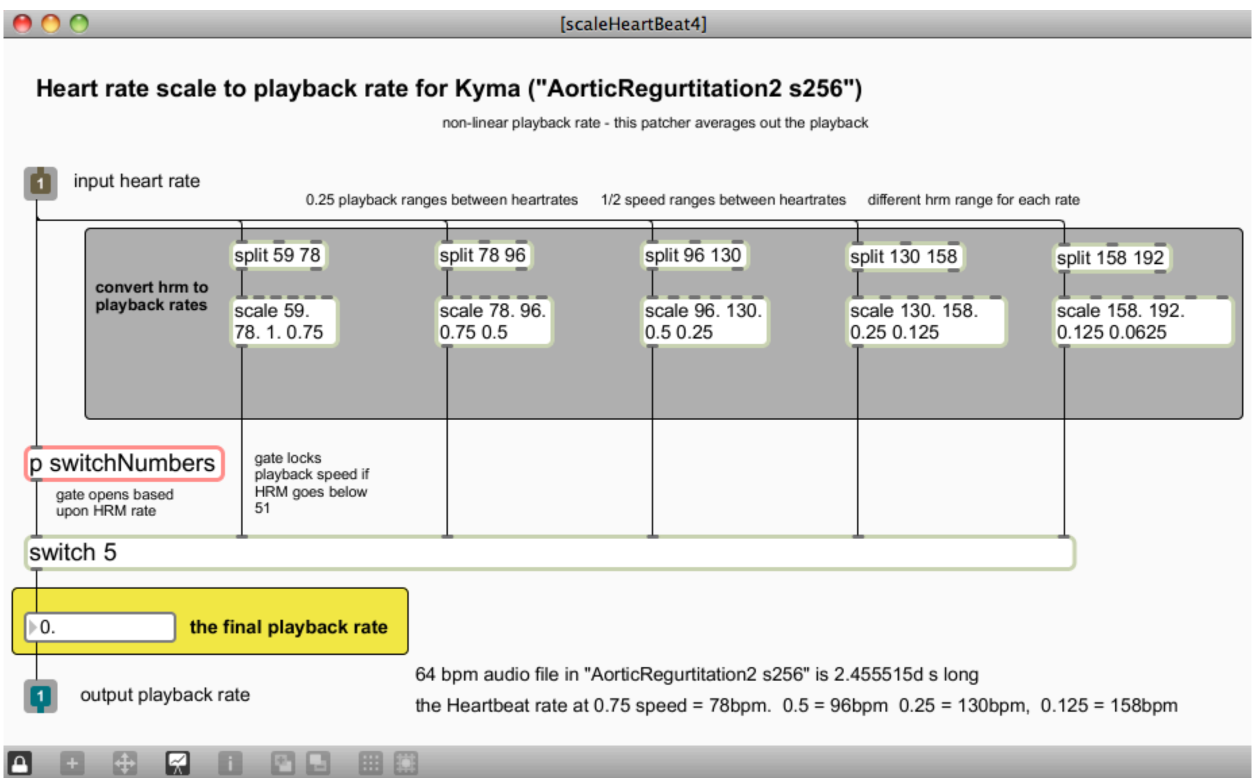



Figure A.1.c.3. Heart Rate Controls Heartbeat/Aorta Audio Playback Patcher

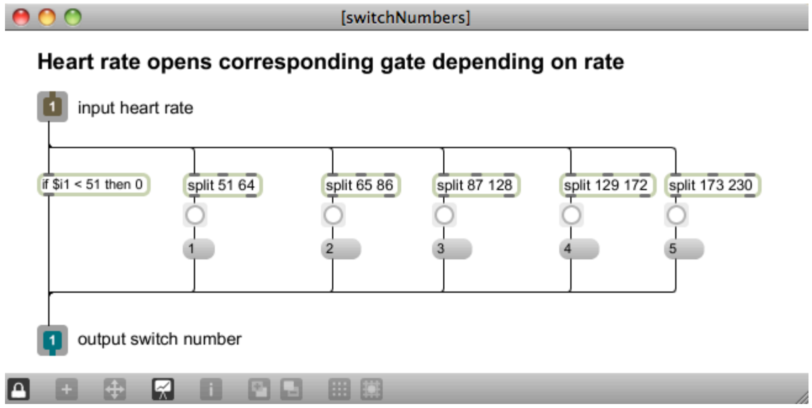



Figure A.1.c.4. Heart Rate Controls Switch Gate Patcher

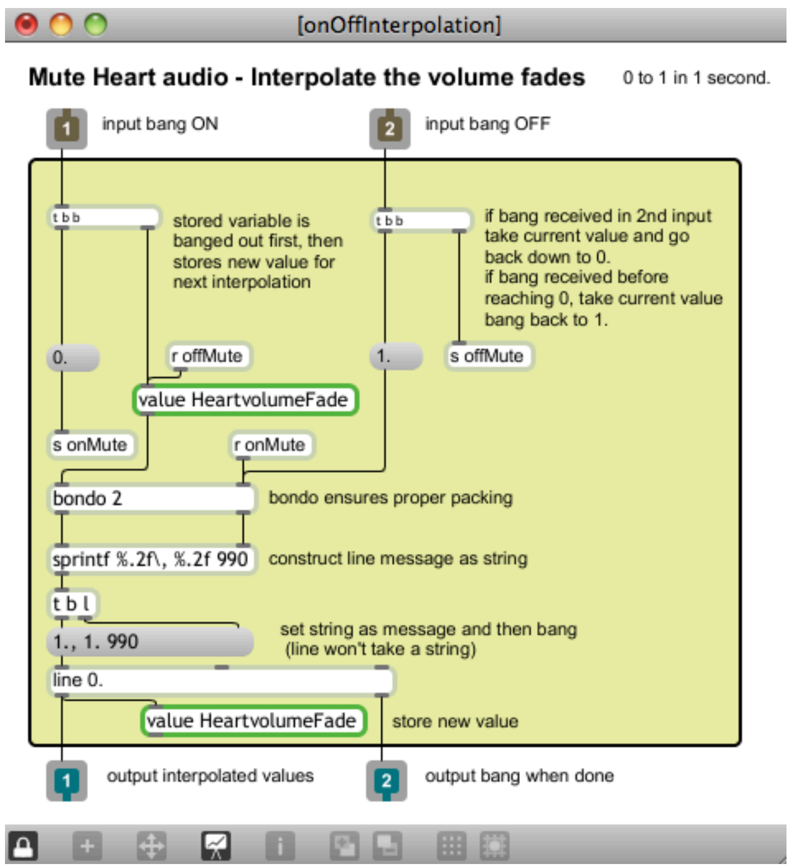



Figure A.1.c.5. Master Heartbeat Audio Volume Control Patcher

A.1.d. Control Window

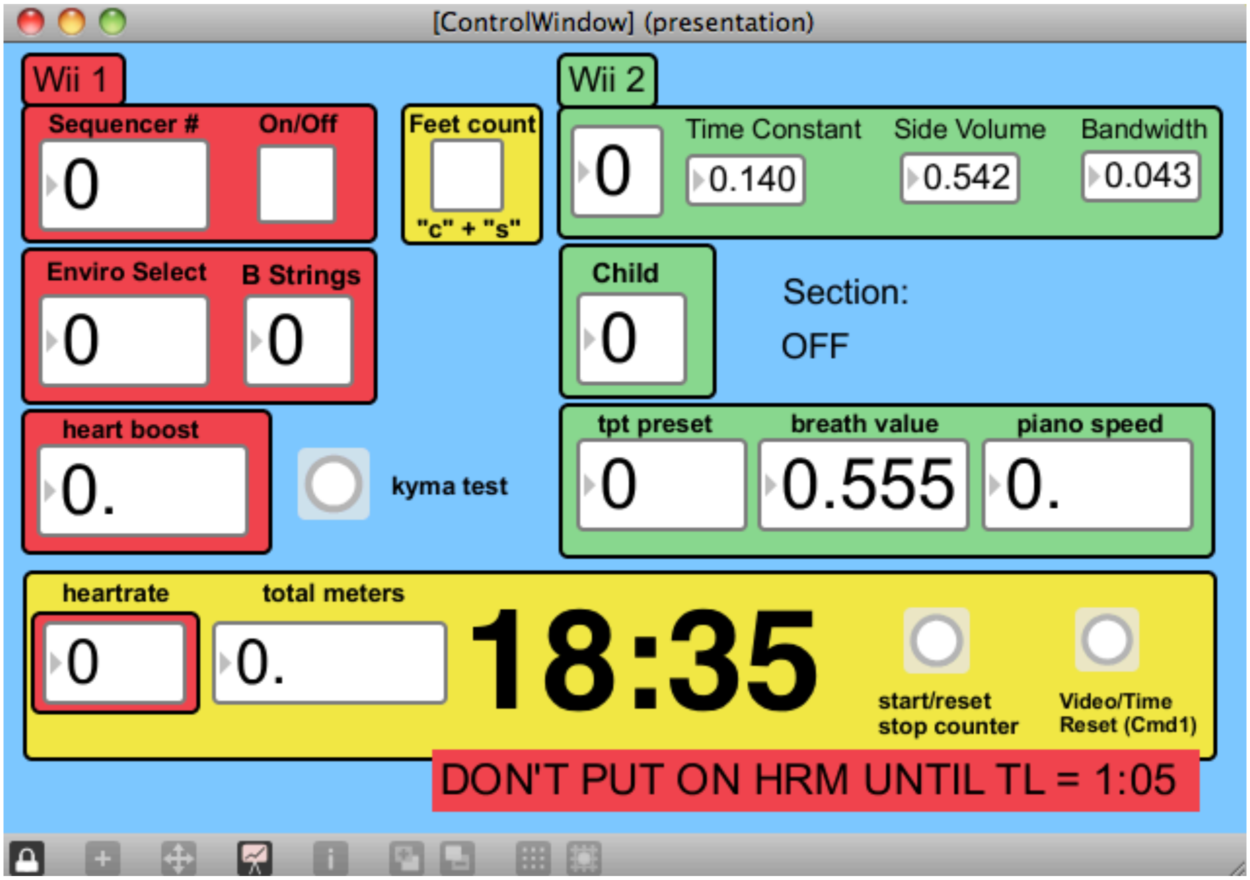



Figure A.1.d.1. Performance Control Patch Window, in Presentation mode

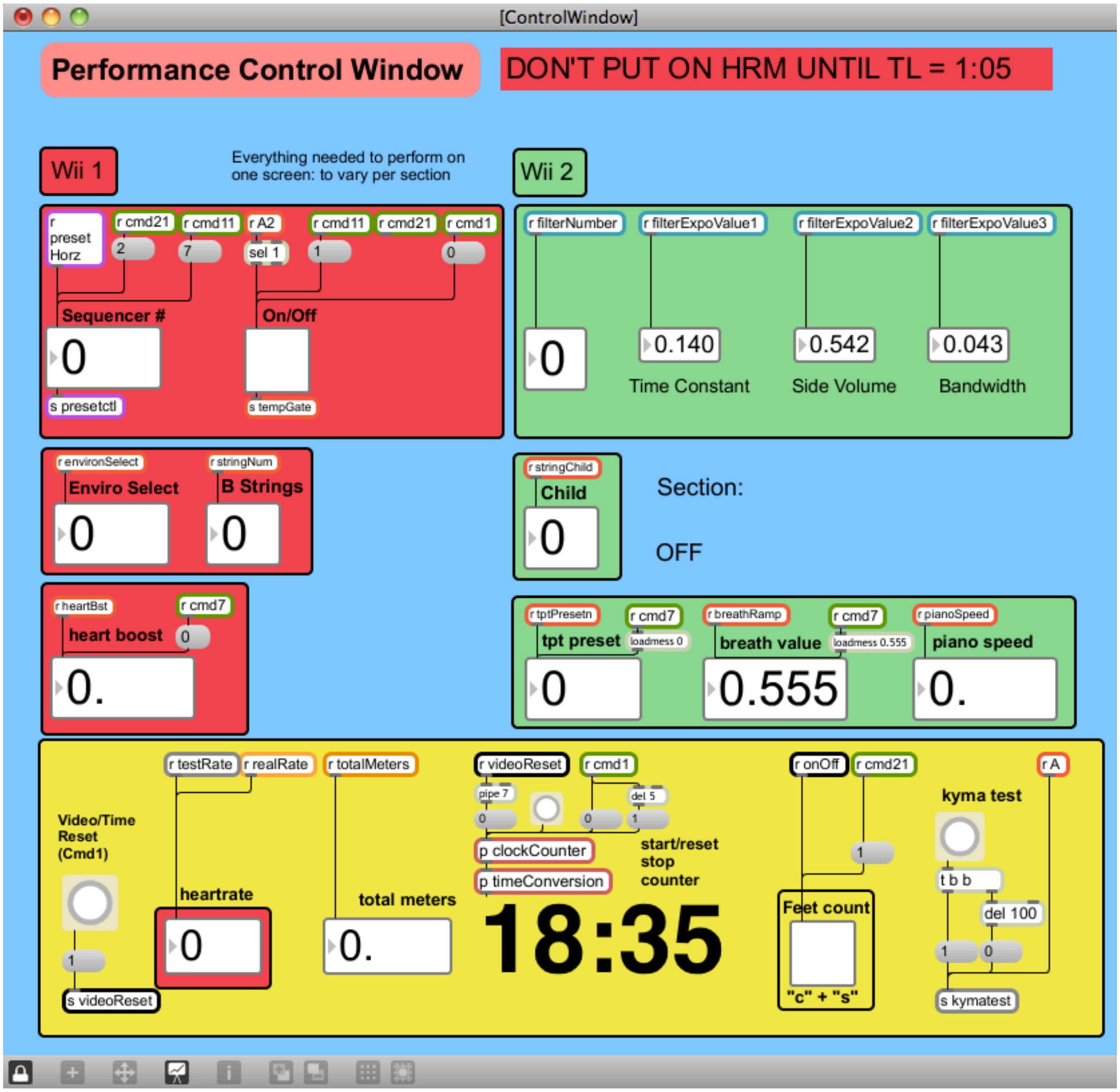



Figure A.1.d.2. Performance Control Patch Window, in Performance mode

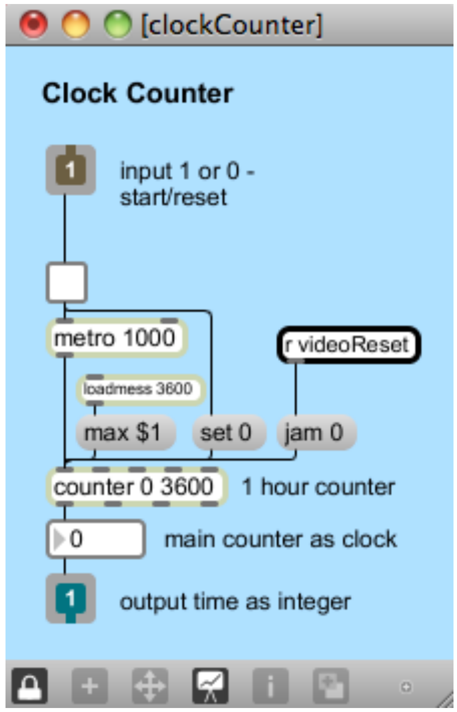



Figure A.1.d.3. Performance Control
Timer as Counter Patcher


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

A.1.e. MIDI

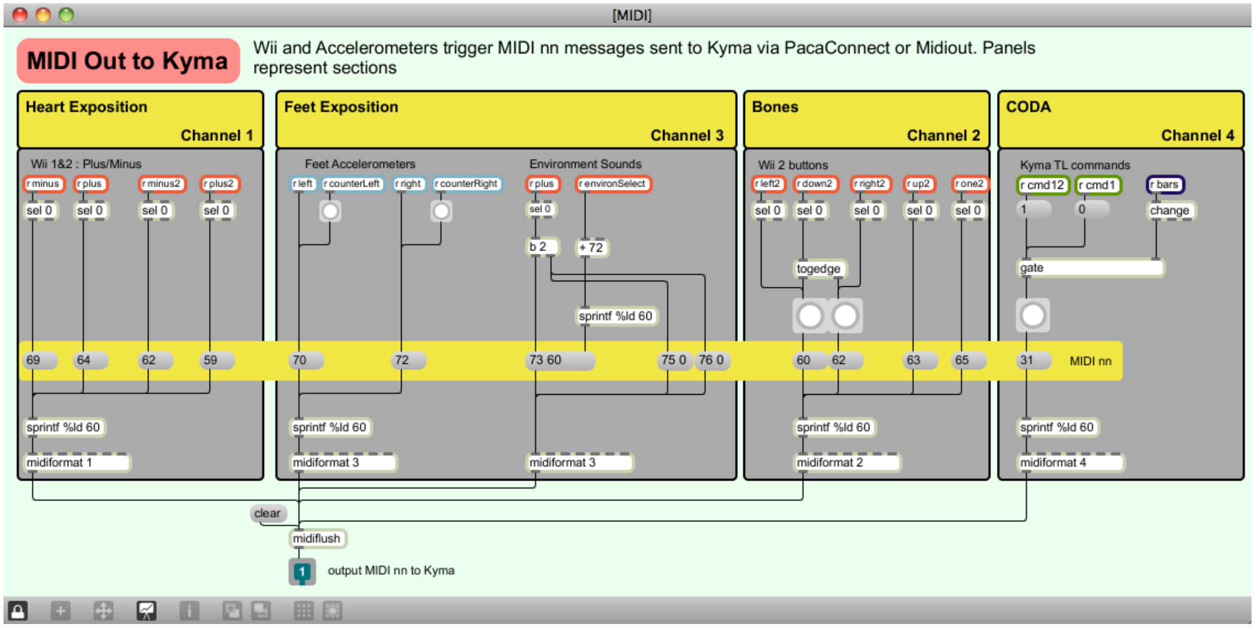



Figure A.1.e.1. MIDI Controls Patch Window, all Make Note messages Sent to Kyma via PacaConnect.

A.1.f. Video Control

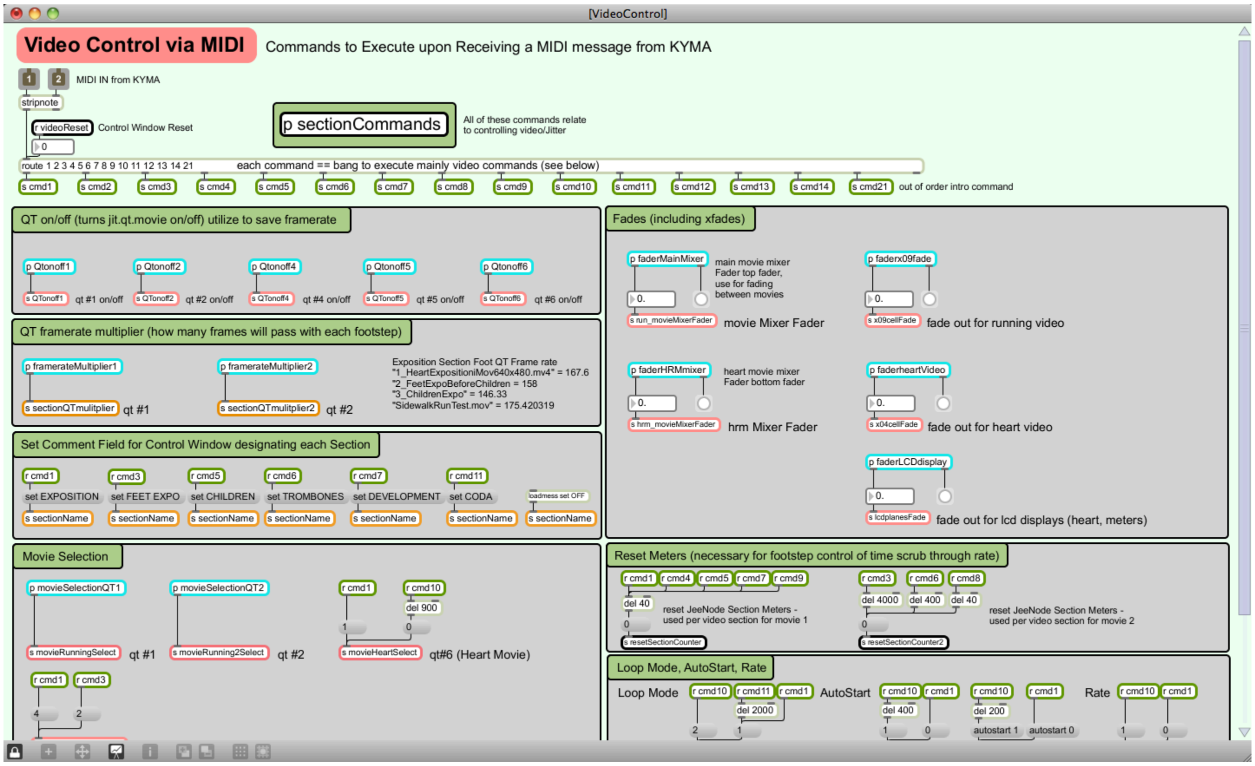



Figure A.1.f.1. Video Control Patch Window, overview of Window layout

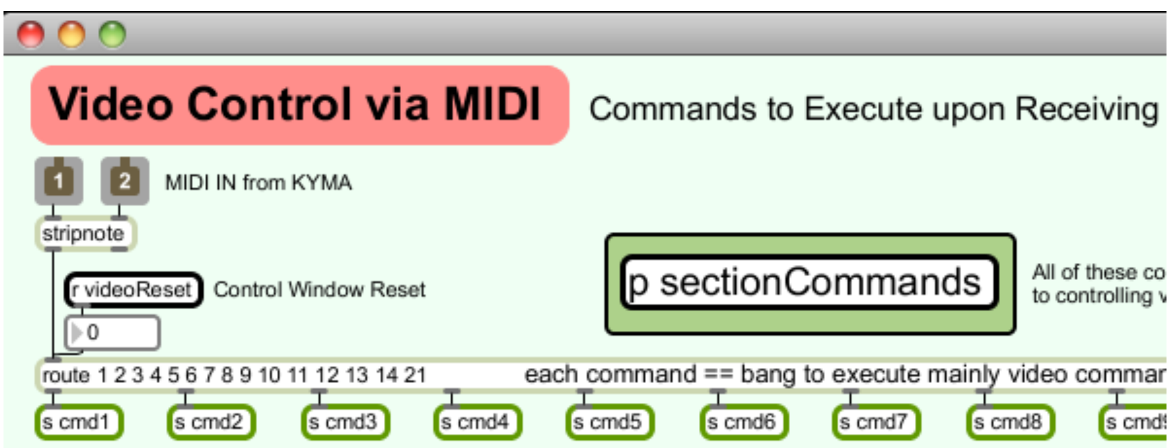



Figure A.1.f.2. Video Control MIDI routing, part 1

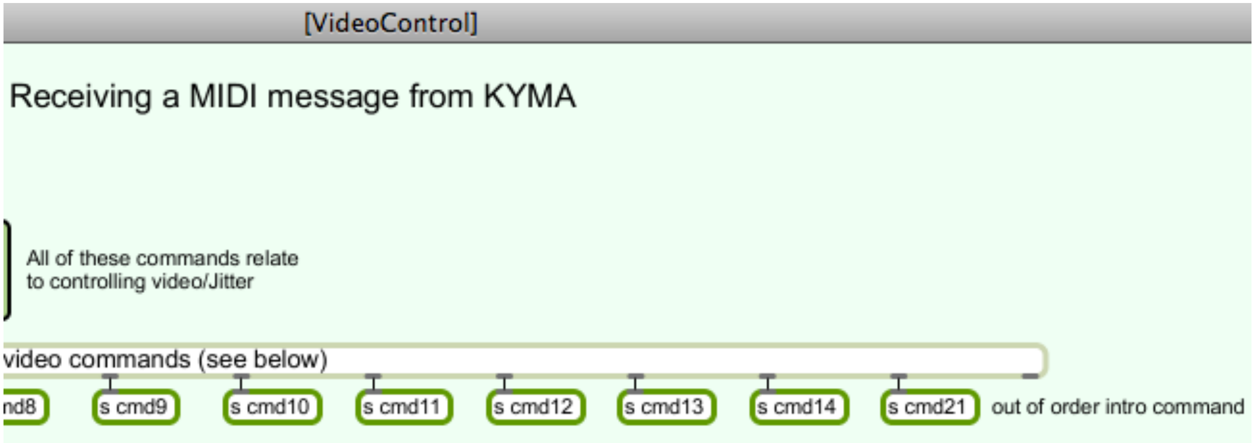



Figure A.1.f.3. Video Control MIDI routing, part 2

[sectionCommands]

## 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.
qt # - 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

3. post HeartExposition
qt # - mov2 on
movie select -mov2= 3, MixerRunningB = src2(#2)
jitter 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

4. before selectASound
qt # - mov1 off
qt multiplier - mov1 = 154.55
movie select -mov1= 0
jitter on/off - mov1 off    turns off movie two
------
reset section meters - mov1 = reset

5. Children Exposition
qt # - 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

6. Trombones
qt # - 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
Fade out movie 1    quick fade in once 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

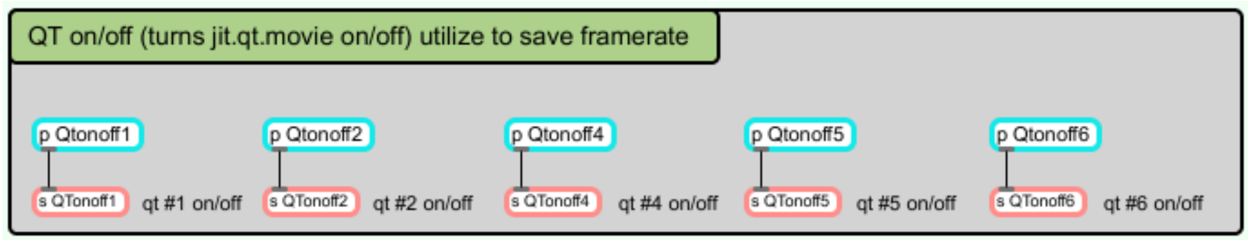



Figure A.1.f.5. QuickTime Movie 'qmetro' Toggle Module

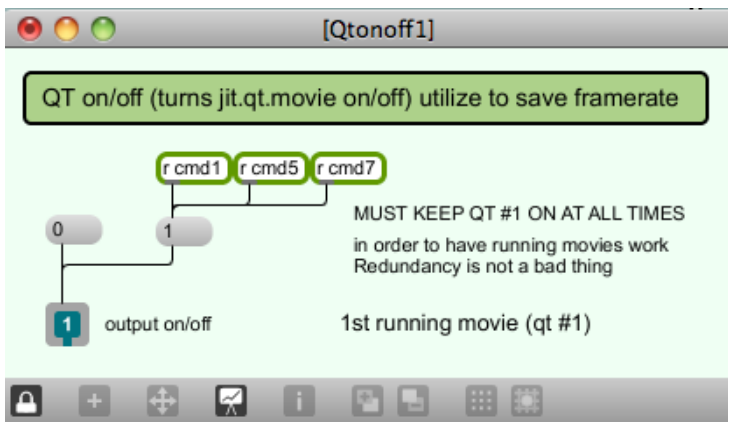



Figure A.1.f.6. QuickTime Movie #1 'qmetro' Toggle Patcher. Controls Running Movie #1.

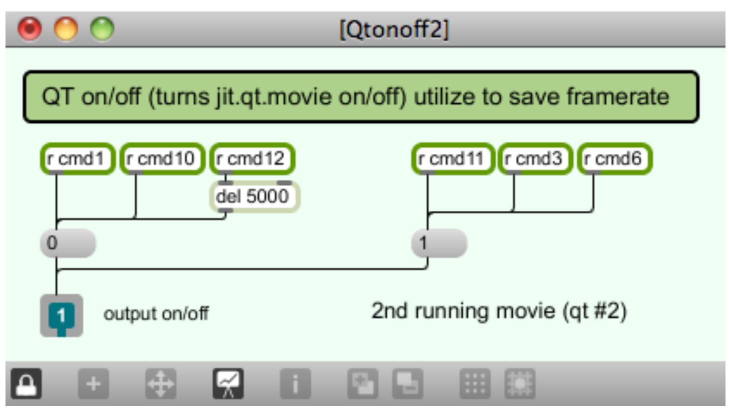



Figure A.1.f.7. QuickTime Movie #2 'qmetro' Toggle Patcher. Controls Running Movie #2.

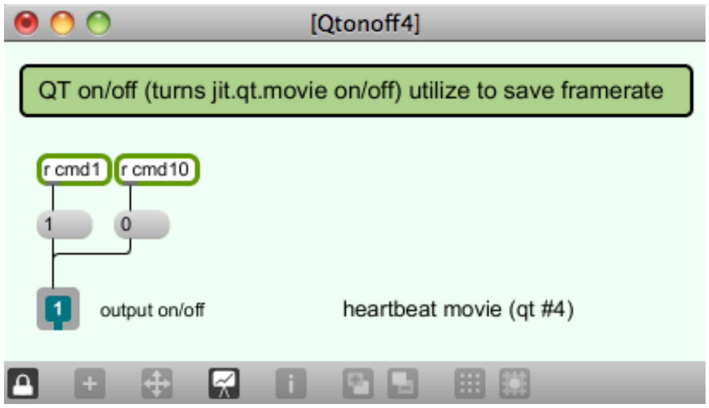



Figure A.1.f.8. QuickTime Movie #4 ‘qmetro’ Toggle Patcher, note Movie #3 does not exist. Movie heartbeat.

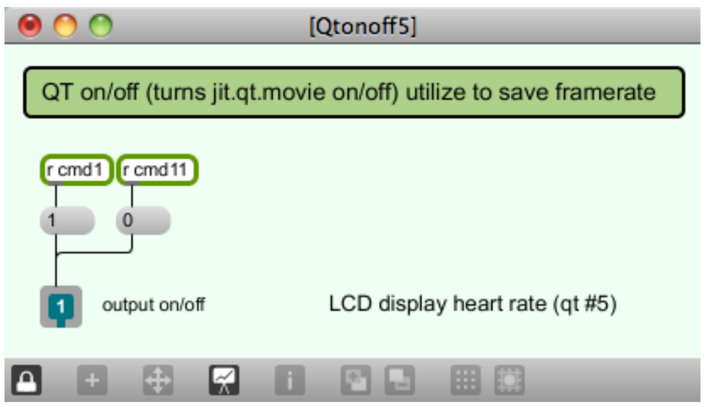



Figure A.1.f.9. QuickTime Movie #5 ‘qmetro’ Toggle Patcher. LCD display heart rate.

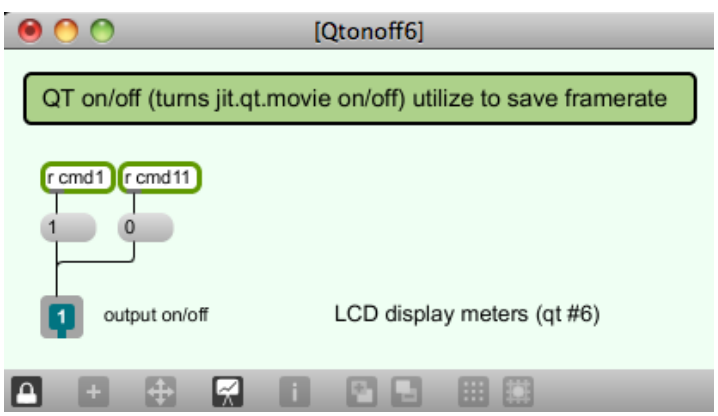



Figure A.1.f.10. QuickTime Movie #6 ‘qmetro’ Toggle Patcher. LCD display meters.

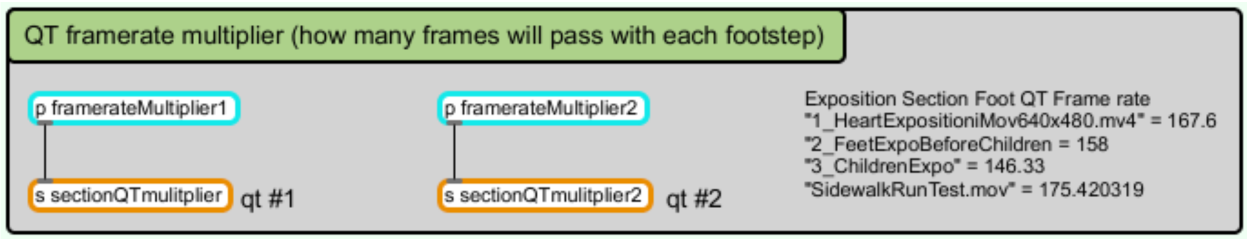



Figure A.1.f.11. QuickTime Frame Rate Multiplier Control Module

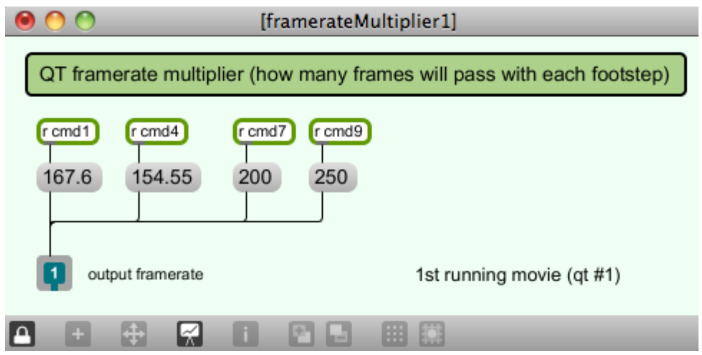



Figure A.1.f.12. QuickTime Movie #1 Frame Rate Multiplier Control Patcher

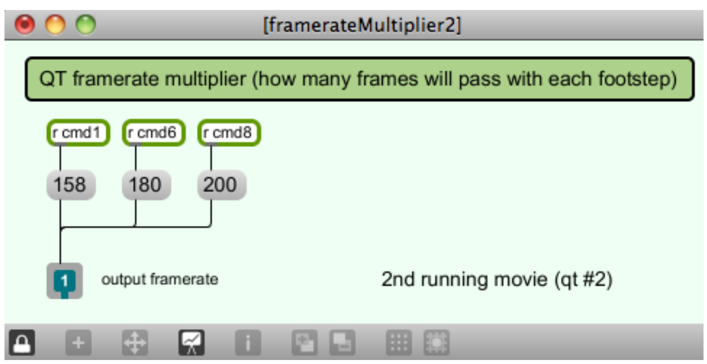



Figure A.1.f.13. QuickTime Movie #2 Frame Rate Multiplier Control Patcher

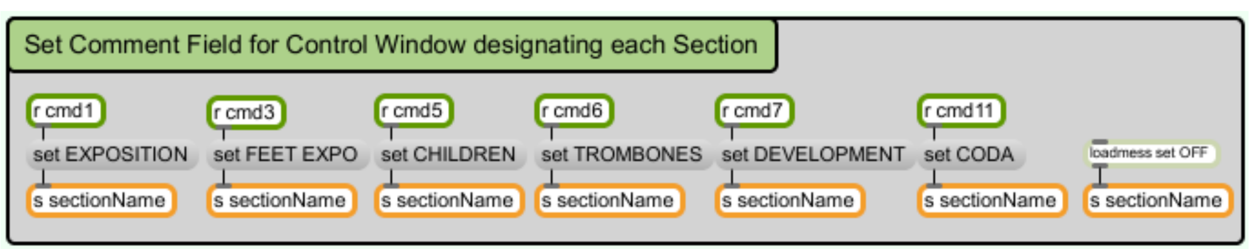



Figure A.1.f.14. Performance Control Window Comment Field Module

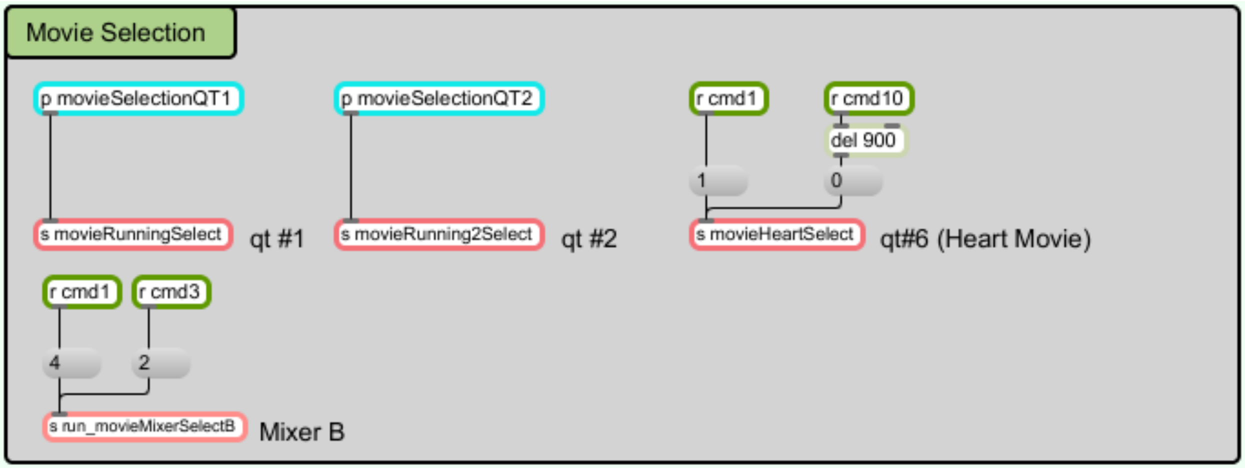



Figure A.1.f.15. QuickTime Movie Selection Module

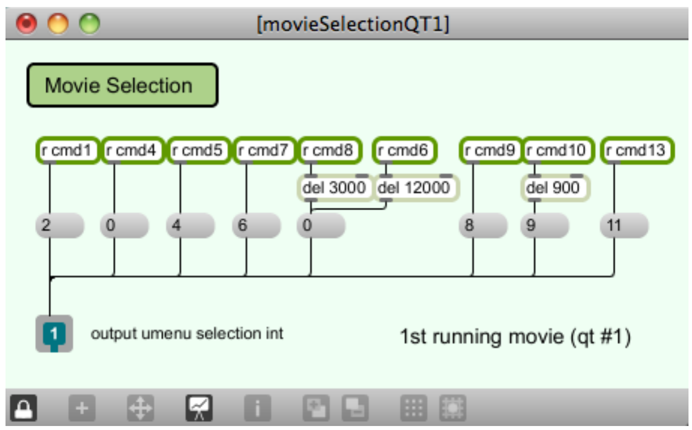



Figure A.1.f.16. QuickTime Movie #1 Selection Patcher

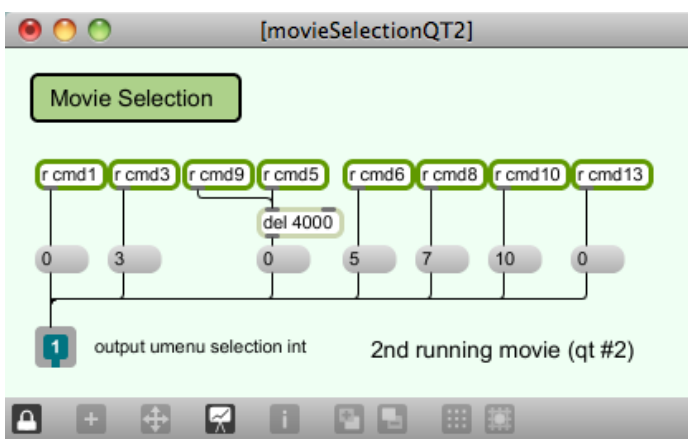



Figure A.1.f.17. QuickTime Movie #2 Selection Patcher

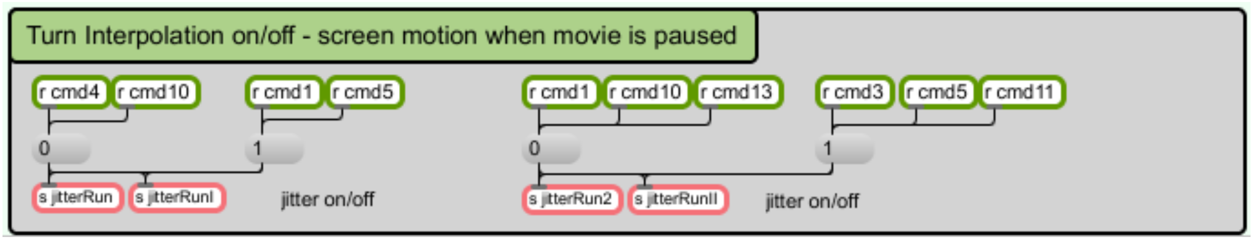



Figure A.1.f.18. QuickTime Movie 'srcrect' Pixel Jitter Toggle Module

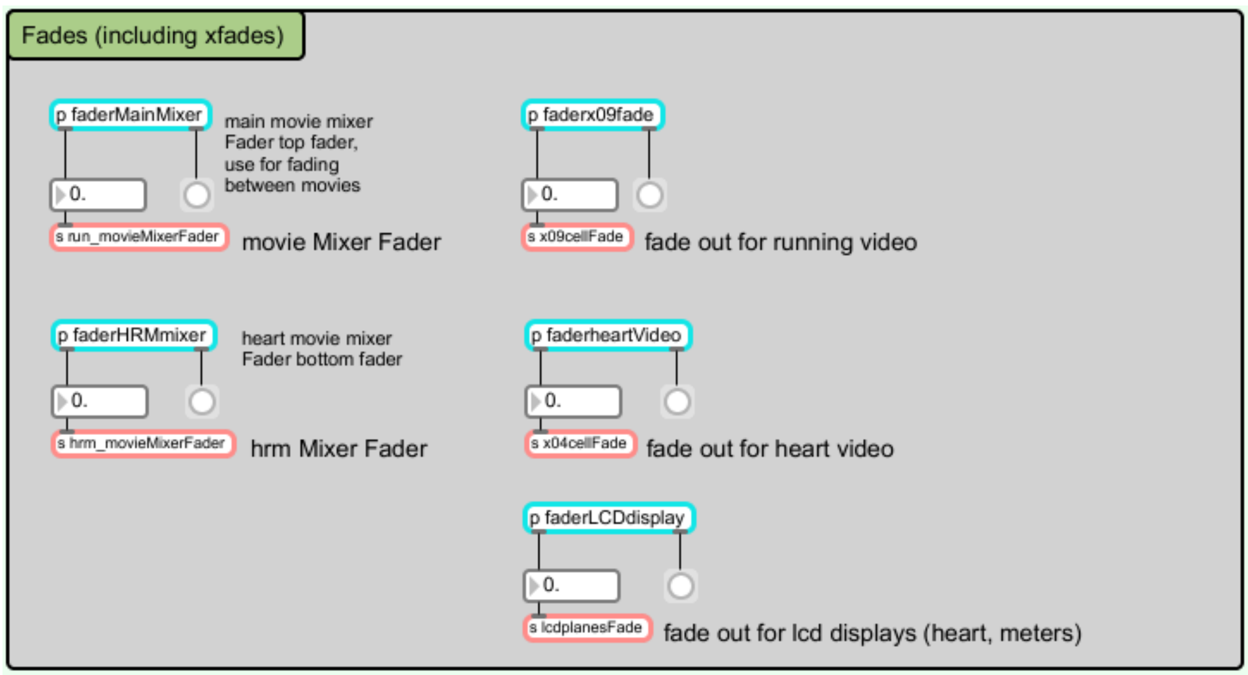



Figure A.1.f.19. QuickTime Movie Fade Control Module

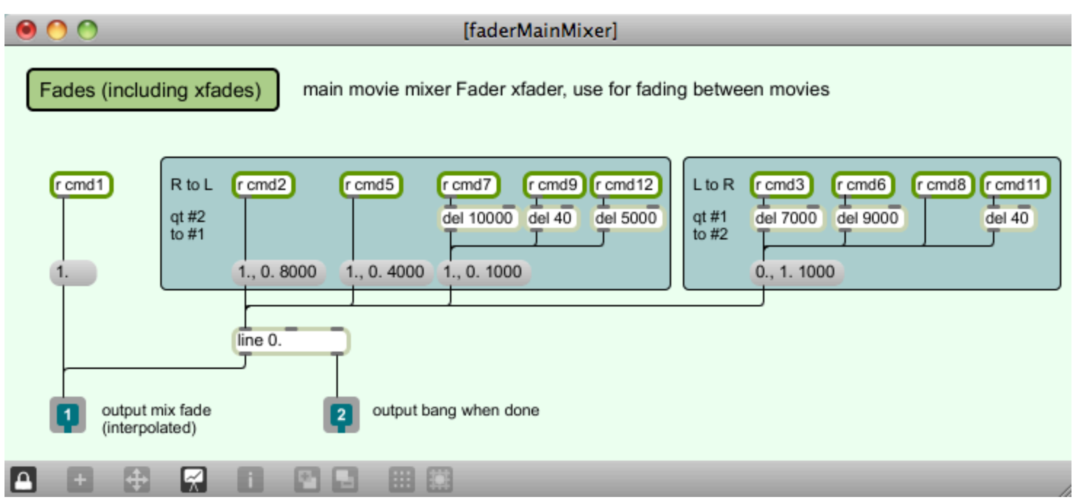



Figure A.1.f.20. QuickTime Movie Main Mixer Fade Control Patcher

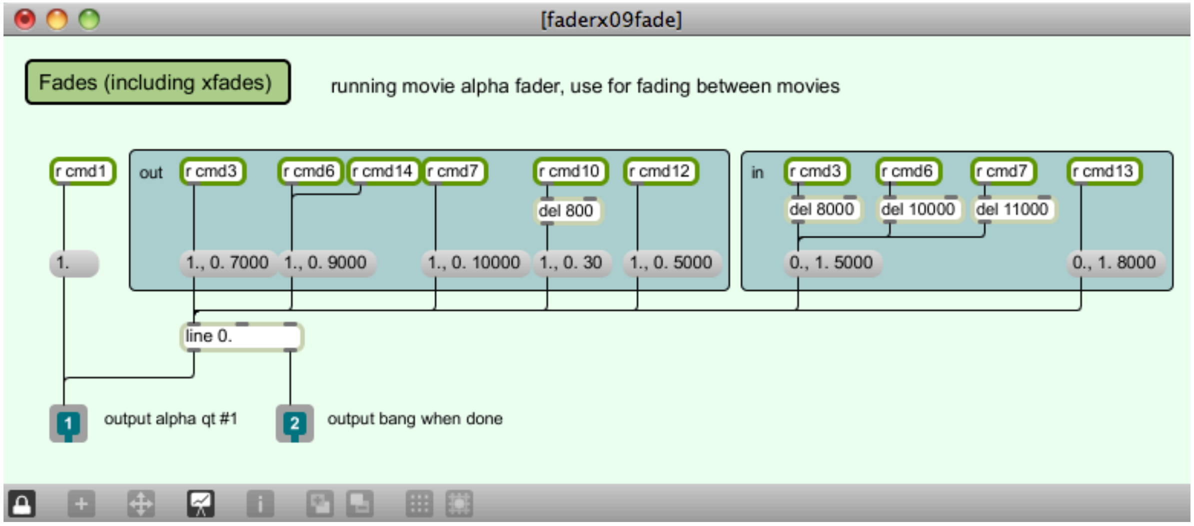



Figure A.1.f.21. QuickTime Movie Running Movie Fade Control Patcher

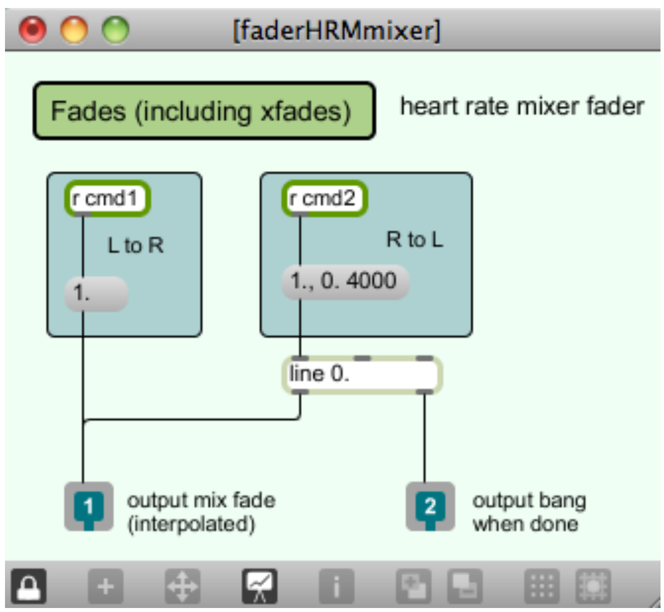



Figure A.1.f.22. QuickTime Movie Heartbeat Movie Mixer Fade Control Patcher

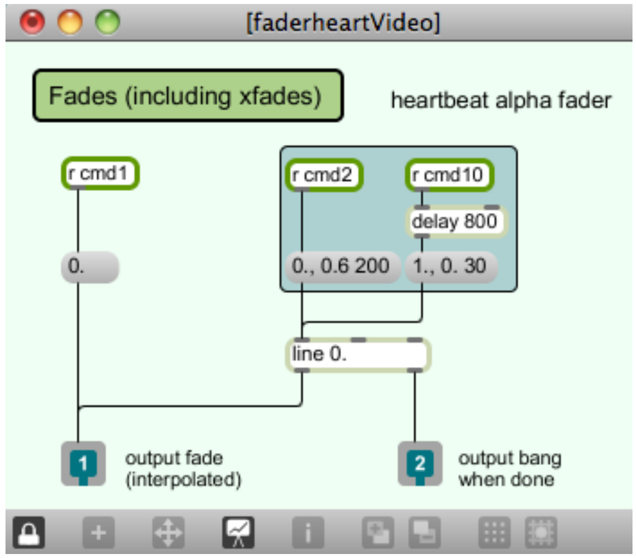



Figure A.1.f.23. QuickTime Movie Heartbeat Movie Fade Control Patcher

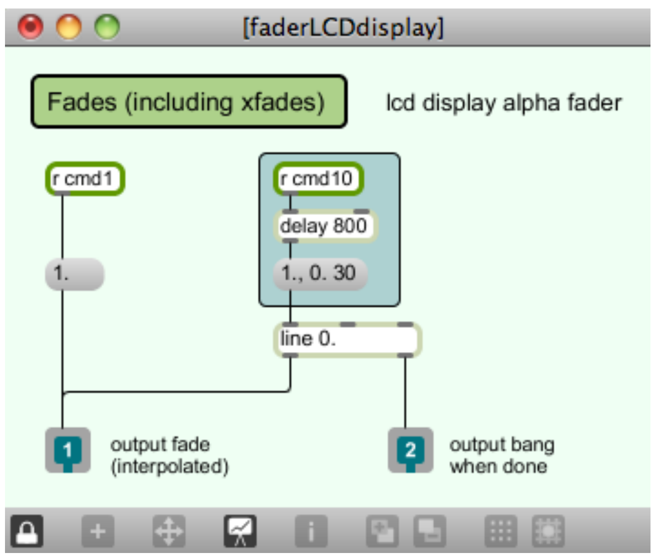



Figure A.1.f.24. LCD Display Fade Control Patcher

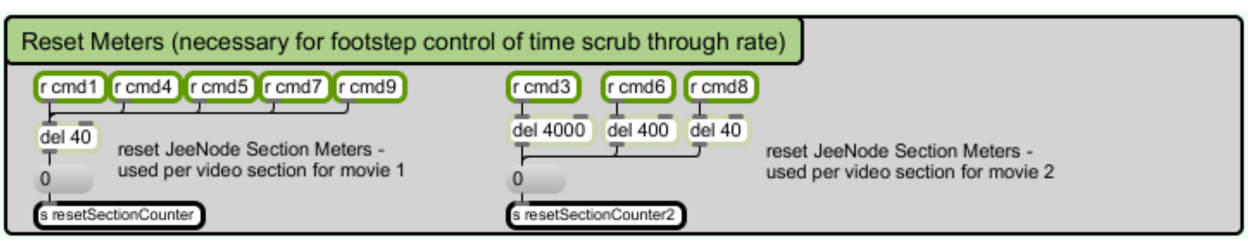



Figure A.1.f.25. Feet Accelerometer Section Distance Counter Reset Module

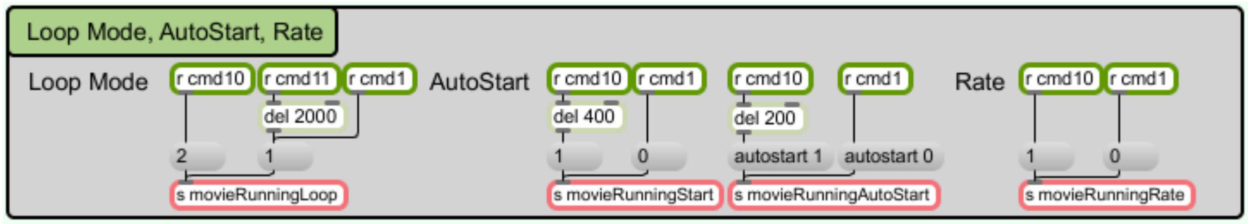



Figure A.1.f.26. Miscellaneous QuickTime Movie Control Module

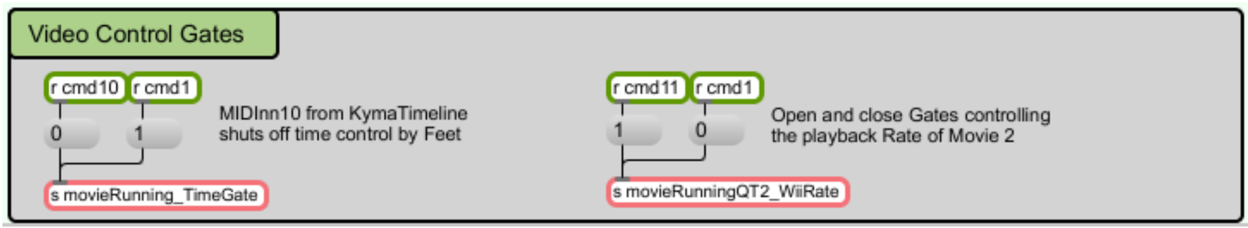



Figure A.1.f.27. Master Video Control Switch Module

## A.1.g. Wiimote Master

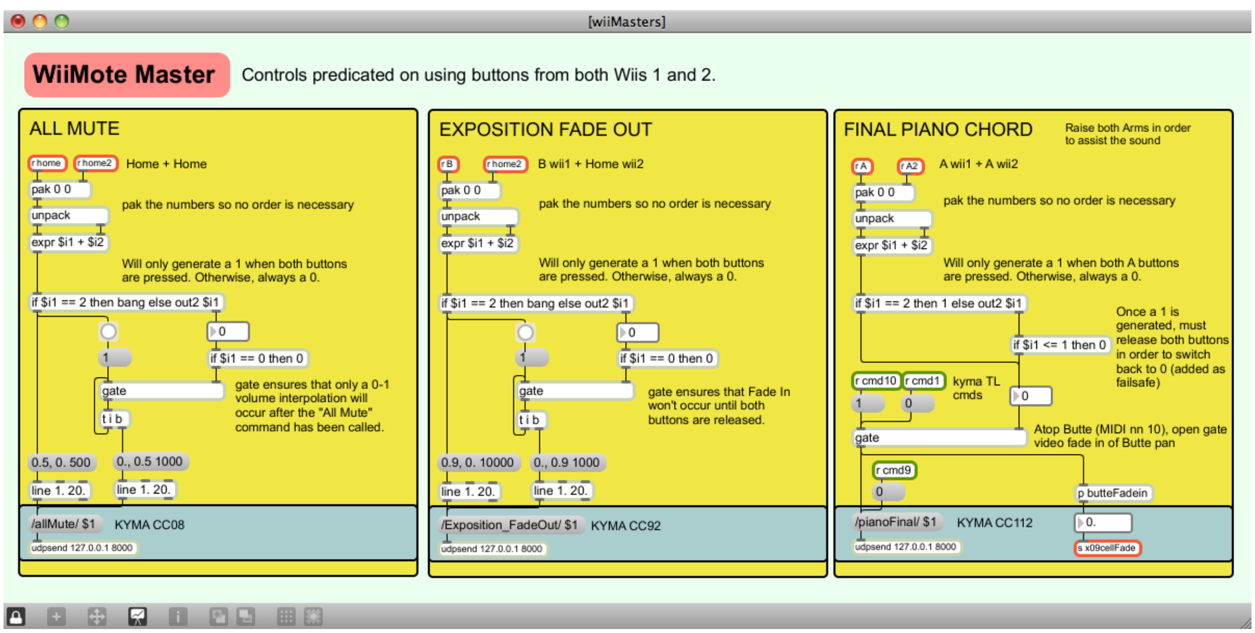



Figure A.1.g.1. Wiimote Master Control Patch Window

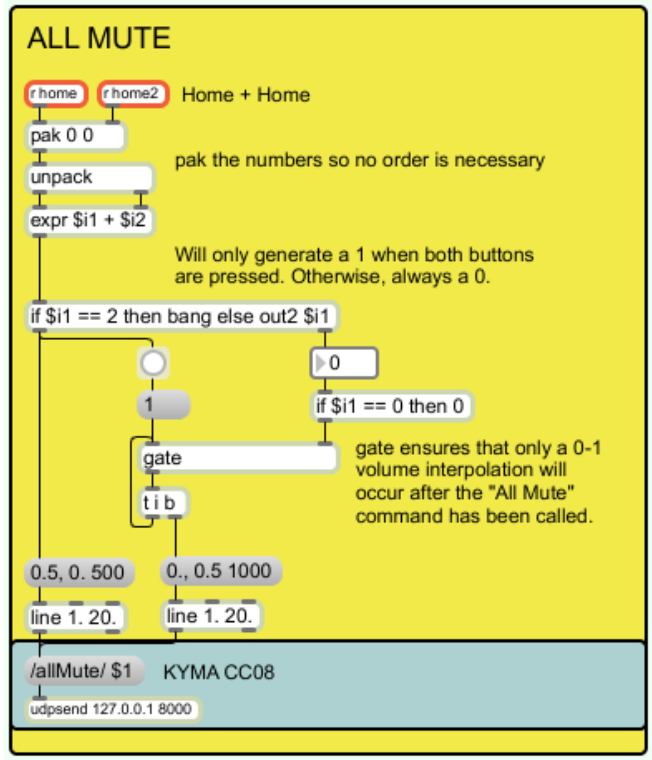



Figure A.1.g.2. All-Mute Wiimote Master Control Module

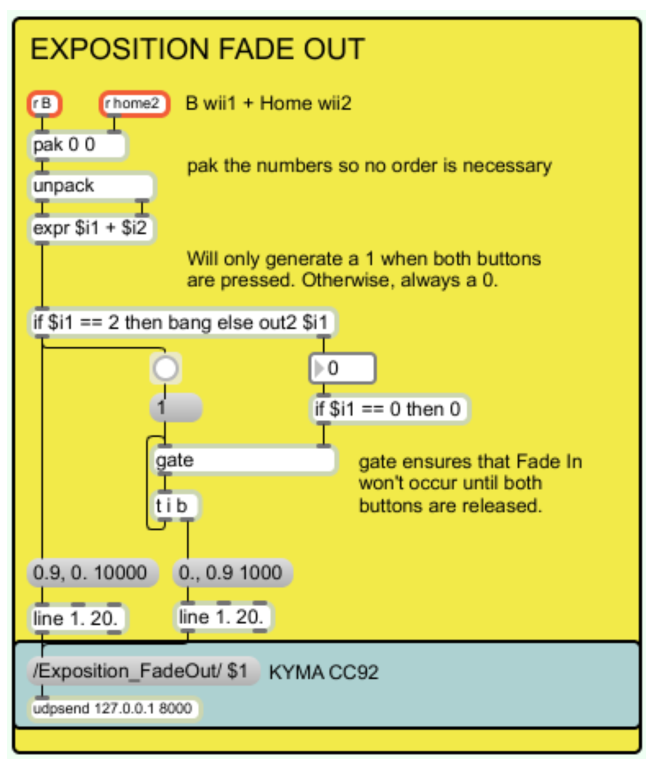



Figure A.1.g.3. Exposition Fade-Out Wiimote Master Control Module

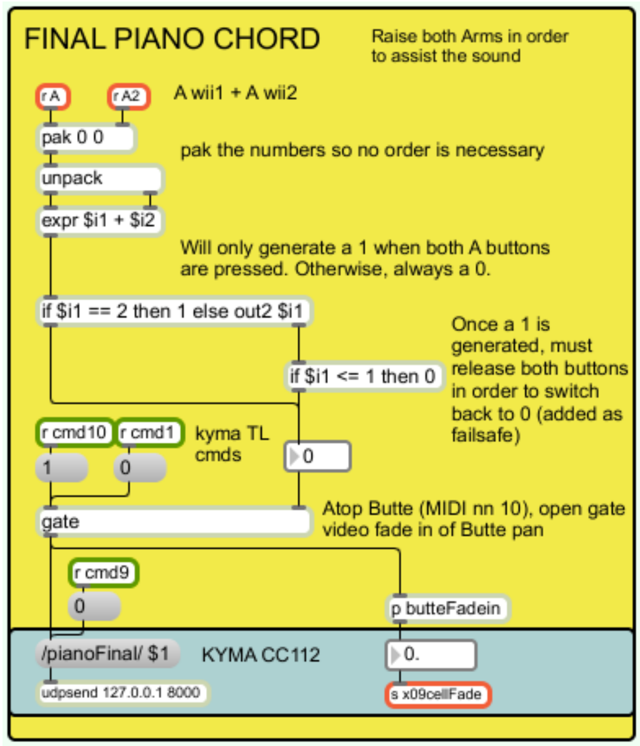



Figure A.1.g.4. Final Piano Chord Wiimote Master Control Module

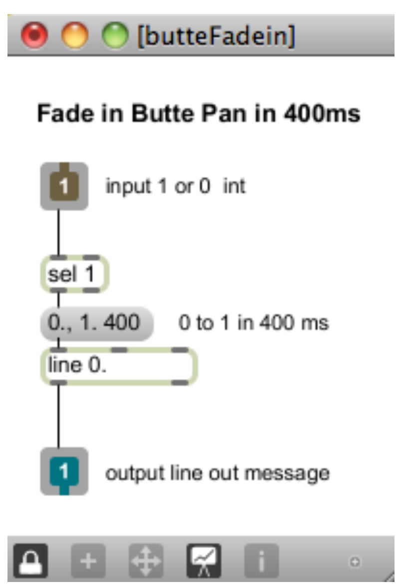



Figure A.1.g.5. Butte Pan Video Fade-In Patcher

## A.1.h. Wiimote 1

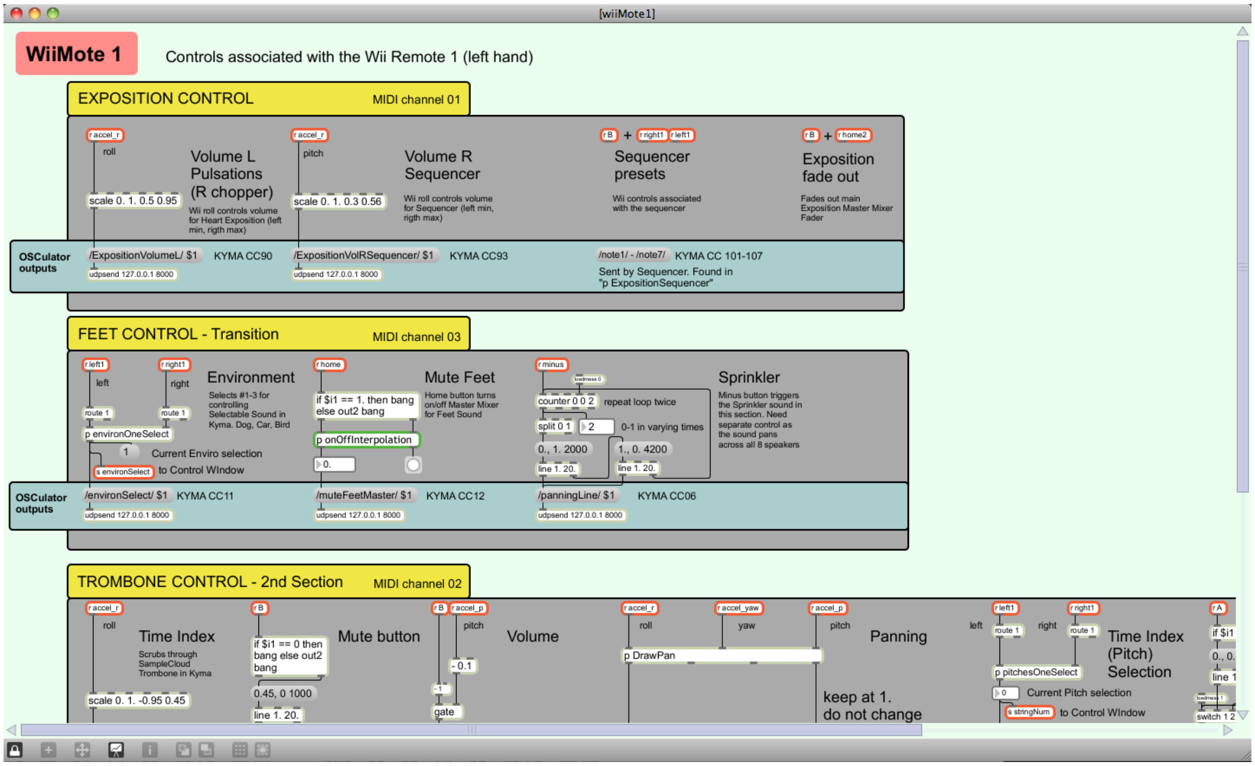



Figure A.1.h.1. Wiimote 1 Control Patch Window, overview of Window layout

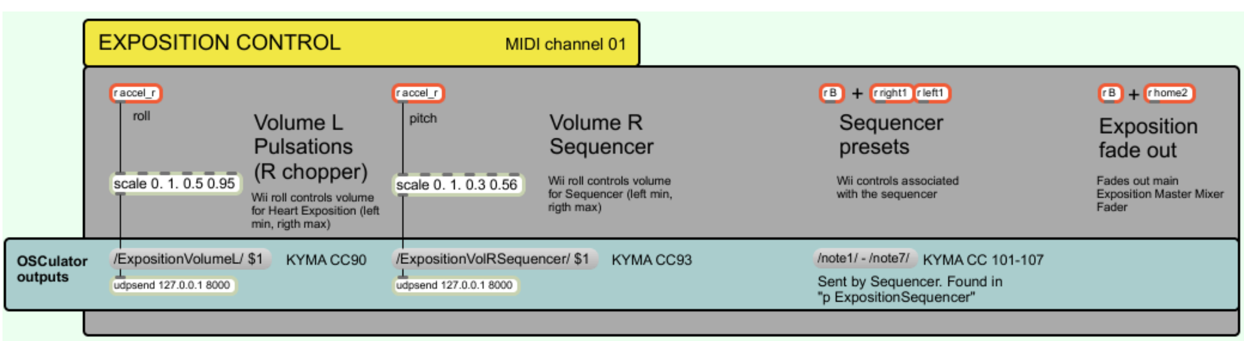



Figure A.1.h.2. Wiimote 1 Heart Rate Monitor Exposition Control Module

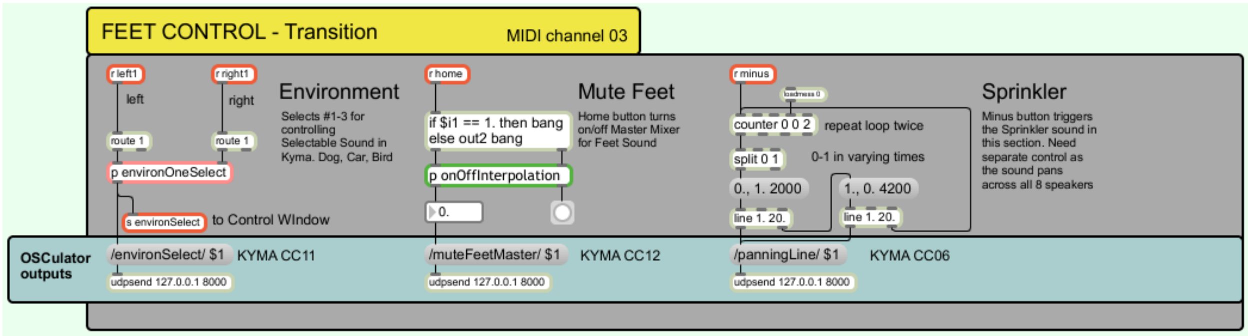



Figure A.1.h.3. Wiimote 1 Feet Exposition Control Module

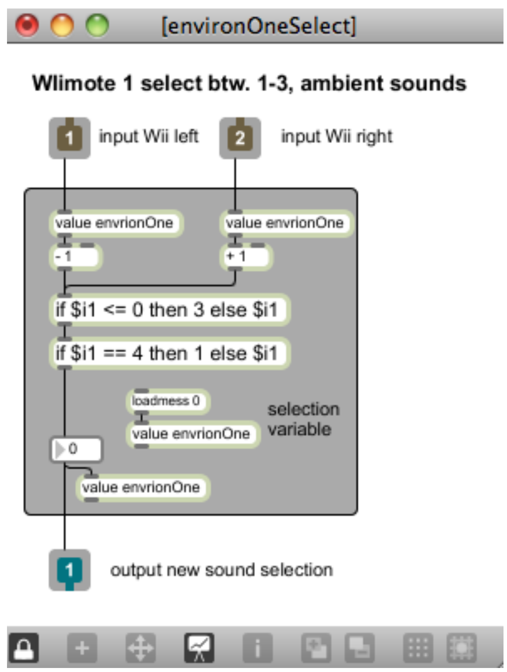



Figure A.1.h.4. Environment Sound Select Patcher, in Feet Exposition

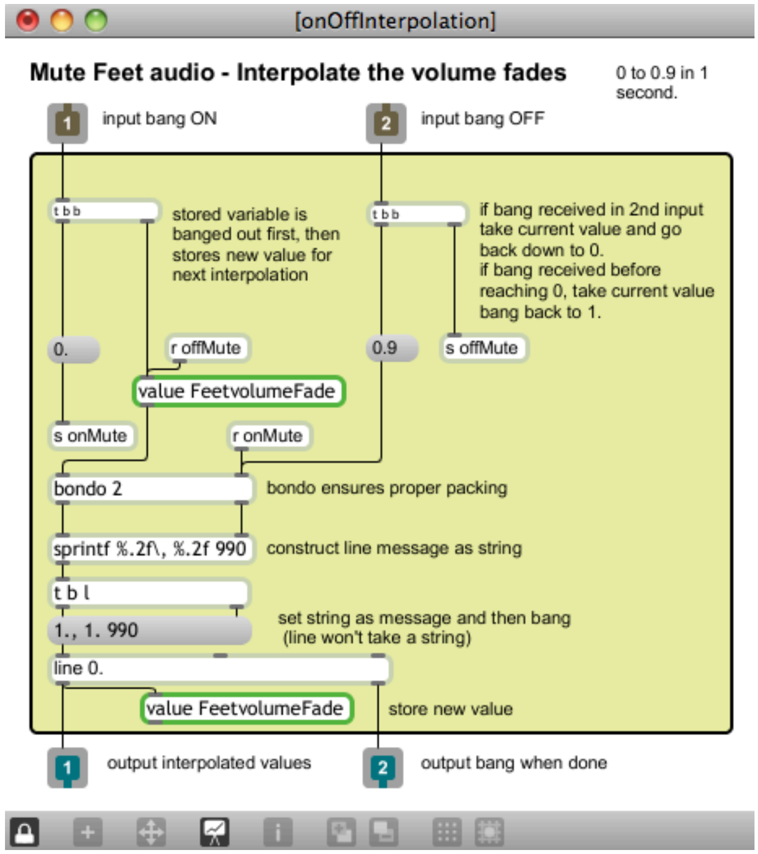



Figure A.1.h.5. Feet Sound Mute Patcher, in Feet Exposition

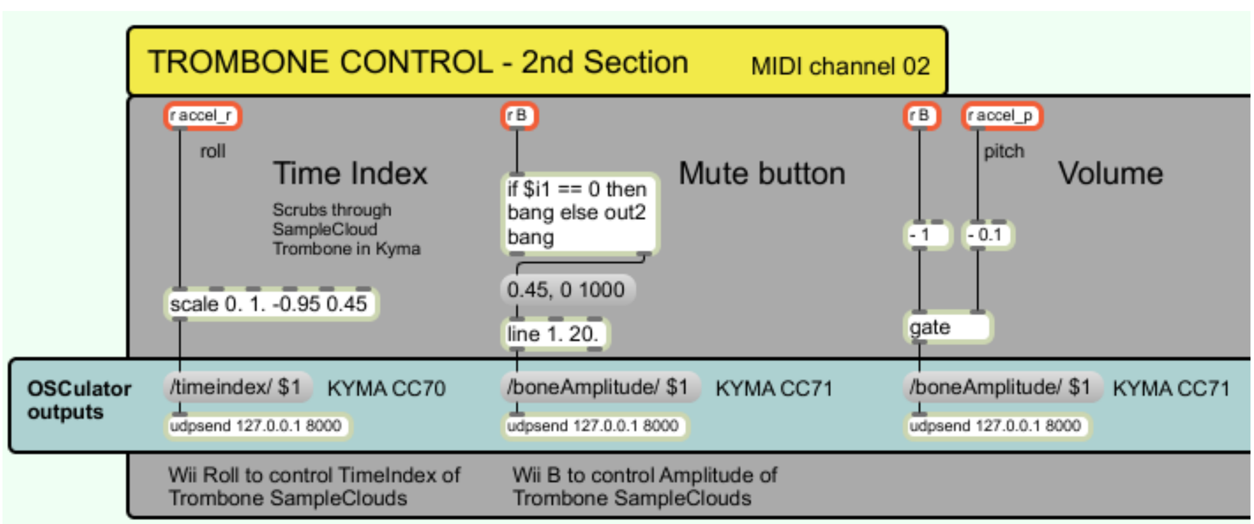



Figure A.1.h.6. Wiimote 1 Development Section Control Module, part 1

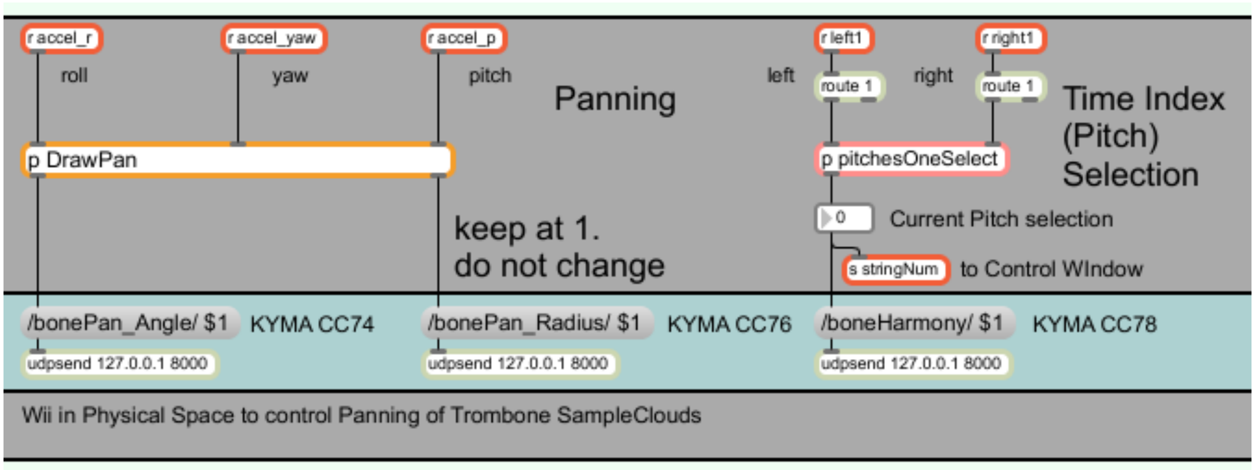



Figure A.1.h.7. Wiimote 1 Development Section Control Module, part 2

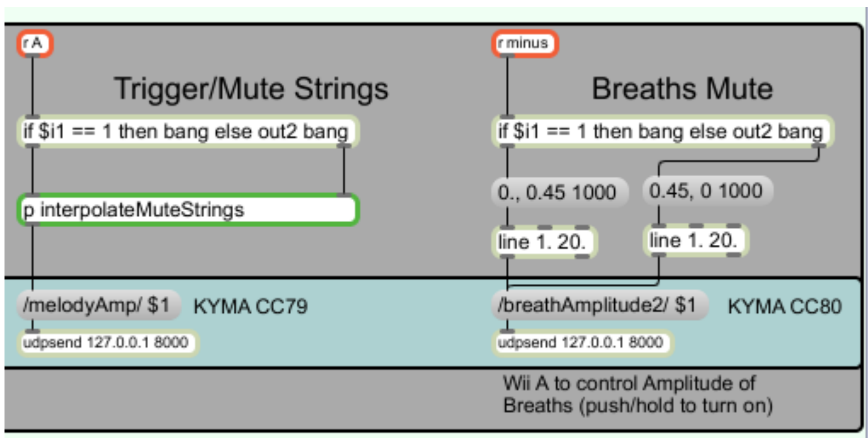



Figure A.1.h.8. Wiimote 1 Development Section Control Module, part 3

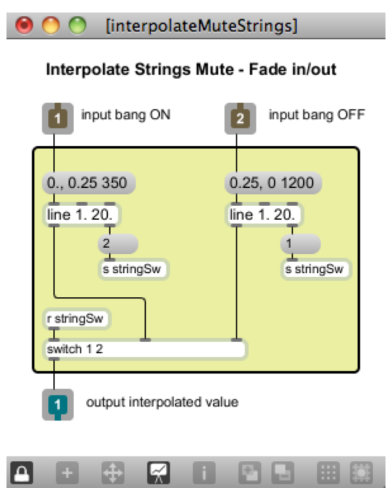



Figure A.1.h.9. String Mute Patcher

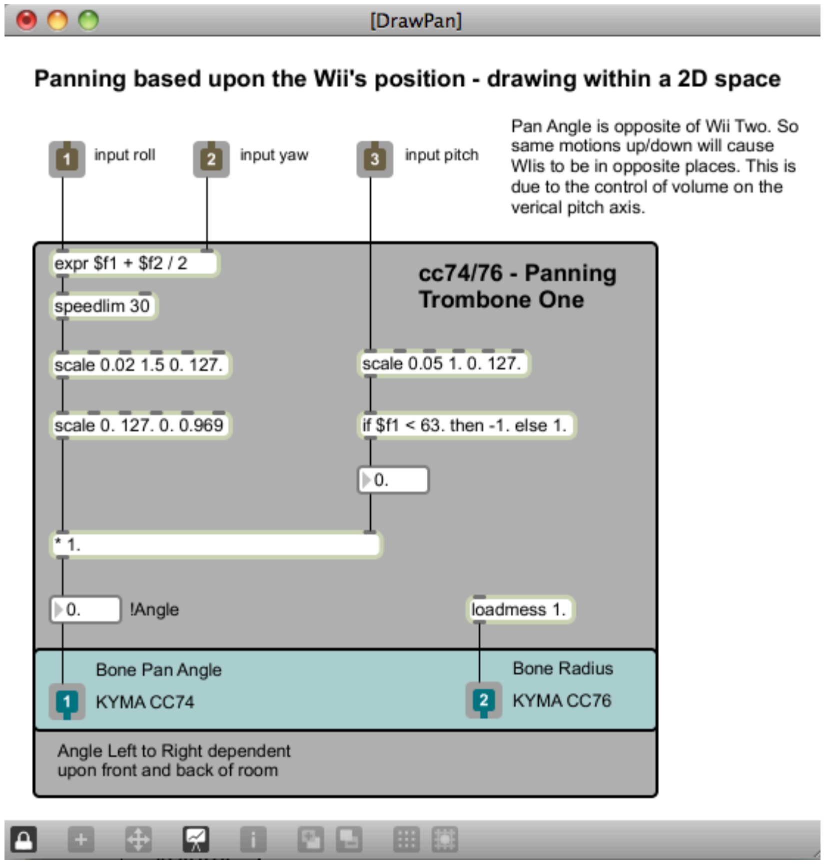



Figure A.1.h.10. Panning of Trombones Control Patcher

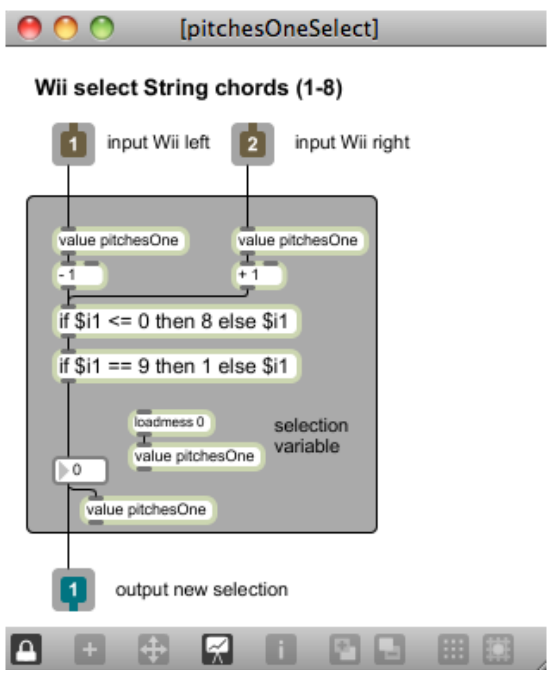



Figure A.1.h.11. String Harmony Pitch Selection Patcher

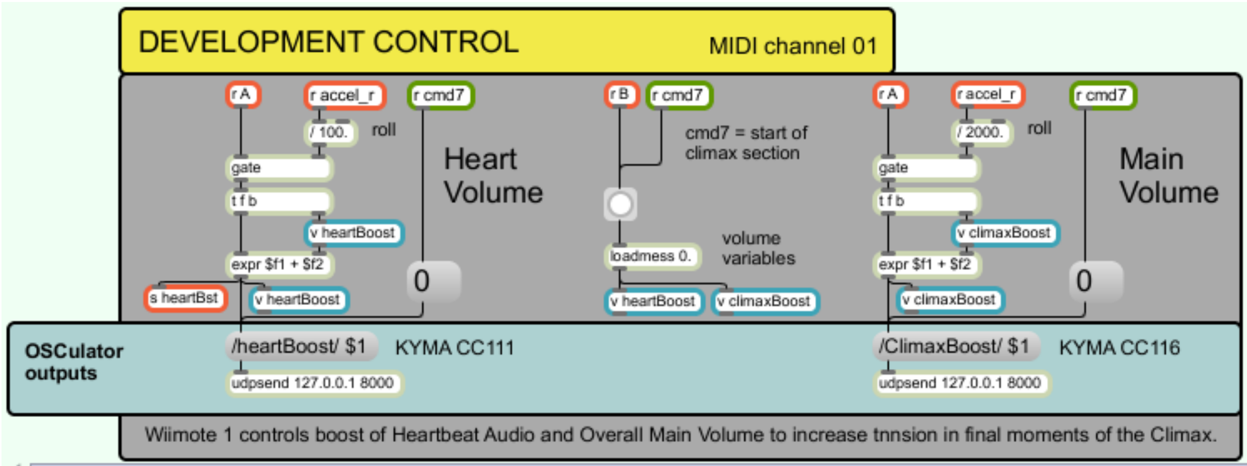



Figure A.1.h.12. Wiimote 1 Development/Climax Section Control Module

## A.1.i. Wiimote 2

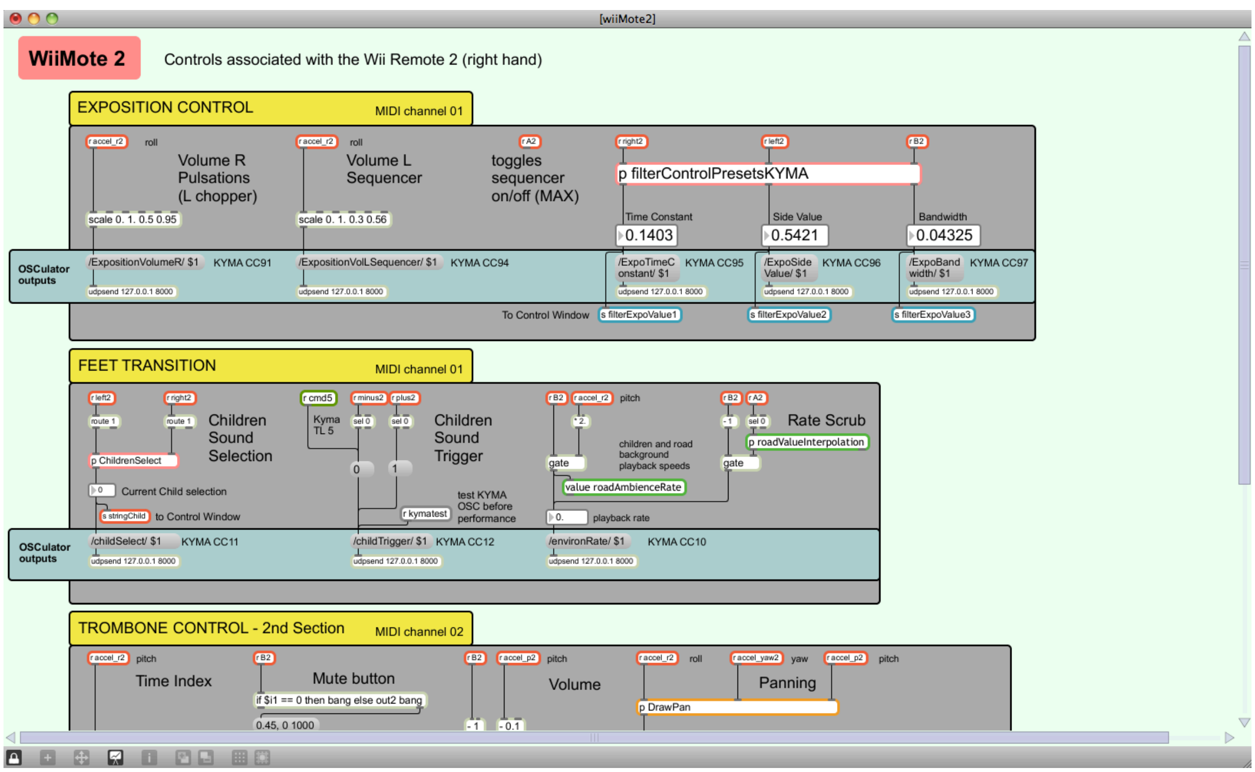



Figure A.1.i.1. Wiimote 2 Control Patch Window, overview of Window layout

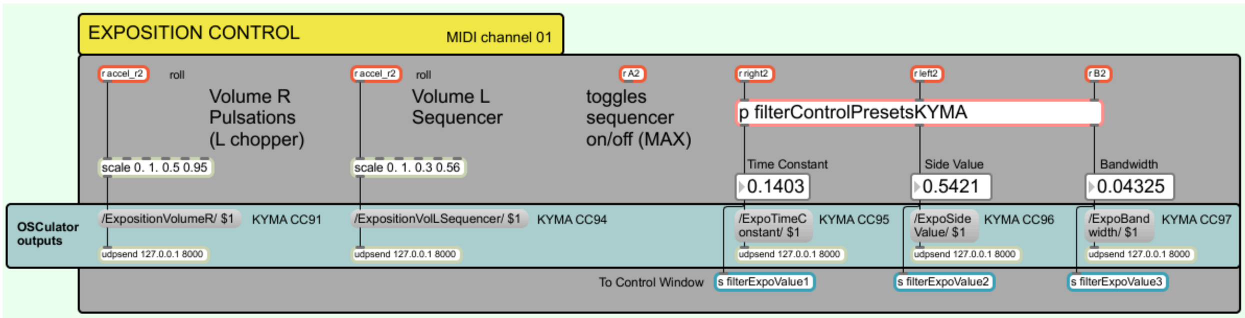



Figure A.1.i.2. Wiimote 2 Heart Rate Monitor Exposition Control Module

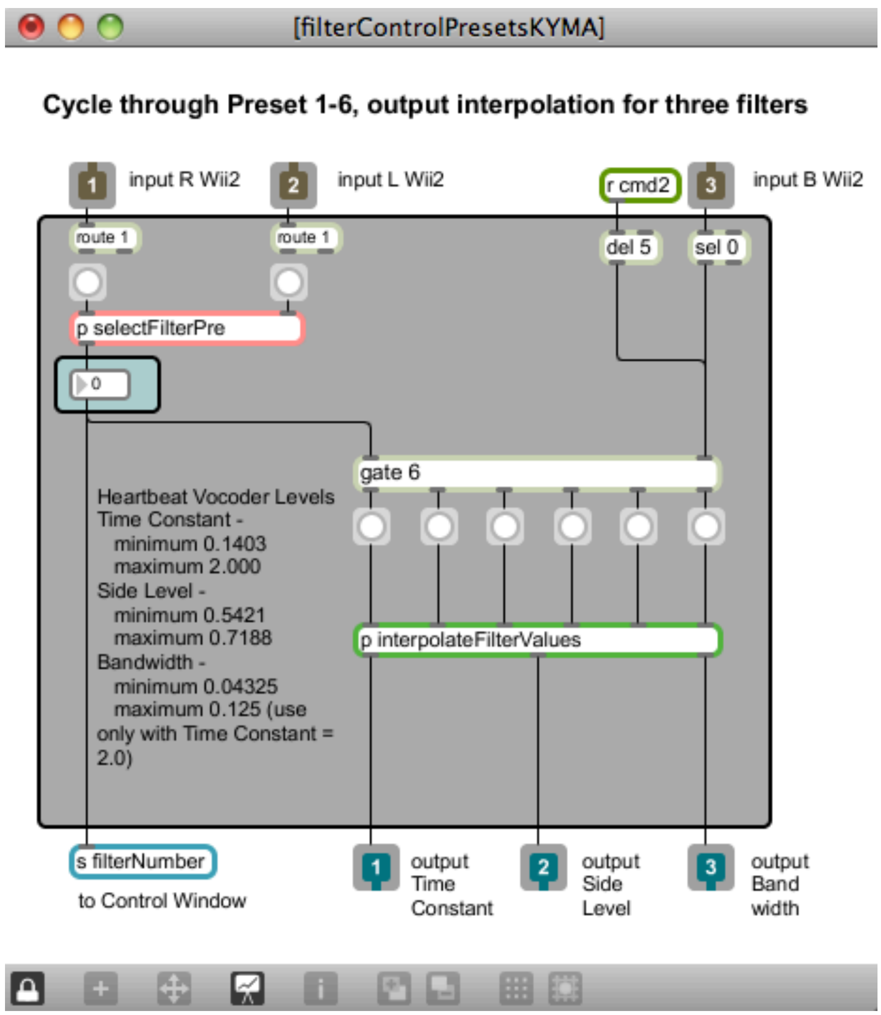



Figure A.1.i.3. Filter Control Presets Patcher

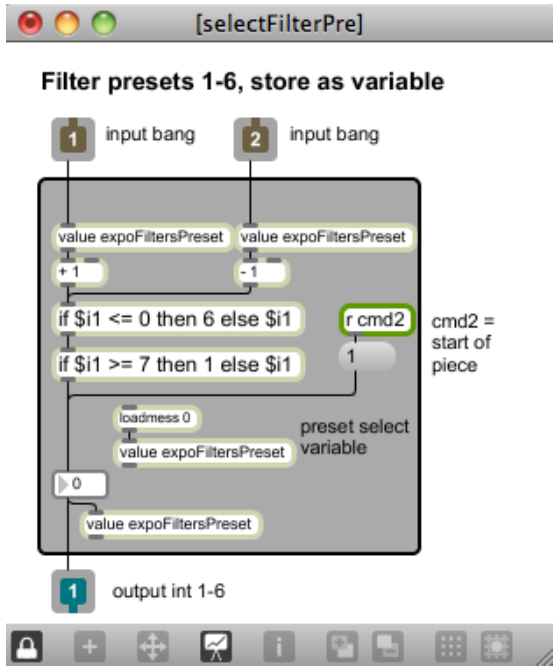



Figure A.1.i.4. Filter Preset Selection Patcher

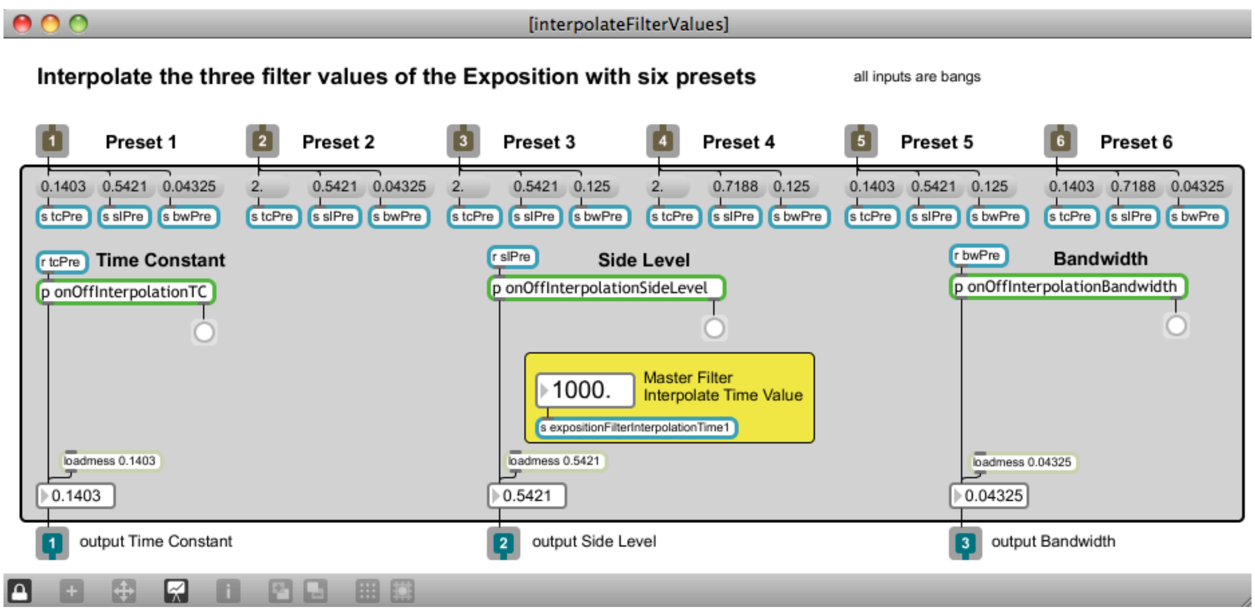



Figure A.1.i.5. Interpolation Between Presets Patcher

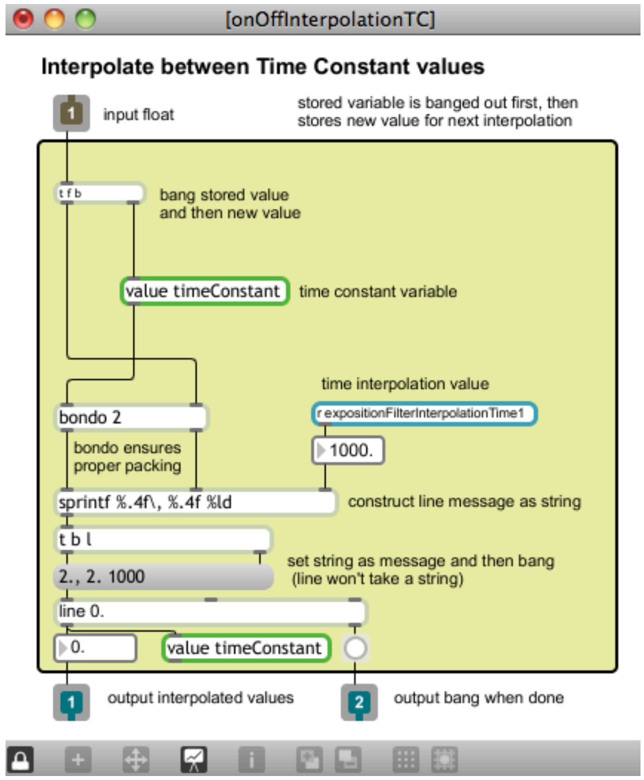



Figure A.1.i.6. Time Constant Parameter Interpolation Patcher

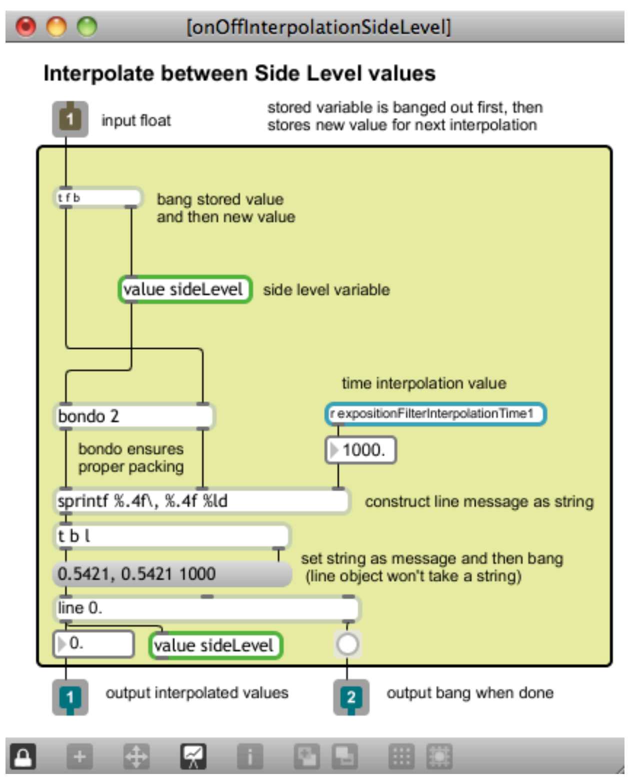



Figure A.1.i.7. Side Level Parameter Interpolation Patcher

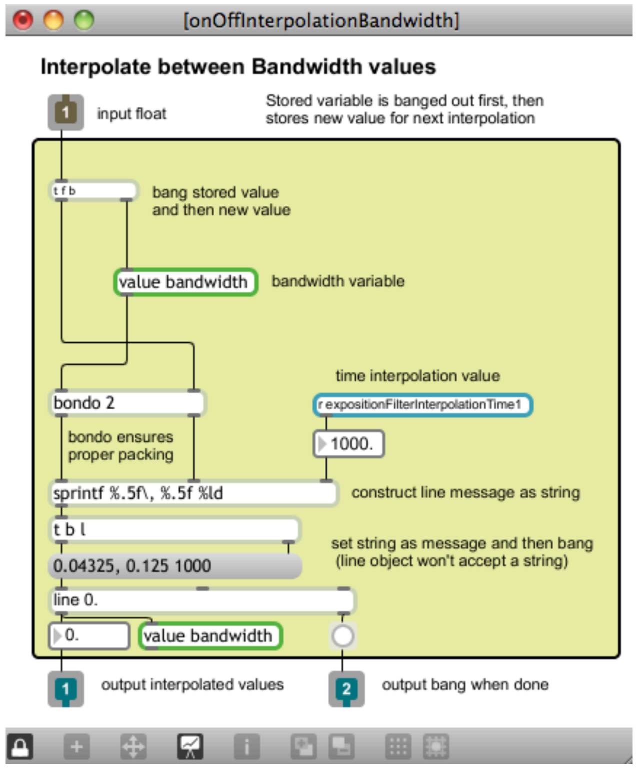



Figure A.1.i.8. Bandwidth Parameter Interpolation Patcher

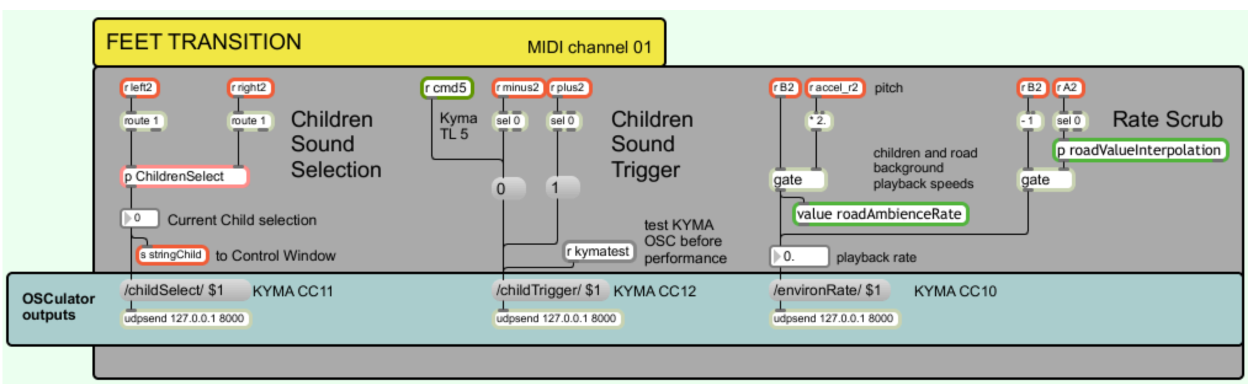



Figure A.1.i.9. Wiimote 2 Feet Exposition Control Module

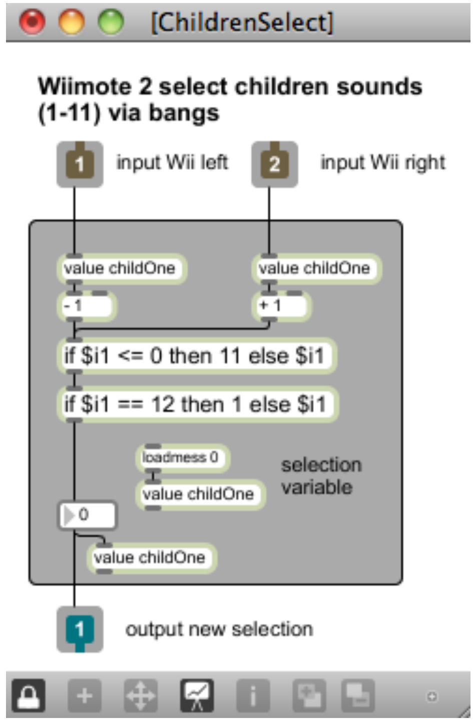



Figure A.1.i.10. Children Audio File Selection Patcher

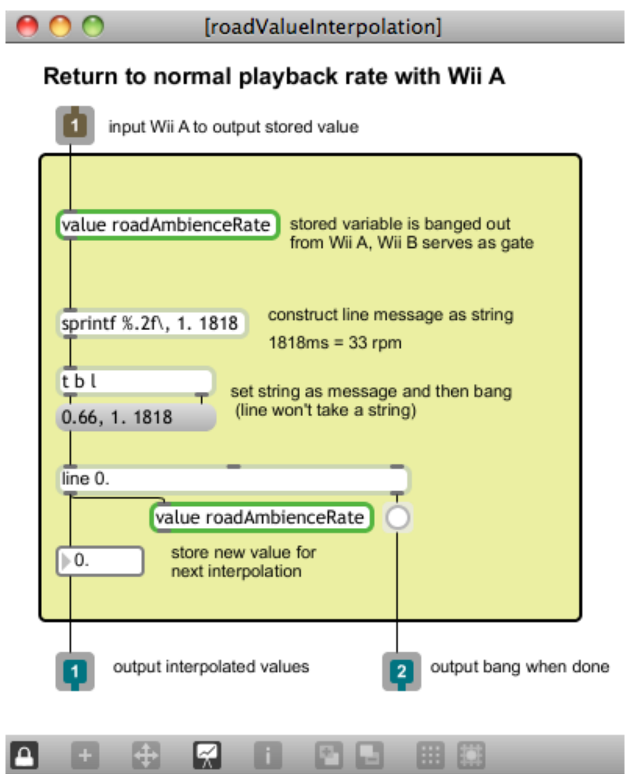



Figure A.1.i.11. Road Ambience Sound Playback Rate Interpolation Patcher

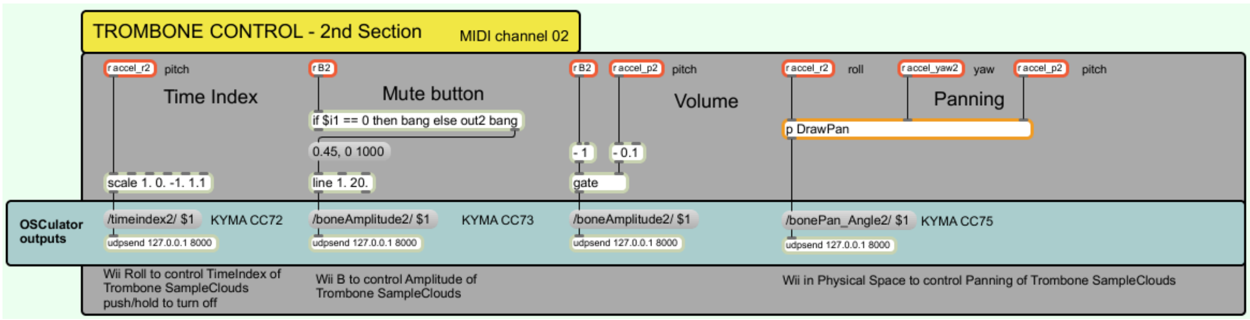



Figure A.1.i.12. Wiimote 2 Development Section Control Module

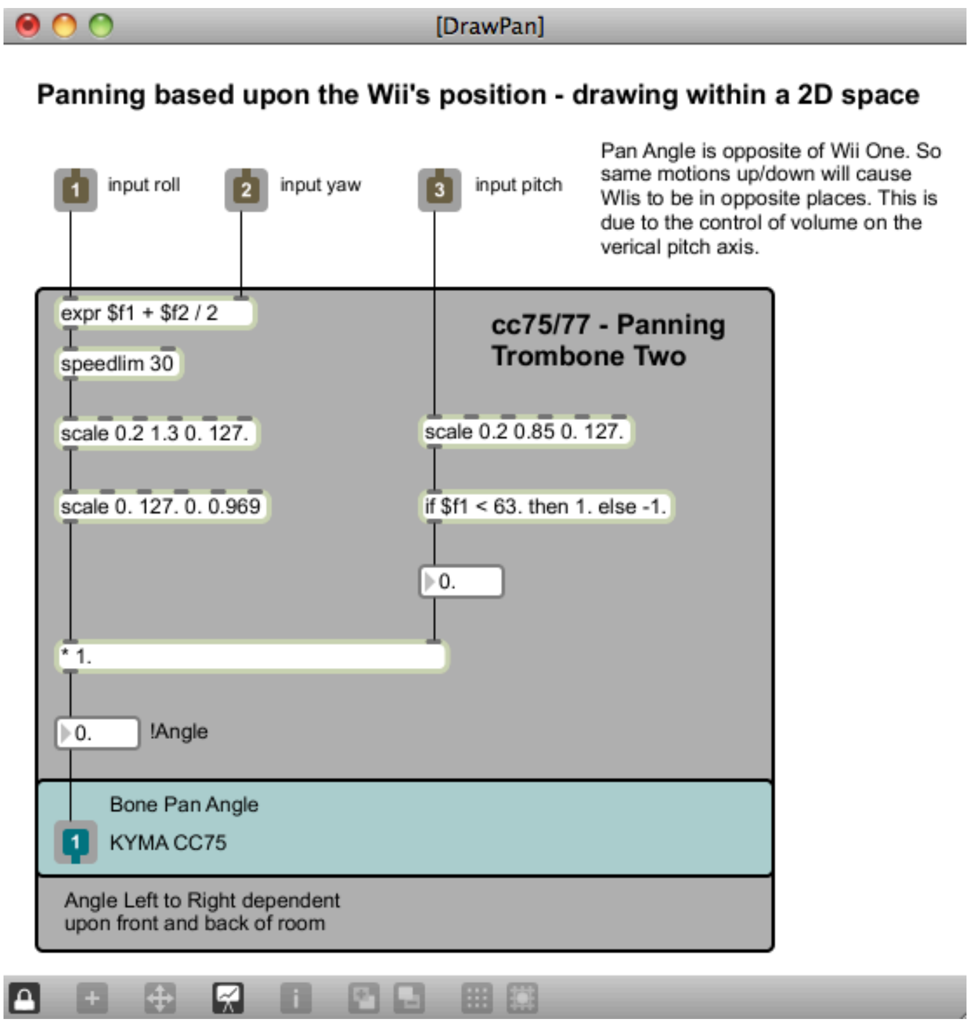



Figure A.1.i.13. Panning of Trombones Control Patcher

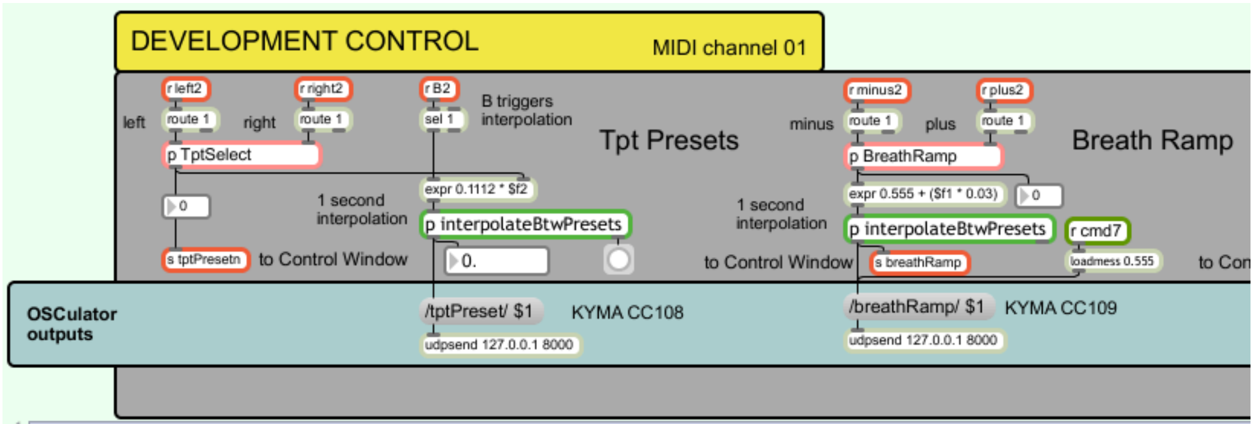



Figure A.1.i.14. Wiimote 2 Development/Climax Section Control Module, part 1

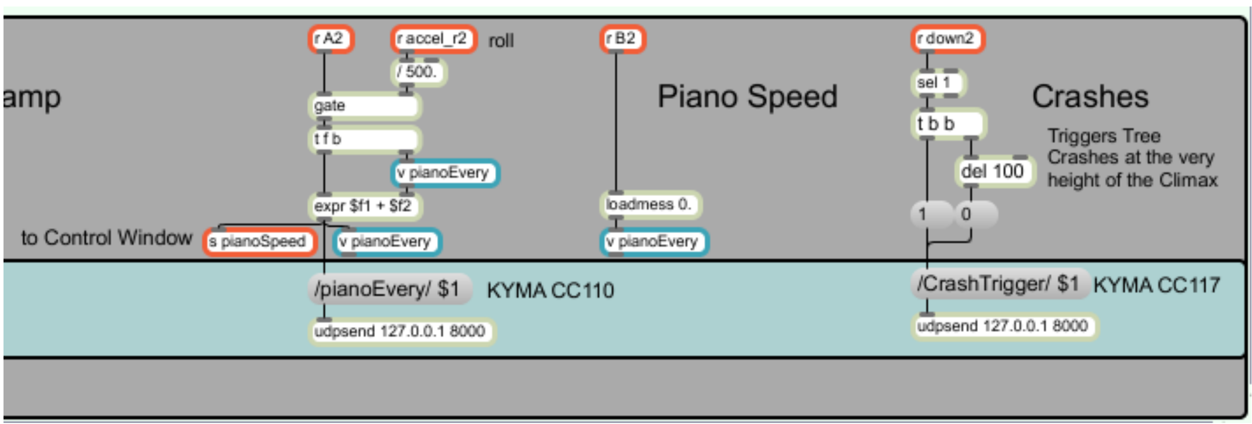



Figure A.1.i.15. Wiimote 2 Development/Climax Section Control Module, part 2

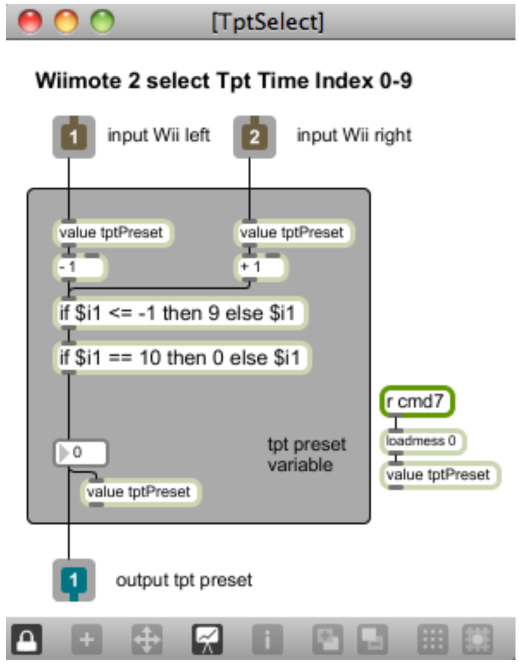



Figure A.1.i.16. Trumpet Time Index Selection Patcher

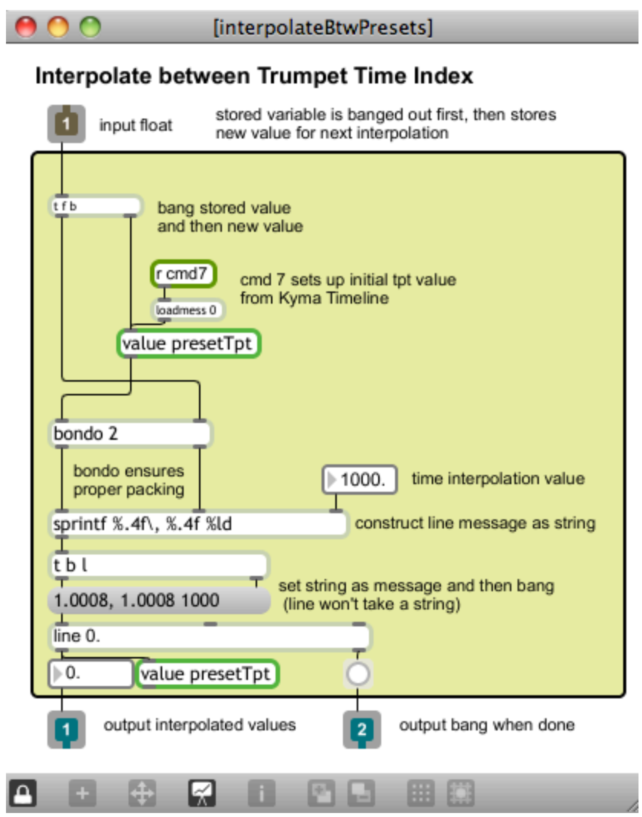



Figure A.1.i.17. Trumpet Time Index Interpolation Patcher

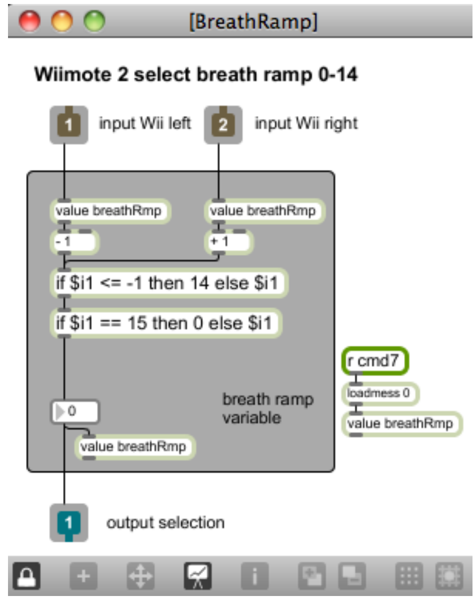



Figure A.1.i.18. Breath Rate Calculator Selection Patcher

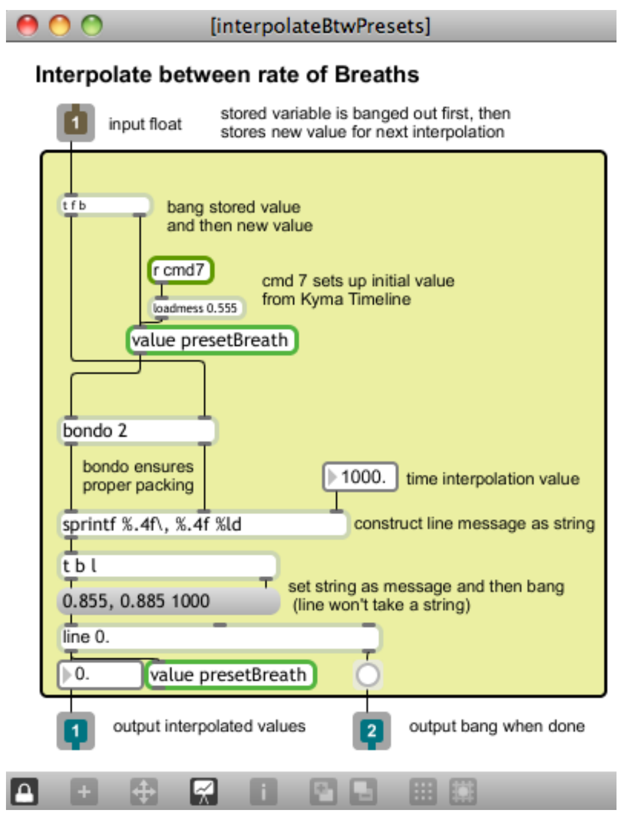



Figure A.1.i.19. Breath Rate Interpolation Patcher

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

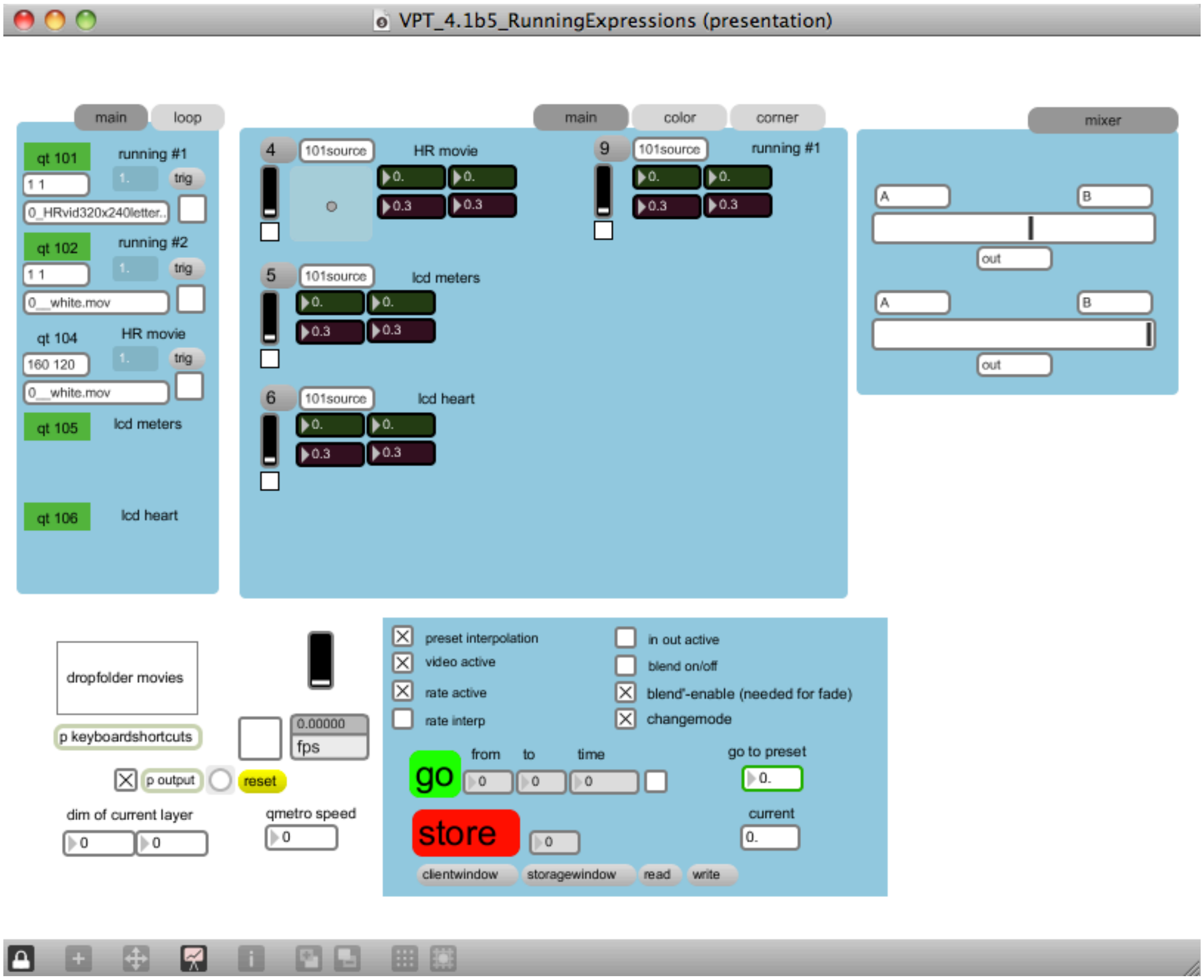



Figure A.2.1. VPT_4.1b5_RunningExpressions.maxpat Main Patch Window, in Presentation mode

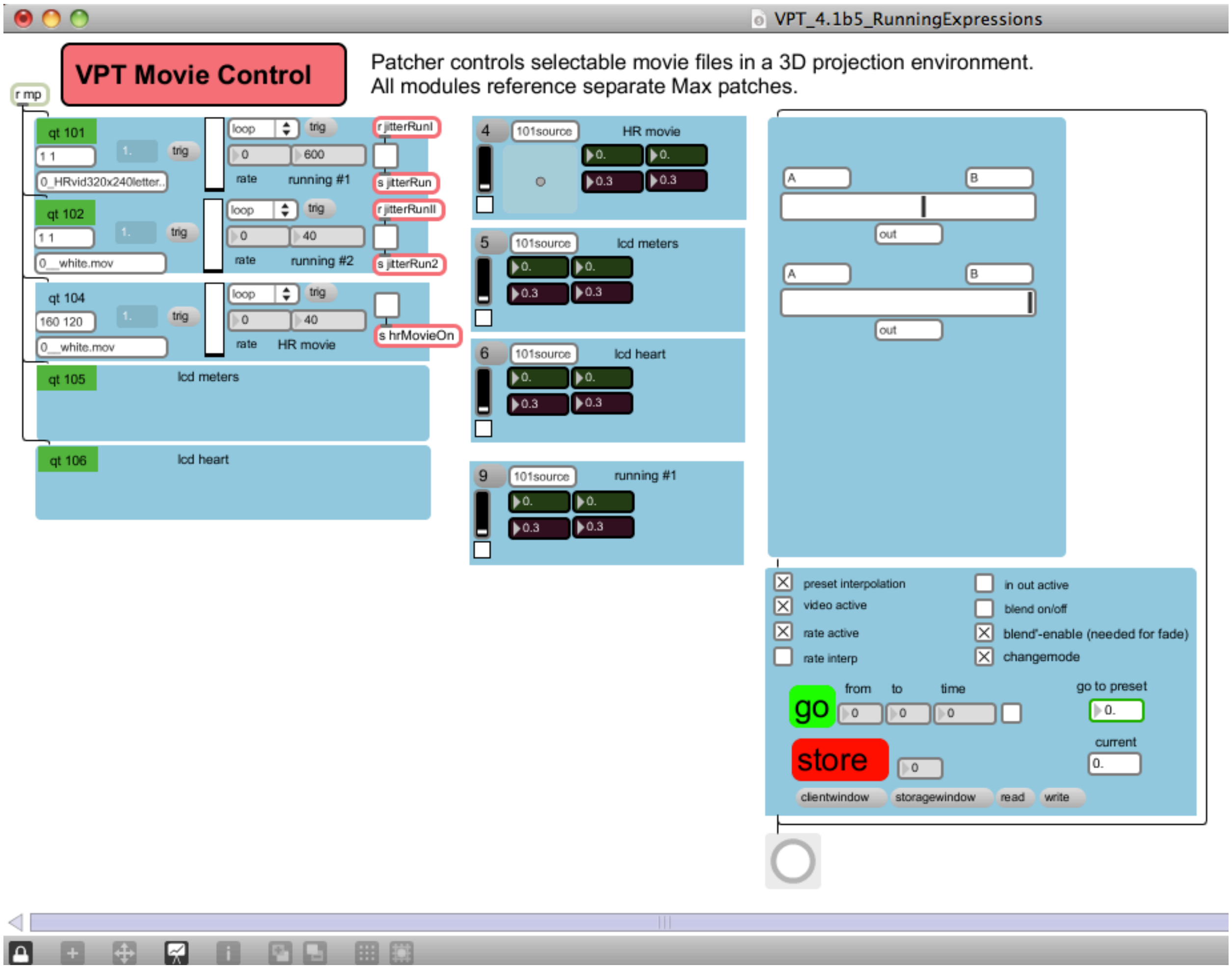



Figure A.2.2. VPT Main Patch Window, in Patcher mode, part 1

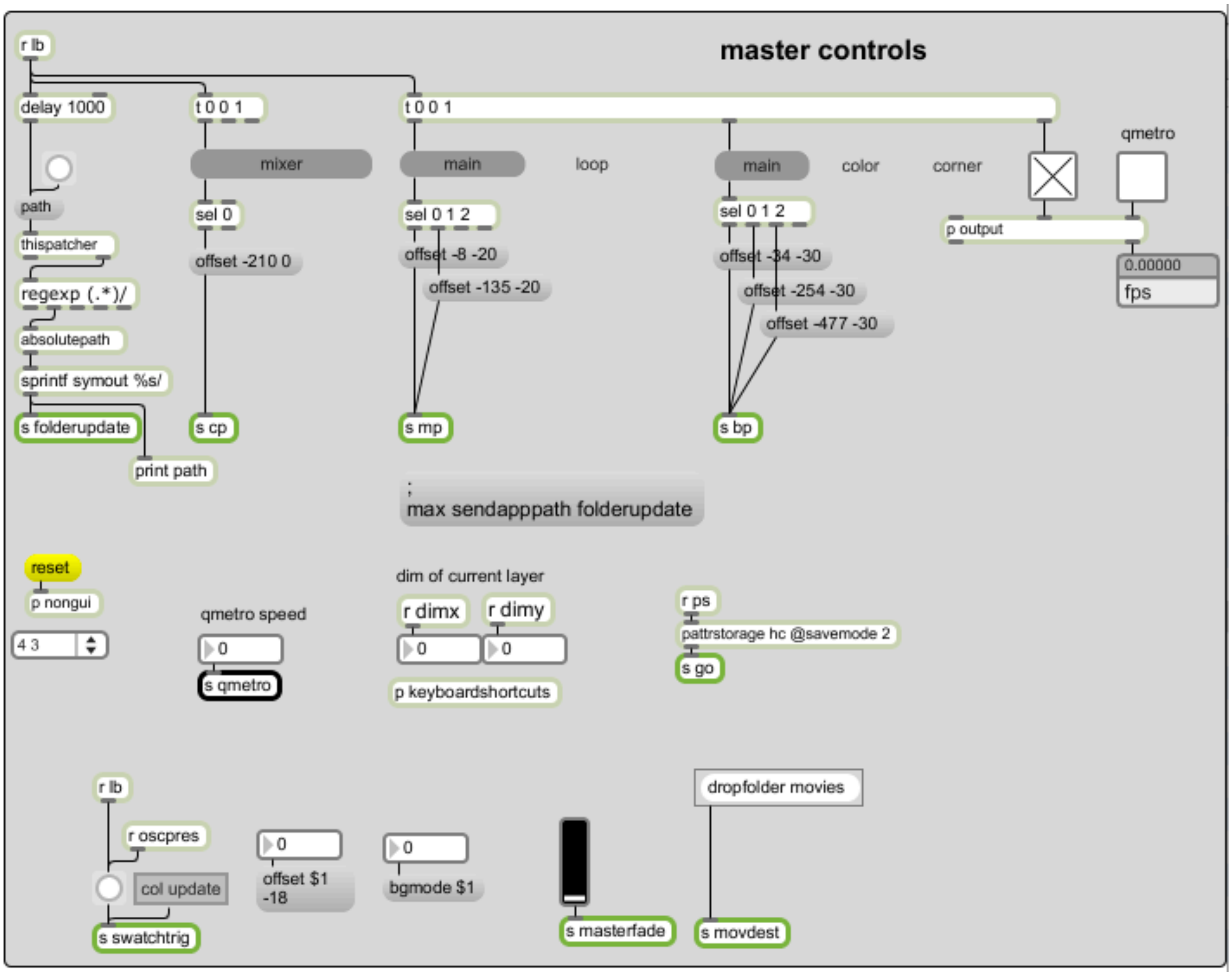



Figure A.2.3. VPT Main Patch Window, in Patcher mode, part 2

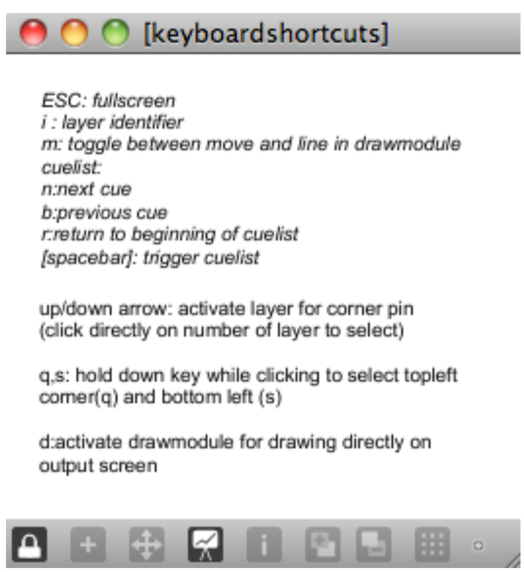



Figure A.2.4. VPT Keyboard Shortcuts

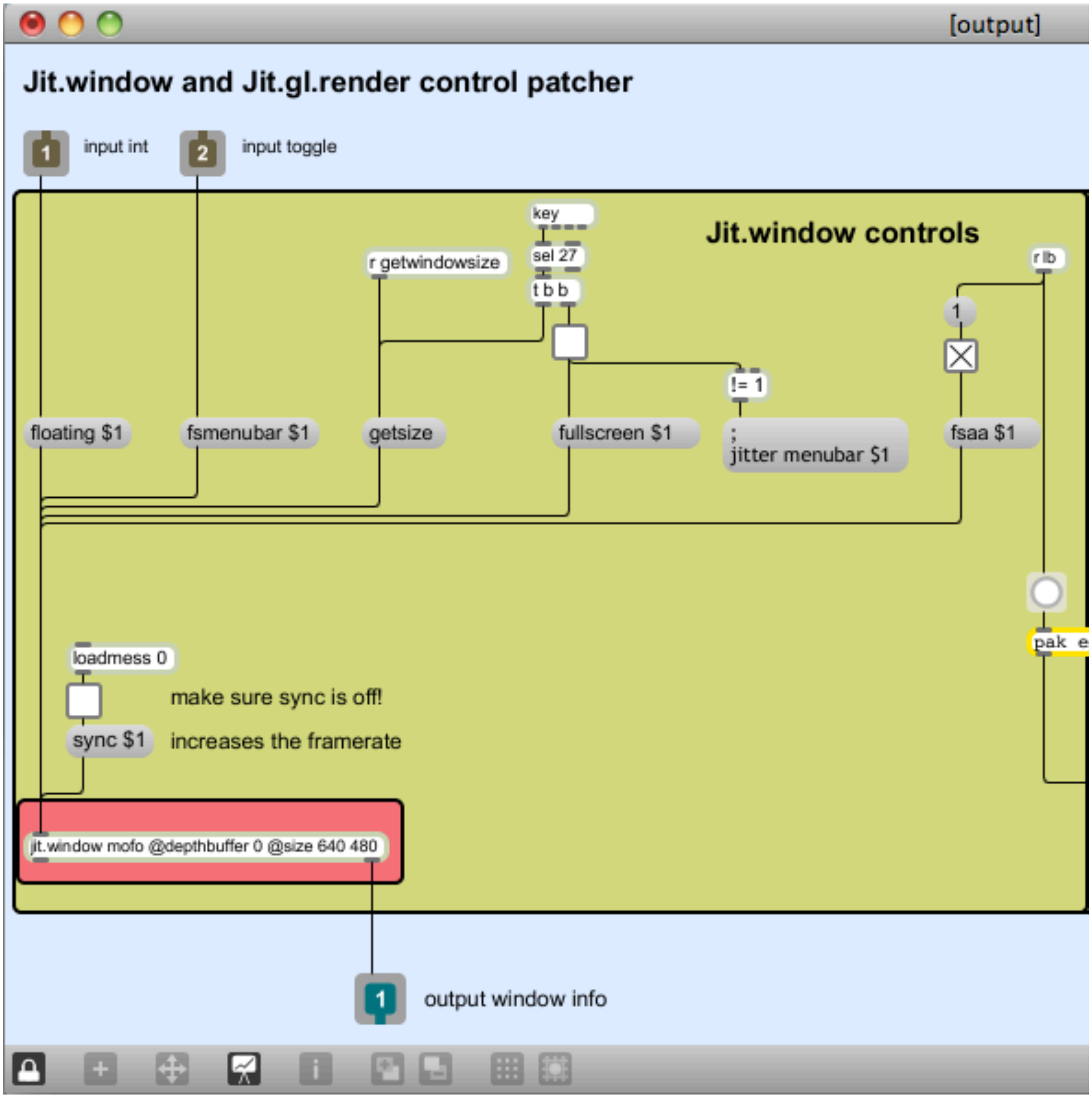



Figure A.2.5. Main ‘jit.window’ and ‘jit.gl.render’ Control Patcher, part 1

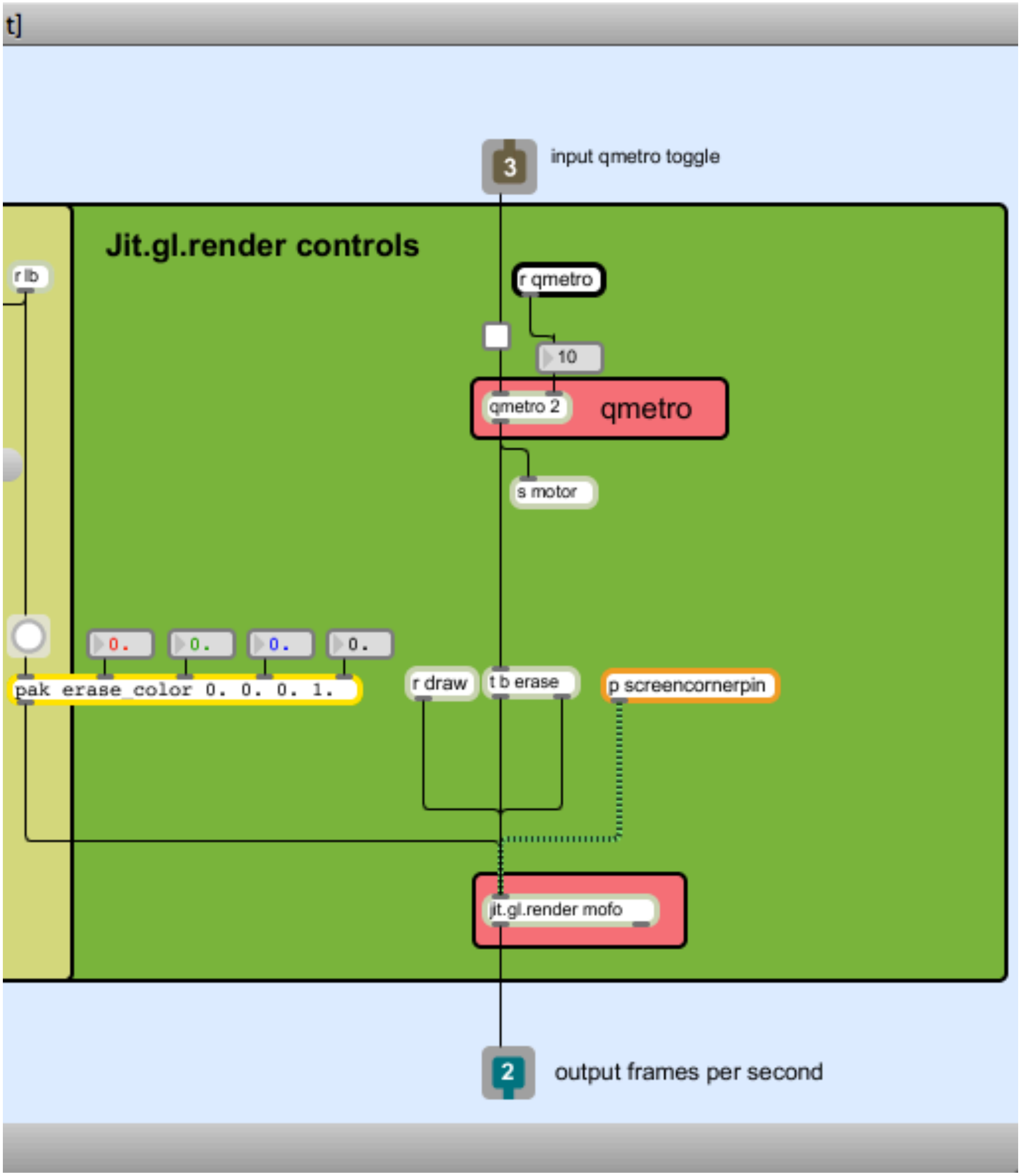



Figure A.2.6. Main ‘jit.window’ and ‘jit.gl.render’ Control Patcher, part 2

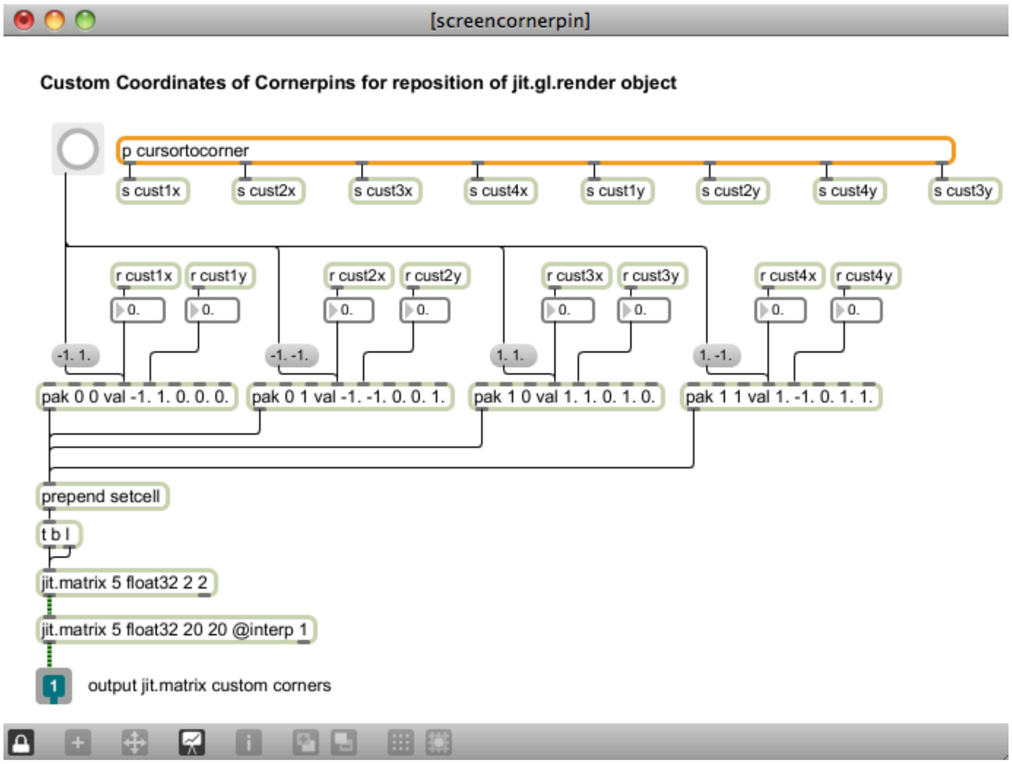



Figure A.2.7. Custom Coordinates Control Patcher

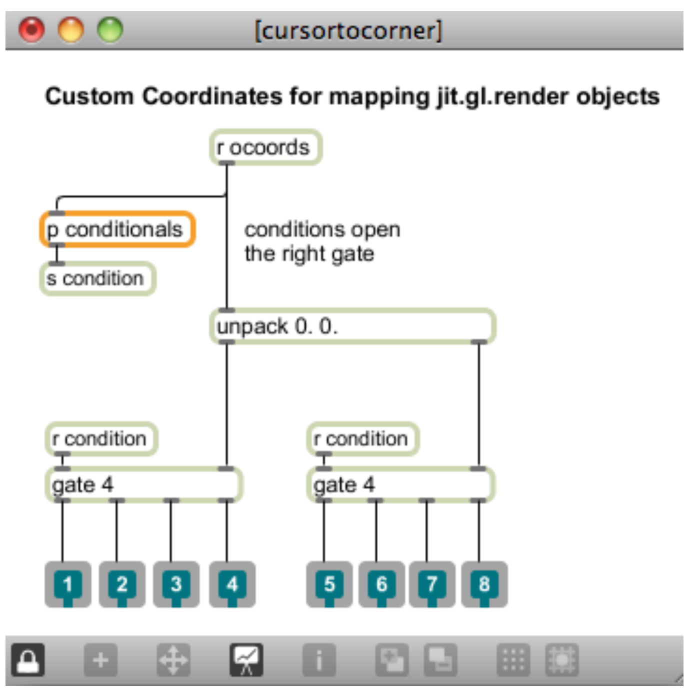



Figure A.2.8. Custom Coordinates Input Patcher

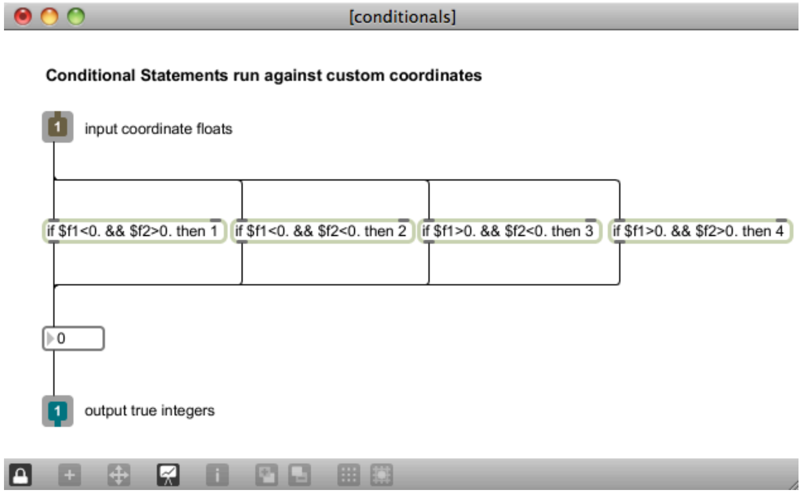



Figure A.2.9. Conditional Statement Custom Coordinates Patcher

## A.2.a. Videoplane Module: Running


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

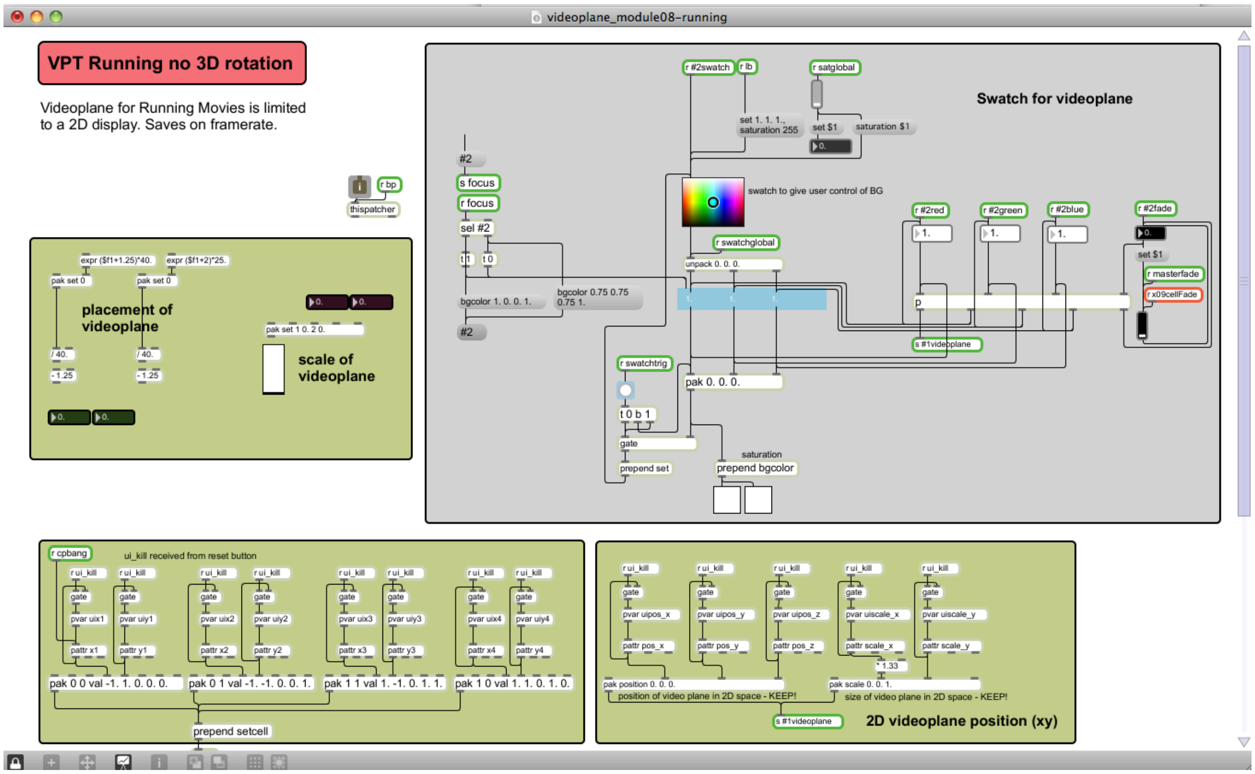



Figure A.2.a.2. Videoplane Running Patch Window, overview of Window layout

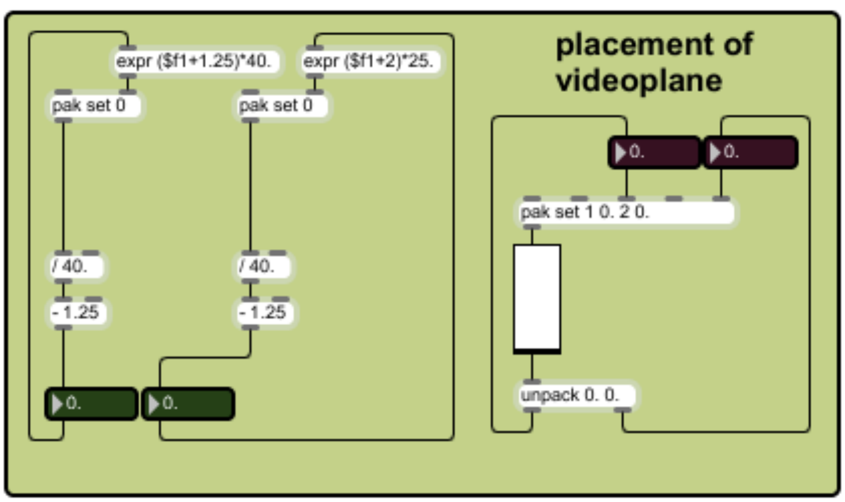



Figure A.2.a.3. Videoplane Position Module

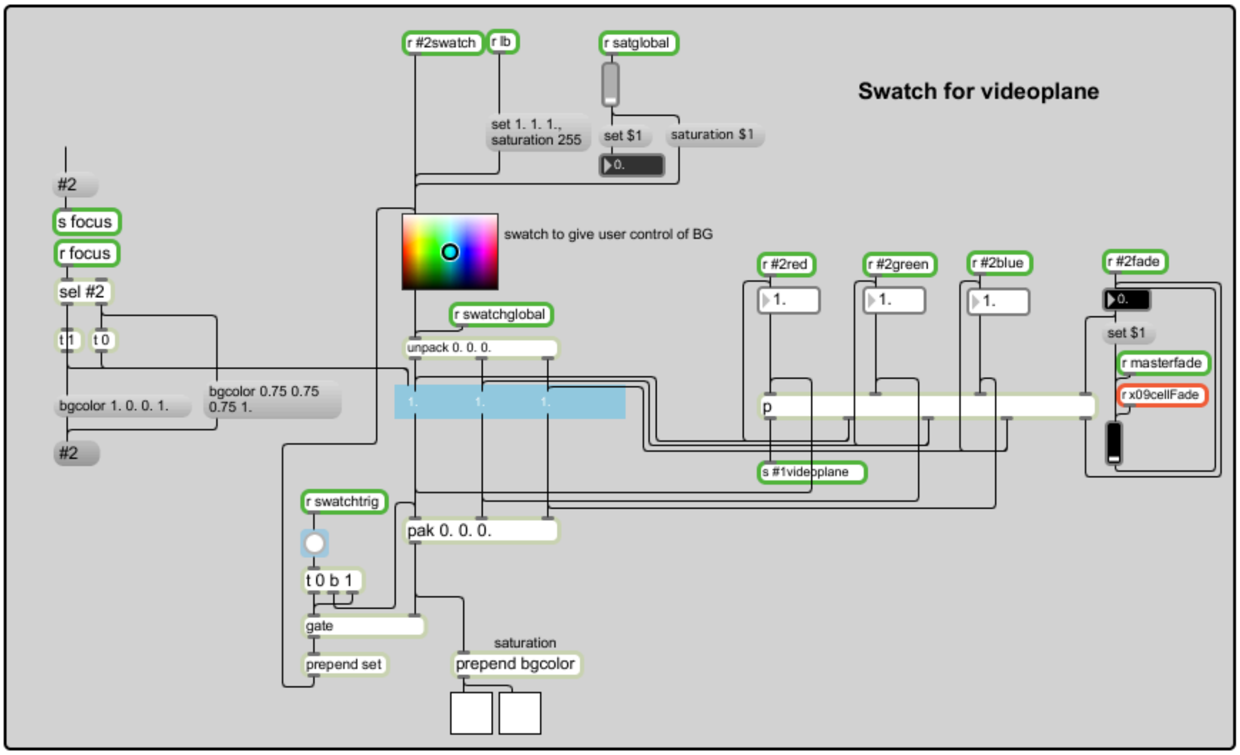



Figure A.2.a.4. Videoplane Color Swatch Module

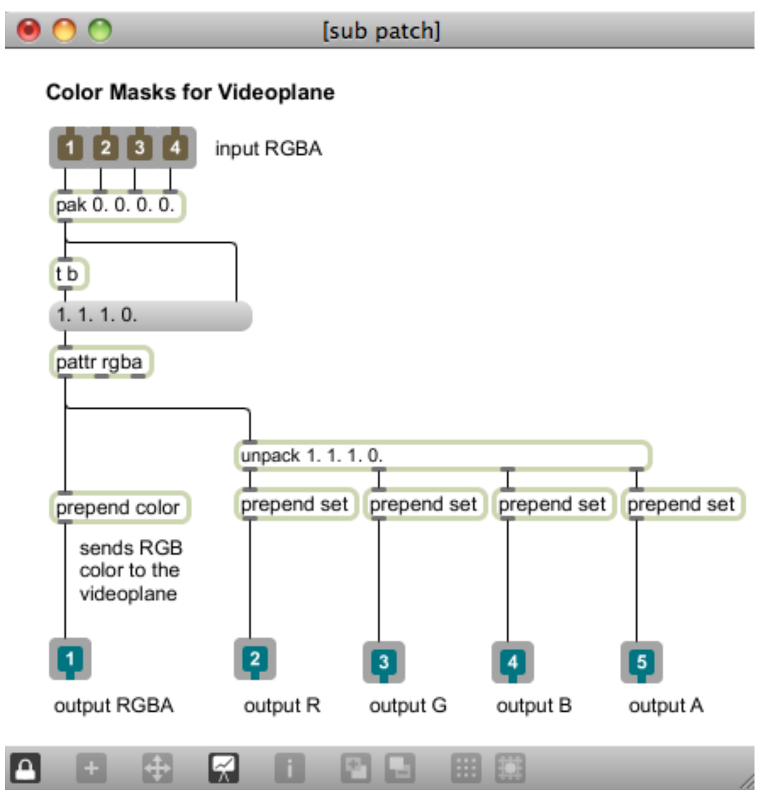



Figure A.2.a.5. Videoplane Color Masks Patcher

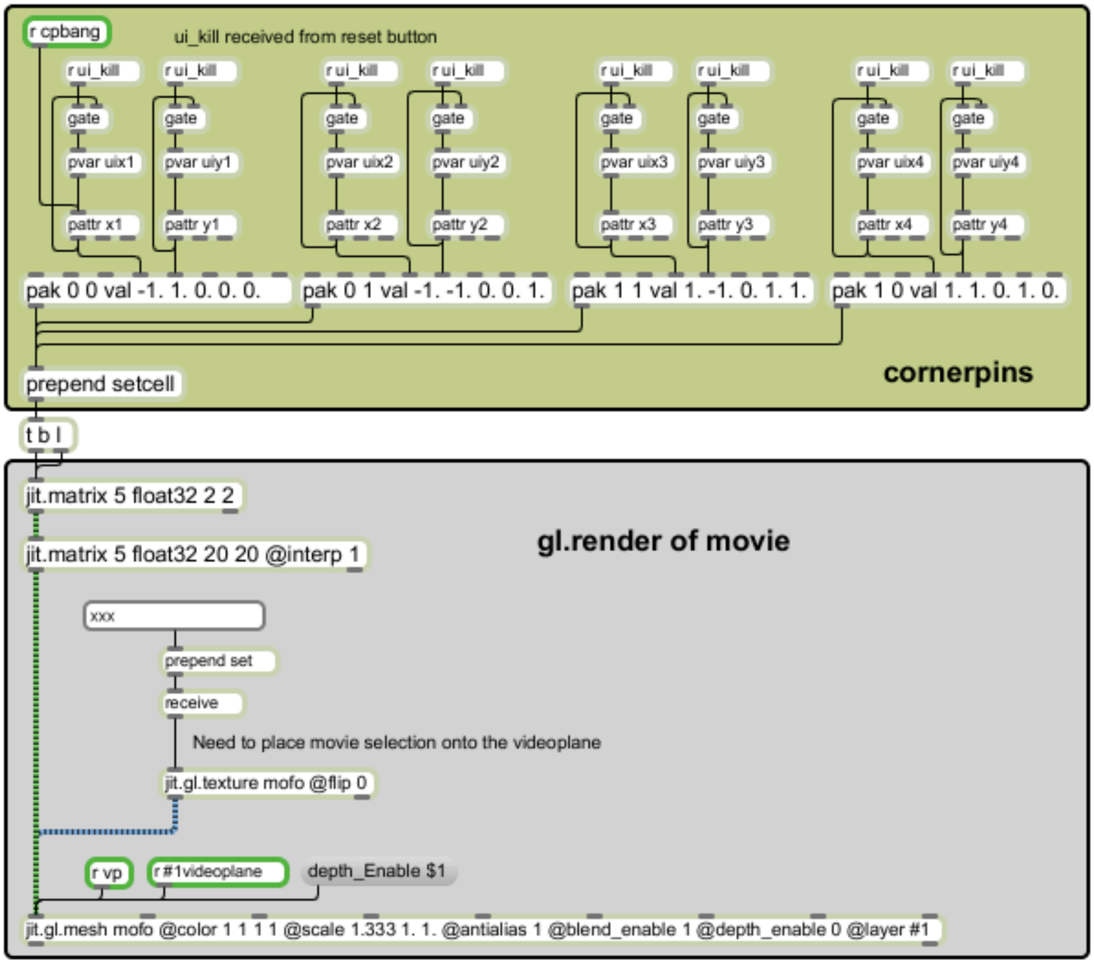



Figure A.2.a.6. Videoplane ‘jit.gl.render’ Control Module

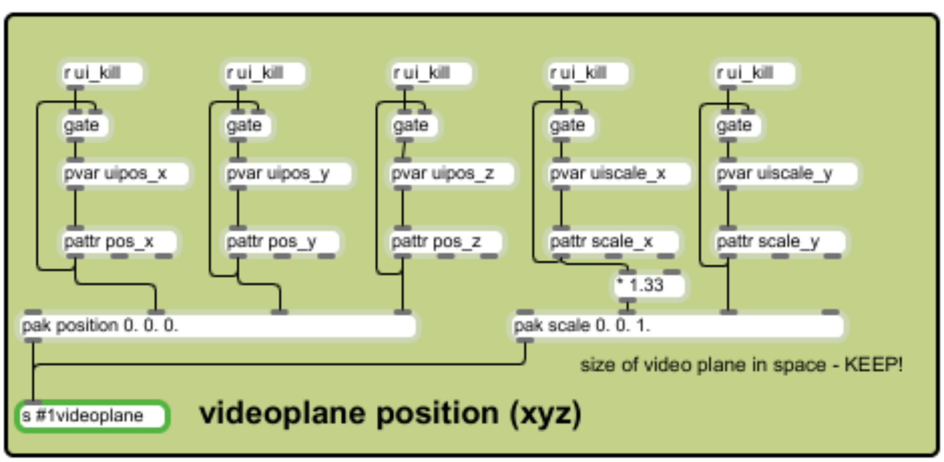



Figure A.2.a.7. Videoplane Positioning Control Module

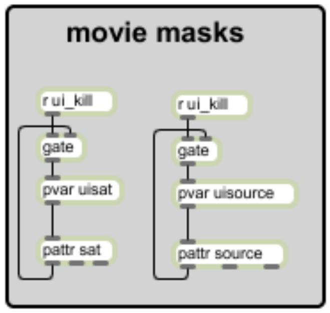



Figure A.2.a.8. Videoplane Movie
Masks Module

## A.2.b. Videoplane Module: Heart Rate

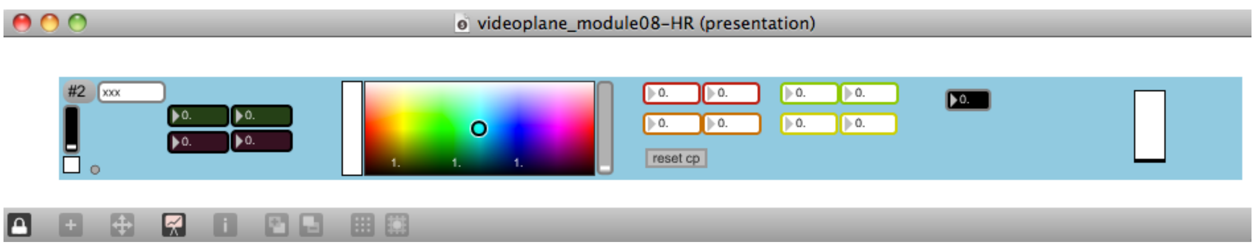



Figure A.2.b.1. Videoplane Heart Rate, in Presentation mode

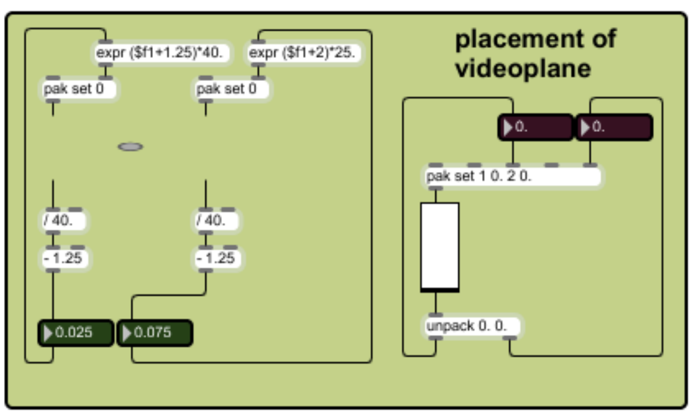



Figure A.2.b.2. Videoplane Heart Rate, Positioning Control
Module, with ‘pictslider’ object

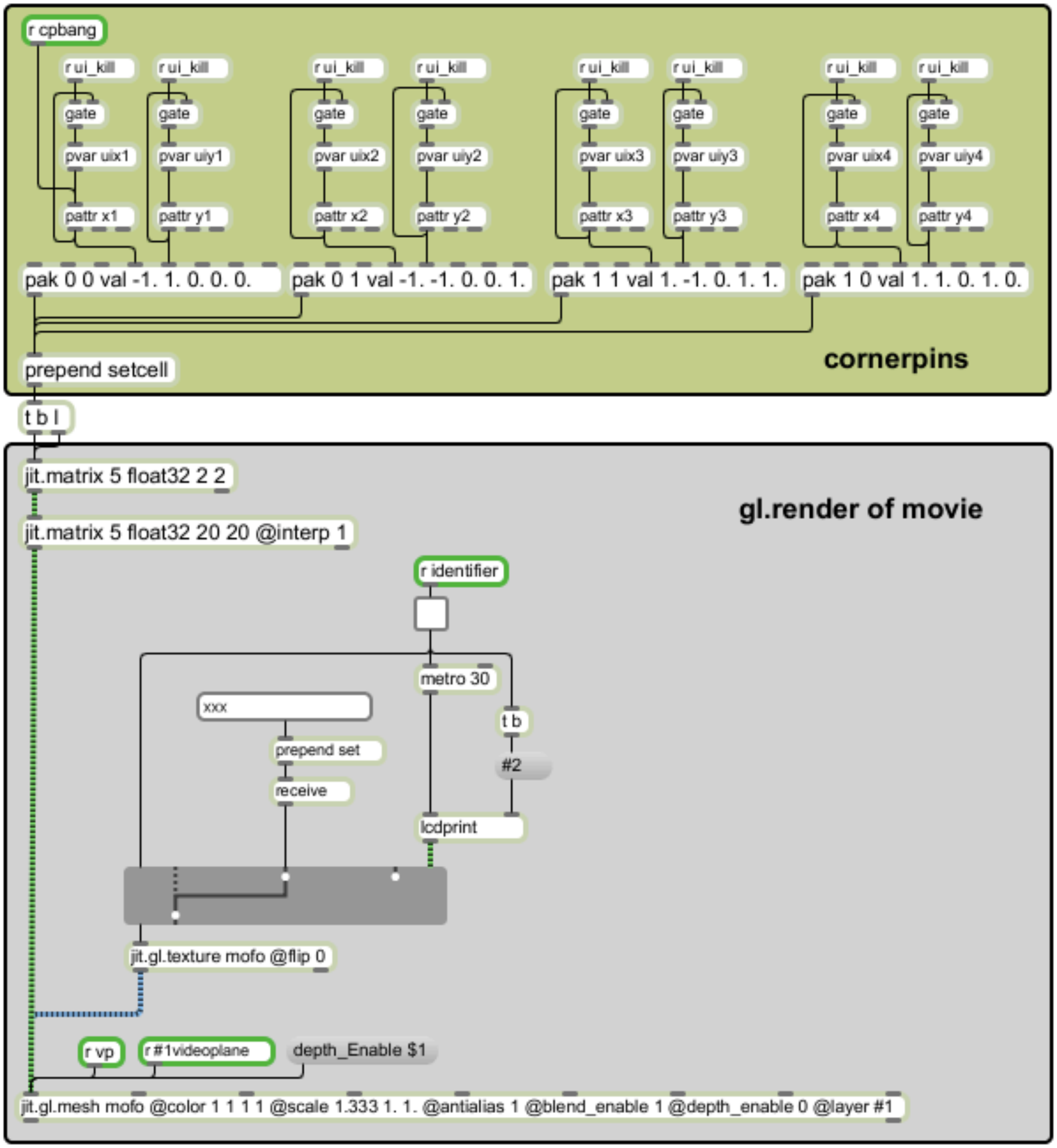



Figure A.2.b.3. Videoplane Heart Rate, ‘jit.gl.render’ Control Module

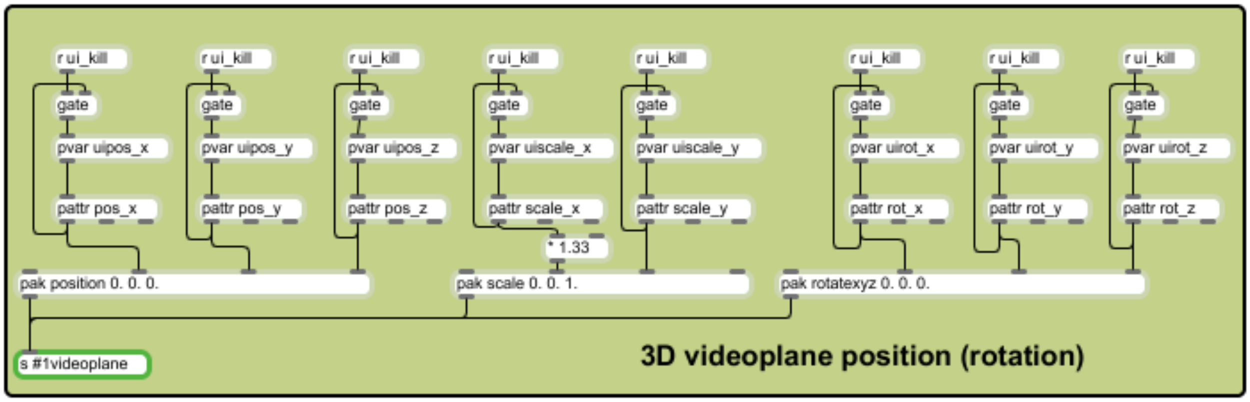



Figure A.2.b.4. Videoplane 3D Positioning Control Module

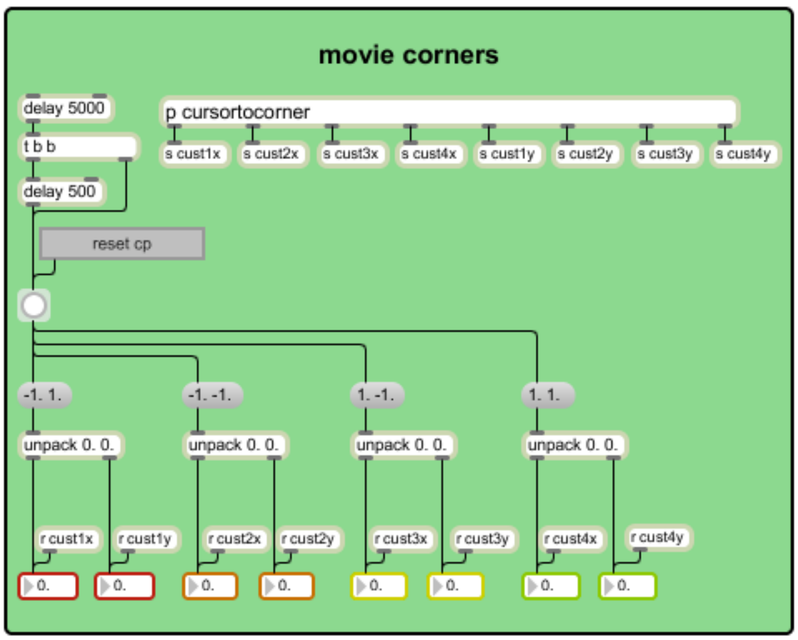



Figure A.2.b.5. Videoplane Custom Corner Positioning Control Module

A.2.c. Videoplane Module: LCD

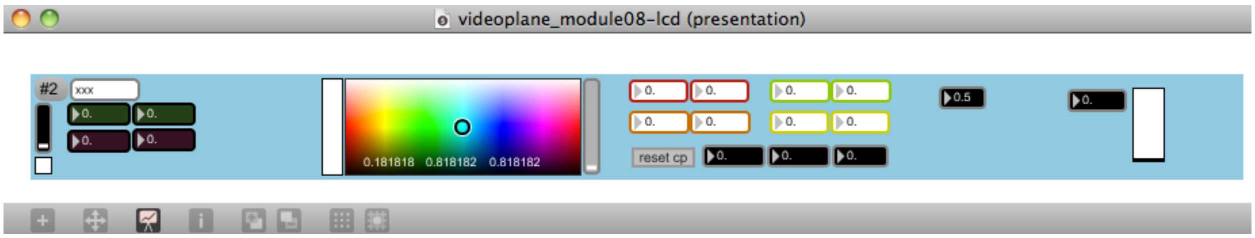



Figure A.2.c.1. Videoplane LCD, in Presentation mode

A.2.d. Preset Module

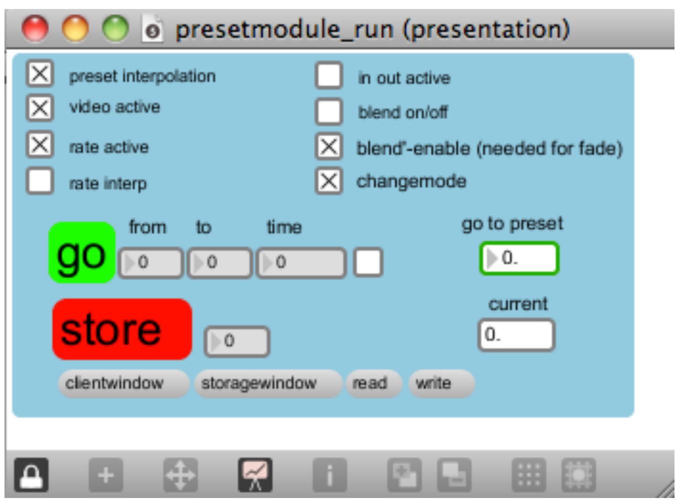



Figure A.2.d.1. Preset Module, in Presentation mode

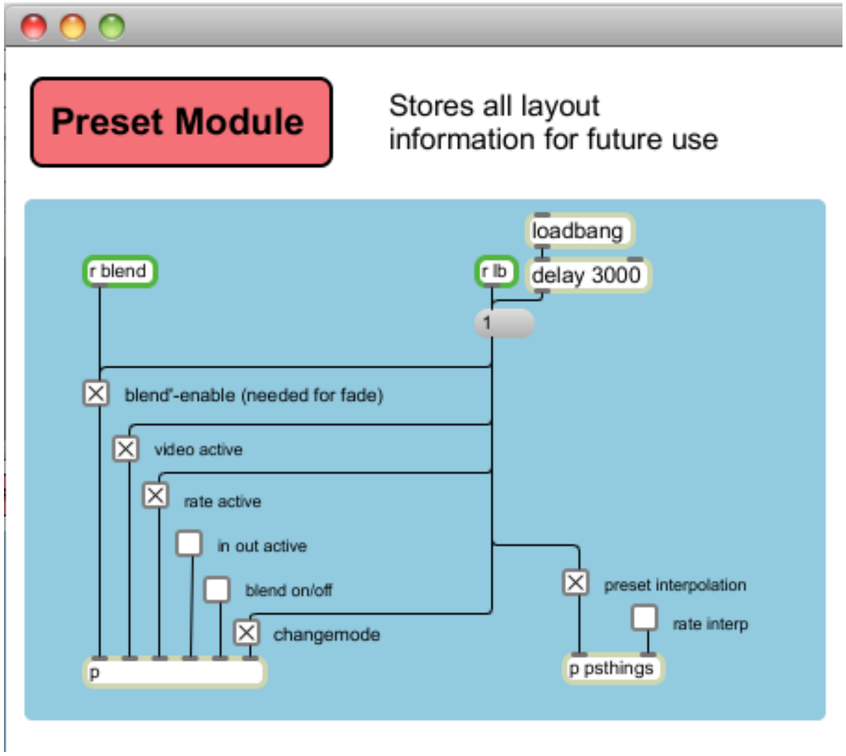



Figure A.2.d.2. Preset Module, in Patcher mode, part 1

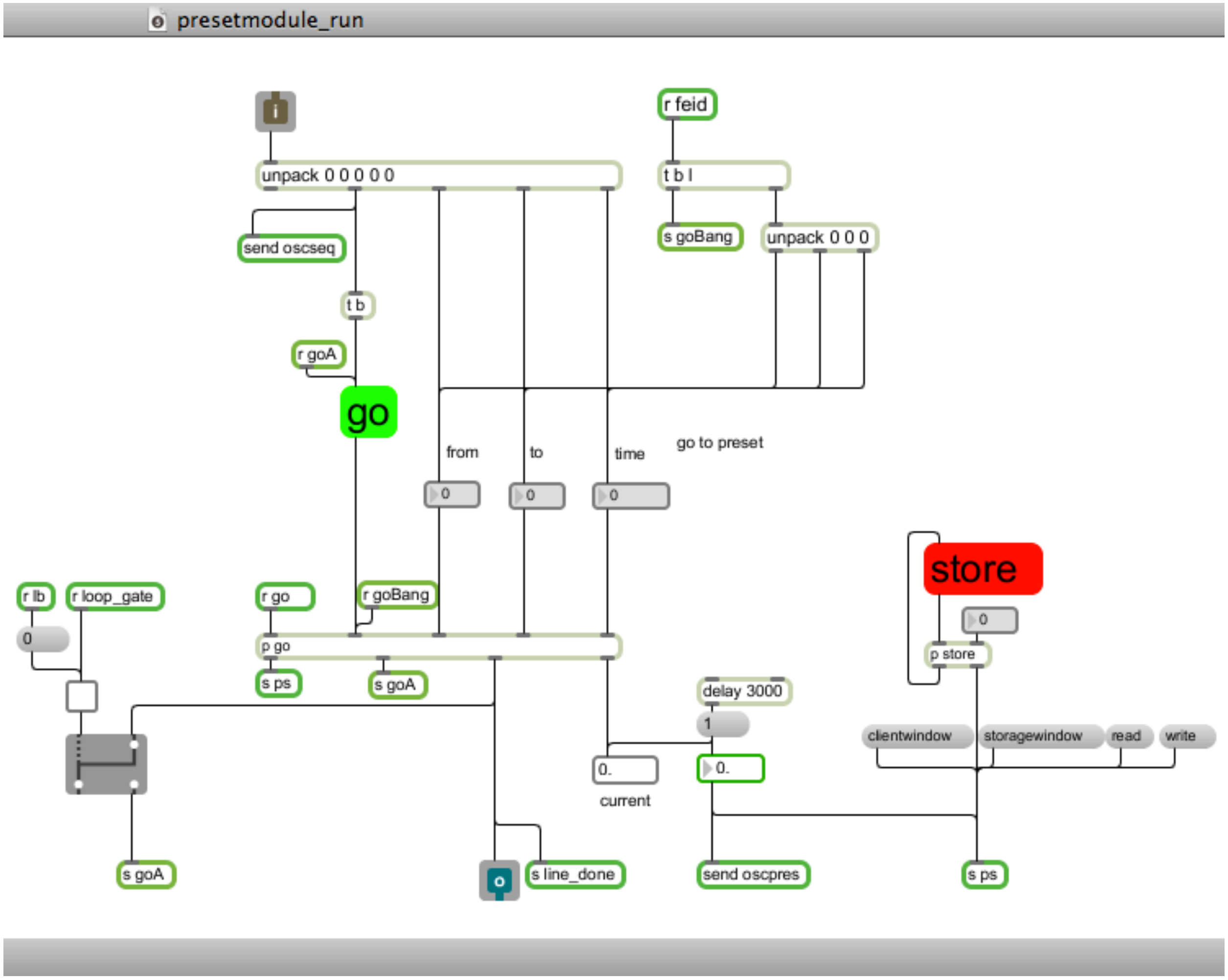



Figure A.2.d.3. Preset Module, in Patcher mode, part 2

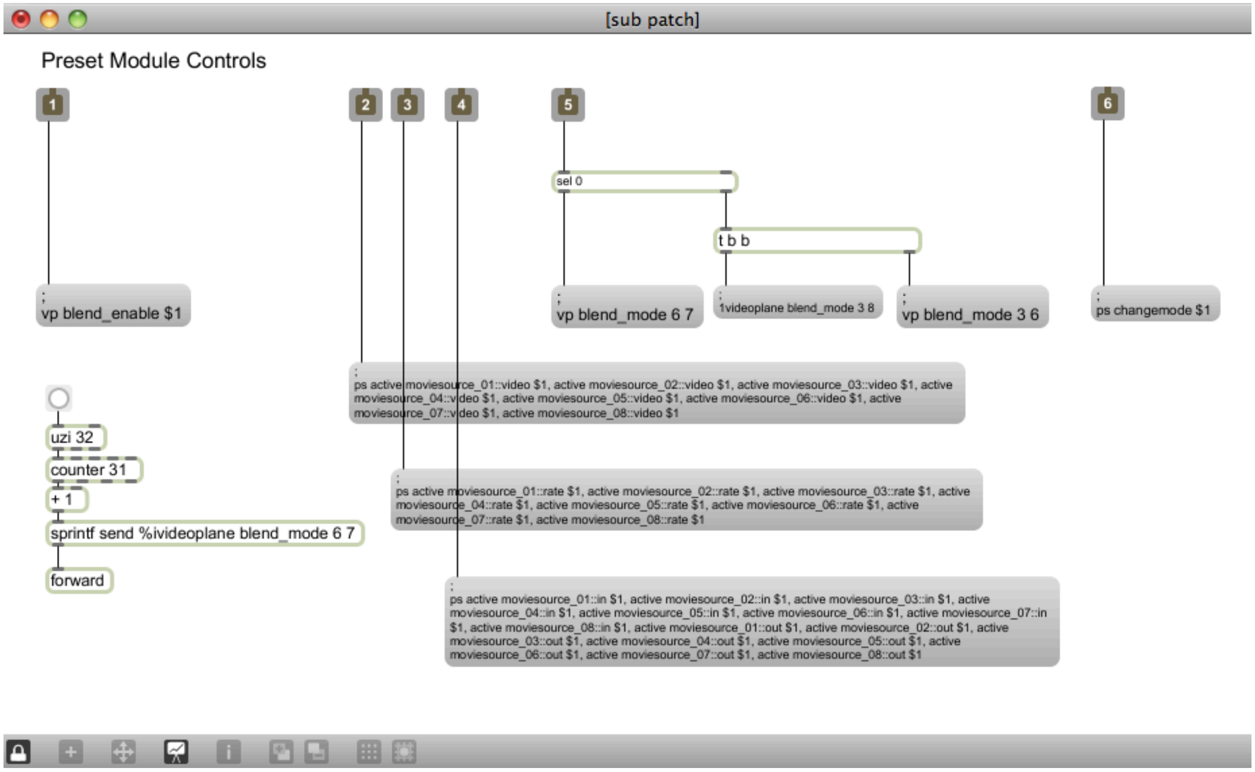



Figure A.2.d.4. Preset Module Controls Patcher

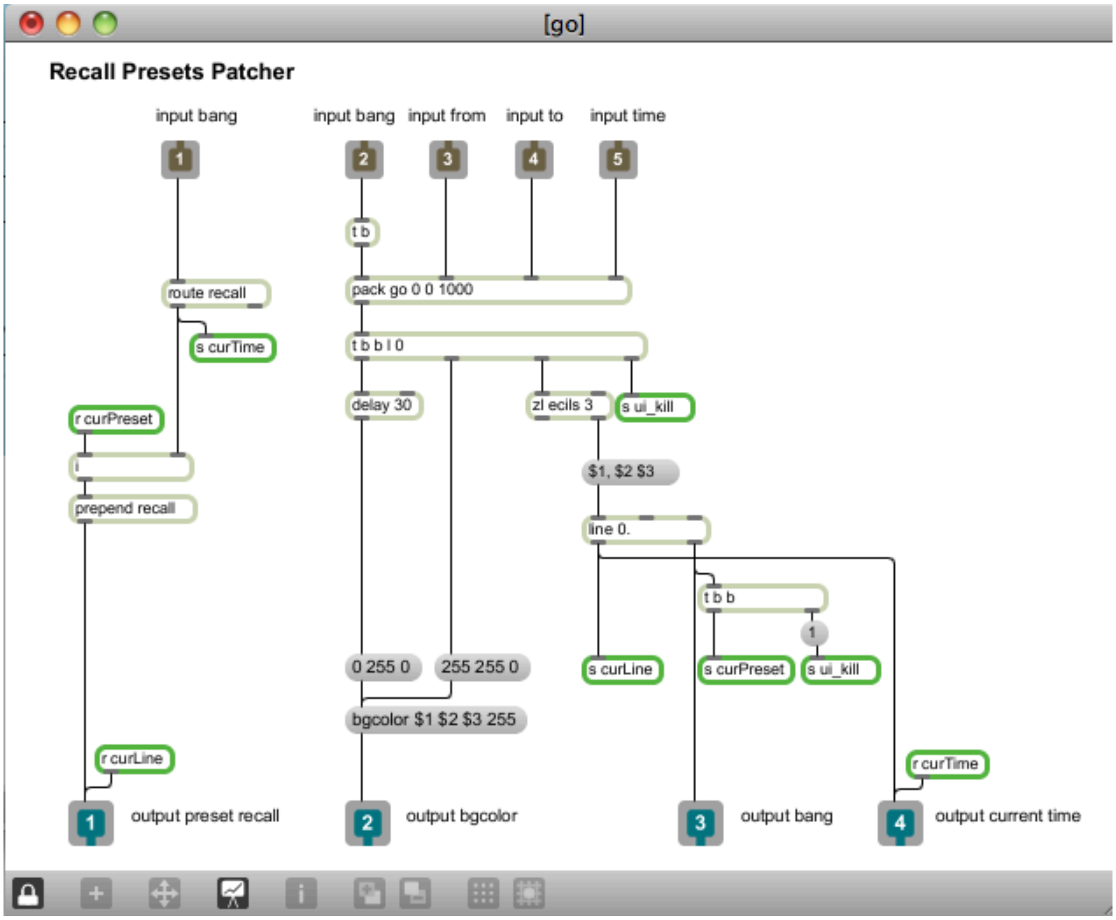



Figure A.2.d.5. Preset Module Recall Patcher

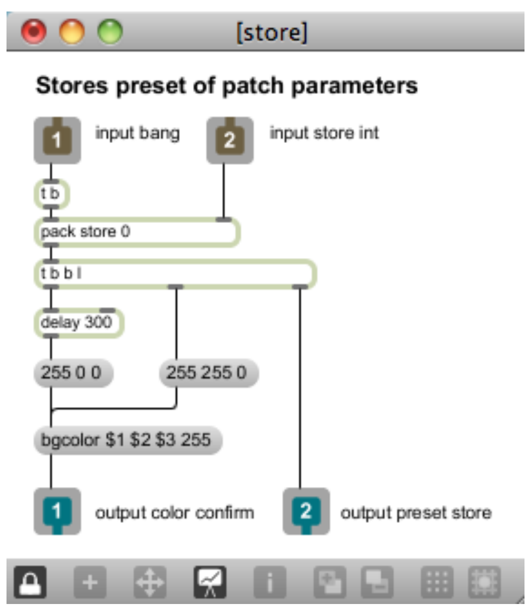



Figure A.2.d.6. Preset Module Data Confirmation Patcher

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

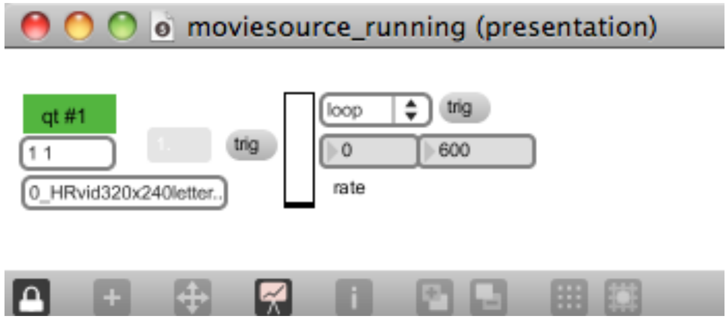



Figure A.2.e.1. Movie Source Running Patch Window, in Presentation mode

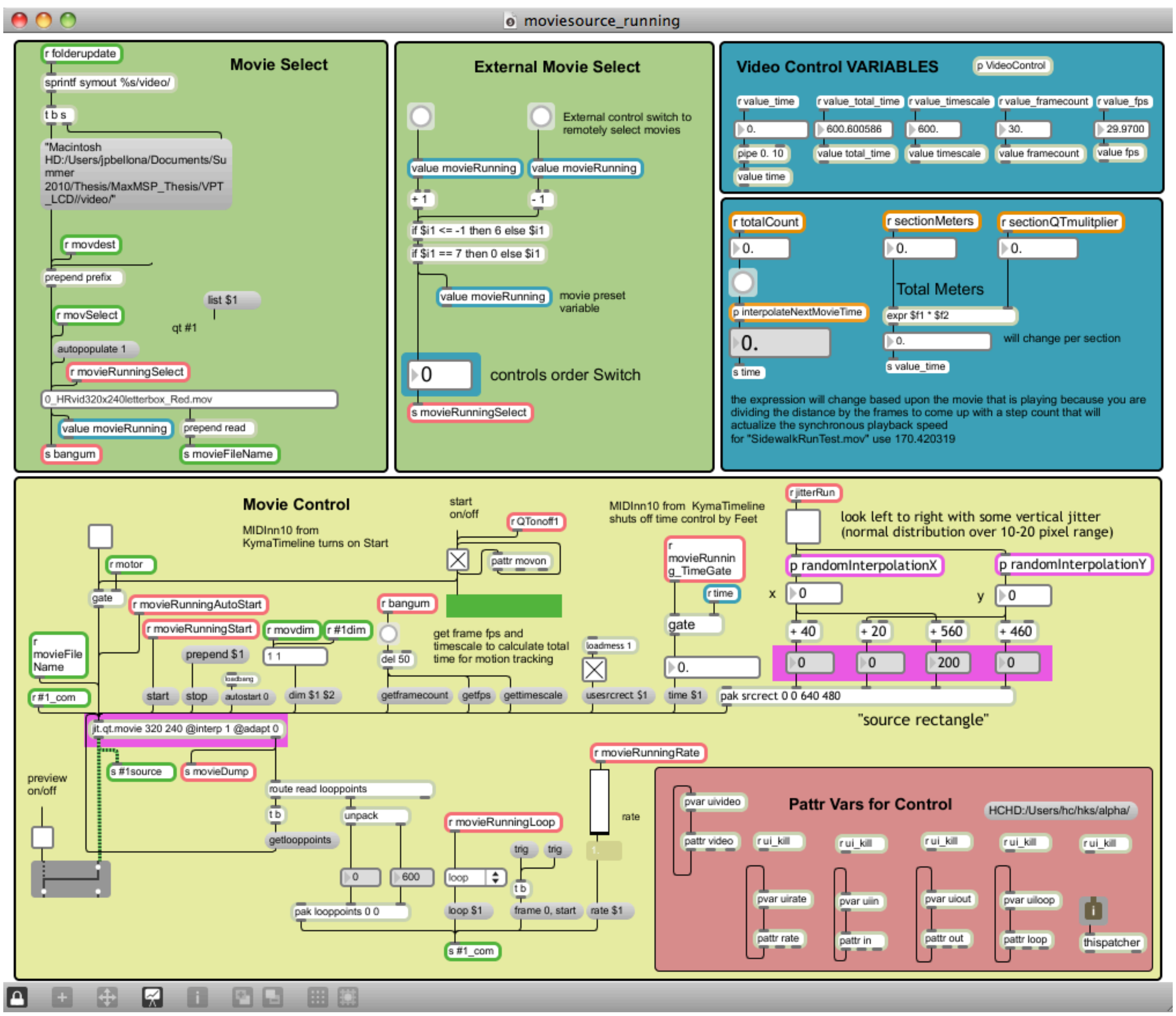



Figure A.2.e.2. Movie Source Running Patch Window, in Patcher mode

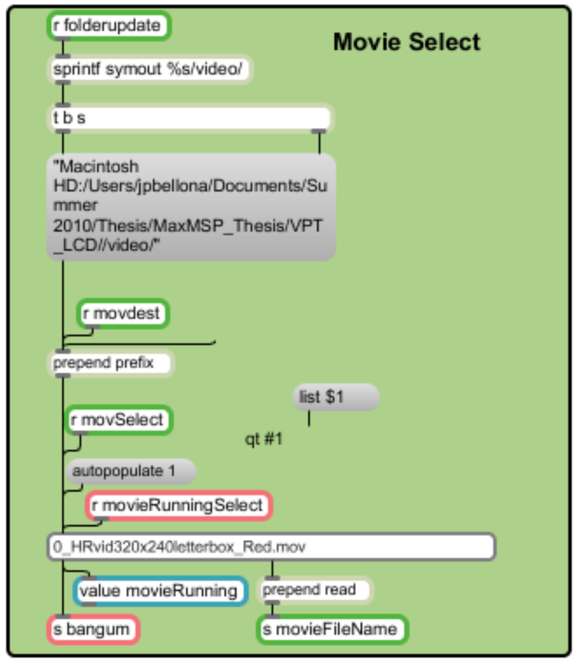



Figure A.2.e.3. Movie Source Select Module

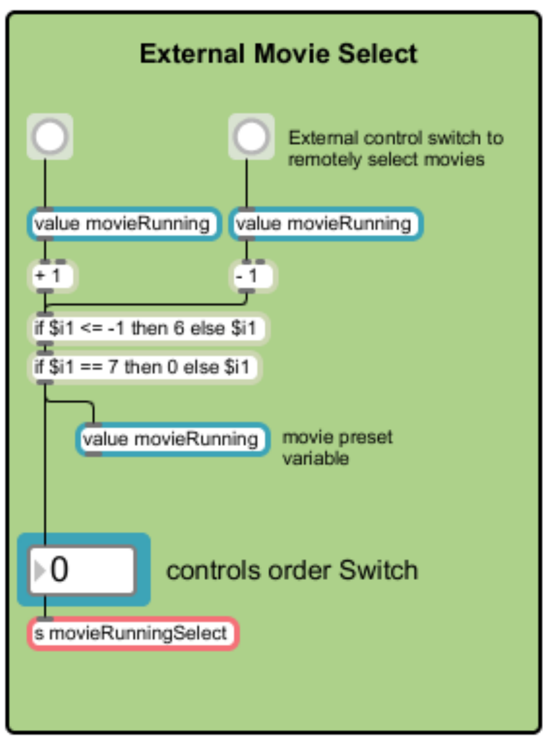



Figure A.2.e.4. Movie Source External Select Control Module

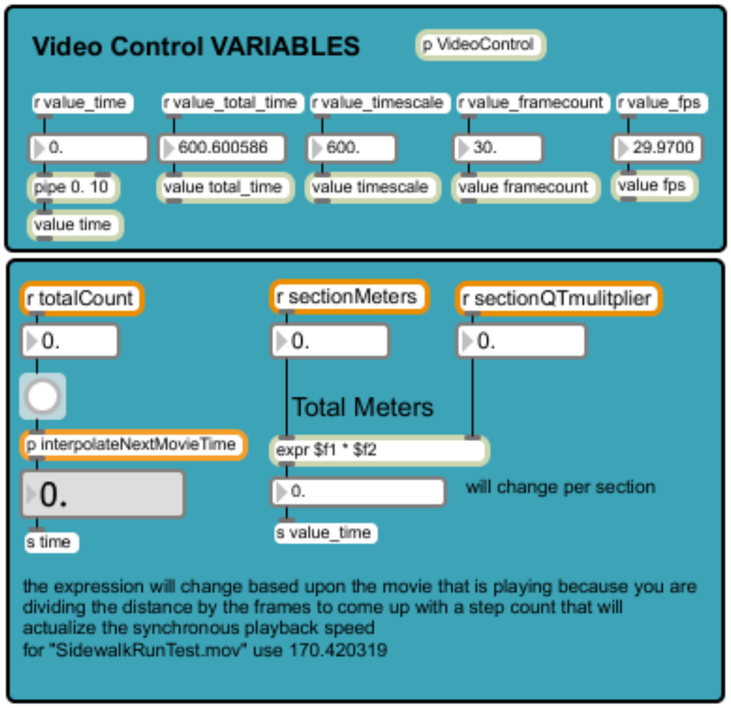



Figure A.2.e.5. Movie Source Video Control Variables Module

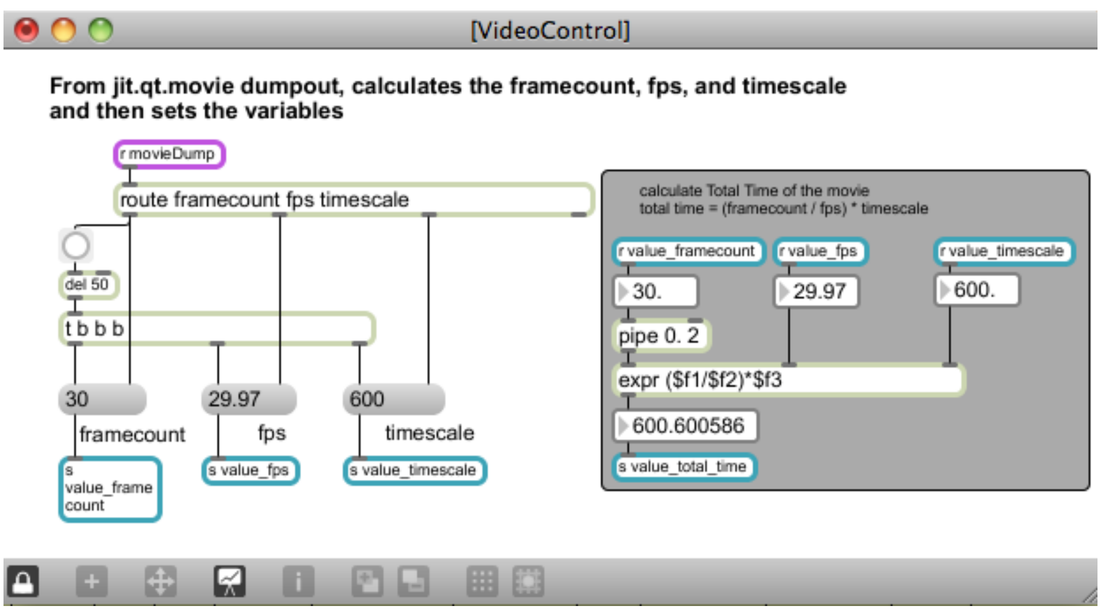



Figure A.2.e.6. Movie Source Variables Assignment Patcher

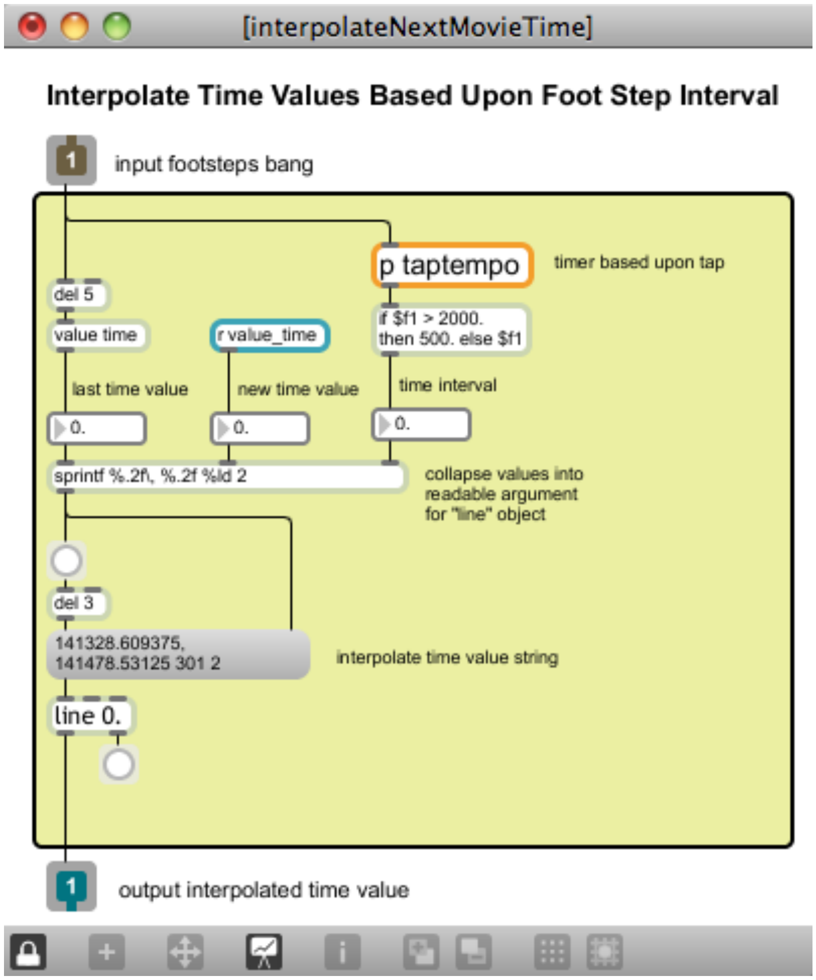



Figure A.2.e.7. Movie Source Video Position Interpolation Calculator Patcher

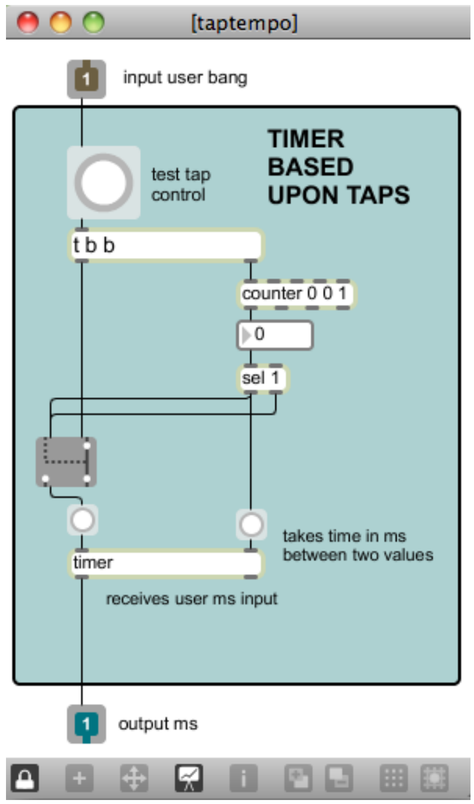



Figure A.2.e.8. Movie Source Position Interpolation Timer Patcher

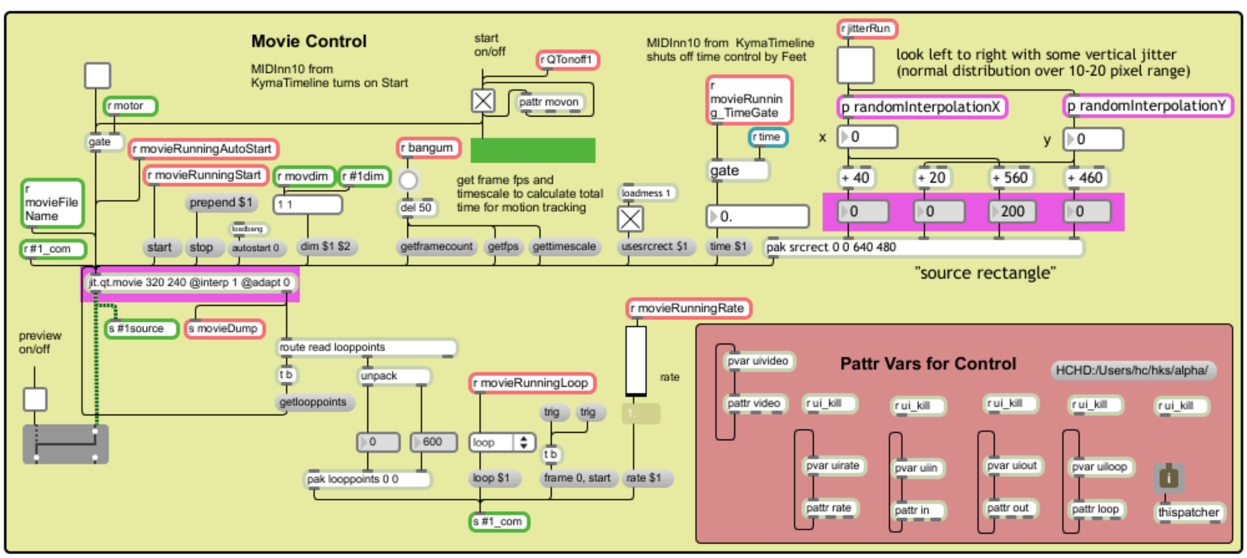



Figure A.2.e.9. Movie Control Module

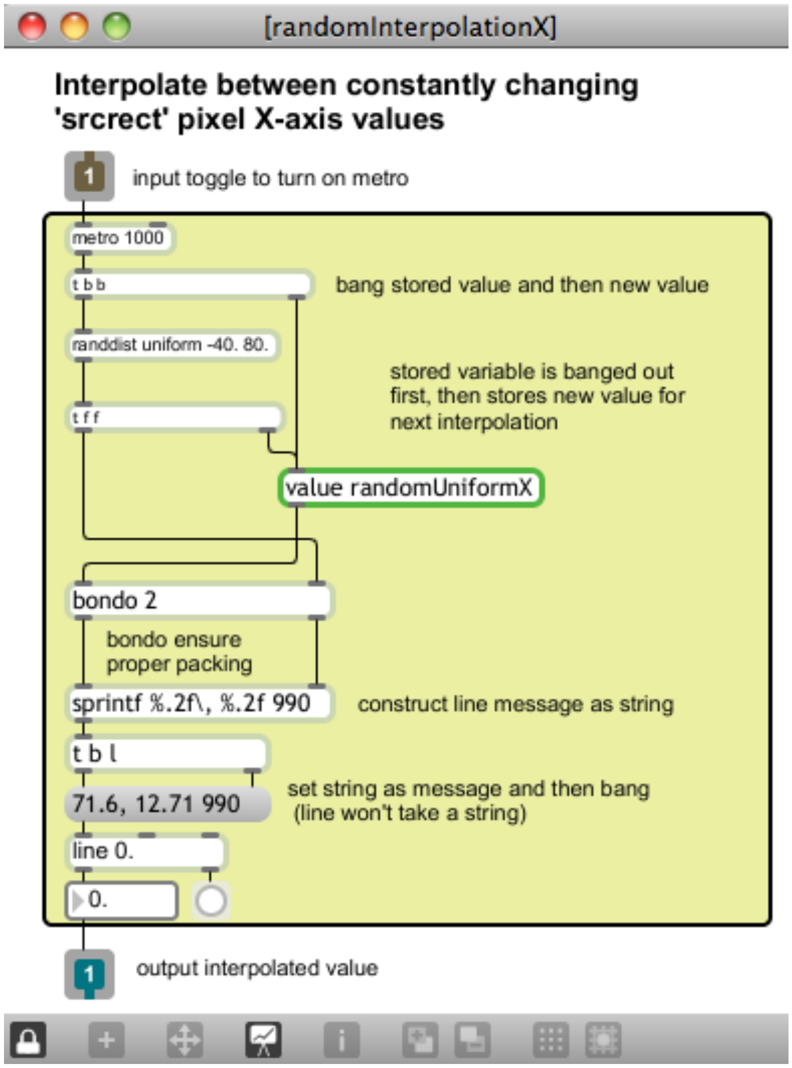



Figure A.2.e.10. Interpolation for ‘srcrect’ X-Axis Jitter Patcher

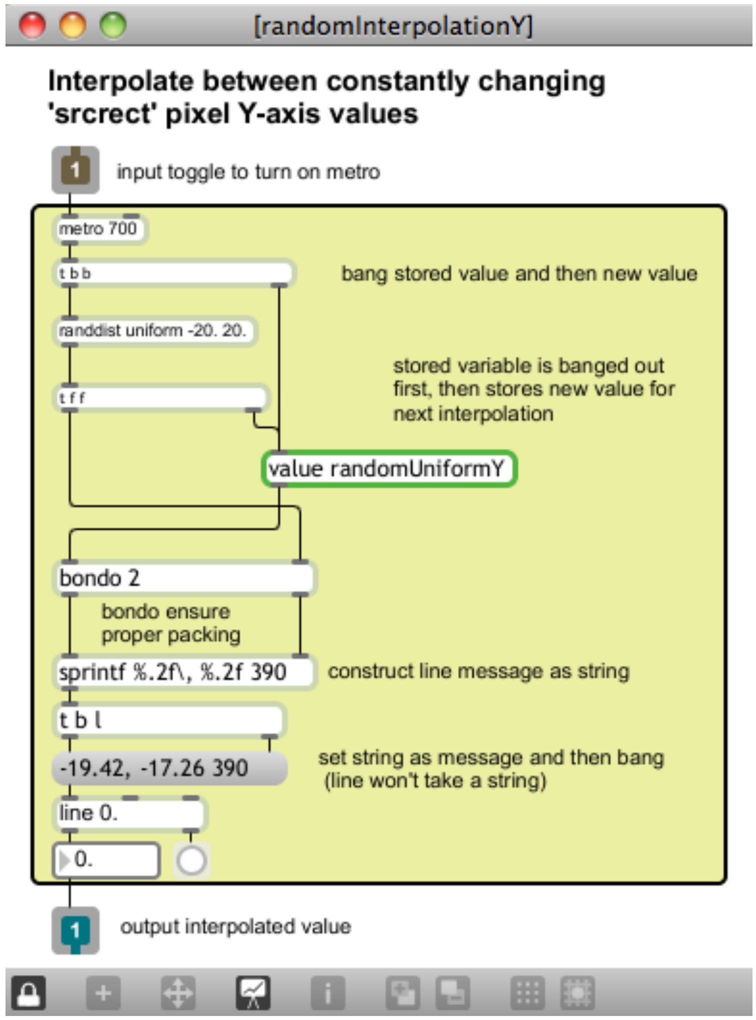



Figure A.2.e.11. Interpolation for 'srcrect' Y-Axis Jitter Patcher

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

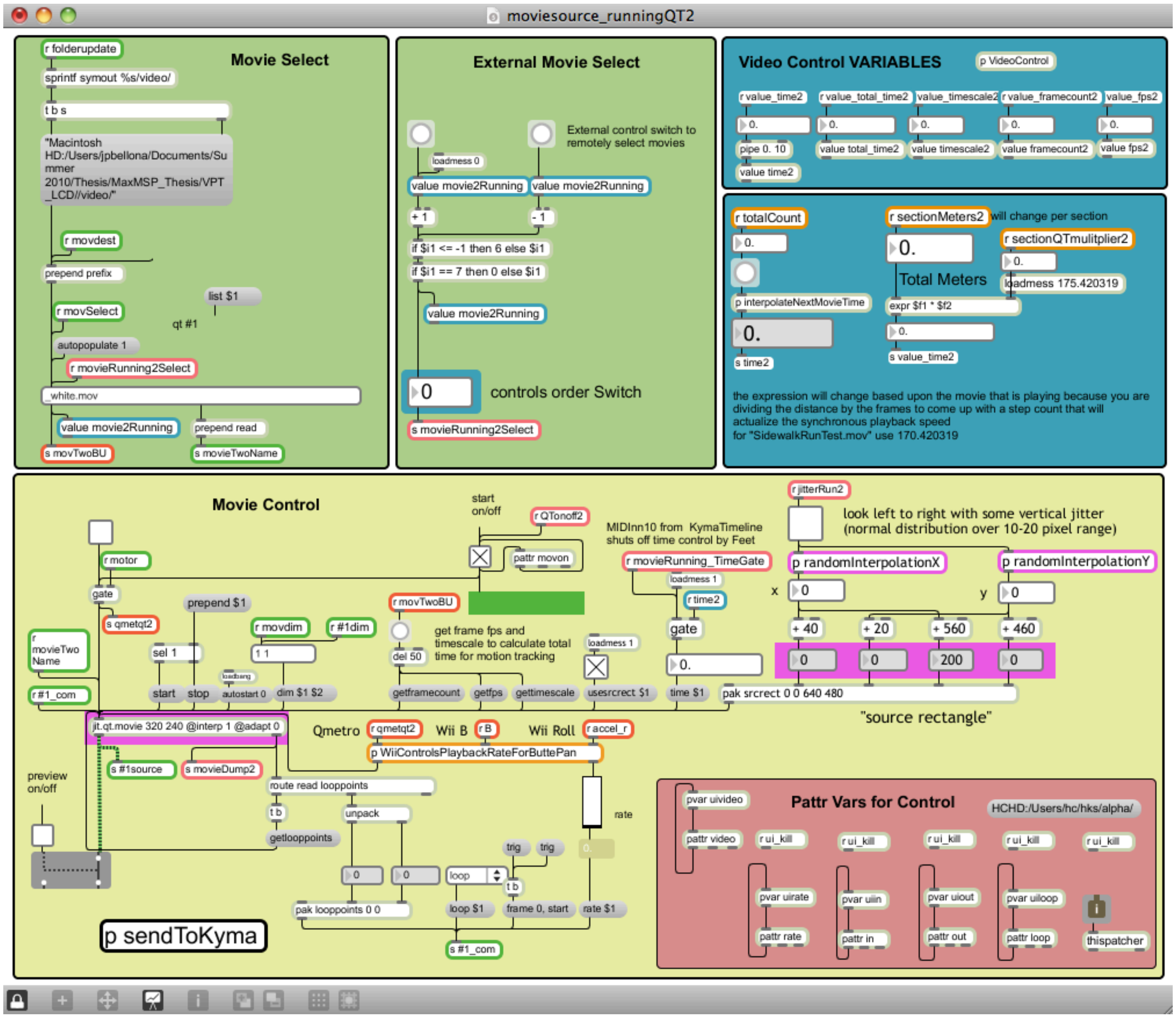



Figure A.2.f.1. Movie Source Running #2 Patch Window, in Patcher mode

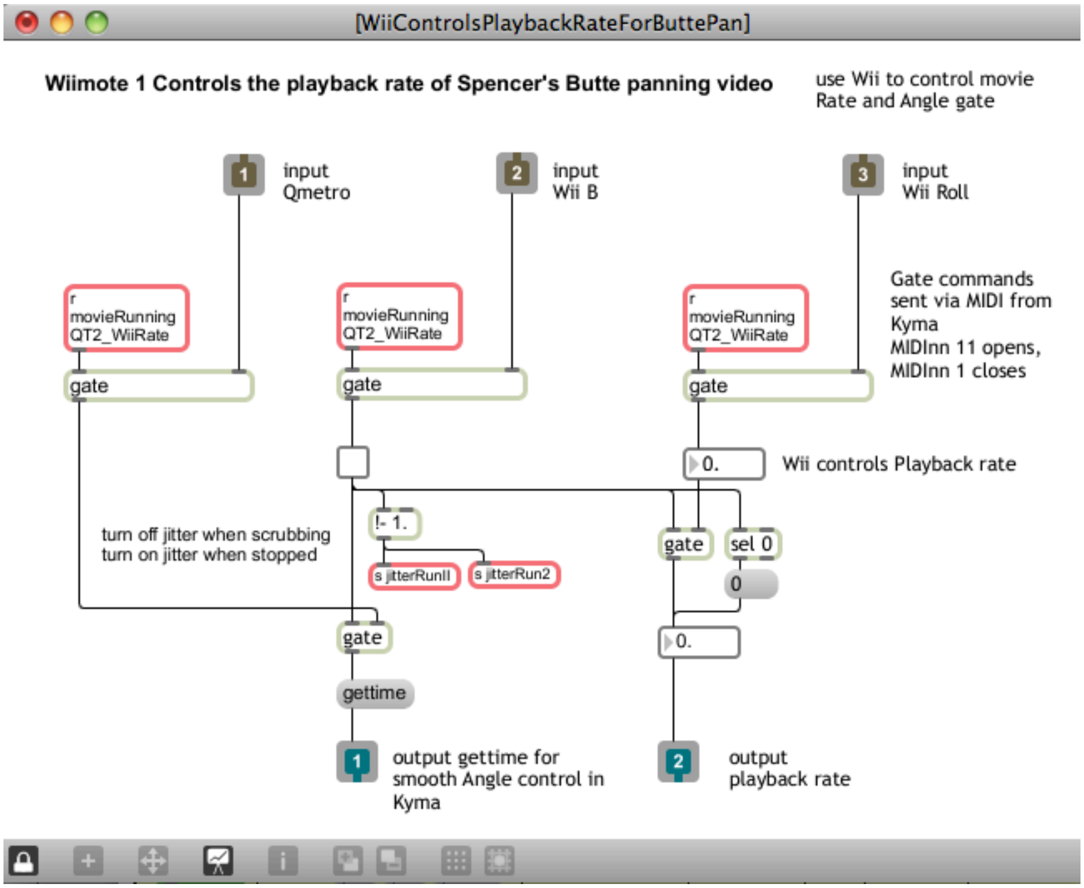



Figure A.2.f.2. Wiimote 1 Controls Panning Video Patcher

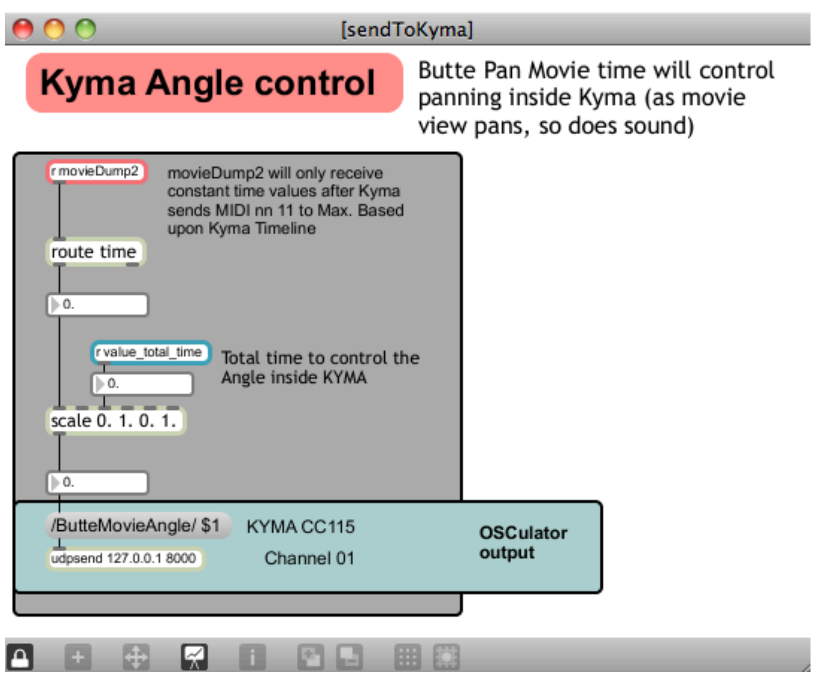



Figure A.2.f.3. Wiimote 1 Controls Kyma 8-channel Panning Patcher

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

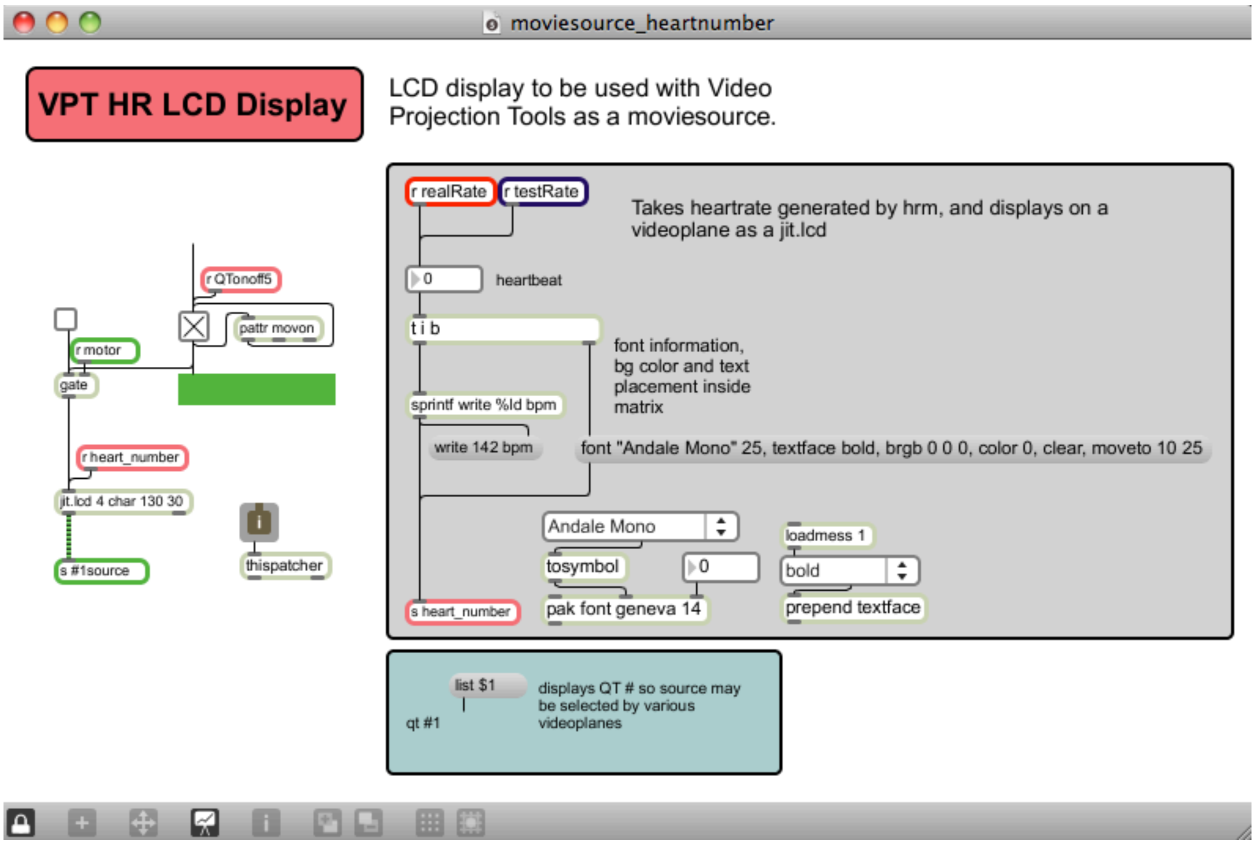



Figure A.2.g.1. Movie Source Heart Rate LCD Display, in Patcher mode

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

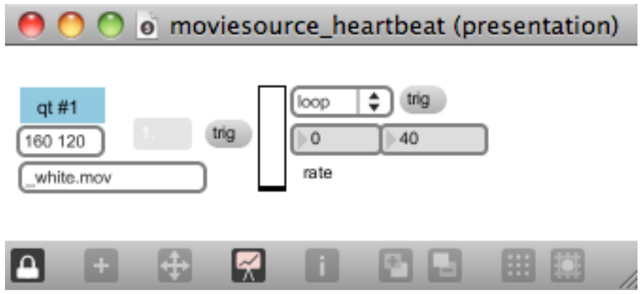



Figure A.2.h.1. Movie Source Heartbeat Patch Window, in Presentation Mode

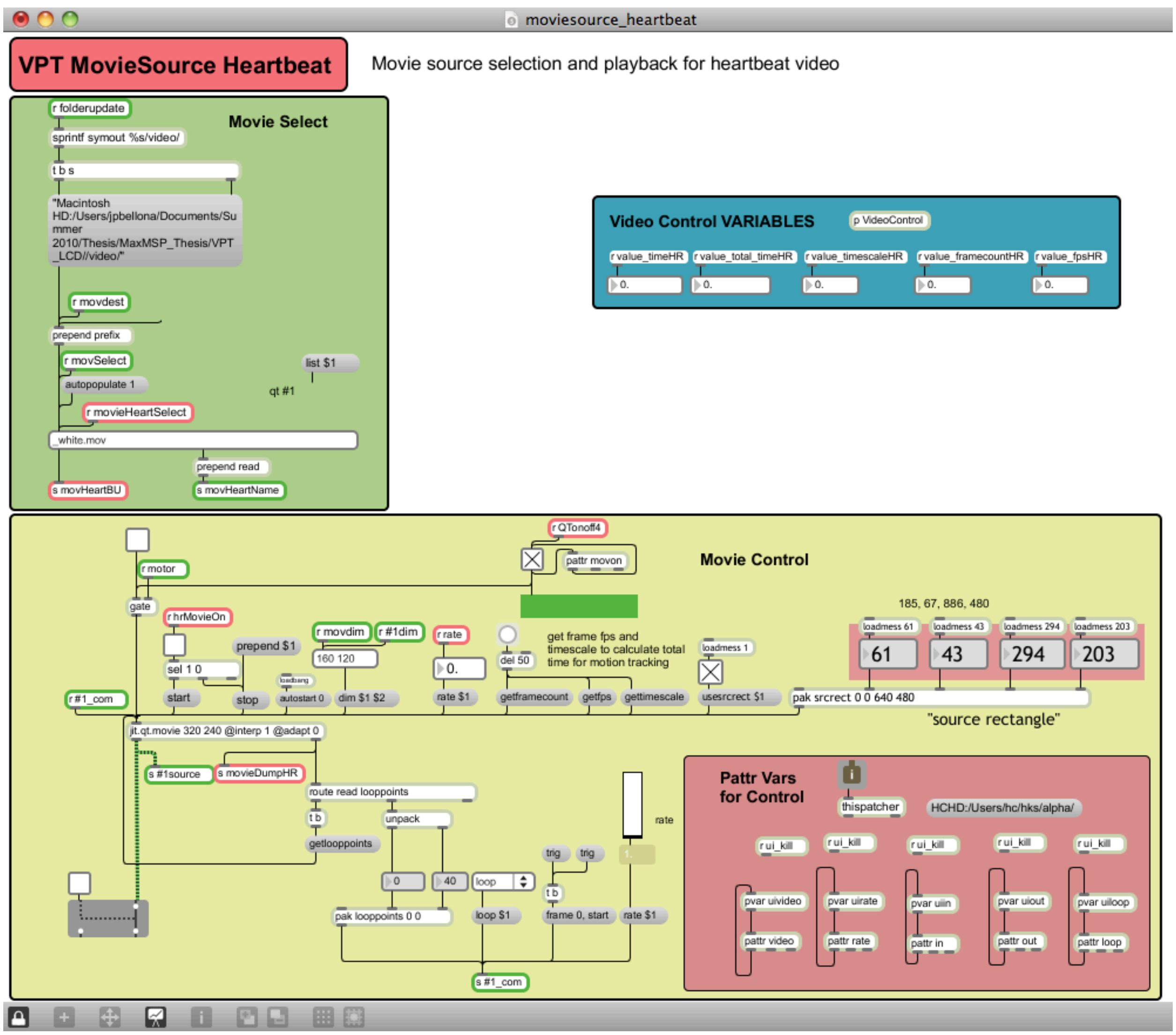



Figure A.2.h.2. Movie Source Heartbeat Patch Window, in Patcher Mode

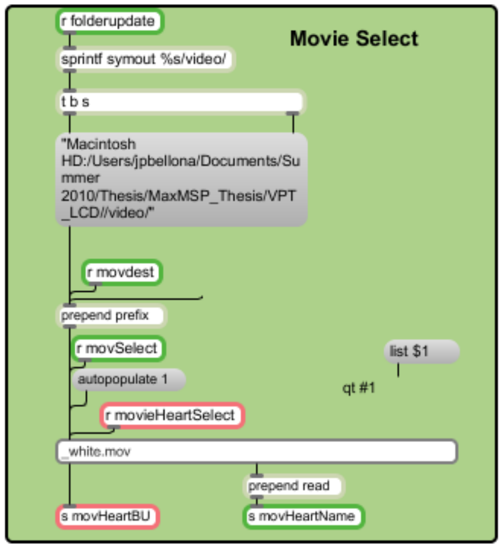



Figure A.2.h.3. Movie Select Heartbeat Module

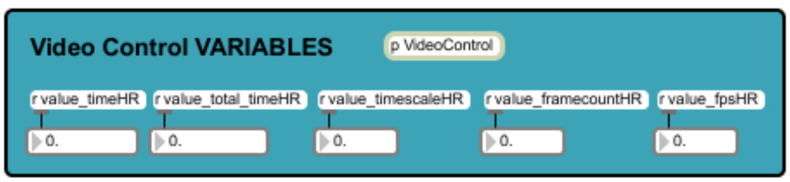



Figure A.2.h.4. Movie Heartbeat Video Control Variables Module

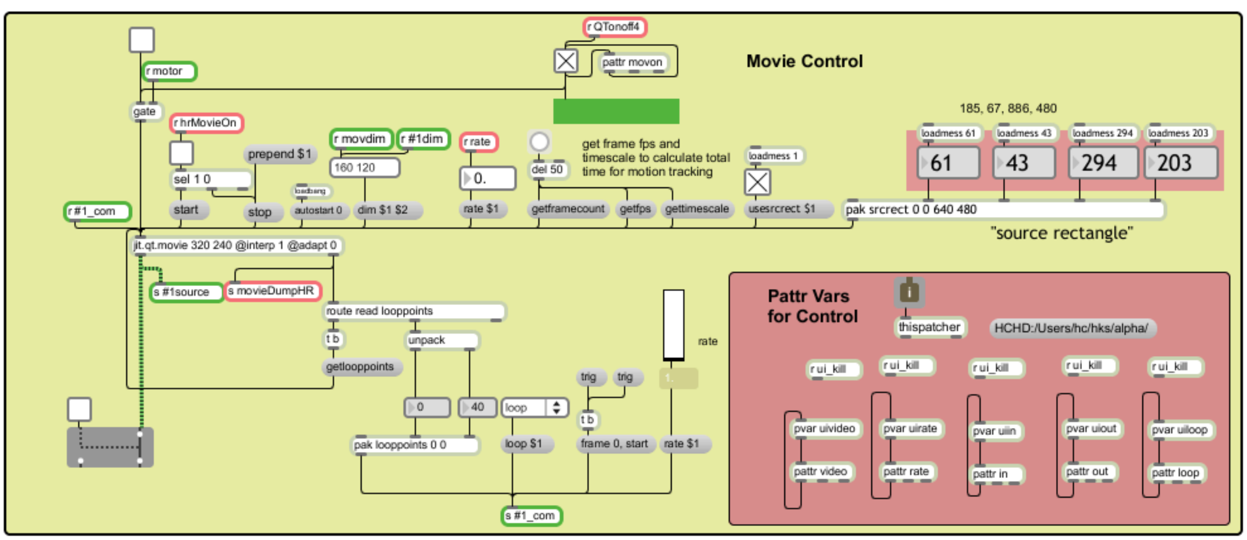



Figure A.2.h.5. Movie Control Heartbeat Module

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

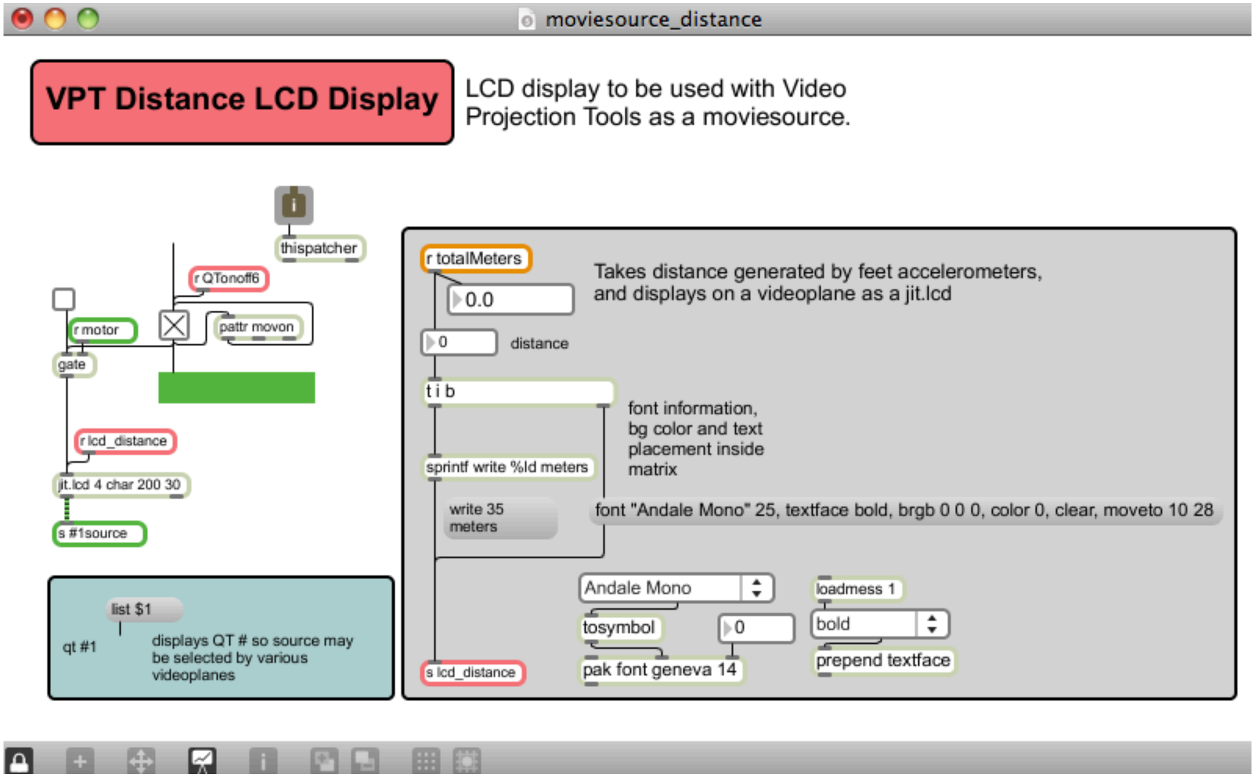



Figure A.2.i.1. Movie Source LCD Patch Window

## A.2.j. Cue List Mixer Module

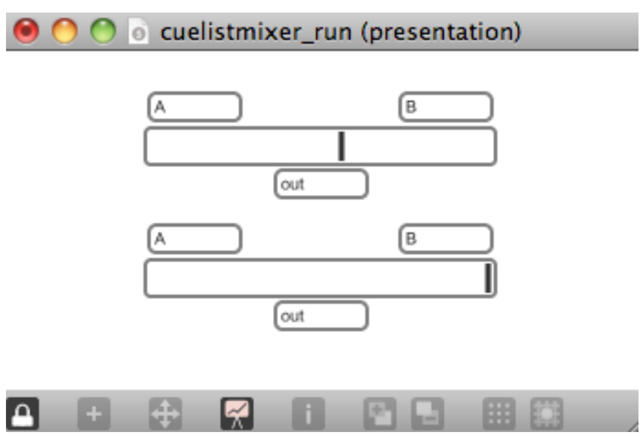



Figure A.2.j.1. Cue List Mixer Patch Window, in Presentation mode

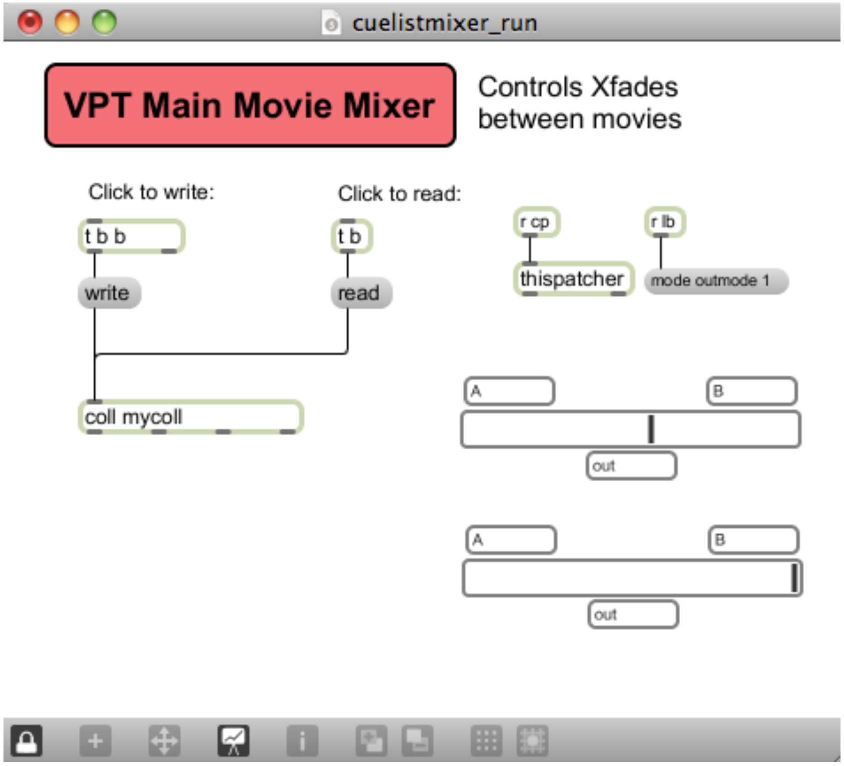



Figure A.2.j.2. Cue List Mixer Patch Window, in Patcher mode

## A.2.k. Mixer Module: Running

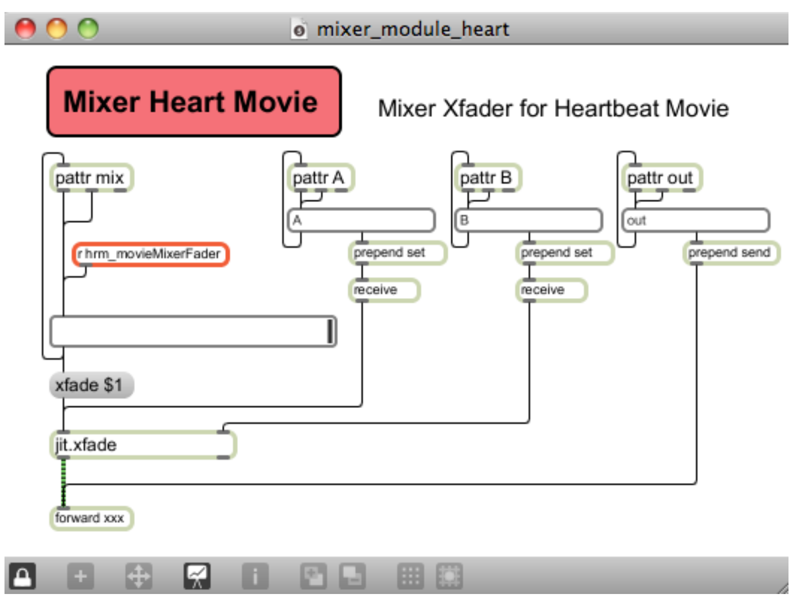



Figure A.2.k.1. Heartbeat Movie Mixer Patch Window

A.2.1. Mixer Module: Heart

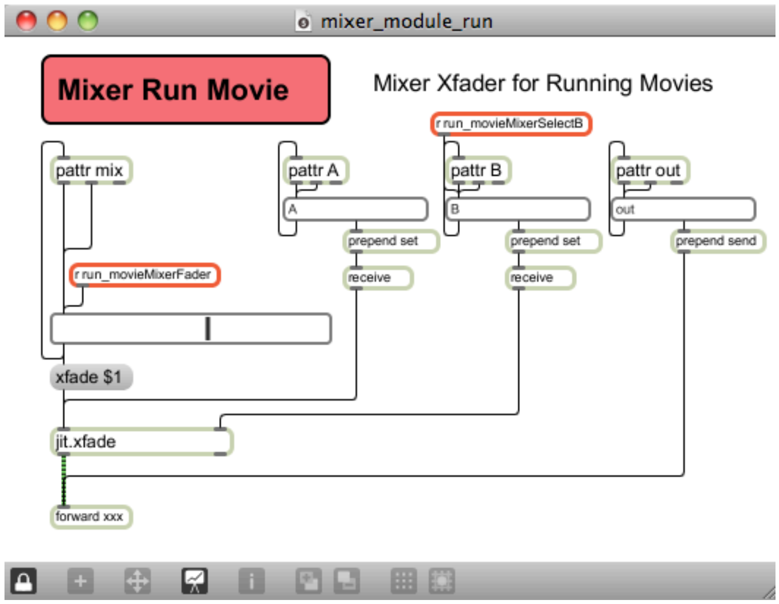



Figure A.2.1.1. Running Movie Mixer Patch Window

## 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 miles[20] 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.

A.4. Graphical Icon Legend

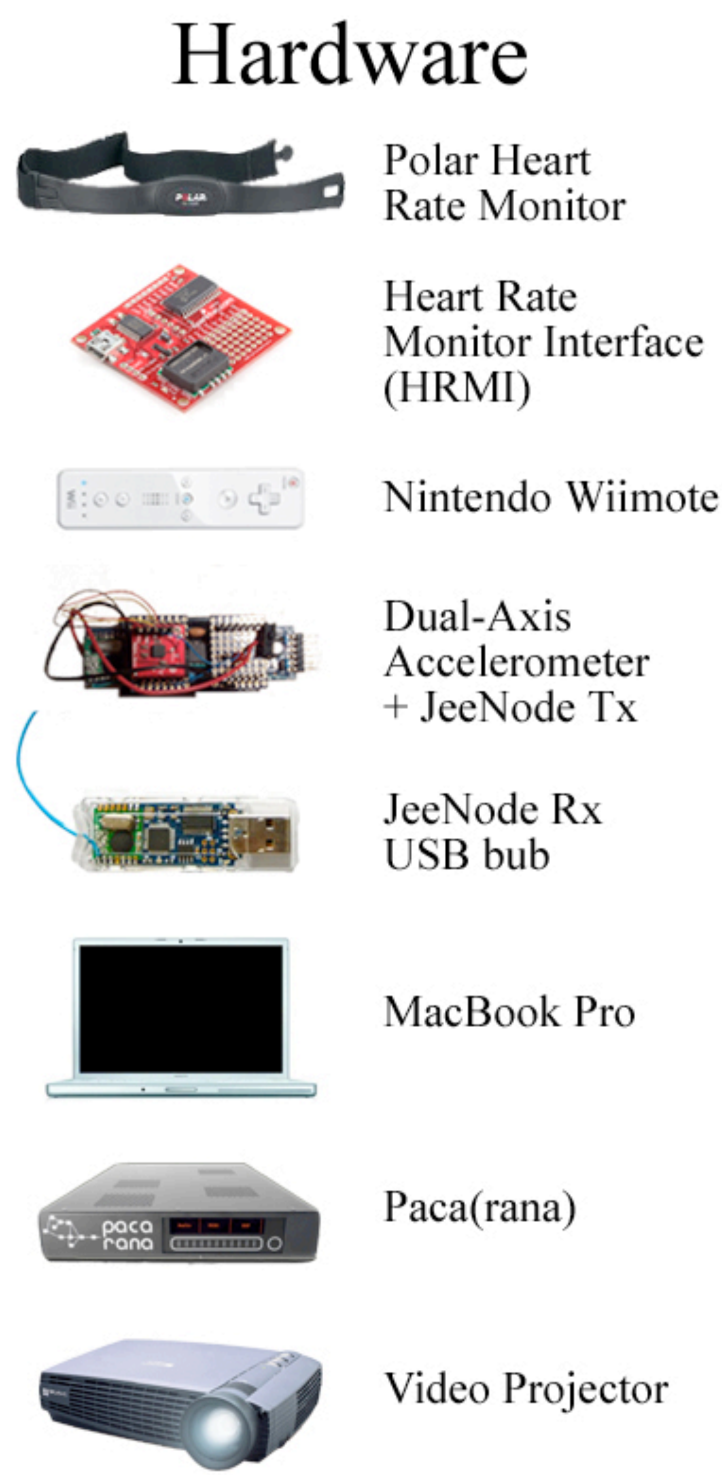



Figure A.4.1. Hardware icons used throughout the documentation

# Connections/Protocols

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: http://www.polarusa.com/us-en/support/downloads?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