21M.380

Music and Technology: Live Electronics Performance Practices

Christopher Ariza

21M.380: Music and Technology: Live Electronics Performance Practices
by Christopher Ariza

Table of Contents

Chapter 1. Meeting 1, Foundations: Live Electronics

1.1. Announcements

- 21M.380: Music Technology: Live Electronics Performance Practices
- Creative, hands-on exploration of contemporary and historical approaches to live electronics performance and improvisation, including basic analog instrument design, computer synthesis programming, and hardware and software interface design.

1.2. Overview

- Technology and instrument building
- The diversity of live electronics practices
- About this course

1.3. Instrument Building and Music Making

- Since humans have moved beyond the voice, music and technology have been closely linked
- The development of new technologies has nearly always led to new instruments
- The development of new instruments has often been linked to the development of new performance practices, ensembles, and musics
- In some traditions, instrument building is fundamental to music making
- With modern technology, instrument making becomes an aesthetic type interface design

1.4. Utility versus Aesthetics in Interface Design

- What are the criteria for evaluating a musical interfaces?

1.5. The Early Histories of Electronic and Computer Music

- At least four main branches of development in electronic and computer music
- All converge in live-electronics practices

1.6. A. Dedicated Electronic Instruments

- The Telharmonium: 1898-1901

(No Model.)

10 Sheets—Sheet 4.

T. CAHILL.

ART OF AND APPARATUS FOR GENERATING AND DISTRIBUTING MUSIC ELECTRICALLY.

No. 580,035. Patented Apr. 6, 1897.

Fig. 5.

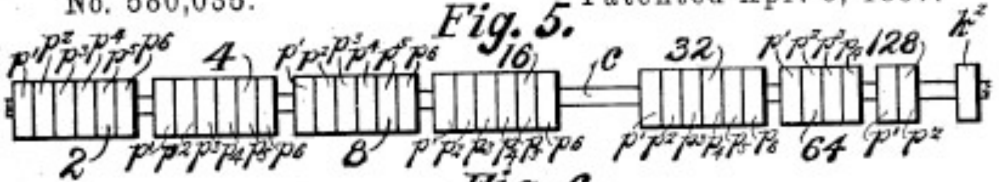


Fig. 6.

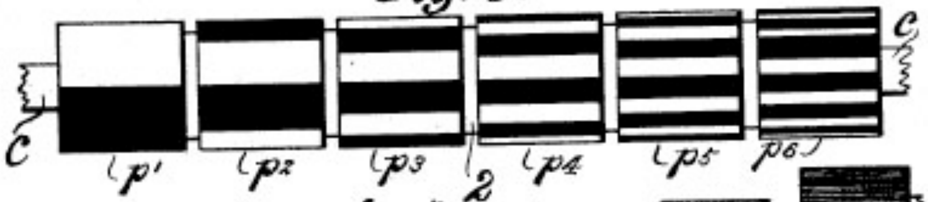


Fig. 7.

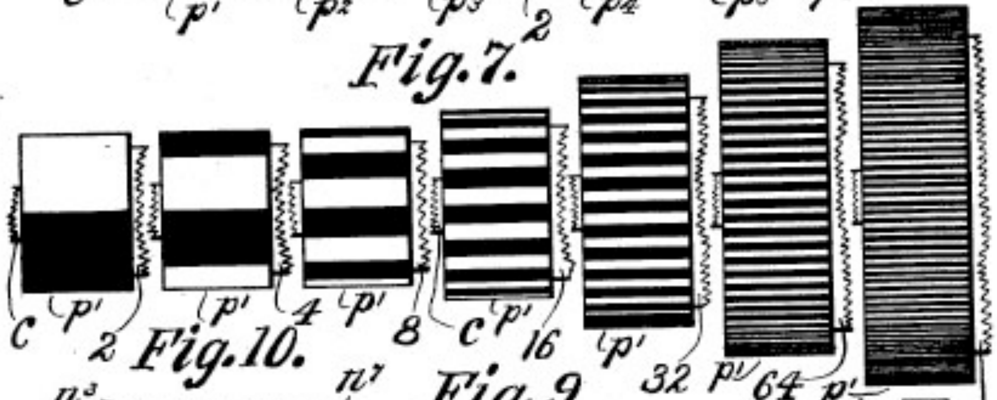


Fig. 10.

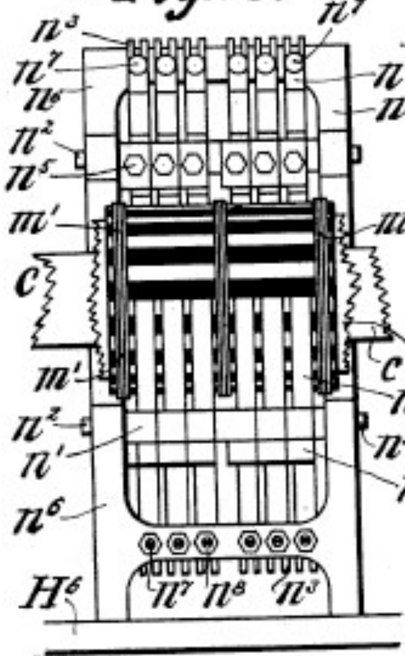


Fig. 9

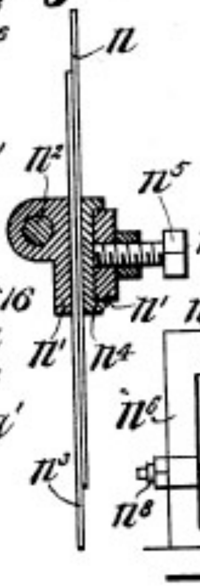
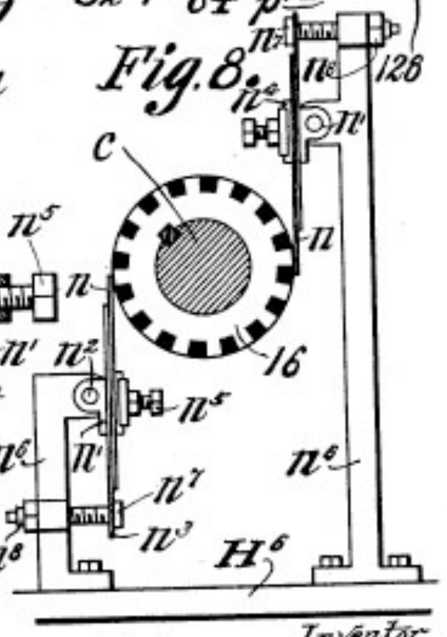
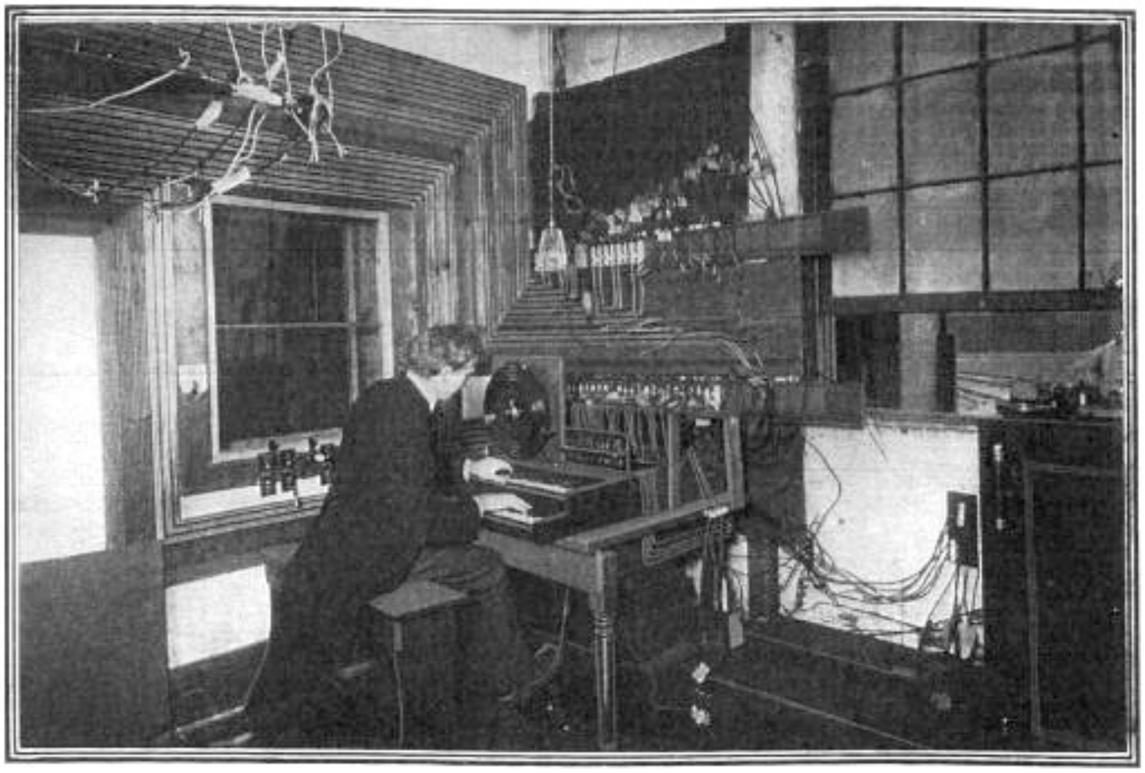


Fig. 8.



Attest:
Arthur J. Cahill.
M. H. Cahill.

Inventor:
Thaddeus Cahill



Public domain image.

- The Theremin: 1920

- The electric guitar and organ
- Turntables, radios, and related electronics

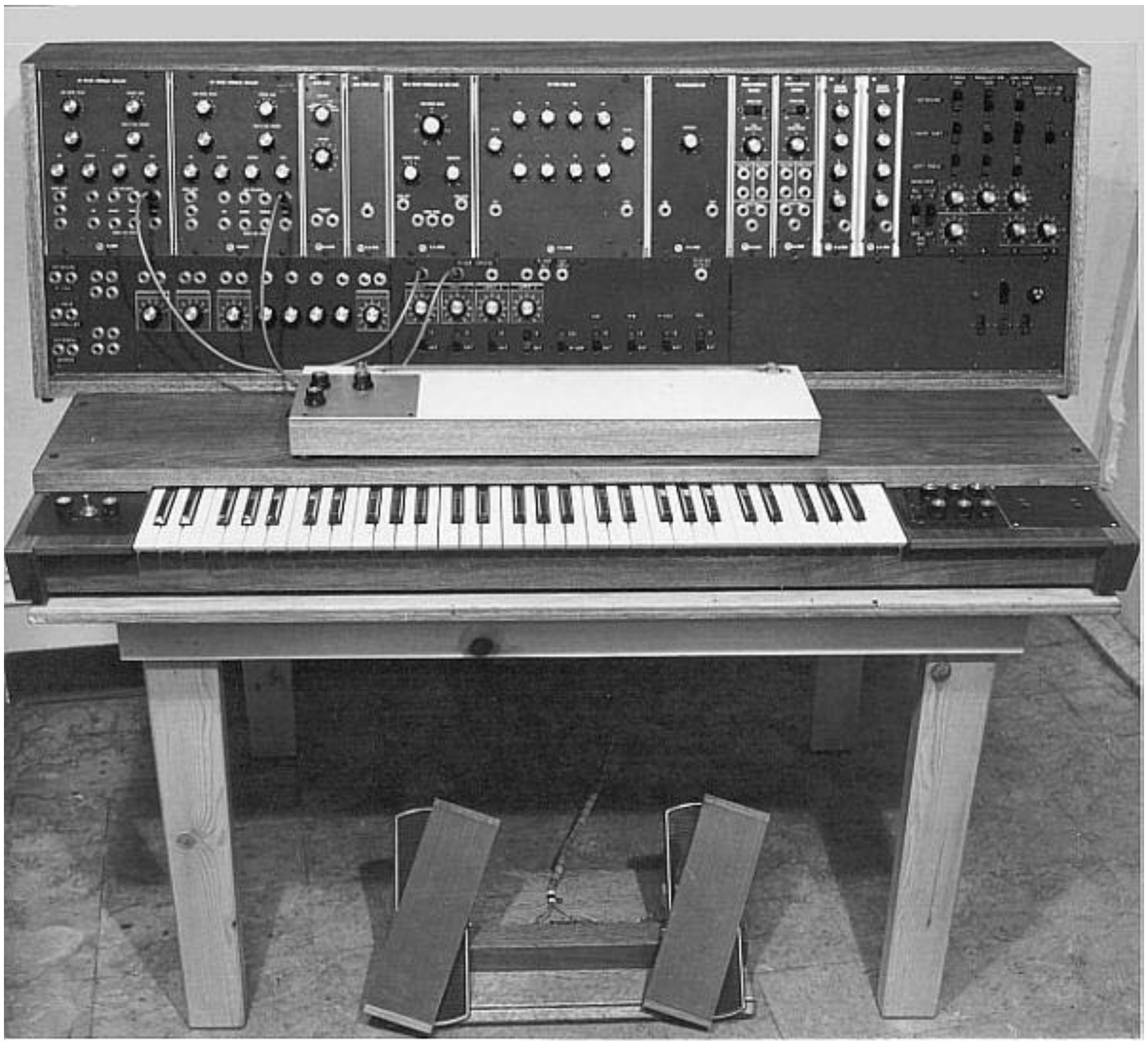
1.7. B. Analog Tape and Analog Synthesis

- Musicians, composers, and researchers repurpose radio equipment for music making

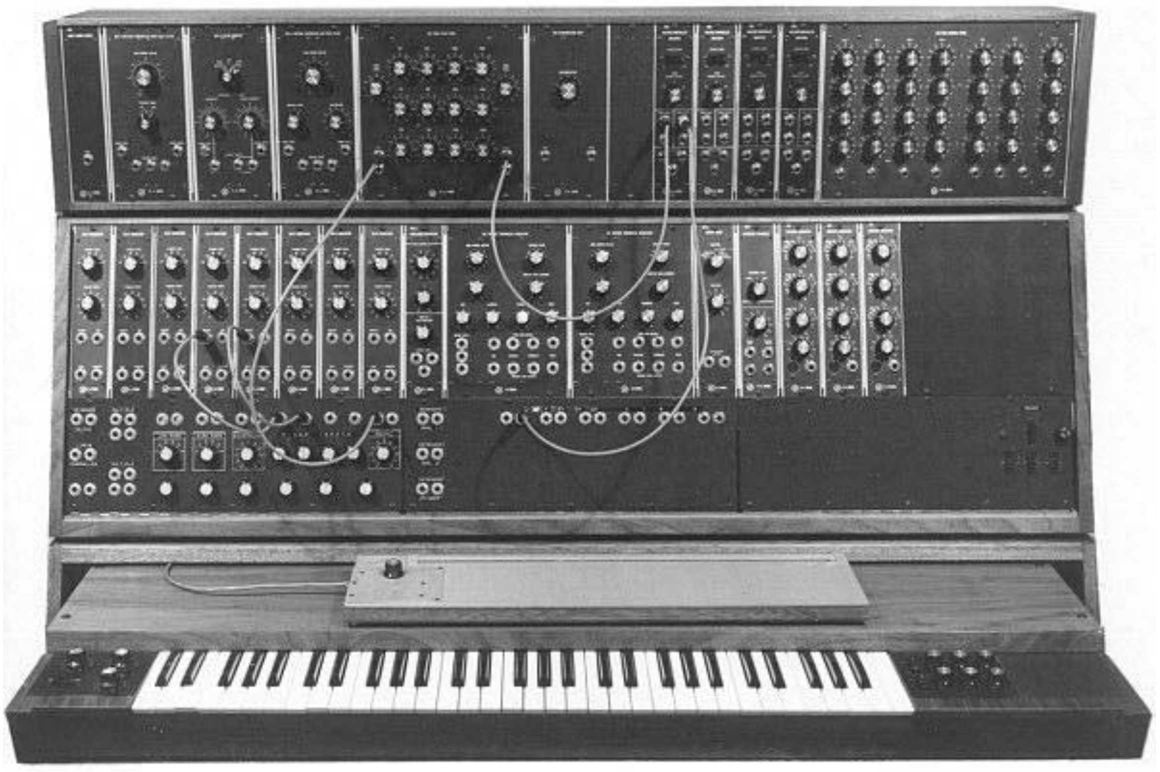
- Musique concrète
    - Pioneered by Pierre Schaeffer in late 1940s and 1950s
    - Developed techniques of manipulating, cutting, splicing, and transforming recorded sounds into music
- Electronic music
    - Pioneered by Werner Meyer-Eppler, Herbert Eimert, and Karlheinz Stockhausen in the 1950s
    - Developed techniques of combining sine tones and generated signals with filters and other modulation techniques
- Both techniques quickly merge
- Computer synthesis techniques routinely combine sample- and synthesis-based techniques

1.8. C. The Modular and Portable Synthesizers

- 1957: RCA Mark II, developed by Belar and Olson, installed
- 1964: Moog, with composer Deutsch, builds first synthesizer prototype
- 1967: Moog releases modular systems I, II, and III



Courtesy of Roger Luther. Used with permission.



Courtesy of Roger Luther. Used with permission.

- Numerous varieties of commercial and installation synthesizers are built in the 1960s
- Modular designs and interfaces provide a lasting legacy in all synthesis systems

1.9. D. Synthesis on Mainframes and Computers

- Early computers in the 1950s are used for synthesizing sound directly with integrated loudspeakers
- 1957: Max Mathews creates Music I on an IBM 704

Courtesy of Lawrence Livermore National Laboratory. Used with permission.

- Mathews develops Music II to V, Music *N* languages develop into to Csound
- Max/MSP and Pd software lineages begin in 1980s
- Other languages and processing frameworks continue to be developed: Supercollider, Impromptu, others

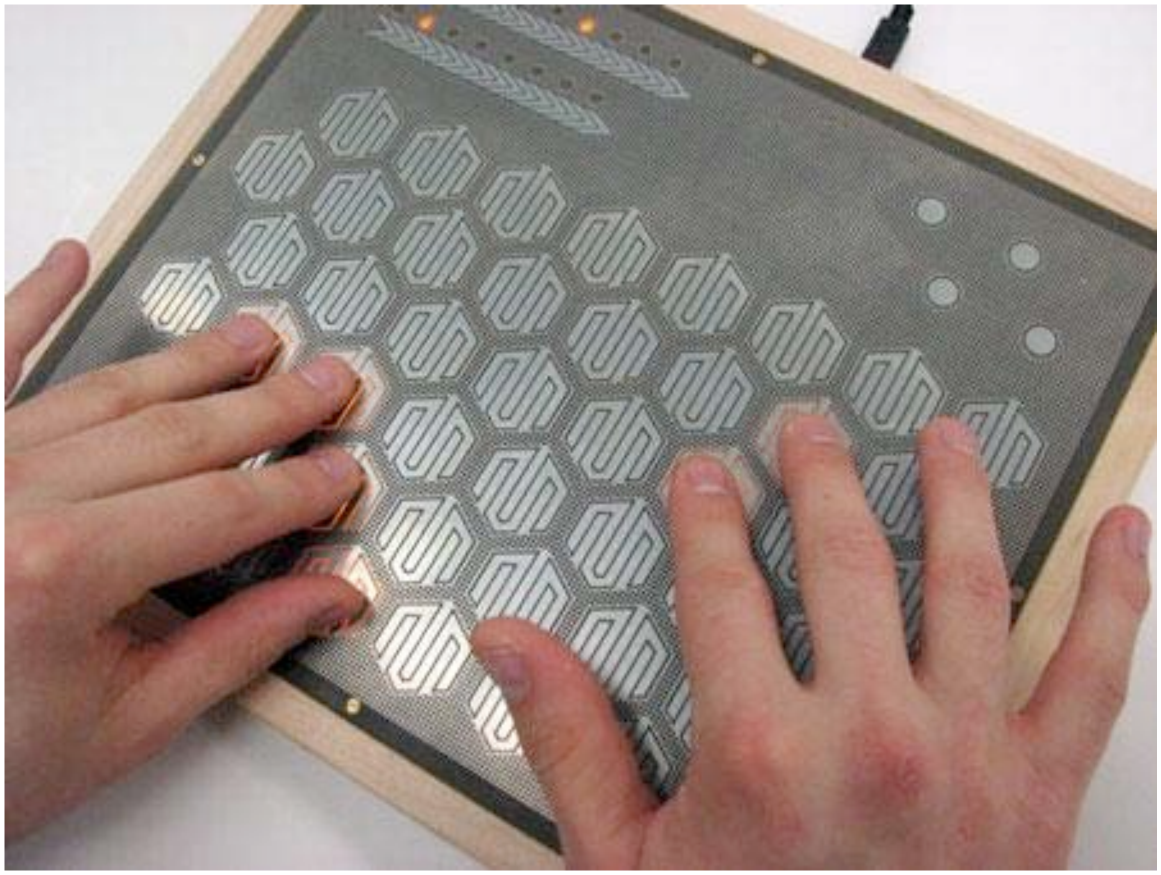
1.10. Live Electronics

- Deployment of these four traditions in performance contexts
- Long tradition of composers and musicians building custom instruments and interfaces
- Long tradition of employing composition and improvisatory practices from jazz and experimental music
- Two basic approaches
    - Develop interfaces for controlling computer synthesis and processing

- Develop interfaces integrated with electro-mechanical devices and electronics

1.11. A. Interfaces to Computer Synthesis

- The laptop: keyboards, trackpads, mice
- Gamepads: joysticks, wii controllers, related
- Touchscreens: iPhone, iPad, etc
- Custom musical controllers
  - Manta

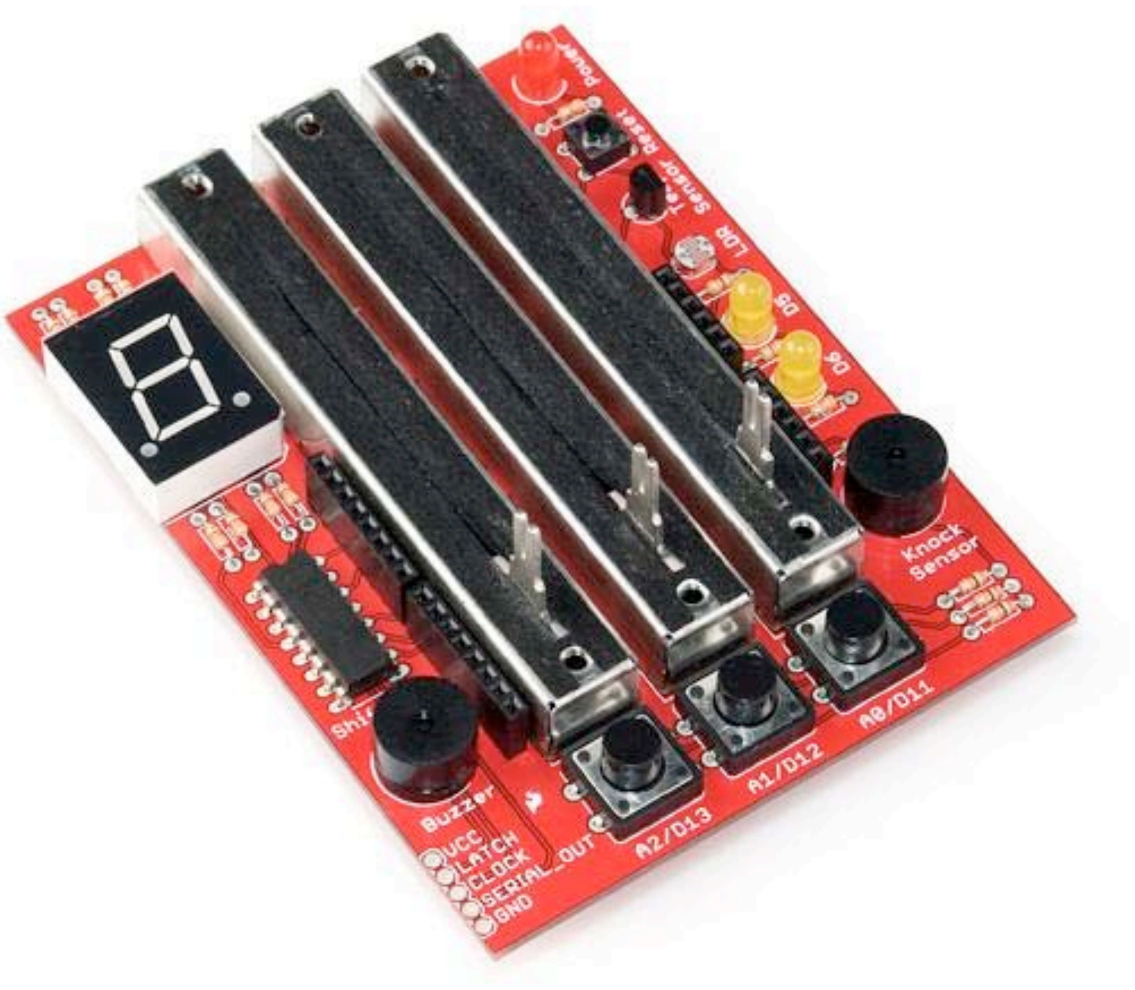


Courtesy of Snyderphonics. Used with permission.

  - Lemur

- Analog sensor input via Arduino or other devices

Photo courtesy of SparkFun Electronics.



Arduino Danger Shield.
Photo courtesy of SparkFun Electronics.

1.12. B. Interfaces with Integrated Sound Sources

- Turntables
- Custom-built circuits
- Manipulated speakers and other transducers
- Manipulated casette decks and tape players

1.13. A Personal Performative Context

- For composers of computer music, the move to live-electronics is a major constraint
- Performing with acoustic instruments is a significant musical and technological challenge

1.14. KIOKU

- Trio of taiko and percussion, alto sax, and live electronics, from 2006 to the present

- Deploy east-asian traditional and folk songs in a free-jazz inspired context
- Led to the development of a comprehensive system employing a number of inexpensive controllers
- Listening: KIOKU: Pinari

1.15. KIOKU Performance System: libOX

- Modular Max/MSP system with a large collection of controllers

- Two dual-analog game pads form the key performance interface

1.16. The Dual Analog Gamepad

- Logitech Dual Action Gamepad

- 2 XY joysticks, 10 buttons, 1 5-position d-pad
- Example: a simple noise instrument
    - Buttons trigger noise sources with different envelopes
    - XY joystick 1: y axis control amplitude, x axis controls high-pass filter
    - XY joystick 2: y axis control amplitude modulation, x axis controls low-pass filter
- An ergonomic, expressive, and inexpensive interface

1.17. 21M.380: Areas of Focus

- Syllabus: download from Stellar
- Historical traditions and practices
- Instrument and interface design
    - Pd synthesis tools

    - Dual-analog game pads
    - Touchscreen controllers with TouchOSC
    - Sensors and physical inputs with Arduino
    - Elementary electro-magnetic instruments
    - Custom-built circuits with basic CMOS ICs
- Composition and improvisation

1.18. 21M.380: Prerequisites

- None but curiosity, willingness to experiment

1.19. 21M.380: Course Meetings

- Two types of meetings
    - Topic meetings: focused on material in readings, listening, and themes, combining lecture, discussion, demonstration, and listening
    - Workshop meetings: improvisation, performance, and discussion
- Bring laptops to all class meetings
- Lecture notes
    - Provided via course website

1.20. 21M.380: Required Course Materials: Software

- Pure Data (Pd), the Pd-extended distribution

  Provides installers for a number of platforms and bundles valuable extension libraries

  http://puredata.info/downloads
- Martingale: a library of Pd resources

http://code.google.com/p/martingale

1.21. 21M.380: Required Course Materials: Hardware

- Hardware resources up to $100 are required
- A portable powered or battery-powered amplifier for keyboards/synthesizers and the appropriate cable (not a guitar amp). The following are good options, in order of preference:

1. Phonic MK15 Keyboard Amp ($80) with 3.5mm stereo to RCA male cable (at least 6 feet long)

2. Behringer Ultratone KT108 15w Keyboard Amplifier ($70 to $75) with 3.5mm stereo to RCA male cable (at least 6 feet long) and two RCA female to 1/4 inch TS (mono) male adapters.

3. Altec Lansing iM-237 Orbit Ultraportable Speaker ($18) or similar (includes attached cable).

4. Numerous alternatives are acceptable: contact me

- A dual-analog game controller

    1. Logitech Dual Action USB Gamepad ($10 to $17)

    2. There are a few alternatives that may work: contact me

1.22. 21M.380: Assignments: Reading

- All reading assignments are posted on the course website.

- One book:

  Collins, N. 2009. *Handmade Electronic Music: The Art of Hardware Hacking*. 2nd ed. New York: Routledge.

- Numerous carefully selected articles and chapters:

  1. Brown, A. R. and A. Sorensen. 2009. "Interacting with Generative Music through Live Coding." *Contemporary Music Review* 28(1): pp. 17-29.

2. Cascone, K. 2004. "Grain, Sequence, System [three levels of reception in the performance of laptop music]." *Intelligent Agent* 4(1).

3. Collins, N. 2003. "Live Coding in Laptop Performance." *Organised Sound* 8(3): pp. 321-330.

4. Cook, P. 2001. "Principles for designing computer music controllers." *Proceedings of the Conference on New Interfaces for Musical Expression.*

5. Dennis, B. 1991. "Cardew's 'Treatise' (Mainly the Visual Aspects)." *Tempo* 177: pp. 10-16.

6. Driscoll, J. and M. Rogalsky. 2004. "David Tudor's 'Rainforest': An Evolving Exploration of Resonance." *Leonardo Music Journal* 14: pp. 25-30.

7. Fiebrink, R. and G. Wang, P. Cook. 2007. "Don't Forget the Laptop: Using Native Input Capabilities for Expressive Musical Control." *Proceedings of the Conference on New Interfaces for Musical Expression* pp. 164-167.

8. Ghazala, Q. R. 2004. "The Folk Music of Chance Electronics: Circuit-Bending the Modern Coconut." *Leonardo Music Journal* 14(1): pp. 97-104.

9. Gresham-Lancaster, S. 1998. "The Aesthetics and History of the Hub: The Effects of Changing Technology on Network Computer Music." *Leonardo Music Journal* 8: pp. 39-44.

10. Holmes, T. 2008. "Live Electronic Music and Ambient Music." In T. Holmes, ed. *Electronic and Experimental Music.* Third ed. New York: Routledge, pp. 376-406.

11. Kuivila, R. 2004. "Open Sources: Words, Circuits and the Notation-Realization in the Music of David Tudor." *Leonardo Music Journal* 14: pp. 17-23.

12. Perkis, T. 2009. "Some Notes on My Electronic Improvisation Practices." In R. T. Dean, ed. *The Oxford Handbook of Computer Music.* Oxford University Press, pp. 161-166.

13. Puckette, M. 2002. "Max at 17." *Computer Music Journal* 26(4): pp. 31-43.

14. Rebelo, P. and A. Renaud. 2006. "The Frequencyliator—Distributing Structures for Networked Laptop Improvisation." *Proceedings of the Conference on New Interfaces for Musical Expression* pp. 53-56.

15. Ryan, J. 1991. "Some Remarks on Musical Instrument Design at STEIM." *Contemporary Music Review* 6(1): pp. 3-17.

16. Smallwood, S. and D. Trueman, P. R. Cook, G. Wang. 2008. "Composing for Laptop Orchestra." *Computer Music Journal* 32(1): pp. 9-25.

17. Tanaka, A. 2009. "Sensor-Based Musical Instruments and Interactive Music." In R. T. Dean, ed. *The Oxford Handbook of Computer Music.* Oxford University Press, pp. 233-257.

18. Trueman, D. 2007. "Why a Laptop Orchestra?." *Organised Sound* 12(2): pp. 171-179.

19. Wanderley, M. M. and N. Orio. 2002. "Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI." *Computer Music Journal* 26(3): pp. 62-76.

20. Wang, G. 2007. "A History of Programming and Music." In N. Collins and J. d'Escriván, eds. *The Cambridge Companion to Electronic Music.* Cambridge: Cambridge University Press, pp. 55-71.

21. Weinberg, G. 2002. "Playpens, Fireflies, and Squeezables: New Musical Instruments for Bridging the Thoughtful and the Joyful." *Leonardo Music Journal* 12: pp. 43-51.

22. Wright, M. 2005. "Open Sound Control: an enabling technology for musical networking." *Organised Sound* 10(3): pp. 193-200.

1.23. 21M.380: Assignments: Listening

- All listening assignments will be posted on the course website.
- Take notes when you listen
- What to listen for without notation: duration, instrumentation, method of production, recording or performance context, notable sonic events, form, temporal design and proportions, aesthetic or historical contexts, and/or critical and subjective responses

1.24. 21M.380: Assignments: Discussion Leaders

- Students are assigned to cover reading and listening assignments for each class
- Must be available to lead discussion, answer questions, and provide a resource to class
- Must post minimal notes on Stellar in the Forum: Reading and Listening Notes
- Need two volunteers for next class

1.25. 21M.380: Assignments: Pd Tutorials

- Short programming exercises in Pd
- Must be completed before the next class

1.26. 21M.380: Assignments: Controller/Interface/Instrument Design Projects

- Controller/Interface/Instrument Design 1
  - Must use Pd and dual-analog controller

    - Must present draft and complete a report
    - Due before spring break
- Controller/Interface/Instrument Design 2
    - May use any interface, design, or approach (as long as there are at least two performative input parameters)
    - Must present proposal, draft, and complete a report
    - Due at end of semester

1.27. 21M.380: Assignments: Performance Framework

- A design for a composition or performance context
- Completed in small groups
- Will be presented on a concert on 4 May

1.28. 21M.380: Assignments: Submission

- All assignments are submitted digitally via email attachment (or as Forum posts)
- Some assignments are due before class, others are due at 11:59:59 PM on due date
- Late within 1 week: 20% reduction; no assignments accepted after 1 week

1.29. 21M.380: Attendance

- Mandatory and essential
- Always communicate with me about needs for excused absences
- More than one unexcused absence incurs a 3% grade reduction

1.30. 21M.380: Exams and Quizzes

- Quizzes will be announced
- All short written answers
- Quizzes will be based on reading, listening, and course content

- No final exam

1.31. 21M.380: Grading

- Reading and Listening Discussion Leader: 10%
- Pd Tutorials: 10%
- Controller/Interface/Instrument Design 1 Implementation and Report: 15%
- Controller/Interface/Instrument Design 1 Draft: 2.5%
- Controller/Interface/Instrument Design 2 Implementation, Report, and Presentation: 20%
- Controller/Interface/Instrument Design 2 Draft: 2.5%
- Controller/Interface/Instrument Design 2 Proposal: 2.5%
- Performance Framework: 10%
- Performance Frameworks Draft: 2.5%
- Quizzes: 15%
- Participation: 10%

1.32. 21M.380: Additional Policies

- Read entire syllabus
- Common courtesies
- Computers in class
- Academic integrity

1.33. 21M.380: Contact

- Email is always best
- Office hours

1.34. For Next Class

- Download and read entire syllabus
- Download and install Pd and Martingale; test Pd installation
- Purchase/order dual-analog controller and amplifiers
- Bring computers

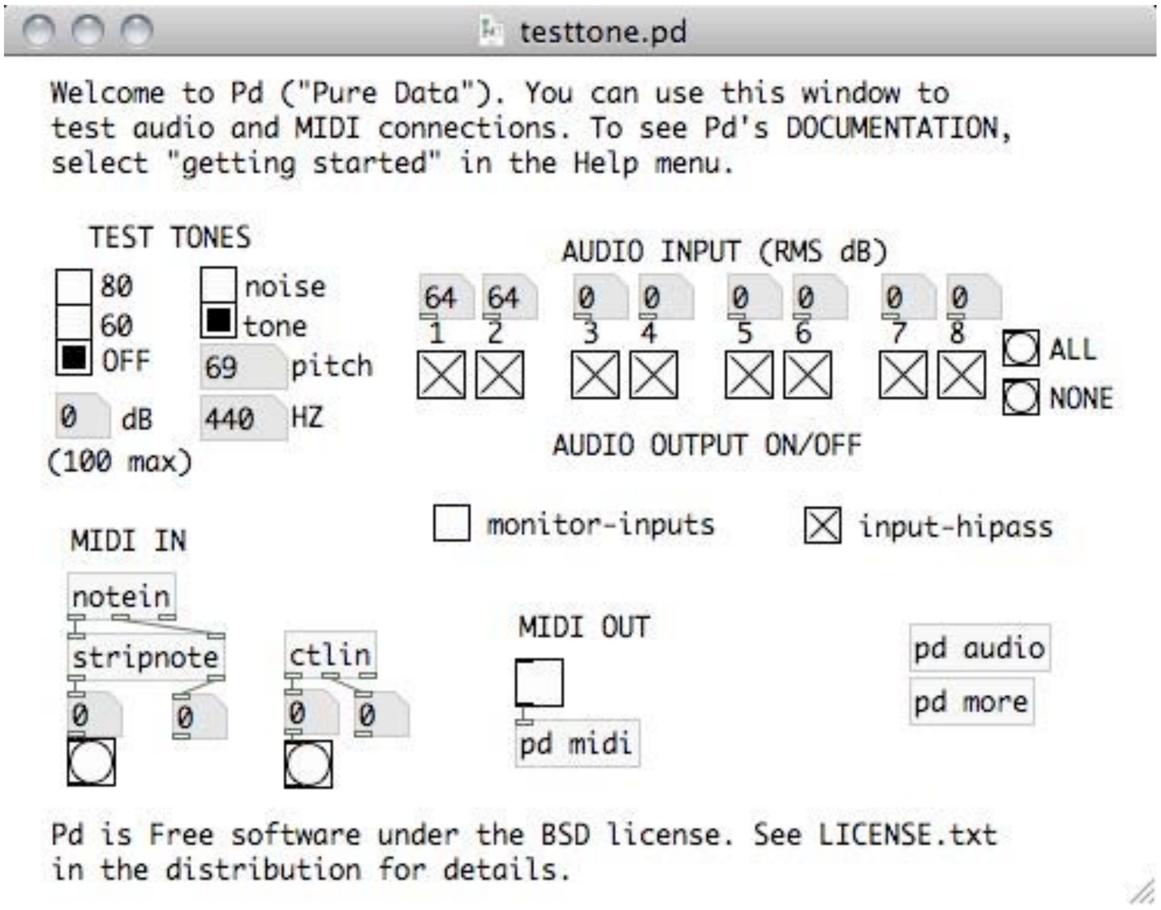
1.35. Testing Pd Installation

- Download and install Pd-extended

  http://puredata.info/downloads
- Launch the Pd application
- Should see the "Pd window"
- Under the "Media" menu, find the "Test Audio and MIDI" option

  Under "TEST TONES," select 80 (dB), listen for a tone

  If no tone is heard, make sure "compute audio" is selected



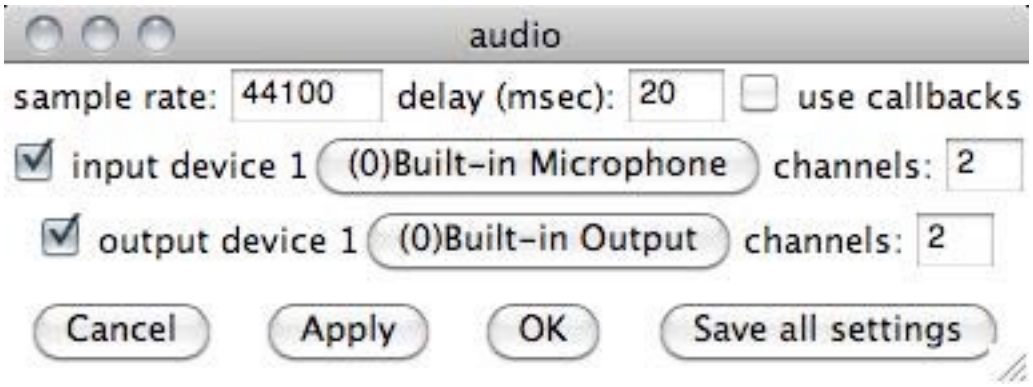
```
Welcome to Pd ("Pure Data"). You can use this window to
test audio and MIDI connections. To see Pd's DOCUMENTATION,
select "getting started" in the Help menu.
```

```
Pd is Free software under the BSD license. See LICENSE.txt
in the distribution for details.
```

- If tone is broken or stutters, configure processing delay

  Find "Audio Settings" options under Preferences

  Increase delay until test tone is continuous



sample rate: 44100 delay (msec): 20 use callbacks

[✓] input device 1 (0)Built-in Microphone channels: 2

[✓] output device 1 (0)Built-in Output channels: 2

- If there is no sound, go to Menu: Preferences > Audio Settings

Try to select a different "output device 1"

- Download Martingale

  http://code.google.com/p/martingale

- Place Martingale in a convenient locations (perhaps a directory where you store all Pd scripts)

- Accessing Martingale resources in Pd will be covered in the next meeting

1.36. Discussion Leader Assignment Schedule

Content removed for privacy reasons.

Content removed for privacy reasons.

Content removed for privacy reasons.

Content removed for privacy reasons.

Chapter 2. Meeting 2, Foundations: Sounds, Signals, Samples, and Encodings

2.1. Announcements

- Reading and Listening Discussion Leader assignments will be posted today
- Note that I will not comment directly on posted notes (but do check that they are there)

2.2. Reading: Wang, A History of Programming and Music

- Wang, G. 2007. "A History of Programming and Music." In N. Collins and J. d'Escriván, eds. *The Cambridge Companion to Electronic Music*. Cambridge: Cambridge University Press, pp. 55-71.
- What non-software programming interfaces for music does Wang describe?
- What are some of the fundmental concepts shared by many music programming languages?
- What is the trajectory of programming languages proposed?

2.3. Reading: Puckette, Max at 17

- Puckette, M. 2002. "Max at 17." *Computer Music Journal* 26(4): pp. 31-43.
- Is there one Max?
- What was the background developement of Max?
- What is max good at? What is it not good at?
- What roles do style and aesthetic play in computer music software design?

2.4. Starting Pd, The Pd Window

- The Pd Window is the destination of all error messages and message sent with [print]
- The "compute audio" toggle
  - Controls all signal processing generation
  - Can also be toggled in Media menu, with key strokes, and also with text commands (to be shown later)

2.5. Basic Components

- Patches: windows or collections of windows
- Object boxes: process or create data or signals
- Message boxes: store data
- Data
    - Can be “event” data or “signal” data
    - Passed via “patch cables” between boxes
- Comments: notes to yourself
- Interface objects: number boxes, slides, signal boxes, etc.

2.6. The Patcher Window

- A window represents a patch
- Windows can communicate between each other
- A patch can be embedded in another patch using [pd name]
- A patch can be stored as a file and loaded into another patch, called an abstraction

2.7. The Patcher Window: Edit and Run Modes

- Patch windows have two modes: edit and run
- Changing modes: Menu: Edit > Edit mode (command E)
- Edit mode: configure objects, create patches, move things around, selected objects are blue
- Run mode: objects do not move, user interface components (knobs, sliders) function
- Example: Put a Vslider; when blue, in edit mode, cannot use slider; in run mode, black, can use slider

2.8. Object Boxes

- An object is generally a computational subroutine
- An object has a class: a paradigm, an archetype

- We can make many instances of the same object, each with its own behaviour and settings
- Example: [random 20], [random 4]

2.9. Object Boxes: Creation

- Use the Put menu: Menu: Put > Object (command 1)
- An empty dotted-line box emerges: this is not an object
- An object has to have at least one creation argument to specify its type
- Additional arguments can be used to configure the object
- Example: [+], [random], [line], [select], [print], [osc~]

2.10. Objects: Types

- There are event (control rate) objects and signal objects
- Event objects process data: [line], [select]
- Signal (tilde) objects process signals: [line~], [osc~]
- There may be two versions of a type of object, one for events, one for signals: [+], [+~]

2.11. Object Inlets

- Inlets provide data or signals into objects (not both)
- White (hollow) inlets are for data, dark (filled) inlets are for signals
- Example: [+], [+~]
- For many event objets, leftmost inlet is hot: output is provided only when values are provided in this inlet
- Example: [+], [pack]

2.12. Object Outlets

- White (hollow) outlets are for data, dark (filled) inlets are for signals
- Example: [+], [+~]

- Outlets almost always provide output from right to left
- Example: [unpack f f f]

2.13. Object Interconnections

- Connections between objects can transmit either signals or event data
- Signal and event data connections are different, and cannot be interconected
- To create a connection: in Edit mode, mouse over outlet until cursor is a circle; click and hold; mouse over desired inlet until cursor is a circle; release click.
- Example: [* 4] to [+ 3], [*~ 4] to [+~ 3]

2.14. Data

- Data can be bangs, numbers, symbols, lists, or signals
- Bangs (b): a trigger, an event, a "do something now"
- Numbers (f): all numbers are floating point values
- Symbols (s): character strings (not in quotes)
- Lists (l): a space separated collection of numbers or symbols
- Signals (v): floating-point number stream at the sampling rate (when "compute audio" is on)

2.15. Data Storage

- Data can be seen (in objects, interfaces, etc) and unseen (in objects, through patch connections)
- Only data that is "seen" is saved with patch

2.16. Data Storage: Object Boxes

- Objects can have additional construction arguments
- These arguments configure how the object performs on initialization
- These arguments can sometimes be overridden by inlet values
- Example: [* 2]

2.17. Data Storage: Message Boxes

- Use the Put menu: Menu: Put > Message (command 2)
- One inlet, one outlet; note curved left side distinguishes message boxes from object boxes
- Store bangs, numbers, symbols, or lists
- Saved with patches
- Provide a user interface: can be clicked in Run mode to provide output
- Example: (bang) to [random 10] to [print]
- Example: (3) and (10) to [+] to [print]

2.18. Interface Objects: Number Boxes

- Can be used to provide numerical inputs to other objects
- Can be used to receive the numbers outputted from objects
- Can be varried as a GUI only in Run mode
- Important: holding down shift permits enter flaoting point values
- Min and max values can be set with object properties

2.19. Interface Objects: Bang

- Can click to send a bang
- When receiving a bang, darkens
- Sending a bang can be replaced by a message box with "bang" specified

2.20. Selecting, Moving, and Copying Objects

- Objects can only be moved in edit mode
- Can click and drag to create a selection area
- Objects (and interconections) can be duplicated and copied
- Copying and pasting overlays existing objects: always duplicate

2.21. Object Help, Documentation, and Tutorials

- Control click on an object and select "help" to view a help patch
- Demo patches (when available) provide examples and explanation
- The PD Glossary http://www.flexatone.net/docs/pdg
- Kreidler, J. 2009. "Programming Electronic Music in Pd." Wolke Publishing House. Available online at http://www.pd-tutorial.com.

2.22. Object Properties

- Control click on a bang interface object and select "properties" to specify visual appearance
- Colors and other attributes can be configured

2.23. Comments

- Comments are notes left to readers of the program
- Comments cannot be used as data in a patch
- Comments are critical and are essential in good code
- Use the Put menu: Menu: Put > Comment (command 5)

2.24. Saving Patches and PD files

- Always save files with a .pd extension at the end
- PD files are text files that specify the interconnections between objects

2.25. Abstractions and Martingale

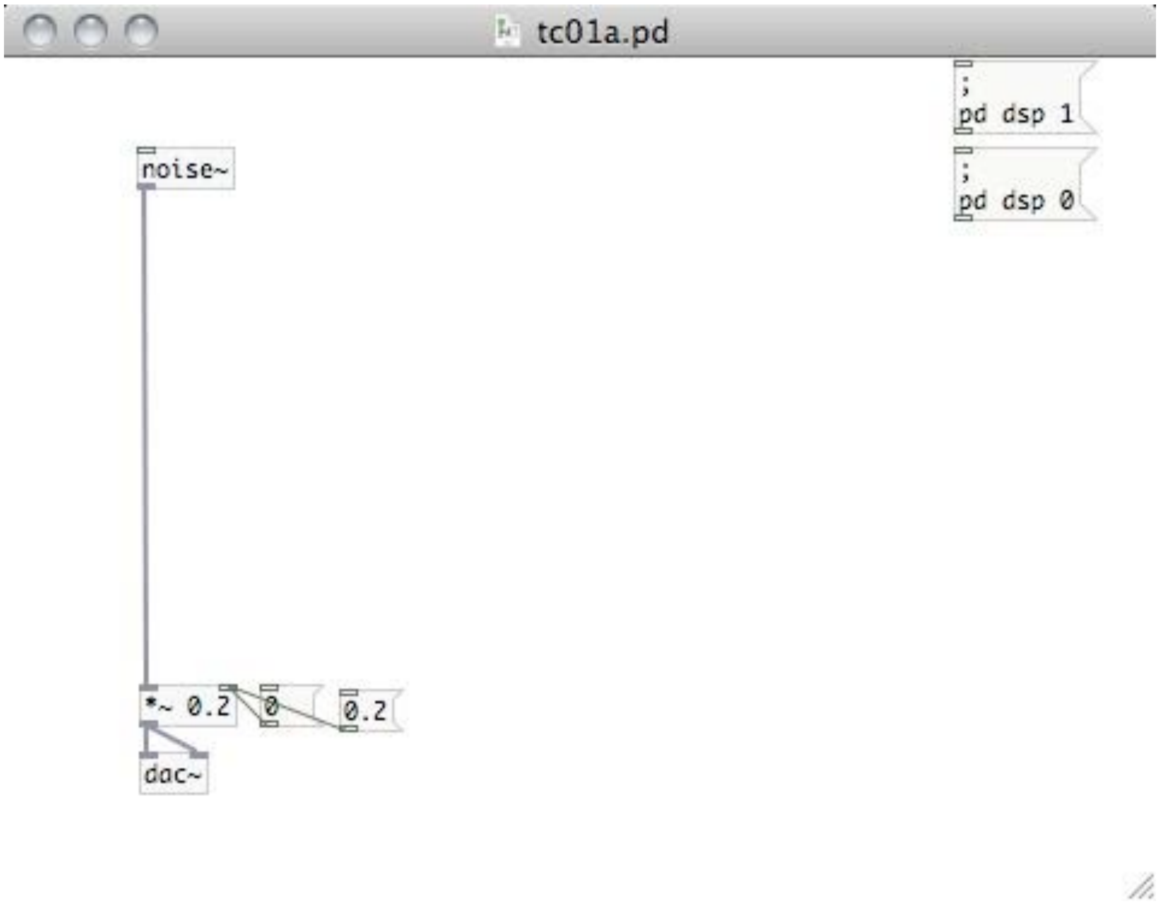
- Abstractions are PD patches that can be used in other PD patches
- Abstractions may have any number of inlets or outlets
- To load an abstraction, it must be placed in a directory that PD knows about
- Download Martingale manually: http://code.google.com/p/martingale/
- Add the "martingale/pd/lib" directory to Preferences > Path; this permits loading abstractions from the martingale library

2.26. Noise

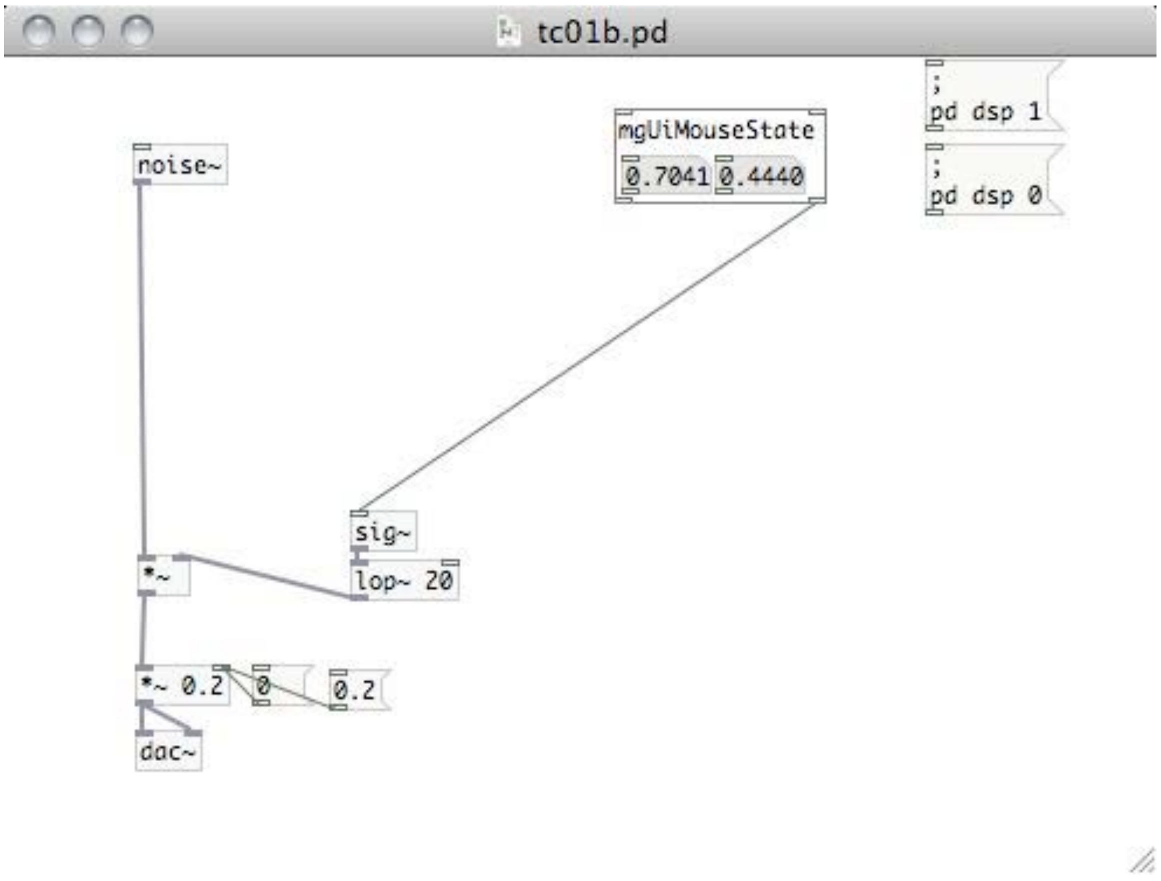
- Noise at the audio rate is random amplitudes, scaled between -1 and 1
- White noise produces equal energy across entire spectrum
- Source of rich signals and randomness
- [noise~] object provides random audio rate values between -1 and 1
- Example: martingale/demo/signalWaveforms.pd

2.27. Mouse State Noise

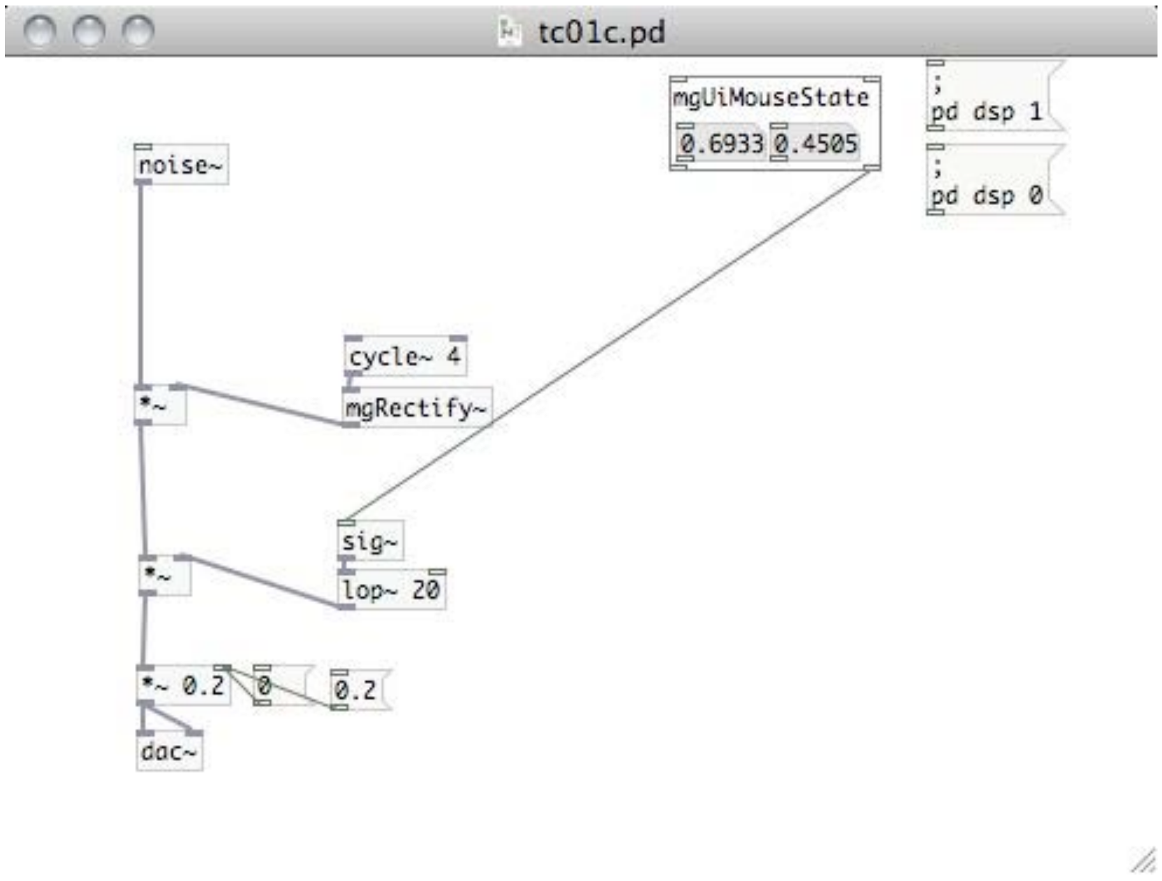
- 1. Connecting Noise to output; scaling amplitude with [*~], turning DSP on and off with message boxes



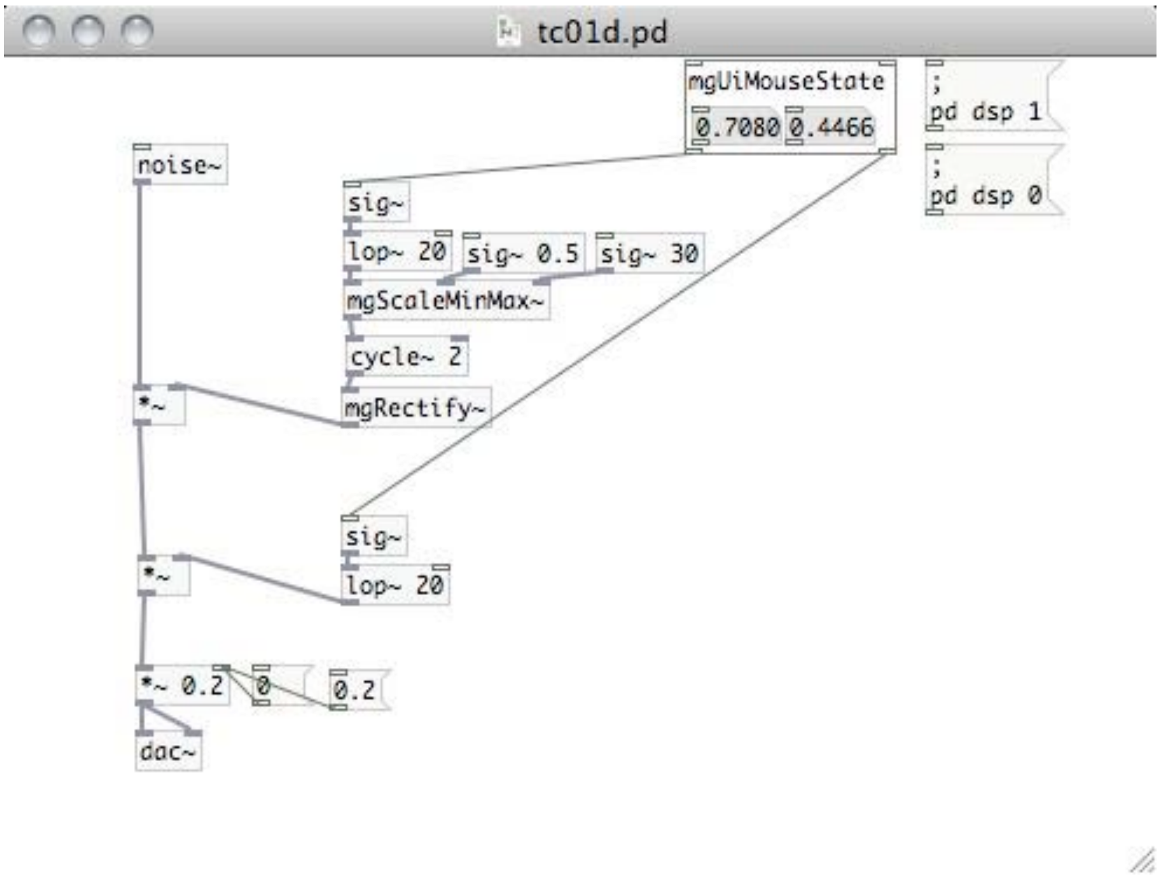
- 2. Smoothly controlling amplitude with [mgUiMouseState], [sig~], and [lop~ 20]; conversion of event data to audio rate data



- 3. Performing subsonic amplitude modulation (tremolo) with [cycle~] and [mgRectify~]



- 4. Scaling unit interval values with [mgScaleMinMax~]

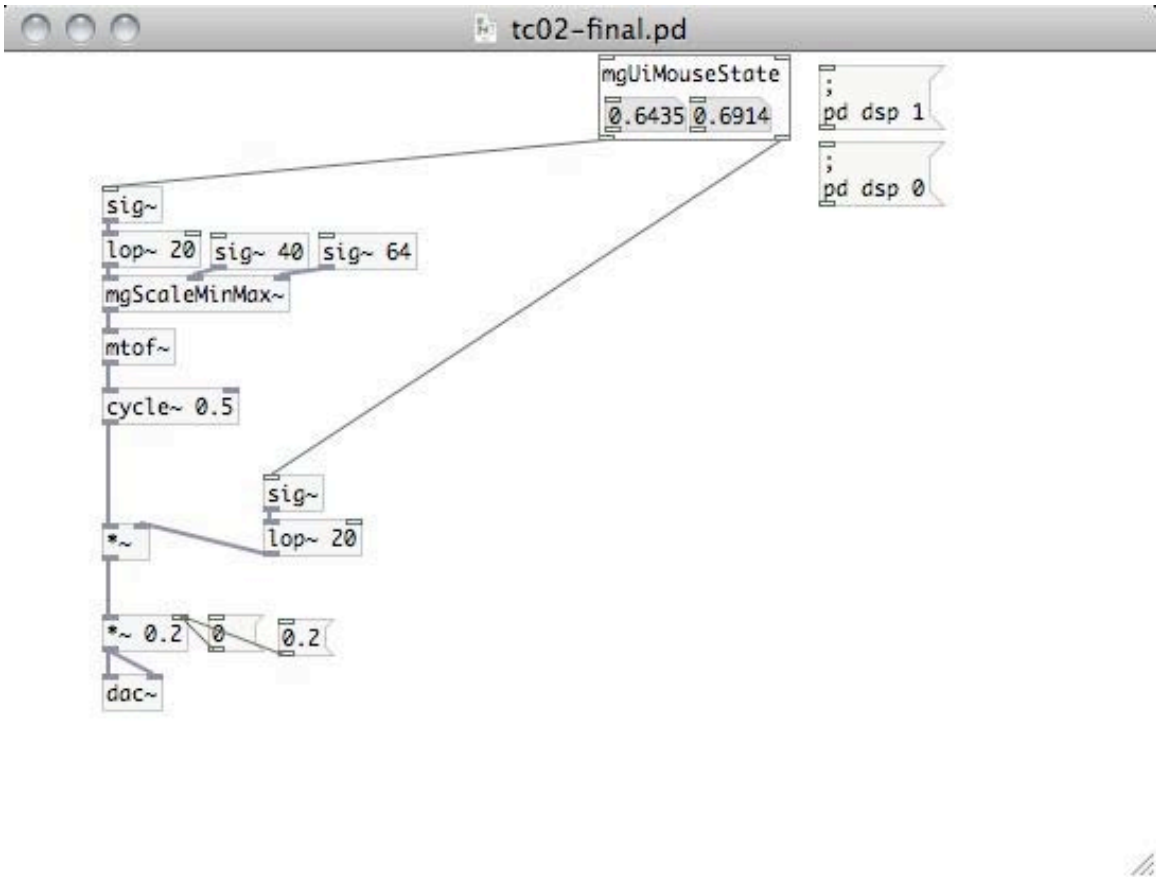


2.28. Sines

- Sine waves provide a perfect circular motion over time
- Produces single, perfect frequency with no overtones
- Example: martingale/demo/signalWaveforms.pd
- Audible range from 20 to 20,000 Hertz
- Example: martingale/demo/earLimits.pd
- Frequency is logarithmically related to pitch; equal pitch values are octaves, or a 2:1 frequency ratio
- Example: martingale/demo/earLogFrequency.pd
- MIDI pitch values are an integer to half-step mapping; can convert from MIDI to frequency with [mtof] and [ftom]

2.29. Mouse State Sines

- A mouse theremin: y axis controls amplitude, x axis control pitch (scaled between MIDI values 40 and 64 and converted with [mtof~])



2.30. Harmonic Waveforms and Wavetables

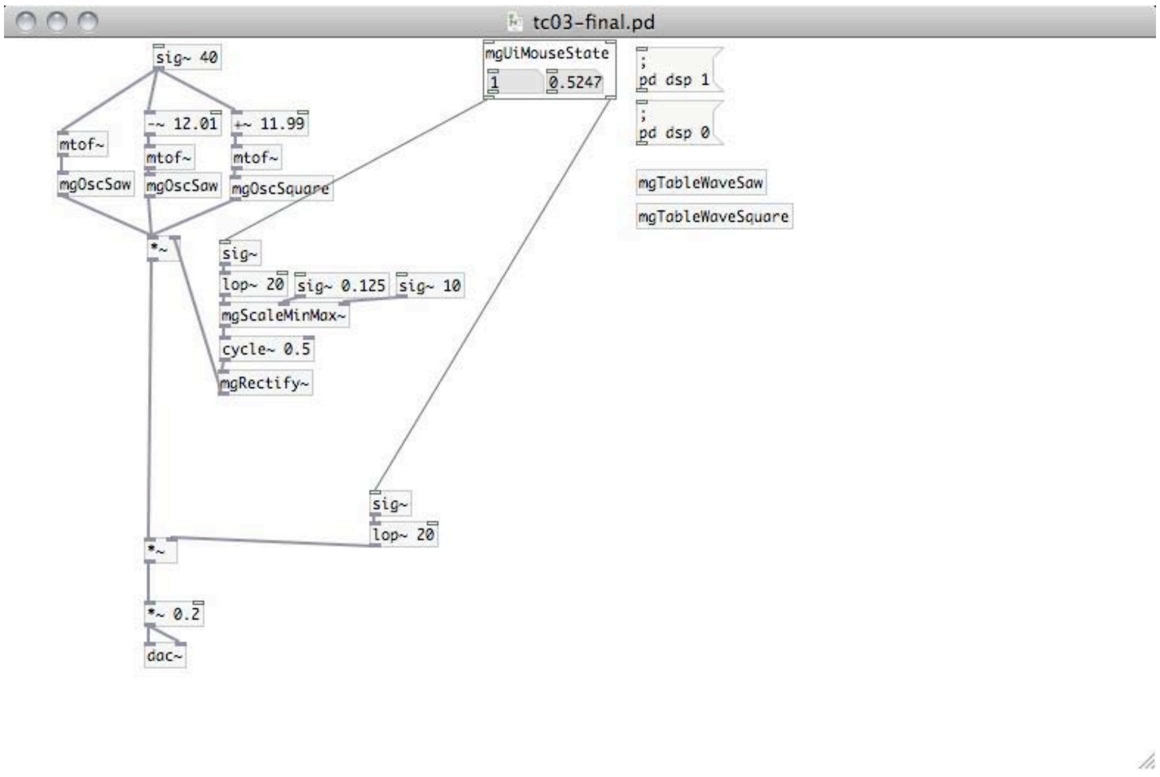
- Anything other than a sine tone has a rich (or richer) spectrum
- Many naturally resonating bodies produce secondary vibrations at whole-number multiples of the base frequency
- Common waveforms represent common arrangements of overtones produced by summing harmonic overtones: triangle, square, and sawtooth
- Example: martingale/demo/sumOfSines.pd

  Example: martingale/demo/signalWaveforms.pd
- A wavetable is an array that stores wave patterns (or other data) and reads them back at variable rates

- Arrays store (generally large) lists of values indexed from zero
- Each array in Pd must have a unique name; names can be provided as arguments

2.31. Mouse State Harmonic Drones

- Mixture of detuned saw and square waves; y axis controls amplitude, x axis controls tremolo



2.32. Listening: Schaeffer

- Listening: Pierre Schaeffer, *Cinq Etudes De Bruits: Etude Violette*, 1948

- Pierre Schaeffer, *Etude aux objets, 2. Objets étendus* , 1959

2.33. Listening: Cage and Oswald

- Listening: John Cage, *Williams Mix*, 1952

-

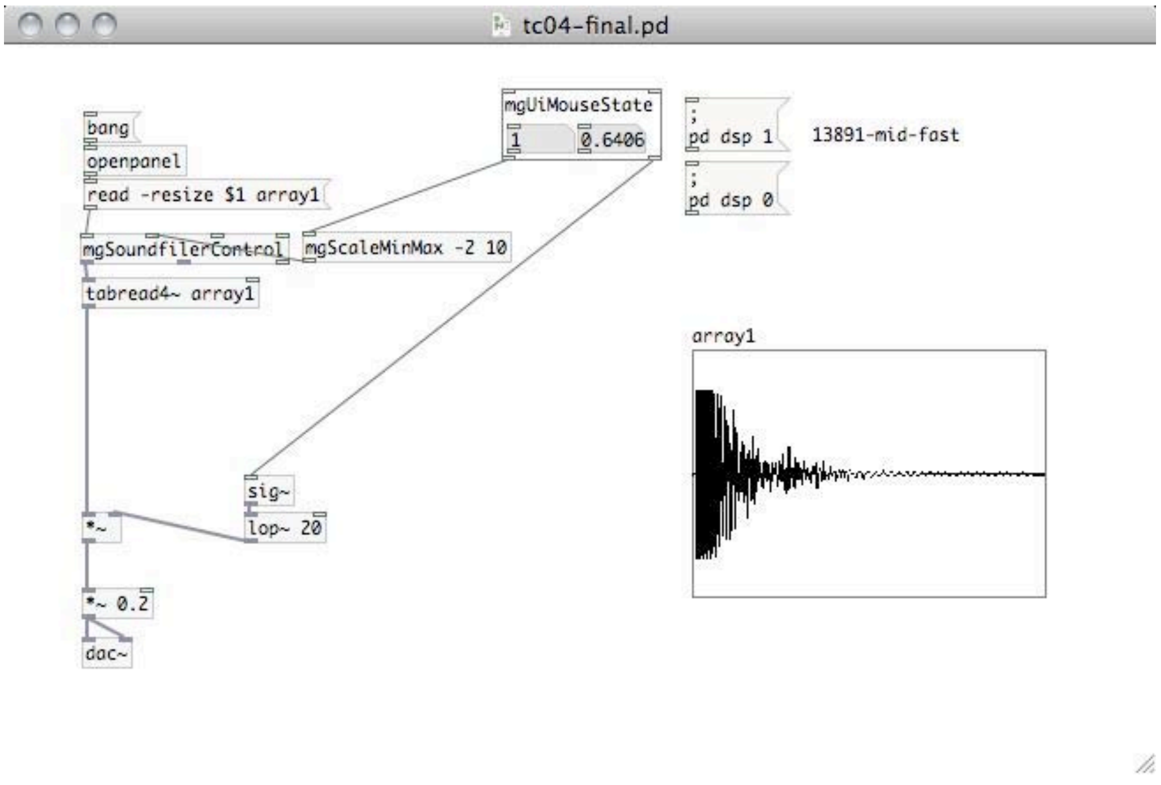
- John Oswald, "Dab," *Plunderphonics 69/96*, 1989

2.34. Stored Digital Audio

- Audio file data can be loaded into arrays and treated as a wavetable
- Audio files may have different bit depths and sampling rates; when loaded in Pd amplitudes range from -1 to 1

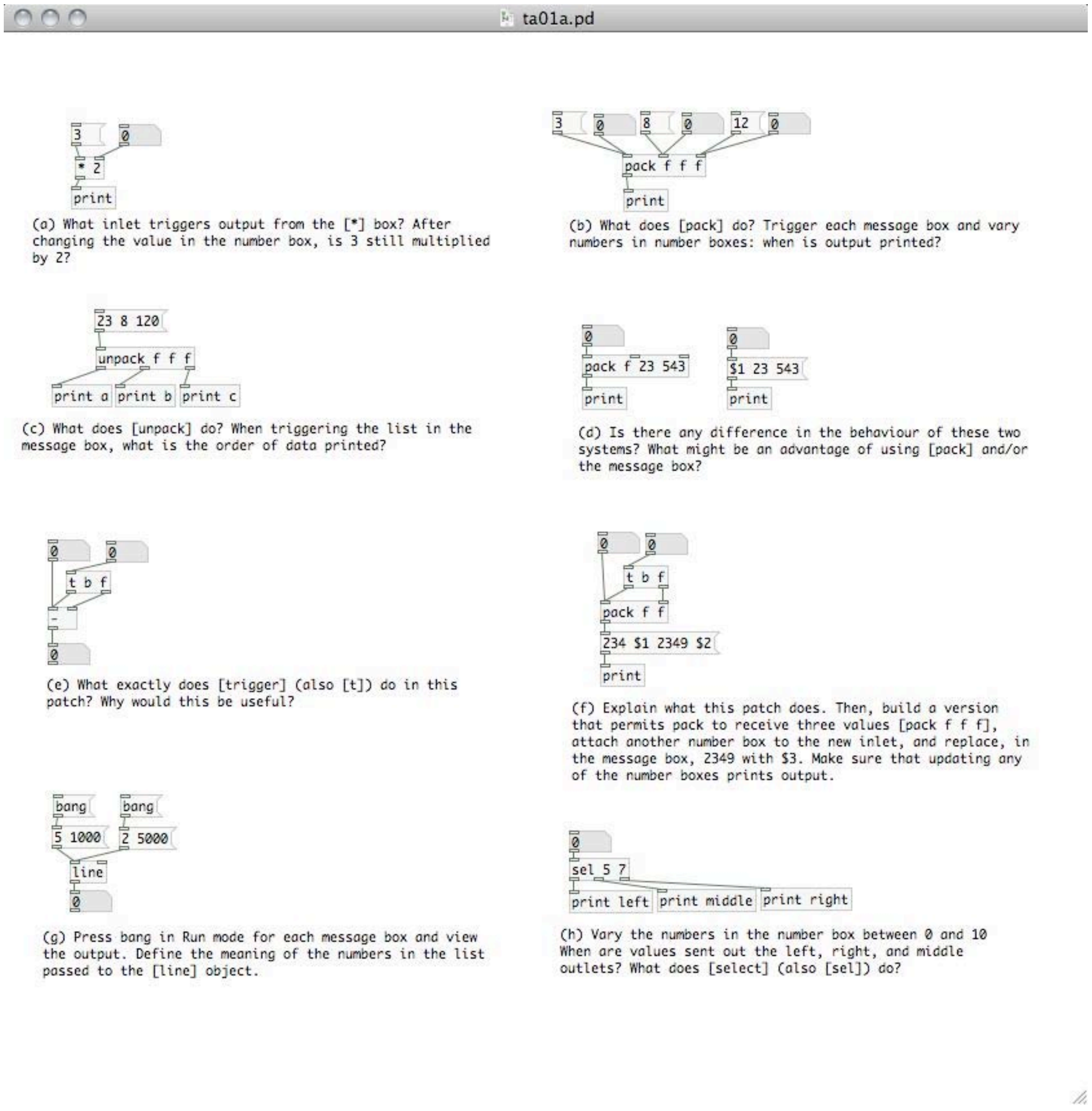
2.35. Mouse State Audio File Looper

- Variable rate audio file looper: y axis controls amplitude, x axis control rate of playback from -2 to 10



2.36. Pd Tutorial 1

1. The following examples demonstrate operations with Pd. Recreate the following patch components in a Pd file and and answer the provided questions as comments in the Pd file.



(a) What inlet triggers output from the [*] box? After changing the value in the number box, is 3 still multiplied by 2?

(b) What does [pack] do? Trigger each message box and vary numbers in number boxes: when is output printed?

(c) What does [unpack] do? When triggering the list in the message box, what is the order of data printed?

(d) Is there any difference in the behaviour of these two systems? What might be an advantage of using [pack] and/or the message box?

(e) What exactly does [trigger] (also [t]) do in this patch? Why would this be useful?

(f) Explain what this patch does. Then, build a version that permits pack to receive three values [pack f f f], attach another number box to the new inlet, and replace, in the message box, 2349 with $3. Make sure that updating any of the number boxes prints output.

(g) Press bang in Run mode for each message box and view the output. Define the meaning of the numbers in the list passed to the [line] object.

(h) Vary the numbers in the number box between 0 and 10 When are values sent out the left, right, and middle outlets? What does [select] (also [sel]) do?

2. In a Pd file, re-create one of the demonstrated Mouse State instruments shown above. Extend the instrument in some way: alter fixed parameters, apply alternate mappings of mouse values, combine different sound sources.

Chapter 3. Meeting 3, Foundations: Envelopes, Filters, Modulation, and Mixing

3.1. Announcements

- Bring controllers (not amps) to next class on Monday; first class with amps and controllers will be meeting 5, Wednesday, 16 February

3.2. Review Waveshapes and Wavetables

- Noise, sine, waveshapes, and audio files
- Using [phasor~] to read a table

3.3. Review Pd Tutorial 1

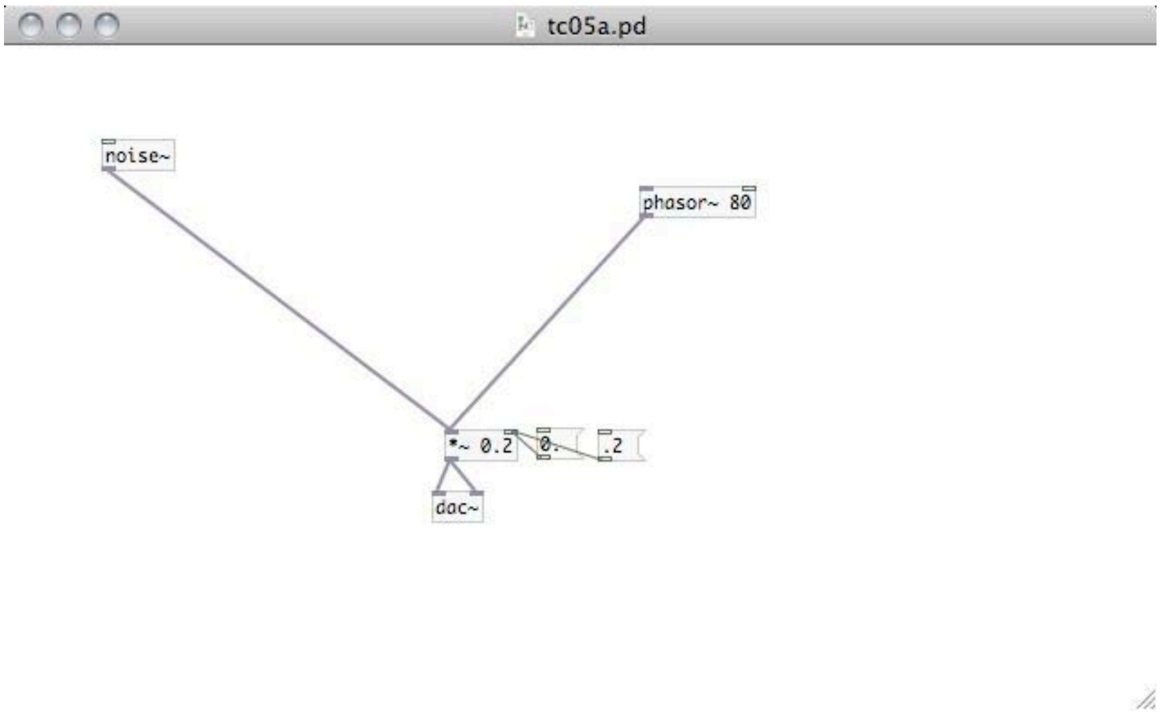
- ?

3.4. Reading: Tanaka: Sensor-Based Musical Instruments and Interactice Music

- Tanaka, A. 2009. "Sensor-Based Musical Instruments and Interactive Music." In R. T. Dean, ed. *The Oxford Handbook of Computer Music*. Oxford University Press, pp. 233-257. (233-243)
- Why might it be problematic to define an instrument?
- Why are instruments not like normal tools?
- Why might the notion of idiomatic performance technique be important when working with new musical interfaces?

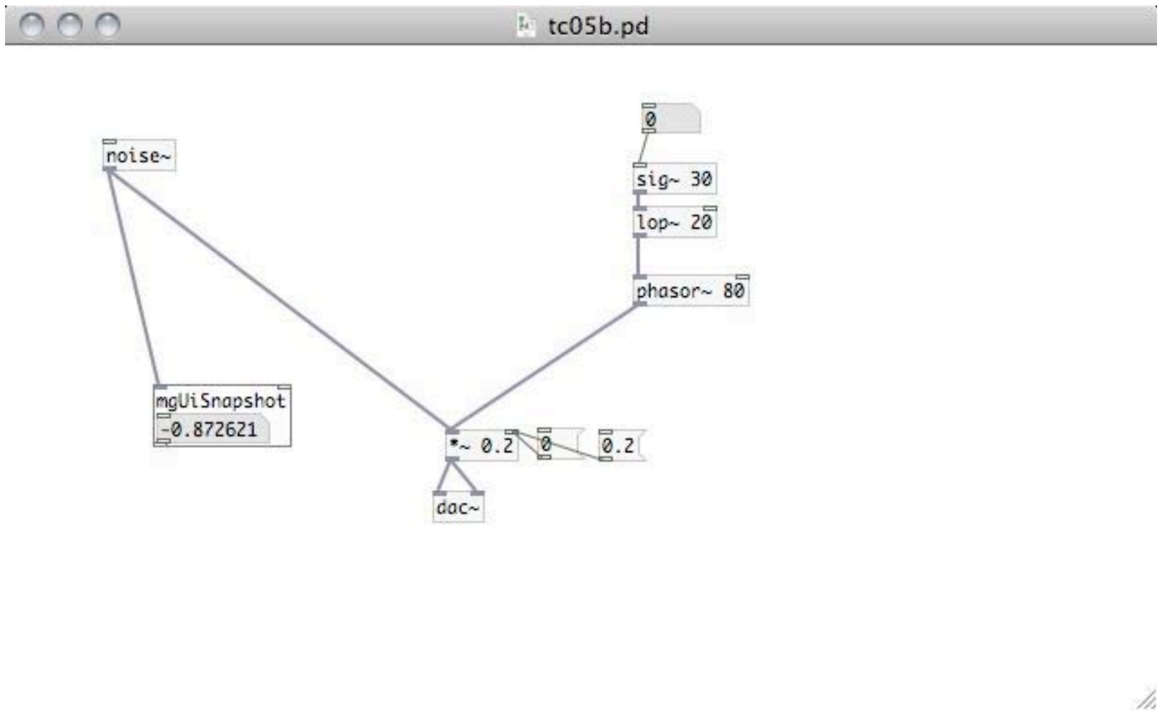
3.5. Adding and Multiplying Signals

- Adding signals [+~] is mixing
- Two signals provided into the same inlet automatically sum
- Multiplying signals [*~] is scaling
- Example: mixing [noise~] and [phasor~]



3.6. Converting Signals to Data and Data to Signals

- When converting from data to a signal, always use [sig~] to [lop~ 20]
- When converting from signal to data, use [snapshot~], [mgSnapshot], or [mgUiSnapshot]
- Example: getting values from [noise~] with [mgUiSnapshot], controlling [phasor~] frequency with [sig~] and [lop~ 20]



3.7. Envelopes

- Envelopes are signal scalars
- Generally unipolar signals that move from 0 to 1 and back to 0
- Can be used as amplitude scalars, or to provide dynamic control values to other parameters
- A number of common shapes define common types of articulations
- Can be classified as mono-triggered or bi-triggered
- Some musical events can be thought of as having a single on trigger: off is implied after a certain duration: other musical events can be thought of as having two triggers: off is trigered after an unknown duration

3.8. Envelopes: Common Shapes

- AR: Attack / Release

  Example parameters: 20 2000

  Examples: [mgEnvlBtAr], [mgEnvlMtAr]
- ADSR: Attack / Decay / Sustain / Release

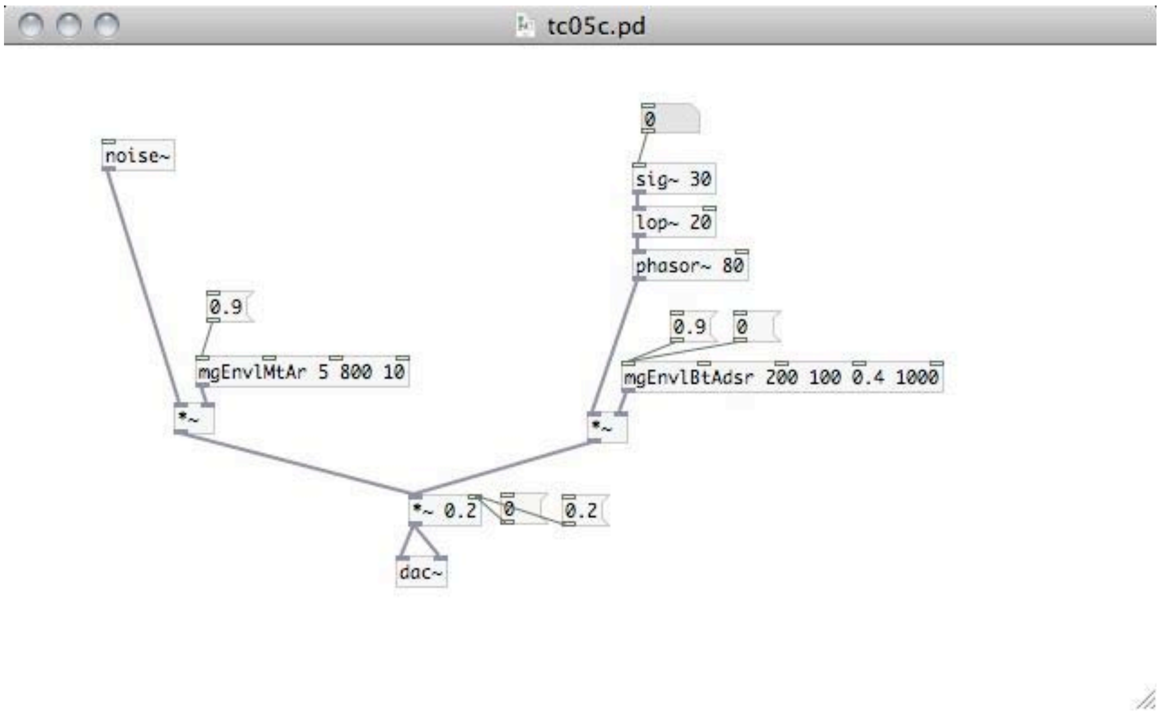
Example parameters: 200 80 .6 1200

Examples: [mgEnvlBtAdsr], [mgEnvlMtAdsr]

- Envelopes might be mixed to produce more interesting shapes
- Example: martingale/demo/envelopes.pd

3.9. Envelopes: Triggering

- Trigger start envelope
- Martingale envelopes use numerical values: non-zero sets peak amp and triggers envelopes; 0 stops envelope
- A mono trigger starts and stops with a single trigger
- A bi-trigger envelope needs a trigger to start and a trigger to end
- Example: martingale/demo/envelopes.pd
- Example: enveloping two signals



3.10. Filtering

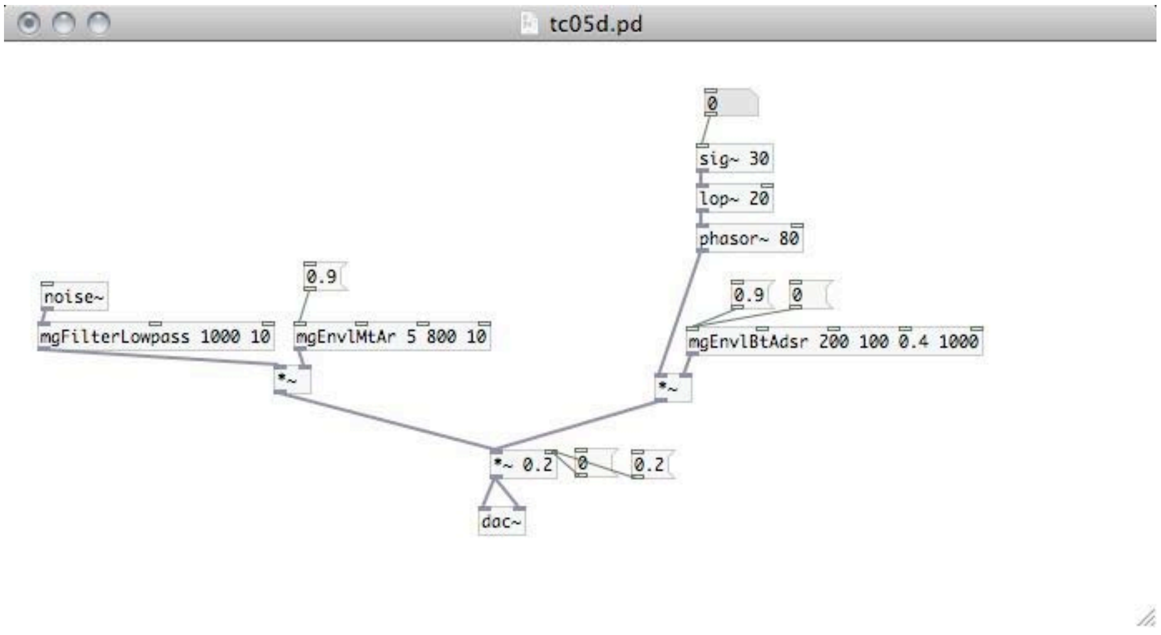
- Filters selectively boost or cut the amplitude of frequency components
- Equalizers are filters (a distinction is not useful)
- Filters cannot add frequencies that are not present in the source
- Filter shapes are depicted with frequency-domain graphs with a 0-centered amplitude change
- Filters can be used on audio signals or on control signals
- Filters can be chained in series or used in parallel

3.11. Filtering: One Parameter

- Cutoff frequency
- High pass filters (HPF), low pass filters (LPF)
- Examples: [lop~] and [hip~]
- Using an [lop~] on a control signal filters out quick changes, smoothing the signal
- Example: martingale/demo/filters.pd
- Note: may need to convert signals to event data with [mgSnapshot] to control parameters

3.12. Filtering: Two Parameter

- LPF and HPF filters with cutoff frequency and resonance
- Examples: [moog~], [mgFilterLowPass], [mgFilterHighPass]
- Examples: adding a [mgFilterLowPass]



- Bandpass and notch filters: center frequency and gain
- Examples: [mgFilterBandPass], [mgFilterNotch]
- Example: martingale/demo/filters.pd

3.13. Filtering: Three Parameters

- Parametric filter: cutoff frequency, gain, and bandwidth
- Examples: [mgFilterParametric]
- High and low shelf filters: cutoff frequency, gain, and shape
- Examples: [mgFilterLowShelf], [mgFilterHighShelf]

3.14. Modulation

- Modulation is varying a parameter over time
- Modulation can be executed with an envelope (for intermittent modulation) or a low-frequency oscillator (LFO; for continuous modulation)

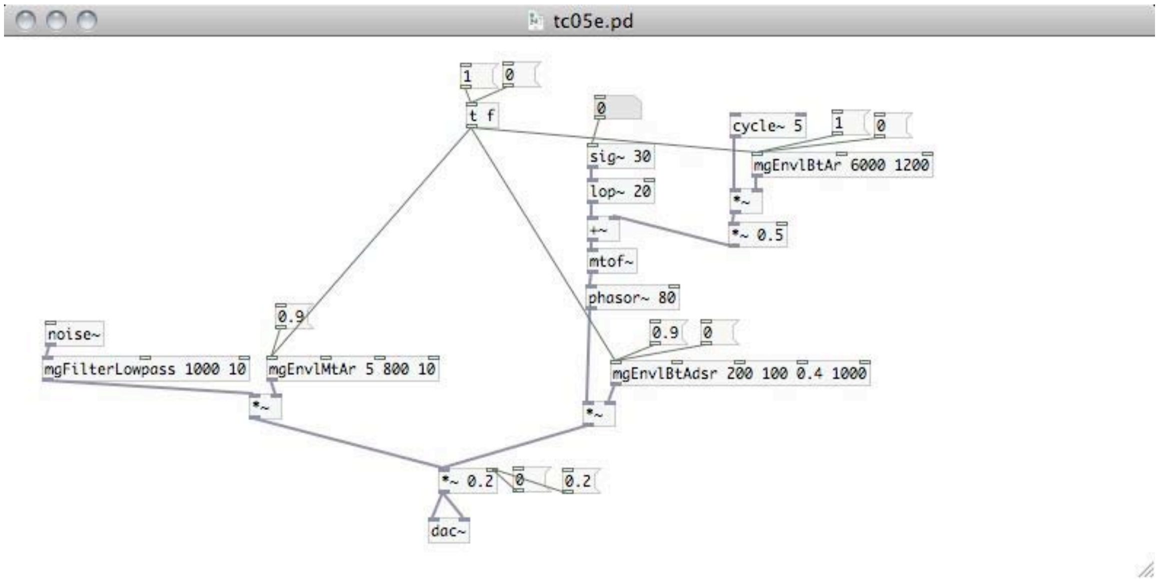
3.15. Modulation: Amplitude

- Sub-sonic AM is tremolo

- Sonic AM produces new overtones (sidebands, or rings)
- Example: martingale/demo/modulationAmBasic.pd

3.16. Modulation: Frequency

- Sub-sonic (less than 30 Hz) FM is vibrato
- Sonic FM produces new overtones
- Sonic FM produces new overtones
- Example: martingale/demo/modulationFmBasic.pd
- Example: creating a vibrato scaled by an envelope



3.17. Listening: Tenney

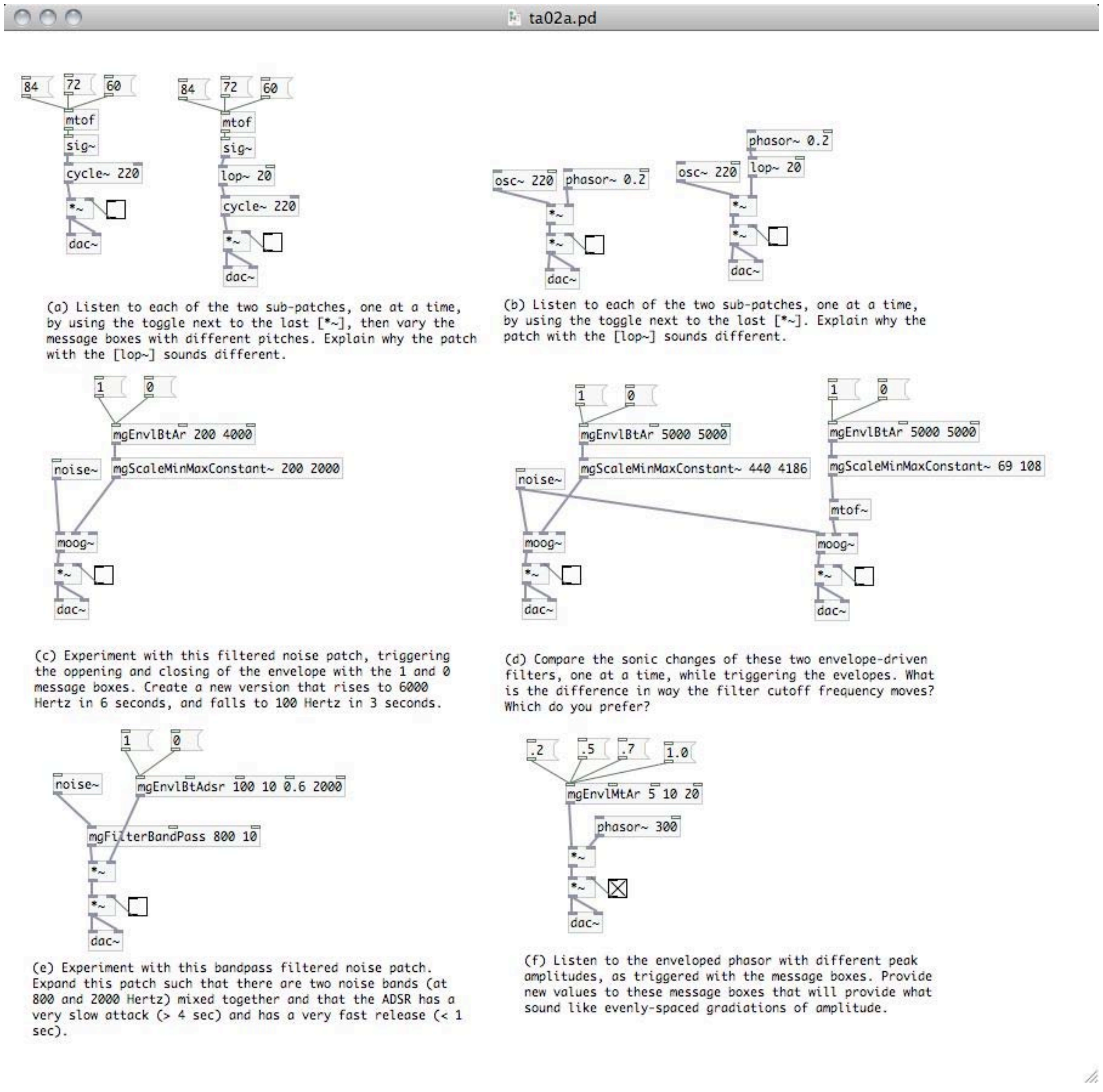
- Mathews: "to my mind, the most interesting music he did at the Laboratories involved the use of random noises of various sorts." (1980, p. 17)
- Listening: James Tenney, *Analog #1: Noise Study*, 1961

3.18. Listening: Koenig

- Listening: Gottfried Michele Koenig, *Funktion Grau*, 1969

3.19. Pd Tutorial 2

1. The following examples demonstrate operations with Pd. Recreate the following patch components in a Pd file and and answer the provided questions as comments in the Pd file.



(a) Listen to each of the two sub-patches, one at a time, by using the toggle next to the last [*~], then vary the message boxes with different pitches. Explain why the patch with the [lop~] sounds different.

(b) Listen to each of the two sub-patches, one at a time, by using the toggle next to the last [*~]. Explain why the patch with the [lop~] sounds different.

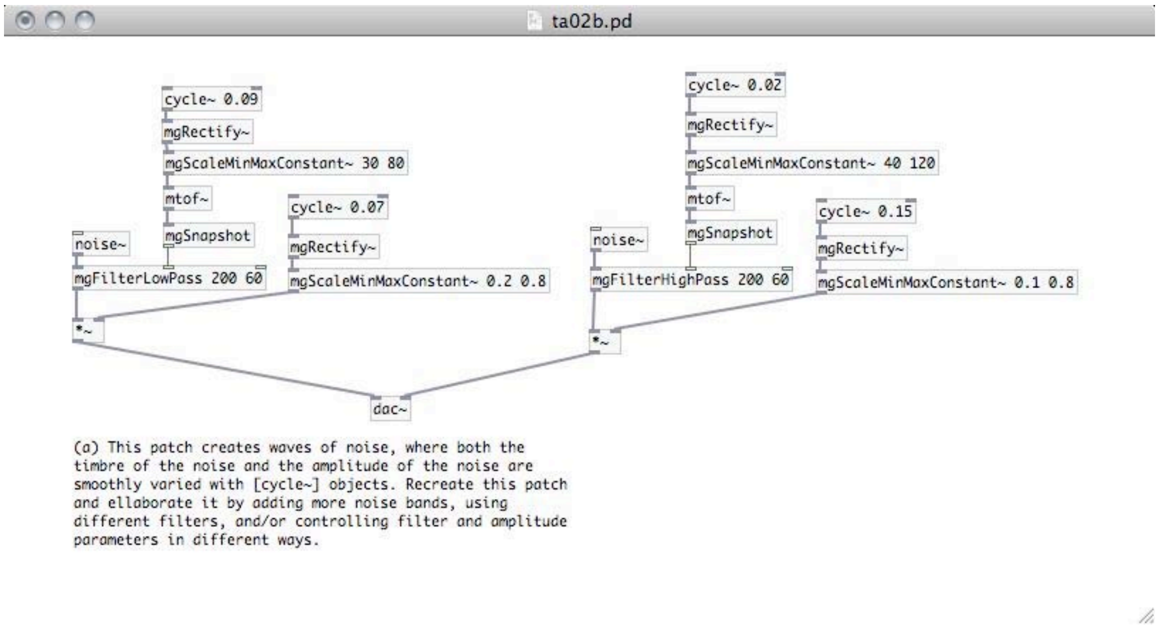
(c) Experiment with this filtered noise patch, triggering the oppening and closing of the envelope with the 1 and 0 message boxes. Create a new version that rises to 6000 Hertz in 6 seconds, and falls to 100 Hertz in 3 seconds.

(d) Compare the sonic changes of these two envelope-driven filters, one at a time, while triggering the evelopes. What is the difference in way the filter cutoff frequency moves? Which do you prefer?

(e) Experiment with this bandpass filtered noise patch. Expand this patch such that there are two noise bands (at 800 and 2000 Hertz) mixed together and that the ADSR has a very slow attack (> 4 sec) and has a very fast release (< 1 sec).

(f) Listen to the enveloped phasor with different peak amplitudes, as triggered with the message boxes. Provide new values to these message boxes that will provide what sound like evenly-spaced gradiations of amplitude.

2. Create and extend the following patch.



(a) This patch creates waves of noise, where both the timbre of the noise and the amplitude of the noise are smoothly varied with [cycle~] objects. Recreate this patch and ellaborate it by adding more noise bands, using different filters, and/or controlling filter and amplitude parameters in different ways.

Chapter 4. Meeting 4, Foundations: Managing Events, Signals, and Data

4.1. Announcements

- Bring controllers and amps to next class on Wednesday

4.2. Review Pd Tutorial 2

- ?

4.3. Reading: Ryan: Some Remarks on Musical Instrument Design at STEIM

- Ryan, J. 1991. “Some Remarks on Musical Instrument Design at STEIM.” *Contemporary Music Review* 6(1): pp. 3-17.
- Why is a desire for immediacy in computer music seen as possibly historically surprising or ironic?
- Why does Ryan reject the word \`interface\` as sufficient to describe musical controllers?
- Why does Ryan suggest that it might be interesting to make musical control as difficult as possible?
- What generalizations can be made about the approach and type of work done at STEIM?
- Whats wrong with general-purpose solutions?

4.4. Overview

- Hardware inputs
- Triggers
- Lists

4.5. Hardware Inputs

- Hardware inputs are usually serial data from USB or a network communication
- There may be platform-specific differences may emerge

4.6. The Dual Analog Interface

- Two joysticks, 8 main buttons + 2 additional buttons (each with 2 values), and dpad (as a button with 5 possible values)
- Plug in USB device, start Pd, and open martingale/pd/lib/mgHwDualAnalog.test.pd
- Find USB device number, and look for output

4.7. The Dual Analog Interface: Windows and Virtual

- Some Windows platforms may need to use a different interface
- Add the martingale/pd/externals/win directory to your Preferences > Path
- martingale/pd/lib/mgHwJoystickDualAnalog.test.pd
- If no other option presently, can use a keyboard based mapping
- martingale/pd/lib/mgHwDualAnalogVirtual.test.pd

4.8. A Hierarchy of Components

- A voice is single type of sound source, and elementary component (will be named *_v.pd)
- A synth is an instrument that takes real-time parameters and loads preset parameters stored in banks
- A performance provides mappings from a controller to one or more synths, and might control large-scale parameter management

4.9. Testing a Synth Voice

- Add the martingale/audio directory to your Preferences > Path
- open martingale/pd/lib/mgSynthBuffer_v.test.pd

4.10. Testing a Performance

- martingale/pd/instruments/dualAnalogPerfA.test.pd
- martingale/pd/instruments/dualAnalogVirtualPerfA.test.pd

4.11. Math and Expressions

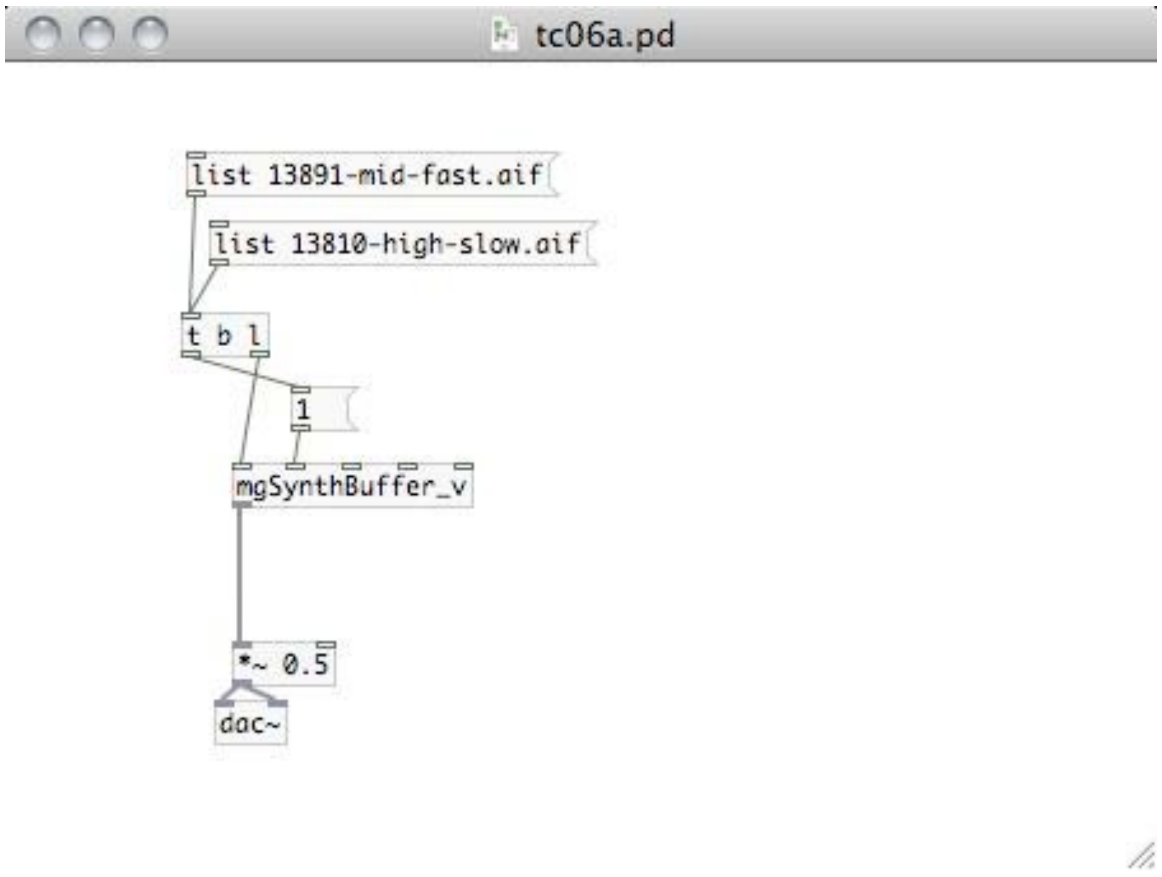
- A number of math objects for data and signals: [+], [-]
- Always favor multiplication over division: [* .01] is better than [/ 100]
- For multiple multiple computations, the [expr] object is conventient
- Operator precedence is multiplication or division, then addition or substraction; Values can be taken to exponents with [expr pow(3, 2)] syntax
- One or more value can be provided into [expr] with numbered numeric variables: $f1, $f2, etc.
- Left-most inlet is always hot: a bang or number is required for output
- The [expr~] object works with signal inlets represented by $v1

4.12. Math: Data Conversions

- MIDI Pitch to frequency: [mtof], [ftom], [mtof~], [ftom~]
- BPM to msec: [mgBpmToMs], [mgMsToBpm]
- Msec to frequency: [mgMsToFq], [mgFqToMs]

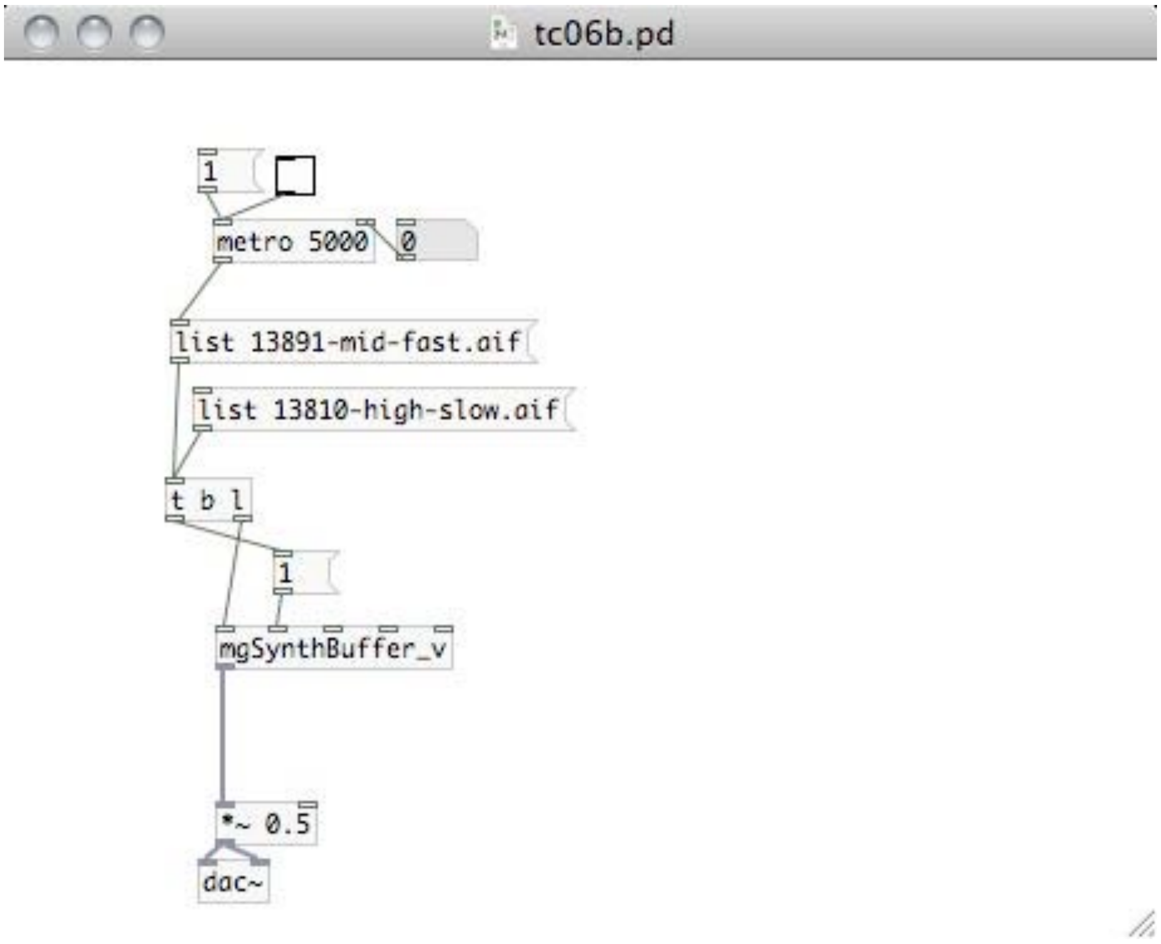
4.13. The Trigger Object

- A [trigger] object can be used to send multiple bangs or messages as fast as possible in a defined order
- A [trigger] can be seen as a way to replicate message and/or bangs in a specific order
- Whenever going from one outlet to many inlets you must use a trigger
- Can use a float, list, or symbol to send a sequence of bangs or copies of the initial data
- Example: send a file path (as an explicit list, not a symbol) paired with a bang to trigger playback

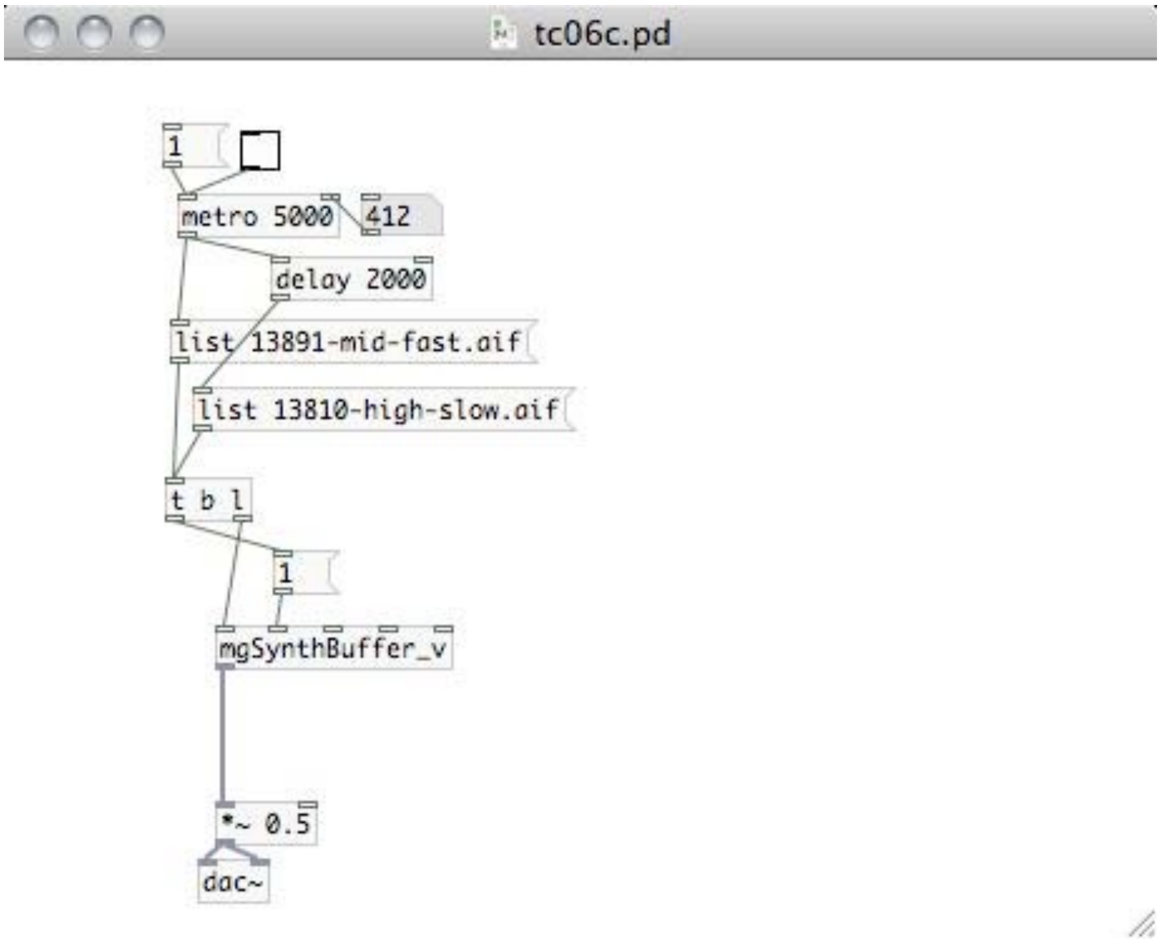


4.14. A Source of Triggers: [metro]

- For performance interfaces, triggers will often come from a controller
- Automated triggers can come from a [metro]: a timed sequence of bangs
- On/off is specified in left inlet with 1/0
- Speed in milliseconds is specified either as a construction argument or through the right inlet; can convert from BPM to msec with [mgBpmToMs]
- Speed can be varied dynamically
- Example: using [metro] to repeatedly trigger an audio file

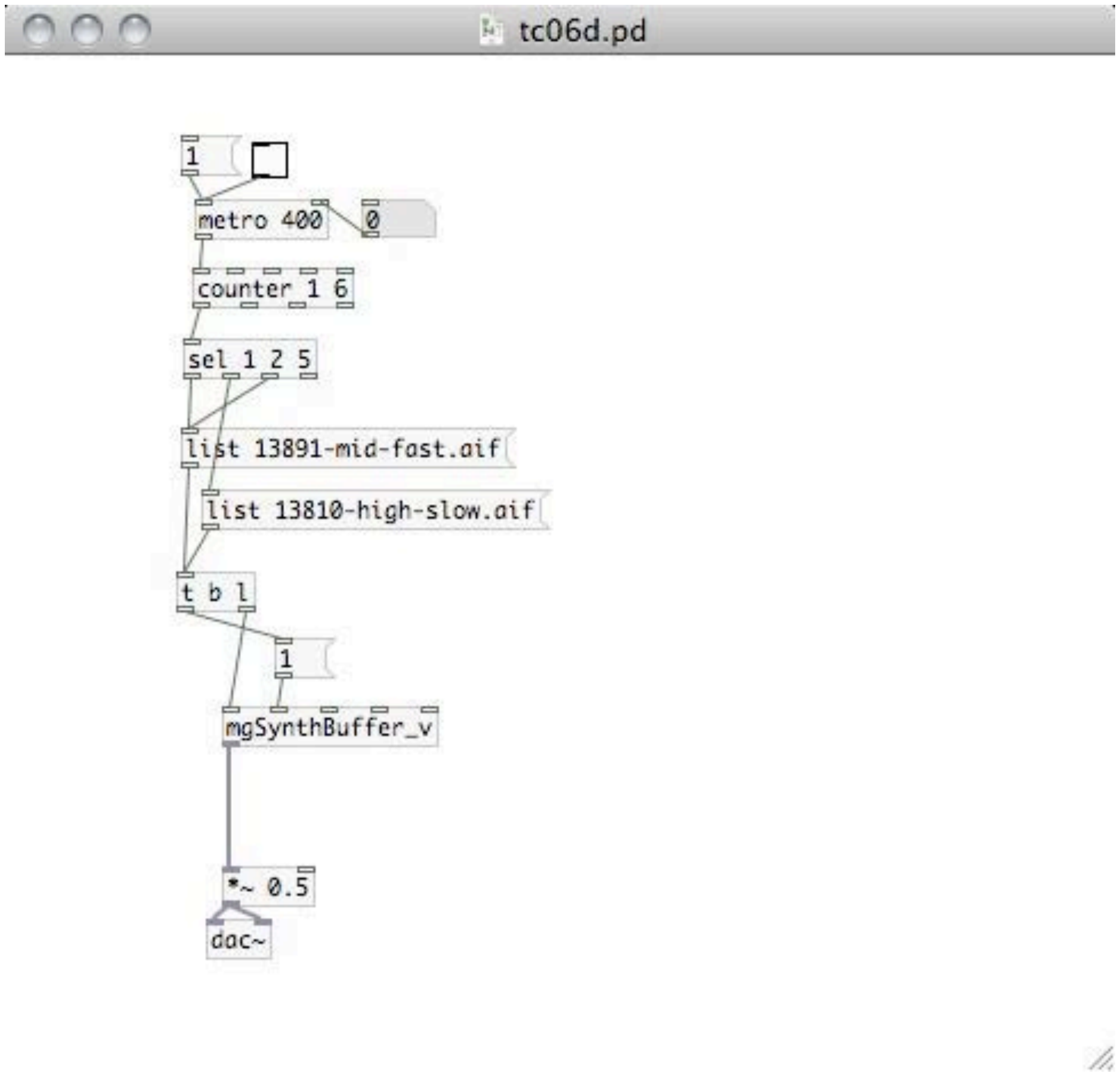


- Bangs can be delayed with [delay]; messages can be delayed withÊ[pipe]
- Example: using [delay] to trigger a second sample after the first



4.15. Counting Events

- [counter] provides a convenient way to count bangs (or other events)
- Start and stop values can be provided as creation arguments: [counter min max]
- Counter can be forced to jump to new values or reset on next bang; counter can go up, down, or back and forth
- Combining [counter] with [sel] is a powerful way to articulate multiples of a base duration
- Example: using [counter] and [sel] to create rhythmic articulations



4.16. A Signal Metronome: [phasor~]

- [metro] produces bangs separted by milliseconds
- [phasor~] ramps from 0 to 1 in Hertz (cycles per second)
- Can use this ramp for reading through a range, or for detecting periodic points in the cycle
- Can convert from ms to frequency with [mgMsToFq]

4.17. Lists and Arrays

- A collection of similar data in a single row accessible by an index

- Lists count from 1

  Lists can be floats or symbols

- Arrays count from 0

  Arrays have to be floats

4.18. Lists

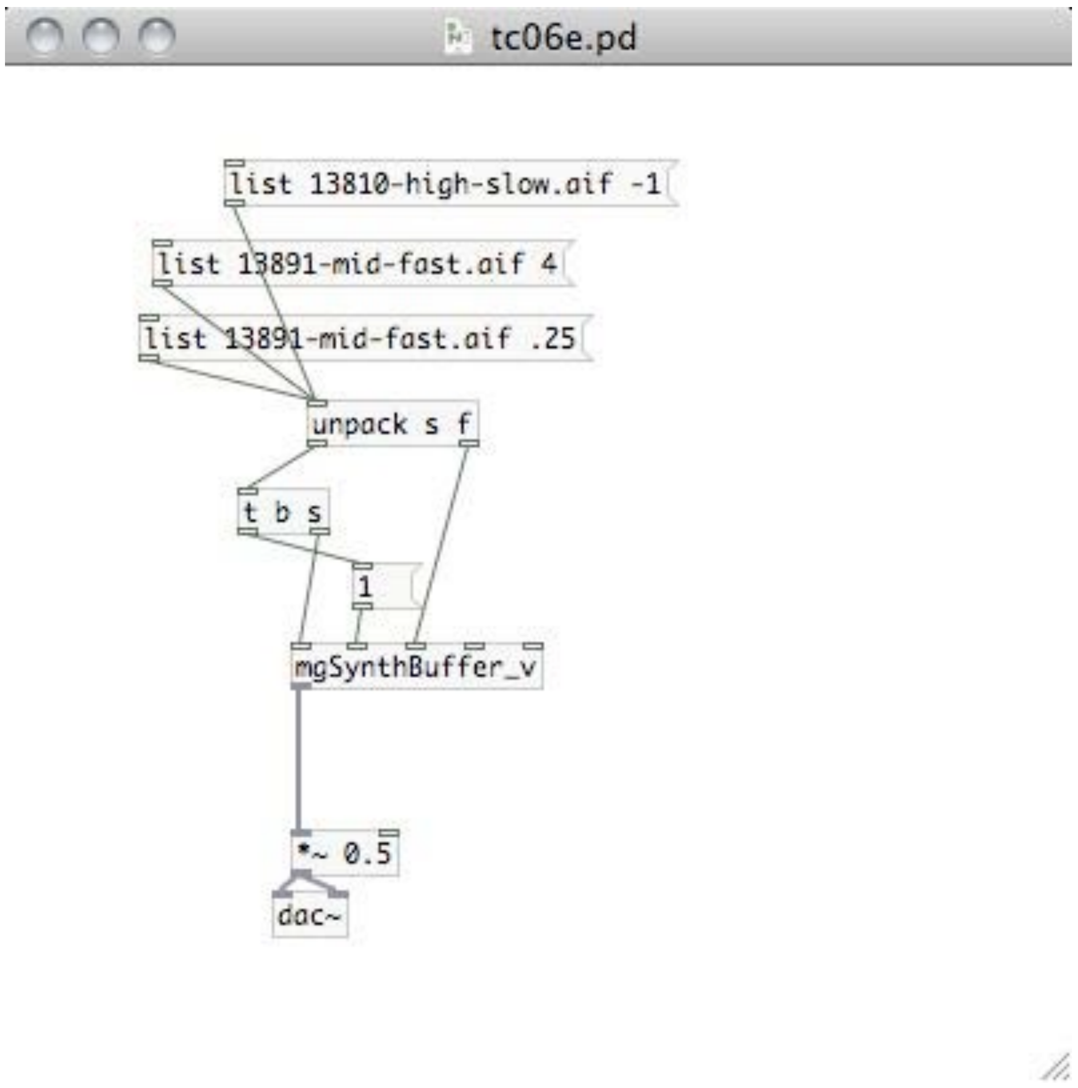
- Can be all numbers in a message box
- Can be all symbols in a message box preceded by "list"
- [zl] tools (there are many), [pack], [unpack] are main processors
- Length: [zl len]

4.19. Storing Data for Later Use

- [f], [zl reg], and [symbol] allow us to store data provided through a right inlet and bang it out later with the left inlet
- Very useful for when we need to hold a value, process those values, and then do something with the original values

4.20. Breaking Lists

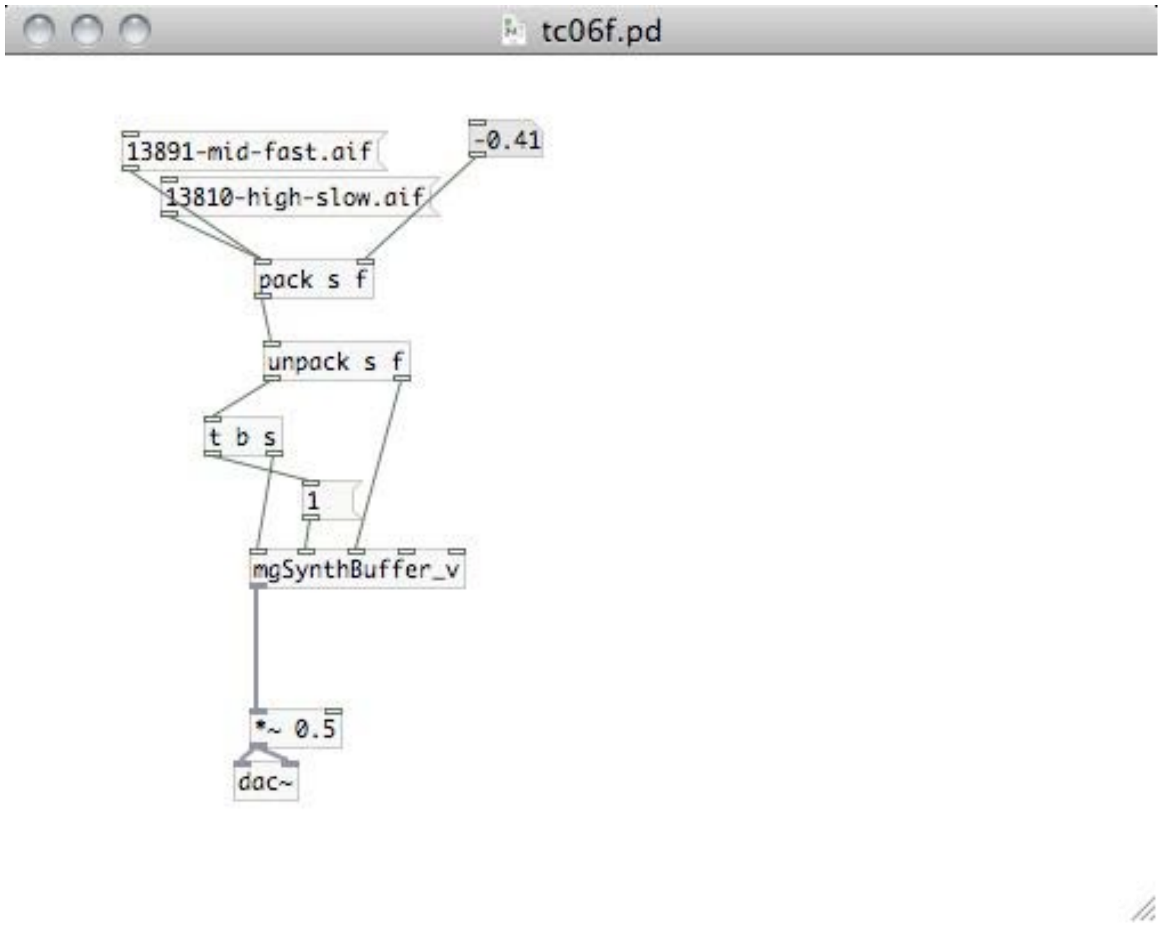
- [unpack] takes a list and provides outlets for each component
- Outlets return values from right to left
- Must declare how many components to include by specifying type of argument (f, s, etc)
- Example: Storing file path and playback rate in a single list and using to trigger playback



tc06e.pd
list 13810-high-slow.aif -1
list 13891-mid-fast.aif 4
list 13891-mid-fast.aif .25
unpack s f
t b s
1
mgSynthBuffer_v
*~ 0.5
dac~

4.21. Building Lists

- [pack] permits building (concatenating) a list of any number of components
- Must declare how many components to include by specifying type of argument (f, s, etc)
- Pack only provides output when a value or bang is received in the left-most inlet: often need to use a trigger to insert value and then bang leftmost inlet
- The [mgPak2] and related send output for every inlets
- Can use [append] and [prepend] to add single elements to lists
- Example: Combing file path and playback rate into a single list and using to trigger playback

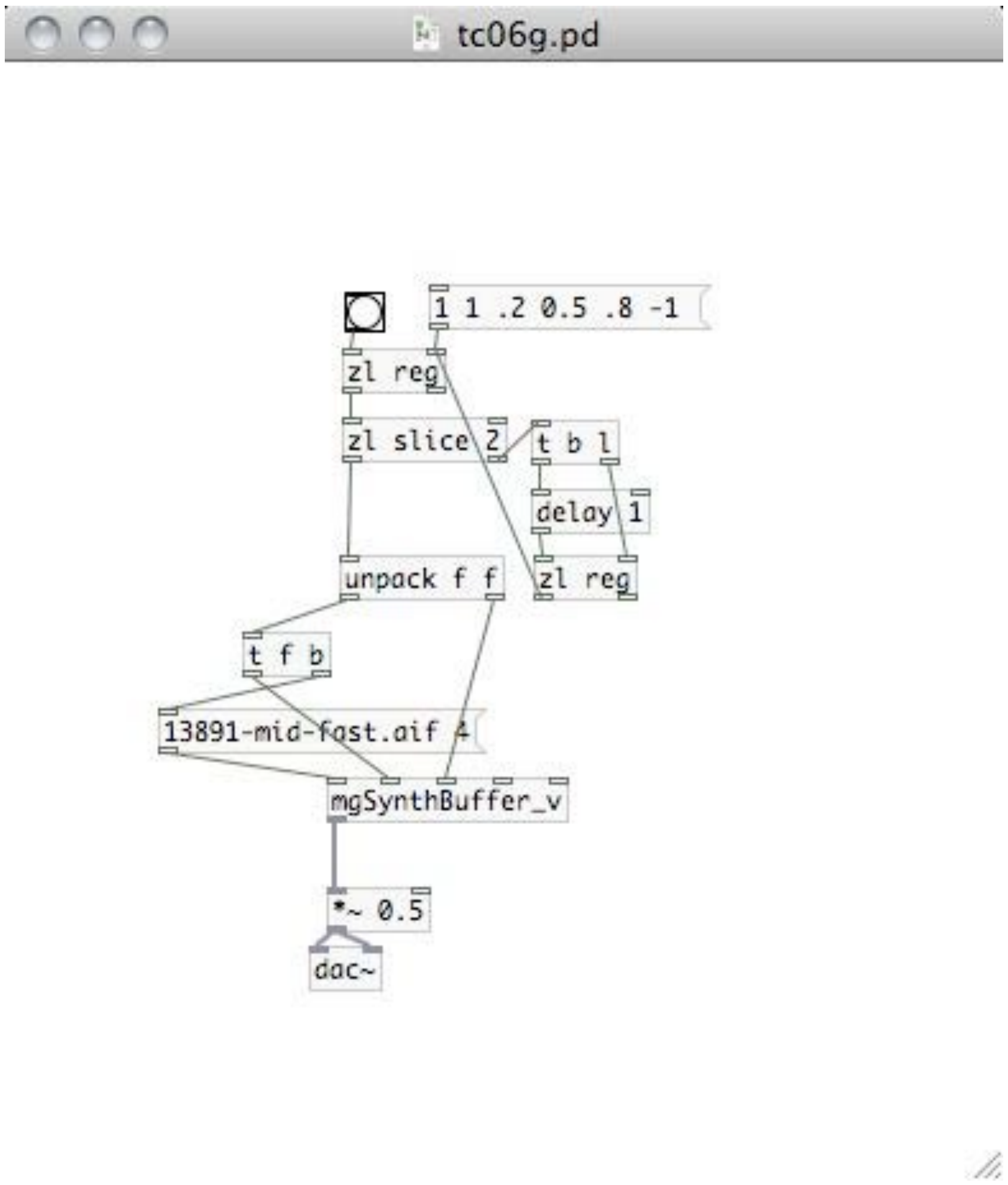


4.22. Lists: Iterating, Grouping, Joining

- Iterating a list one (or more) elements at a time, as fast as possible: [zl iter]
- Wait until a number of items have been received, output them as a list: [zl group]
- Combine two lists: [zl join]

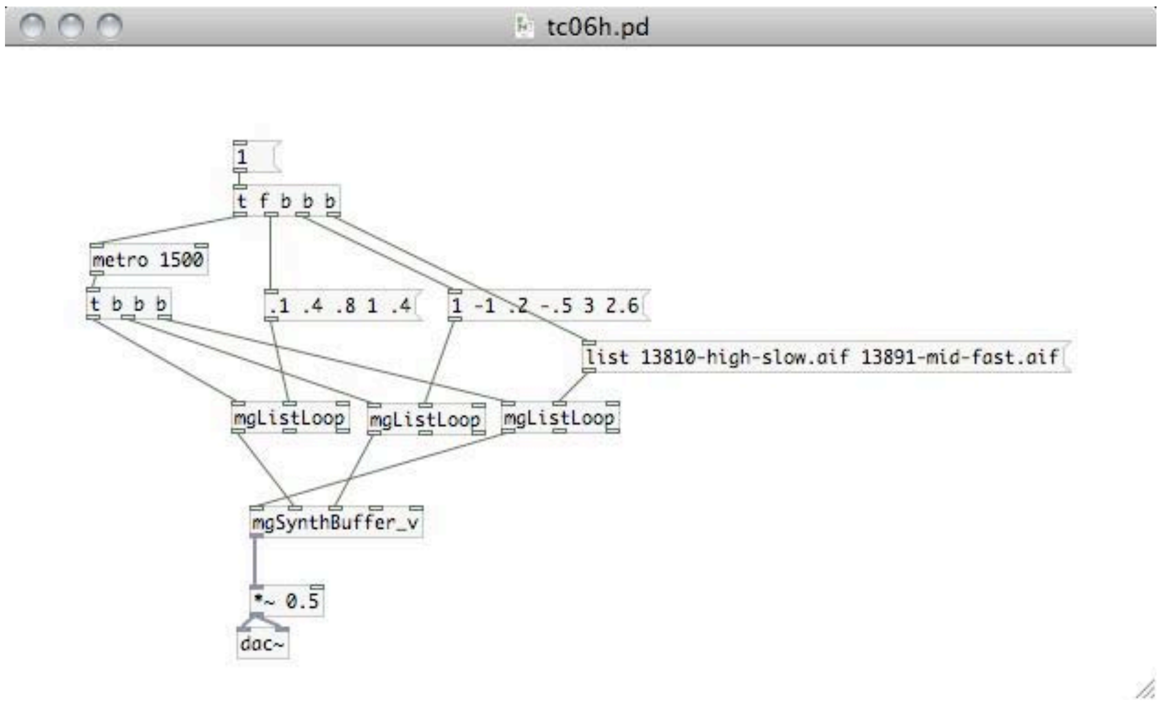
4.23. Lists: Slicing, Rotating, and Reversing

- Slicing from the front and back: [zl slice] and [zl ecils]
- Rotating and reversing: [zl rot], [zl rev]
- Example: Storing playback peak amplitude and playback rate into a single list to trigger playback



4.24. Lists: Accessing By Index

- [zl nth]
- Indices start from 1
- Must provide index first (right inlet) and then complete list (left inlet)
- Alternative functionality available from [mgListLoop]: given a list, provide bangs to loop values



4.25. Listening: Fennesz

- Listening: Fennesz, “Traxdata,” Hotel Paral.lel, 2004

- Listening: Fennesz, “The Point of It All,” Venice, 2004

4.26. Listening: Ikue Mori

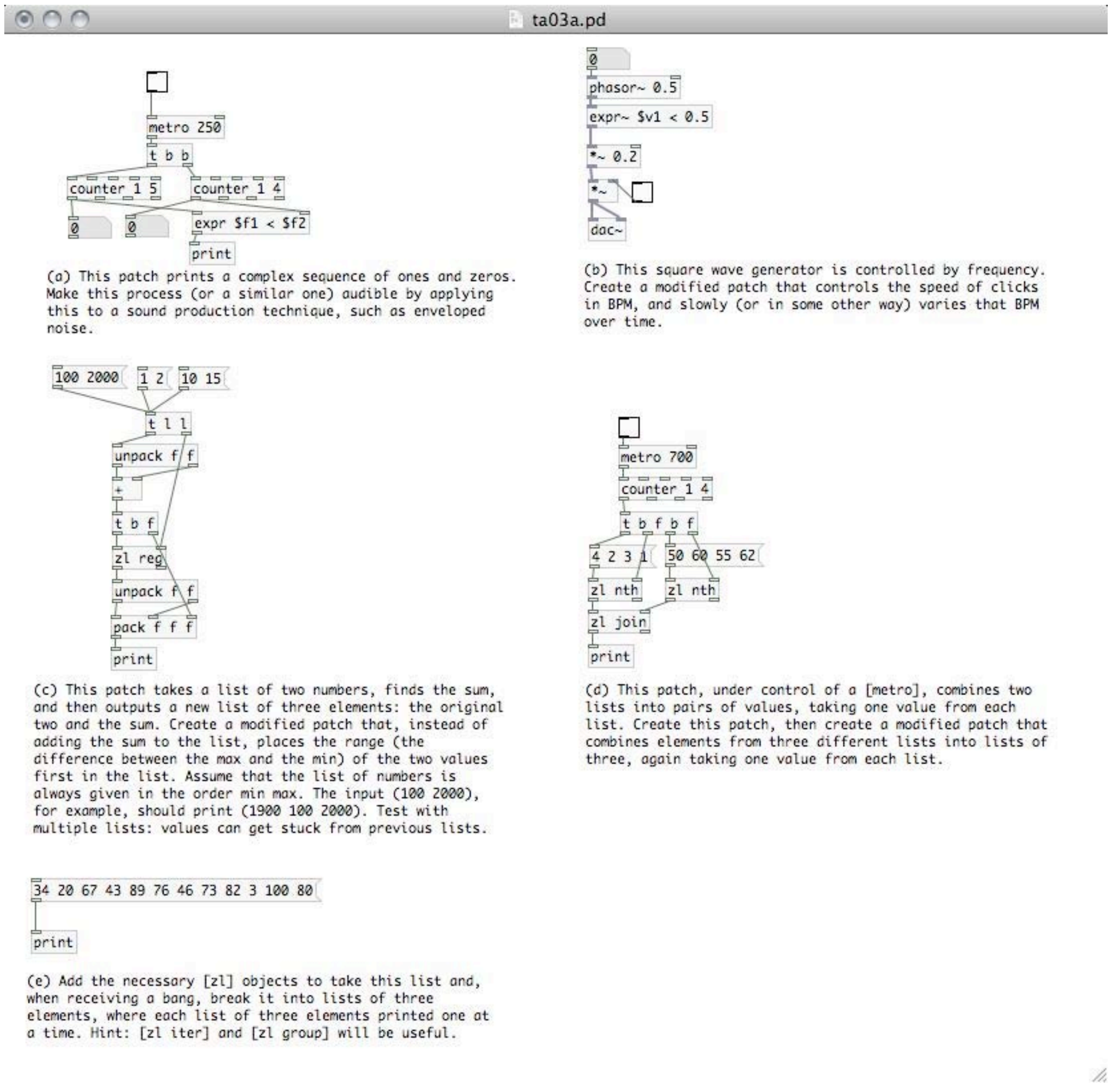
- Listening: Ikue Mori, “The Pit & The Pendulum”, Garden, 1996

- Listening: Ikue Mori, “Civitella Daze”, Labyrinth, 2001

- Listening: Ikue Mori, “Day of Locusts,” Labyrinth, 2001

4.27. Pd Tutorial 3

1. The following examples demonstrate operations with Pd. Recreate the following patch components in a Pd file and and answer the provided questions as comments in the Pd file.



(a) This patch prints a complex sequence of ones and zeros. Make this process (or a similar one) audible by applying this to a sound production technique, such as enveloped noise.

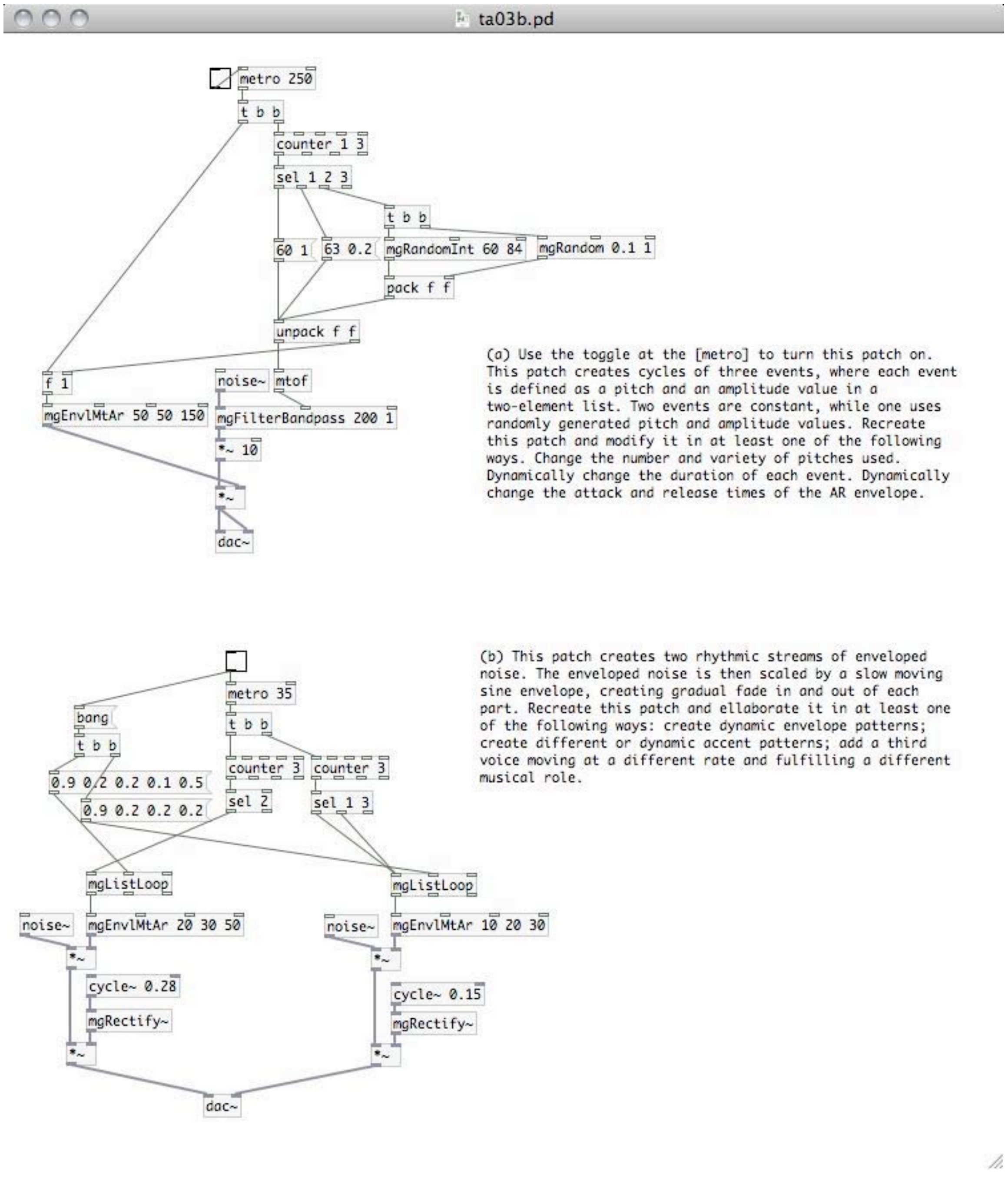
(b) This square wave generator is controlled by frequency. Create a modified patch that controls the speed of clicks in BPM, and slowly (or in some other way) varies that BPM over time.

(c) This patch takes a list of two numbers, finds the sum, and then outputs a new list of three elements: the original two and the sum. Create a modified patch that, instead of adding the sum to the list, places the range (the difference between the max and the min) of the two values first in the list. Assume that the list of numbers is always given in the order min max. The input (100 2000), for example, should print (1900 100 2000). Test with multiple lists: values can get stuck from previous lists.

(d) This patch, under control of a [metro], combines two lists into pairs of values, taking one value from each list. Create this patch, then create a modified patch that combines elements from three different lists into lists of three, again taking one value from each list.

(e) Add the necessary [zl] objects to take this list and, when receiving a bang, break it into lists of three elements, where each list of three elements printed one at a time. Hint: [zl iter] and [zl group] will be useful.

2. Create and extend the following patch.



(a) Use the toggle at the [metro] to turn this patch on. This patch creates cycles of three events, where each event is defined as a pitch and an amplitude value in a two-element list. Two events are constant, while one uses randomly generated pitch and amplitude values. Recreate this patch and modify it in at least one of the following ways. Change the number and variety of pitches used. Dynamically change the duration of each event. Dynamically change the attack and release times of the AR envelope.

(b) This patch creates two rhythmic streams of enveloped noise. The enveloped noise is then scaled by a slow moving sine envelope, creating gradual fade in and out of each part. Recreate this patch and ellaborate it in at least one of the following ways: create dynamic envelope patterns; create different or dynamic accent patterns; add a third voice moving at a different rate and fulfilling a different musical role.

Chapter 5. Meeting 5, Workshop: Performance and Improvisation

5.1. Announcements

- Class meets next week on Tuesday and Wednesday!
- No tutorial due on Tuesday

5.2. Listening: Supersilent

- Listening: SuperSilent, "5.1," 5, 2001
- Listening: SuperSilent, "6.2," 6, 2001 [PhonCD P Sup76 sup6]

5.3. Reading: Cook: Principles for Designing Music Controllers

- Cook, P. 2001. "Principles for designing computer music controllers." *Proceedings of the Conference on New Interfaces for Musical Expression*.
- Why is programmability is a curse?
- What is meant by saying that some players have spare bandwidth?
- Why is it valuable to make a piece, not an instrument or controller?

5.4. Reading: Trueman: Why a Laptop Orchestra?

- Trueman, D. 2007. "Why a Laptop Orchestra?." *Organised Sound* 12(2): pp. 171-179.

- Are there good reasons for calling an ensemble of laptops an orchestra?
- What makes PLOrk different from other laptop orchestras?
- What makes PLOrk different from other laptop orchestras?
- How does distance and spatial orientation affect performance practice? Are these differences changed with electronic instruments?
- What are some approaches to overcome the lacking visual elements of a laptop-orchestra?

5.5. An Ensemble, not an Orchestra

- No hierarchy of instruments
- Non-uniformity in equipment
- Improvisation and performance at the foreground, not recreating fixed works
- Performer-directed, rather than conductor-directed
- The orchestra is an old historical European organization; our practice derives from other musical traditions and ensemble compositions

5.6. Approaching Experimental Music and Improvisation

- All musical ideas made with integrity are valid (not necessarily aesthetically pleasing, but valid)
- Attention and seriousness is required; humor happens, but is subtle
- A shared sense of creative opportunity (Ballou 2009)
- Communicate with body and instrument: do not speak, ask questions, mumble, or apologize (musical exclamations are possible)
- No undo or redo: make it work (mistakes are opportunities)
- Less is often more; louder is not better

5.7. Mono Performance B (mgSynthNoiseFilter)

- Eight different noise textures on buttons 1-8
- Y1: amplitude; Y2: AM modulation
- X1: high pass filter; X2: low pass filter

- D-pad 1 (left): slow echo; D-pad 2 (down): fast echo; D-pad 3 (right): long reverb;
- Ideas
  - Can create gestures with filters X1 and X2
  - Can create fades and rhythmic articulations with Y1 amplitude
- Cautions
  - Moving Y1 greater the .5 will result in a boost that may peak
  - Careful with Y2: excessive AM is not desirable

5.8. Exercise: Pass-the-Gesture Solo

- Load: mono performance-b.test.pd

  martingale/pd/instruments/dualAnalog*/mono/performance-b.test.pd

  SynthNoiseFilter
- Performer A: Create a memorable musical gesture, lasting between 5 and 15 seconds
- Performer B: Re-create the musical gesture of Performer A, not necessarily using the same sound source, but maintaining temporal and spectral outline.
- Continue process from performer to performer, moving around in a circle

5.9. Exercise: Pass-the-Gesture Ensemble

- Load: mono performance-b.test.pd

  martingale/pd/instruments/dualAnalog*/mono/performance-b.test.pd

  SynthNoiseFilter
- Ensemble: Create a subtle, low frequency background texture, contributing a small bit to the total sound

  Performer A: Create a memorable musical gesture, lasting between 5 and 15 seconds; repeat it twice, with variable space in-between gestures.
- Performer B: Re-create the musical gesture of Performer A, not necessarily using the same sound source, but maintaining temporal and spectral outline.

  Ensemble: Continue

- Continue process from performer to performer, moving around in a circle
- Variation: Performers A and B selected in pairs: A and B in a dialog or argument
- Variation: Performer B responds to A not in similarity, but maximal contrast, producing the opposite idea

5.10. Poly Performance B (mgSynthBuffer8, mgSynthBuffer8)

- Eight different samples buttons 1-8
- Y1: amplitude; Y2: LPF resonance
- X1: NC; X2: low pass filter cutoff frequency
- D-pad 1 (left): slow echo; D-pad 2 (down): fast echo; D-pad 3 (right): long reverb;
- Poly
    - Instrument 1: percussive tones, lower on keys 1-4, higher on keys 5-8
    - Instrument 2: longer, lower tones on keys 1-4, long, ambient samples on keys 5-8
- Ideas
    - Can create gestures with filter; can pick out pitches with resonance of filter
    - Can create fast and short articulations
    - Can create fades and rhythmic articulations with Y1 amplitude
    - Can create spectral fades by bringing in low-pass filter slowly
- Cautions
    - Releasing a key closes the envelope; striking a key always start it at the beginning
    - Careful with Y1: samples have large dynamic range
    - Careful with Y2: excessive resonance can distort

5.11. Exercise: Pass-the-Rhythm Solo

- Load: poly performance-b.test.pd

  martingale/pd/instruments/dualAnalog*/poly/performance-b.test.pd
- Performer A: Create a memorable musical rhythm, lasting between 5 and 15 seconds

- Performer B: Re-create the musical rhythm of Performer A, not necessarily using the same sound source, but maintaining rhythmic, temporal, and spectral outline.

- Continue process from performer to performer, moving around in a circle

5.12. Exercise: Eight On / Eight Off

- Load: poly performance-b.test.pd

  martingale/pd/instruments/dualAnalog*/poly/performance-b.test.pd

- Ensemble: Articulate 8 beats in-time, on major beats or divisions (not necessarily uniform)

- Ensemble: Silence for 8 beats, or short rhythms or gestures in-time within 8 beat span or less

- Repeat

5.13. Exercise: Pass-the-Rhythm Ensemble

- Load: poly performance-b.test.pd

  martingale/pd/instruments/dualAnalog*/poly/performance-b.test.pd

- Ensemble: Articulate a regular pulse by articulating a simply 4 or 8 beat pattern

  Performer A: Create a memorable musical rhythm, lasting between 5 and 15 seconds

- Performer B: Re-create the musical rhythm of Performer A, not necessarily using the same sound source, but maintaining rhythmic, temporal, and spectral outline.

  Ensemble: Continue

- Continue process from performer to performer, moving around in a circle

Chapter 6. Meeting 6, Foundations: Processing and Transforming Sounds

6.1. Announcements

- Bring amps and controllers to next class (tomorrow)
- Quiz next Monday
- Complete discussion leader assignment schedule is posted

6.2. Review Pd Tutorial 3

- ?

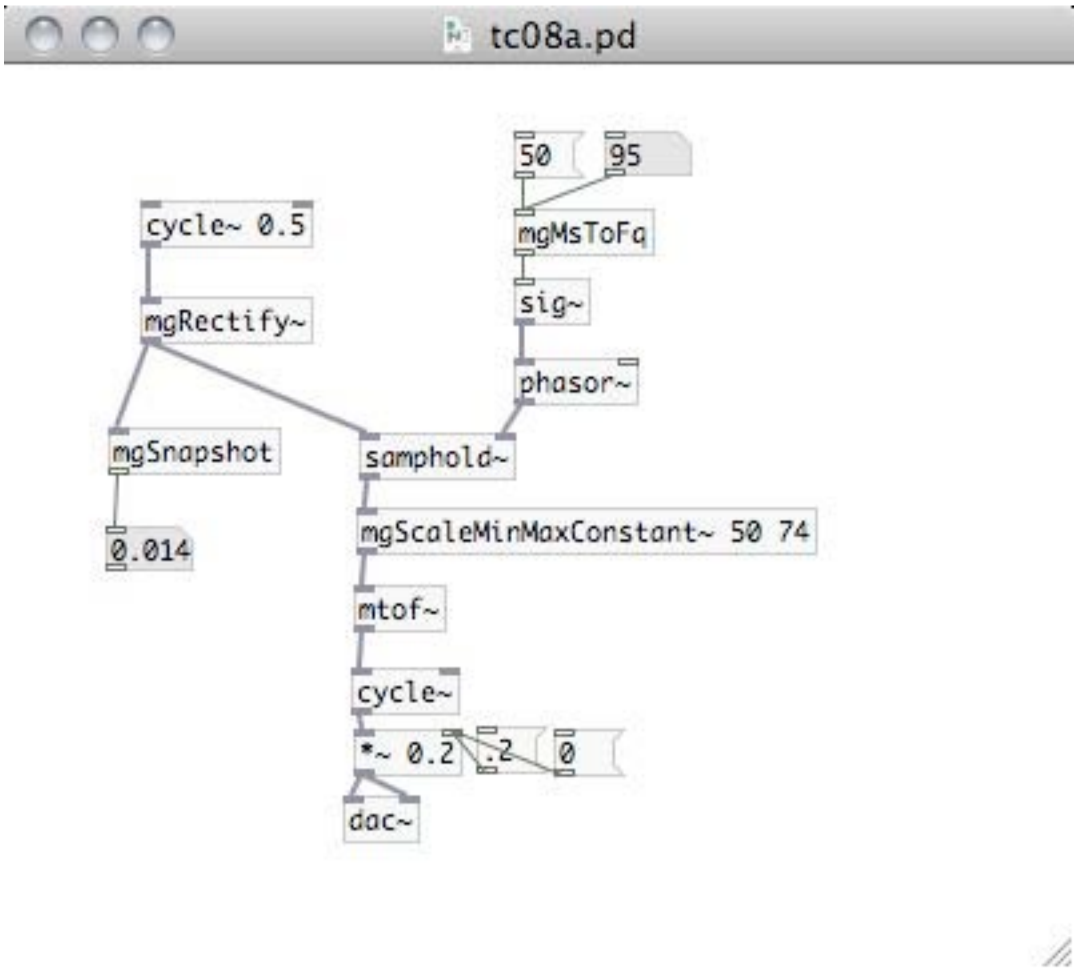
6.3. Reading: Wanderley and Orio: Evaluation of Input devices for Musical Expression: Borrowing Tools from HCI

- Wanderley, M. M. and N. Orio. 2002. "Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI." *Computer Music Journal* 26(3): pp. 62-76.
- What are some categories of input devices for musical expression the authors offer?
- What are some musical and aesthetic problems in evaluating musical controllers?
- What is Fritts's law? Does it relate to music making? Does it relate to the dual analog controller?
- How are musical controllers like controlling a vehicle?
- What are some contexts of interactive computer music mentioned?
- What ways can you think of quantitatively evaluating musical controllers?

6.4. Slowing Signals

- The audio rate is fast (44100 samples per second)
- [mgSnapshot]: periodically samples a signal and returns a value the event rate (once every 50 msec)
- Sometimes we want to sample a signal and return the value as a signal
- [samphold~]: sample a signal and output the value until triggered

- Trigger is when value in right inlet decreases (such as during one cycle of [phasor~])
- Using [samphold~] to map discrete values of a sine tone to the pitch of another sine tone.



6.5. Lists and Arrays

- A collection of similar data in a single row accessible by an index
- Lists count from 1

  Lists can be floats or symbols
- Arrays count from 0

  Arrays have to be floats

6.6. Arrays and Tables

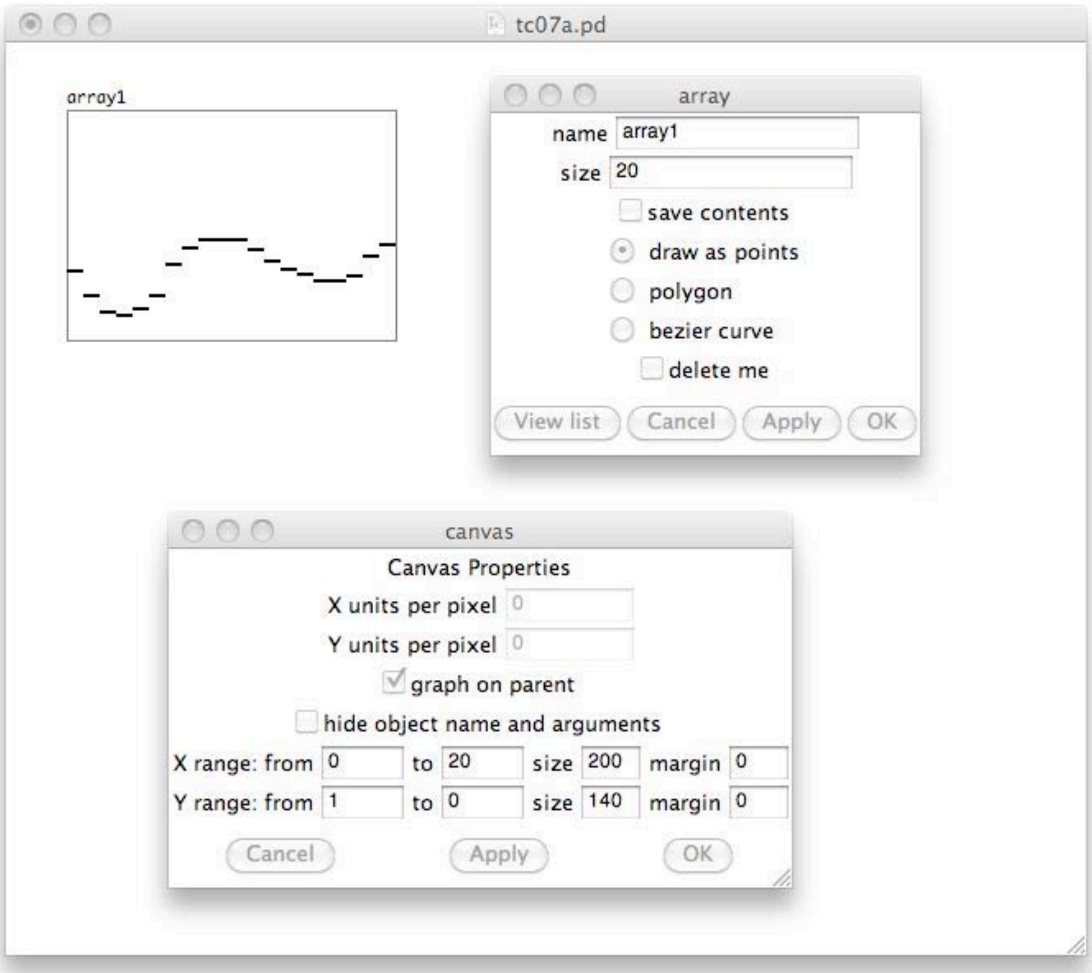
- Arrays store data in an object, and then access and write data into that object through other objects
- Data can be small collections of parameters, or millions of samples (an audio file)
- All arrays are named, though array names can be randomly generated with each patch using $0
- Array: visual representation of an array through a Canvas object
- Table: an Array Graph hidden in a sub-patch: can be given a dynamic name: [table name]

6.7. Arrays: Creating and Configuring

- Use the Put menu: Menu: Put > Array
- Will get windows for properties; can configure later by option-clicking
- Canvas parameters
    - Values here are for visual presentation only: does not limit the range of values that can be stored
    - Size is pixels on the screen
    - Range is mapping of values inside of the display
    - Y range is entered from max to minimum (not as expected)
- Array parameters
    - Name must be unique

      Duplications result in an error: “warning: array1: multiply defined”

      Array names can be accessed from any open patch
    - Size: number of data points
    - Draw as points / polygon / bezier curve refers to presentation alone
    - Save content: stores data in patch (not reccommended: stores raw data in Pd patch)
- Use [table] for all arrays that do not need to be seen (most)
- Create an array with 20 points (array properties), and (under canvas properties) an X range from 0 to 20 and a Y range from 1 to 0 (note order of values)

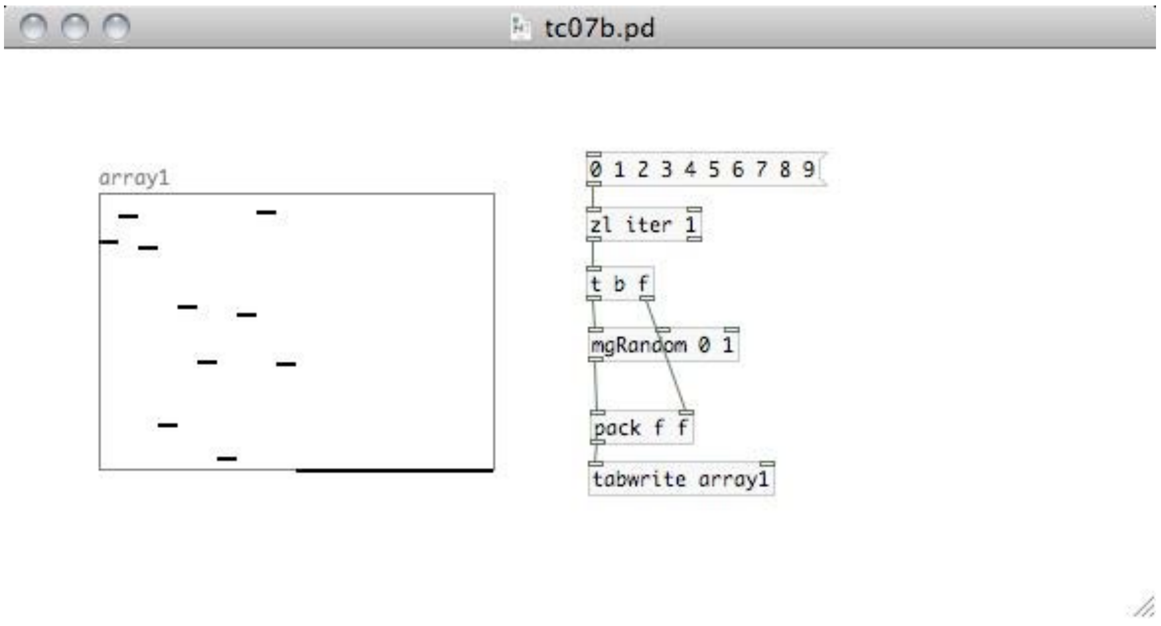


name: array1
size: 20
save contents: [ ]
[✓ draw as points] polygon bezier curve
delete me: [ ]

X units per pixel: 0
Y units per pixel: 0
graph on parent: [✓]
hide object name and arguments: [ ]
X range: from 0 to 20 size 200 margin 0
Y range: from 1 to 0 size 140 margin 0

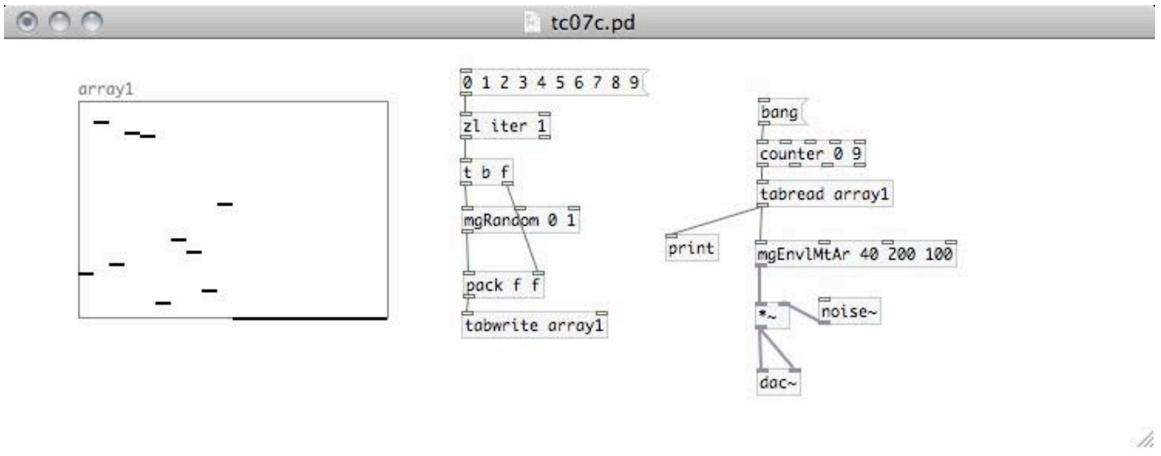
6.8. Adding Data

- Can directly edit graphical representation: click and drag in run mode
- [tabwrite name]: provide value, index pairs (as a list); must include array name as object argument
- [soundfiler]: given an array name, can load and resize an array for an audio file (we will use the more powerful [mgSoundfilerControl])
- Can dynamically build 2 element lists of value, index position:



6.9. Reading Array Data at the Event Rate

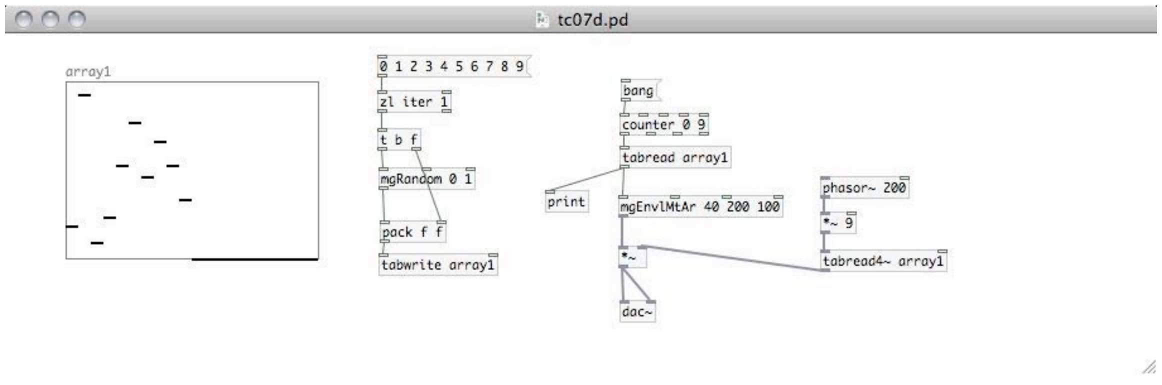
- [tabread name]: given an array name, provide index, get value
- Can read values from an array and employ them for peak envelope values and triggers



6.10. Reading Array Data at the Audio Rate

- Can read data from an array at the audio rate, using arrays to store wavetables or audio samples
- Input is a signal providing the index value, not a number; a scaled [phasor~] (from 0 to the size of the table-1) is the most common approach
- [tabread~ name]: read array data at the audio rate

- [tabread4~ name]: read array data at the audio rate with interpolation
- Using a small array as a wavetable with [tabread4~]



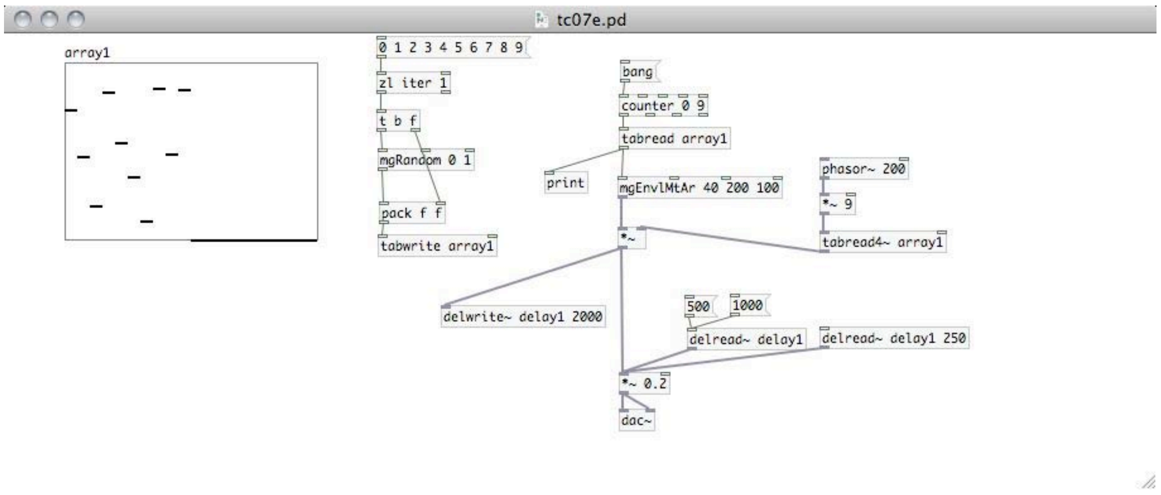
6.11. Delays and Reverbs

- Many common effects are created through combinations of delays
- Echos and reverbs are all built from delays
- Very short reflections blur together
- Echos interfere with the original signal, changing its timbre
- Example: martingale/demo/processorsDelay.pd

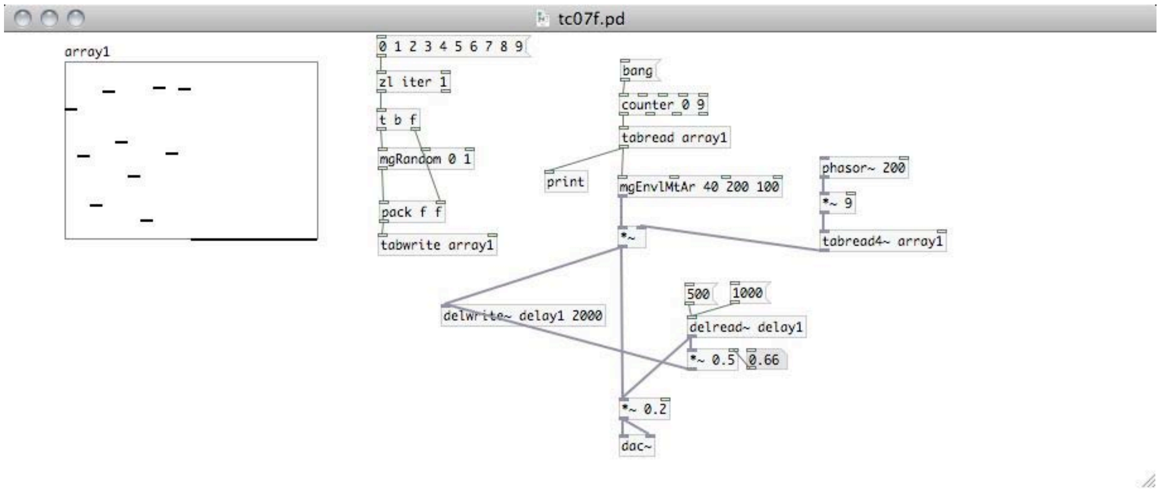
  Example: martingale/demo/processorsDelayDense.pd

6.12. Creating and Reading from Delay Lines

- Delay lines are stored memory for signals
- Must be declared with a name and a maximum delay time: [delwrite name 1000]
- Can be read from at numerous different times, and time can be provided as an argument: [delread~ name 200]



- Can create multiple echos by scaling signal and returning to delay line (feedback)



6.13. Listening: Musica Elettronica Viva (MEV)

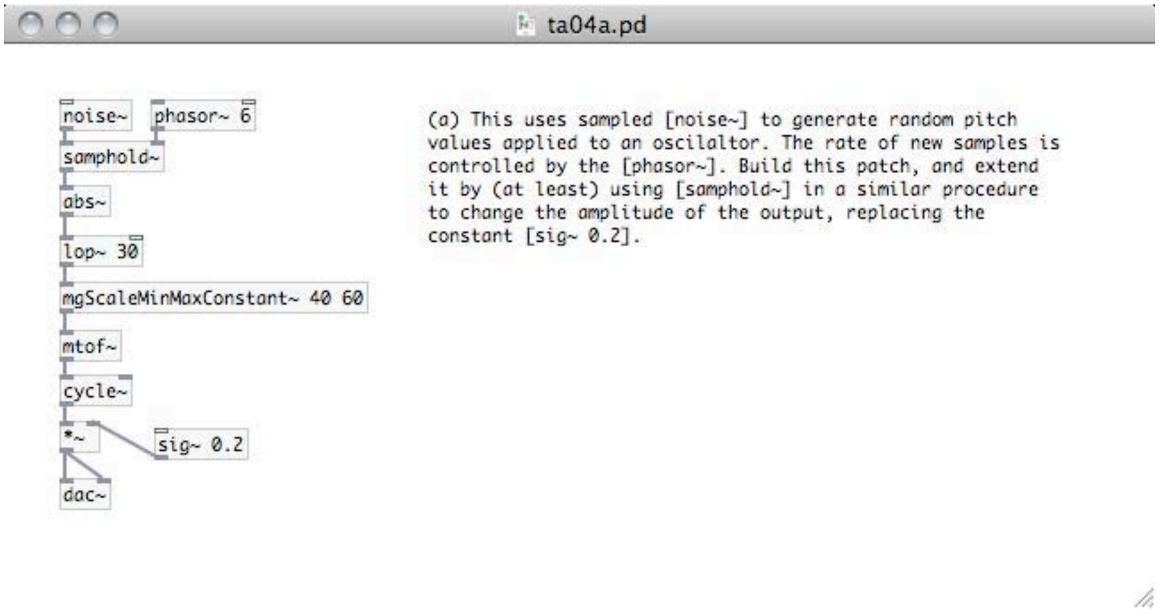
- Listening: Musica Elettronica Viva (MEV), *Spacecraft*, 1967

6.14. Listening: AMM

- Listening: AMM, “The Great Hall,” *Laminal*, 1982

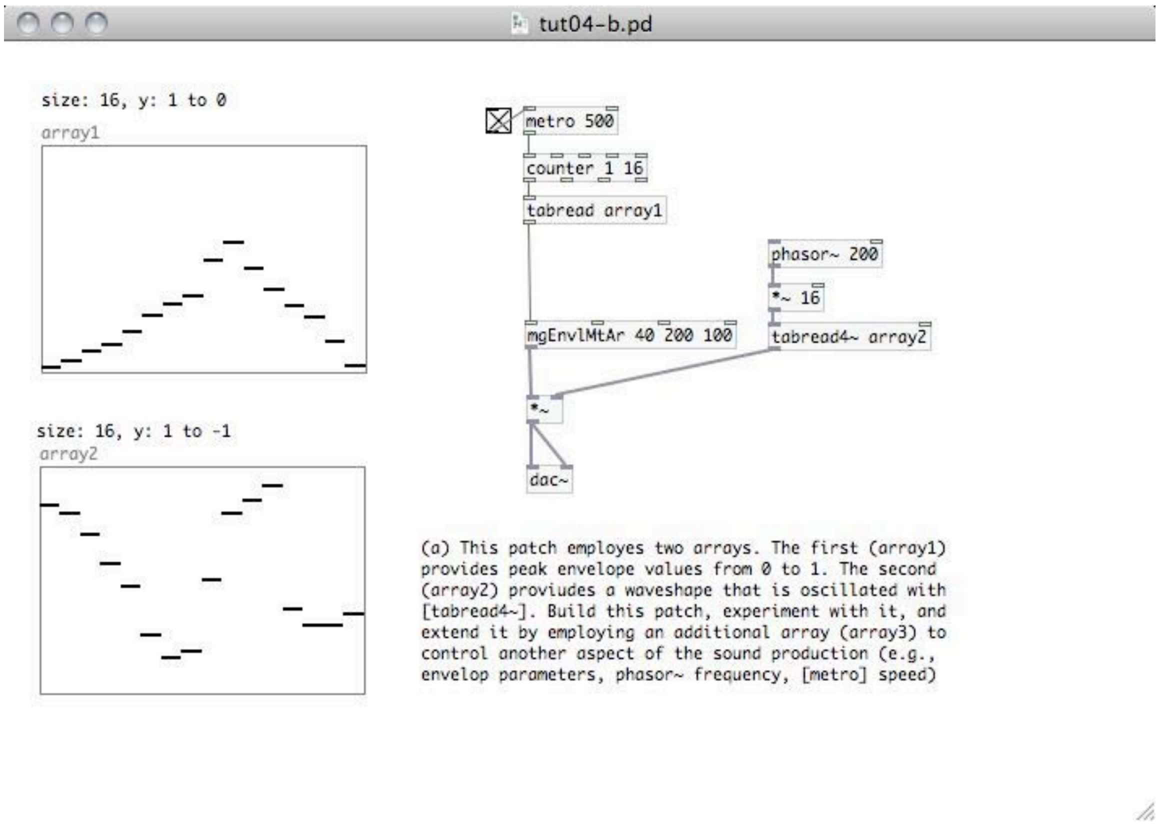
6.15. Pd Tutorial 4

1. Create and extend the following patch.



(a) This uses sampled [noise~] to generate random pitch values applied to an oscilaltor. The rate of new samples is controlled by the [phasor~]. Build this patch, and extend it by (at least) using [samphold~] in a similar procedure to change the amplitude of the output, replacing the constant [sig~ 0.2].

2. Create and extend the following patch.



(a) This patch employes two arrays. The first (array1) provides peak envelope values from 0 to 1. The second (array2) proviudes a waveshape that is oscillated with [tabread4~]. Build this patch, experiment with it, and extend it by employing an additional array (array3) to control another aspect of the sound production (e.g., envelop parameters, phasor~ frequency, [metro] speed)

Chapter 7. Meeting 7, Workshop: Performance and Improvisation

7.1. Announcements

- Quiz on Monday
- Planning a concert for Wednesday May 4: what time?

7.2. Listening: Cage

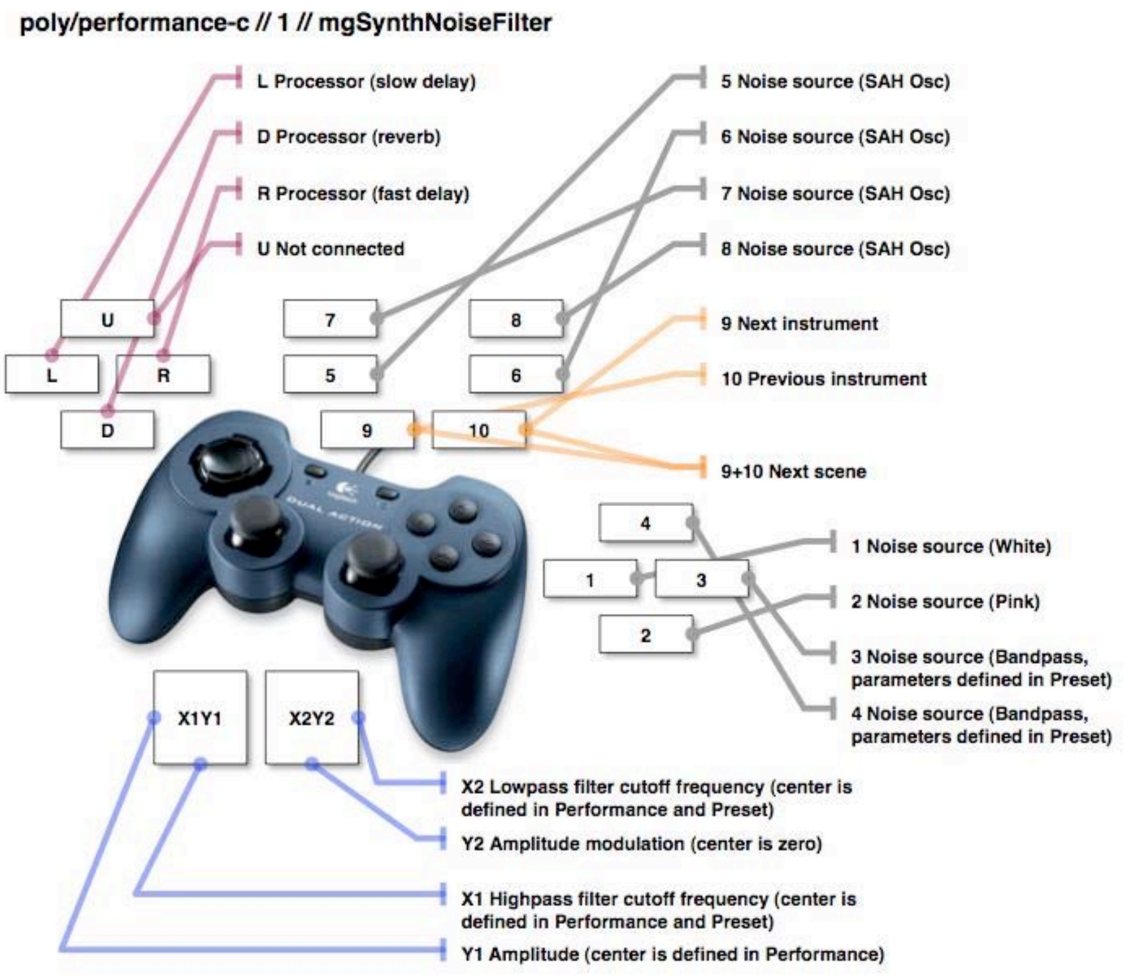
- Listening: John Cage, Cartridge Music, 1960
- Listening: John Cage, Imaginary Landscape No. 1, 1939

7.3. Reading: Holmes: Live Electronic Music and Ambient Music

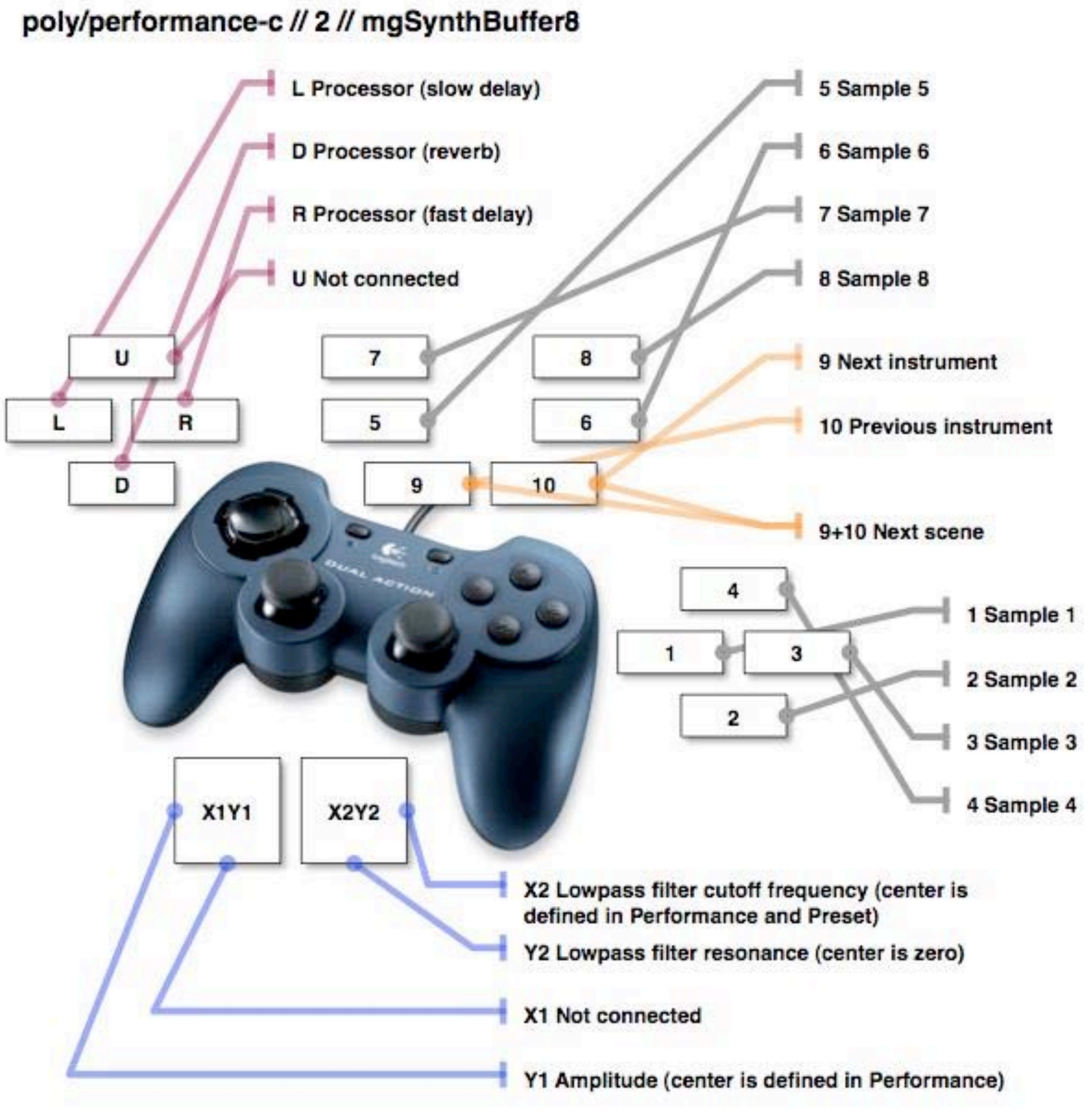
- Holmes, T. 2008. "Live Electronic Music and Ambient Music." In T. Holmes, ed. *Electronic and Experimental Music.* Third ed. New York: Routledge, pp. 376-406.
- What was the artistic backlash to loudspeaker music?
- What motivated Cage to work with a dance ensemble?
- What can we learn from early performance ensembles and festivals?

7.4. Poly Performance C (mgSynthNoiseFilter, mgSynthBuffer8, mgSynthSaw)

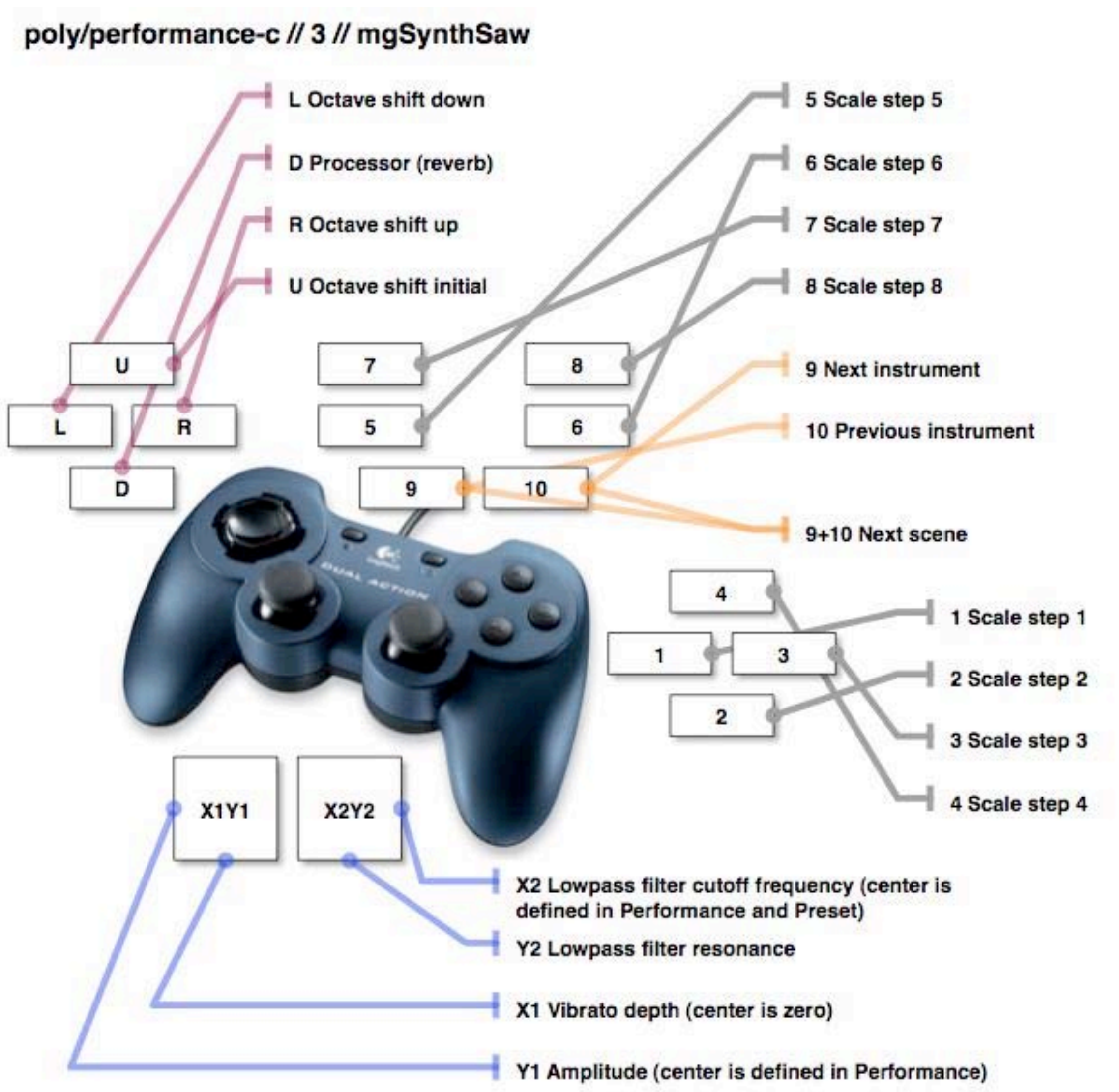
- Poly: 3 instruments
  - Instrument 1: mgSynthNoiseFilter


poly/performance-c // 1 // mgSynthNoiseFilter
L Processor (slow delay)
D Processor (reverb)
R Processor (fast delay)
U Not connected
5 Noise source (SAH Osc)
6 Noise source (SAH Osc)
7 Noise source (SAH Osc)
8 Noise source (SAH Osc)
9 Next instrument
10 Previous instrument
9+10 Next scene
1 Noise source (White)
2 Noise source (Pink)
3 Noise source (Bandpass, parameters defined in Preset)
4 Noise source (Bandpass, parameters defined in Preset)
X2 Lowpass filter cutoff frequency (center is defined in Performance and Preset)
Y2 Amplitude modulation (center is zero)
X1 Highpass filter cutoff frequency (center is defined in Performance and Preset)
Y1 Amplitude (center is defined in Performance)
U
L
R
D
7
5
8
6
9
10
4
1
3
2
X1Y1
X2Y2

  - Instrument 2: mgSynthBuffer8



- Instrument 3: mgSynthSaw



7.5. Instruments and Scenes

- A scene defines present for all instruments in the poly instrument
- Advancing the scene (buttons 9+10 together) changes all presets
- Generally used for large-scale changes

7.6. Exercise: Smooth Fades

- Load: poly performance-c.test.pd

  martingale/pd/instruments/dualAnalog*/poly/performance-c.test.pd

- Ensemble: instrument 1: fade in over 8 counts, fade out over 8 counts
- Variation: in over 4, out over 4 + 8

  Variation: in over 8 + 4, out over 4

7.7. Exercise: Scales

- Load: poly performance-c.test.pd

  martingale/pd/instruments/dualAnalog*/poly/performance-c.test.pd
- Ensemble: instrument 3: scene 2: ascending and descending scale, repeating top note
- Variation: improvise with timber, filters, dyanmics

  Variation: legato v. staccato

  Variation: shift octave

  Variation: open filter in ascent, close filter in descent

7.8. Exercise: Scales

- Load: poly performance-c.test.pd

  martingale/pd/instruments/dualAnalog*/poly/performance-c.test.pd
- Ensemble: instrument 3: scene 2: ascending and descending scale, repeating top note
- Variation: improvise with timber, filters, dyanmics

  Variation: legato v. staccato

  Variation: shift octave

  Variation: open filter in ascent, close filter in descent

7.9. Exercise: Drone Swells

- Load: poly performance-c.test.pd

  martingale/pd/instruments/dualAnalog*/poly/performance-c.test.pd
- Performer A: instrument 3: scene 1: select a pitch and sustain, adding dynamic and timbral variation

- Ensemble: instrument 3: scene 1: each performer enters and finds the same pitch (possible at the same octave)
- Variation: Performer A chooses one pitch, Performer B chooses a second pitch, ensemble duplicates
- Variation: Performer A, B, and C each choose different pitches; ensemble then enters

7.10. Exercise: Eight On / Eight Off

- Load: poly performance-c.test.pd

  martingale/pd/instruments/dualAnalog*/poly/performance-c.test.pd
- Ensemble: instrument 2: scene 2, 3, or 4: articulate 8 beats in-time, on major beats or divisions (not necessarily uniform)
- Ensemble: Silence for 8 beats, or short rhythms or gestures in-time within 8 beat span or less
- Variation: change instruments during 8 beats and create alternative gestures

7.11. Exercise: Sixteen Beat Patterns

- Load: poly performance-c.test.pd

  martingale/pd/instruments/dualAnalog*/poly/performance-c.test.pd
- Ensemble: instrument 2: scene 2, 3, or 4: articulate all numbered beats
- Beats with dynamics
  - 1 x 3 x 1 2 3 x 1 x 3 x 1 x x x

    1 2 3 4 1 x 3 x 1 2 3 4 x x 3 4

    1 2 x 4 1 x 3 4 x 2 x 4 x 2 3 x
- Variation: selected performers take solos over beat

  Variation: divide in two: group A does on beats, group B does background with instrument 1

  Variation: divide in two: group A does on beats, group B does drones with instrument 3: scene 2

  Variation: divide in two: group A does on beats, group B does off beat

  Variation: divide in four: group A does off beats, group B does on beats, group C does drones with instrument 3: scene 2, group D does does background with instrument 1

7.12. Exercise: Pass-the-Gesture Solo

- Load: poly performance-c.test.pd

  martingale/pd/instruments/dualAnalog*/poly/performance-c.test.pd

- Performer A: Create a memorable musical gesture, lasting between 5 and 15 seconds

- Performer B: Re-create the musical gesture of Performer A, not necessarily using the same sound source, but maintaining temporal and spectral outline.

- Continue process from performer to performer, moving around in a circle

7.13. Exercise: Pass-the-Gesture Ensemble

- Load: poly performance-c.test.pd

  martingale/pd/instruments/dualAnalog*/poly/performance-c.test.pd

- Ensemble: Create a subtle, low frequency background texture, contributing a small bit to the total sound

  Performer A: Create a memorable musical gesture, lasting between 5 and 15 seconds; repeat it twice, with variable space in-between gestures.

- Performer B: Re-create the musical gesture of Performer A, not necessarily using the same sound source, but maintaining temporal and spectral outline.

  Ensemble: Continue

- Continue process from performer to performer, moving around in a circle

- Variation: Performers A and B selected in pairs: A and B in a dialog or argument

- Variation: Performer B responds to A not in similarity, but maximal contrast, producing the opposite idea

Chapter 8. Meeting 8, Practices: The Early History of Live Electronics

8.1. Announcements

- Begin to prepare a 1 to 2 minute solo or duo improvisation with Performance C.

8.2. Review Pd Tutorial 4

- ?

8.3. Quiz

- 10 Minuets

8.4. Landmarks in Early Live Electronics

- 1950s to 1960s: John Cage, David Tudor
- 1961 to 1965: ONCE Festivals: Robert Ashley, Gordon Mumma, Pauline Oliveros, David Behrman
- 1966: Sonic Arts Union: Robert Ashley, Gordon Mumma, David Behrman, Alvin Lucier
- Late 1960s: MEV (Curran, Teitelbaum, Rzewski, Lacy, others) and AMM (Prevost, Rowe, Cardew, others)
- 1970s: League of Automatic Composers: Jim Horton, John Bischoff, Tim Perkis, David Behrman, Paul DeMarinis, others
- 1985: first Hub concerts

8.5. Reading: Gresham-Lancaster, The Aesthetics and History of the Hub

- Gresham-Lancaster, S. 1998. "The Aesthetics and History of the Hub: The Effects of Changing Technology on Network Computer Music." *Leonardo Music Journal* 8: pp. 39-44.
- The author describes David Tudor's *Rainforest* as collaborative live performance: explain this.

- How did the Hub’s technology change? How did this alter their aesthetic? How did this affect their ability to build repertory?
- The author describes at times reveling in surprise, and at other times being grateful for system stability: is this a contradiction?
- What is a blackboard system, and how might it be used for collaborative music making?
- Describe a few approach to making a score for the Hub.

8.6. Listening: David Tudor/Pauline Oliveros, Applebox Double

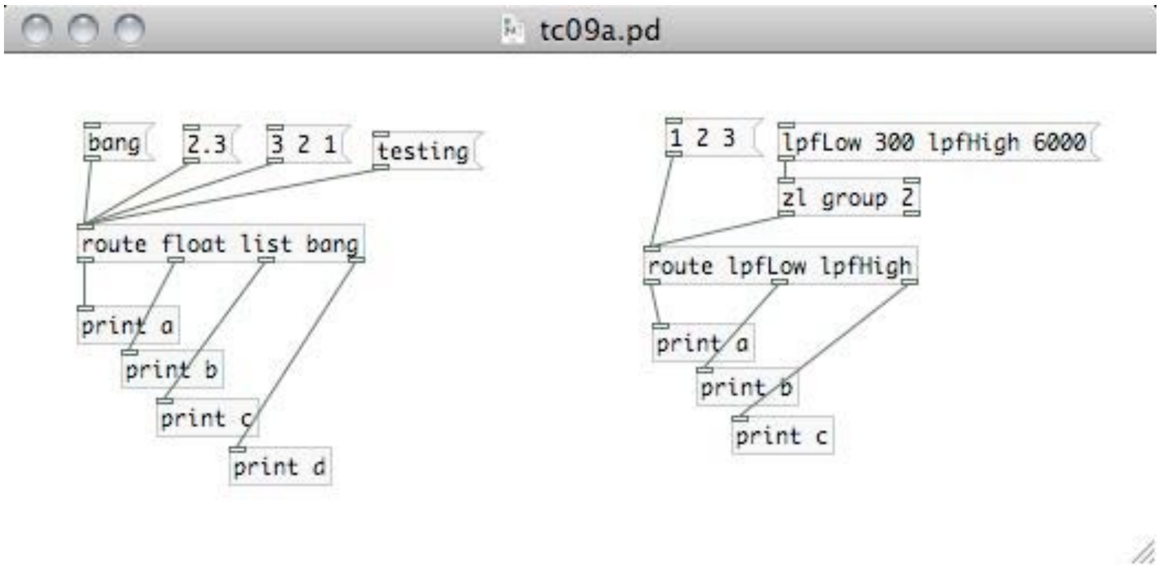
- Part of a collection: music from the ONCE festival, 1961 to 1966
- Listening: David Tudor/Pauline Oliveros, “Applebox Double,” 2003

8.7. Listening: League of Automatic Composers

- Listening: League of Automatic Composers, “Oakland One,” *League of Automatic Music Composers 1978-1983*

8.8. Managing Data Streams

- Can use [route] to move data to different outlets by type
- Can be data types: bang, symbol, float, or list
- Can be first symbol of a two element list
- Example



8.9. Building Patches and Abstractions

- Each small reusable and/or testable component should be abstracted into a component
- Use [inlet], [outlet], [inlet~], [outlet~] to create inlets and outlets
  - Add details as to what inlets are in inlet boxes
  - Assume that order of non-signal inlets might mawtter
- Save as a .pd file in a location Pd can find
- Use [loadbang] bang to provide initialization bangs on startup
- Create a test file
  - Abstraction should always work at startup with no configuration
  - Test should give an idea of what they object can do

8.10. Patches and Resources

- Pd can find any Pd or audio file that is in its list of paths
- Pd can find any file that is in the same directory as the originating path

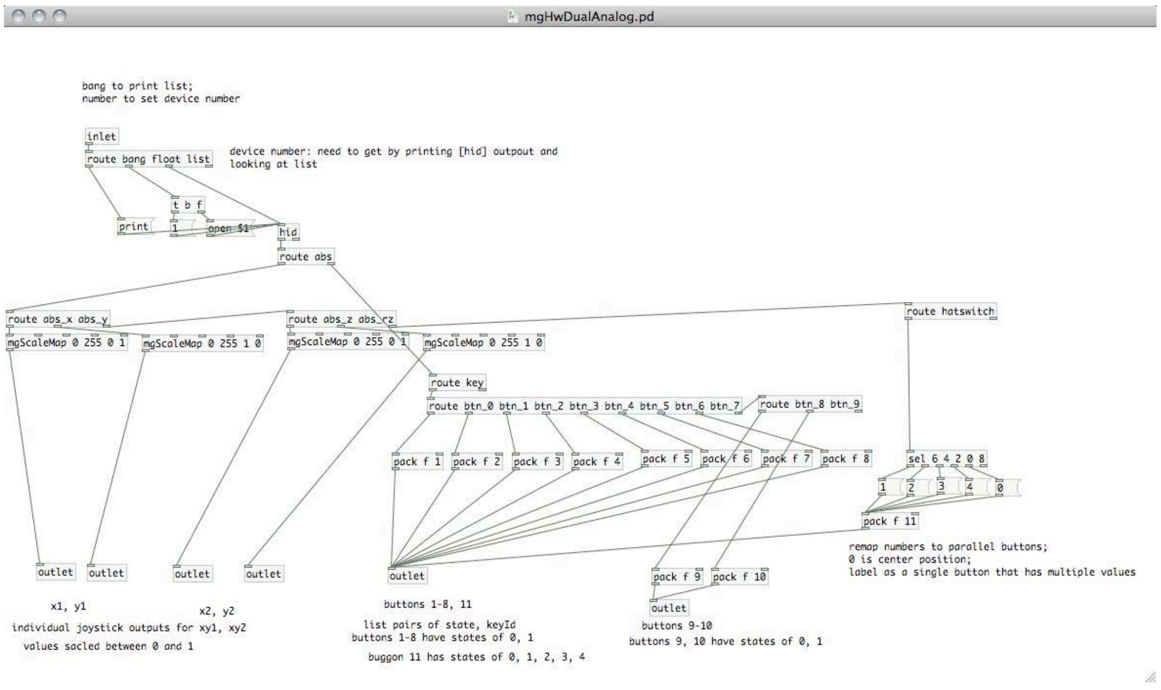
8.11. A Sample Instrument

- Hardware abstraction of Dual Analog polymorphic control

- Voice, instrument, parameters, performance (VIPP) design pattern

8.12. Hardware Abstraction of the Dual Analog

- [mgHwDualAnalog]



bang to print list;
number to set device number

device number: need to get by printing [hid] outpout and
looking at list

remap numbers to parallel buttons;
0 is center position;
label as a single button that has multiple values

x1, y1

x2, y2

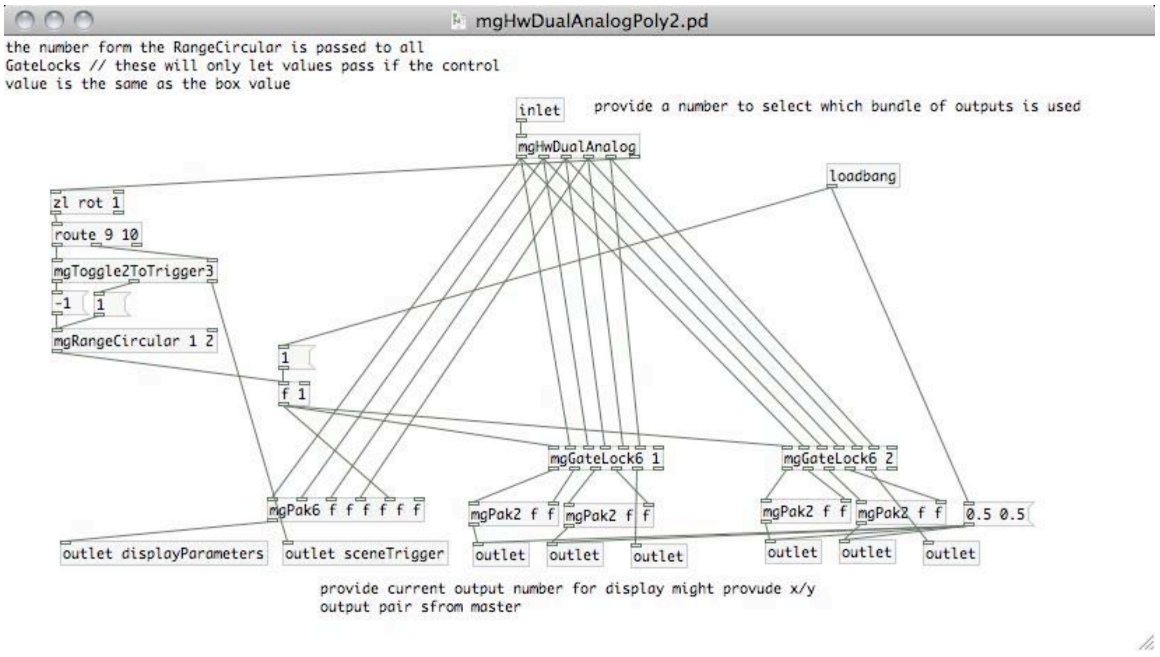
individual joystick outputs for xy1, xy2
values sacled between 0 and 1

buttons 1-8, 11

list pairs of state, keyId
buttons 1-8 have states of 0, 1
buggon 11 has states of 0, 1, 2, 3, 4

buttons 9-10
buttons 9, 10 have states of 0, 1

- Polymorphic control: [mgHwDualAnalogPoly2]



the number form the RangeCircular is passed to all
GateLocks // these will only let values pass if the control
value is the same as the box value

provide a number to select which bundle of outputs is used

provide current output number for display might provude x/y
output pair sfrom master

Chapter 9. Meeting 9, Practices: Extending Common Physical Controllers

9.1. Announcements

- Prepare a 1 to 2 minute solo or duo improvisation with Performance C.
- Next class: bring amps and controllers
- Due on Monday: Controller/Interface/Instrument Design 1 Draft

  Bring to class and prepare to demonstrate

9.2. Quiz Review

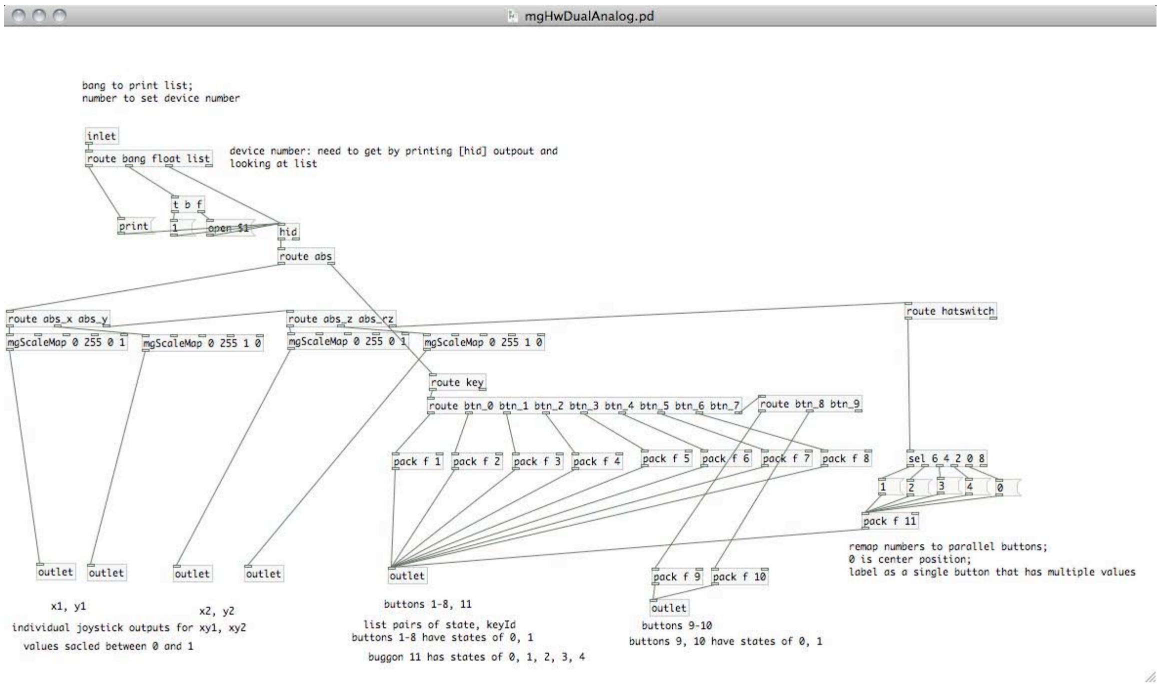
- ?

9.3. Approaching an Improvisation

- Think in simple forms: A B, A B A
- Consider approaches to transitions to new sounds or presets; alternate between two instruments or sound sources
- Embrace silence and space
- Repetition (often) establishes meaning
- Compose short ideas that can be returned to, repeated, and elaborated

9.4. Hardware Abstraction of the Dual Analog

- [mgHwDualAnalog]



bang to print list;
number to set device number

device number: need to get by printing [hid] outpout and
looking at list

remap numbers to parallel buttons;
0 is center position;
label as a single button that has multiple values

x1, y1

x2, y2

individual joystick outputs for xy1, xy2

values sacled between 0 and 1

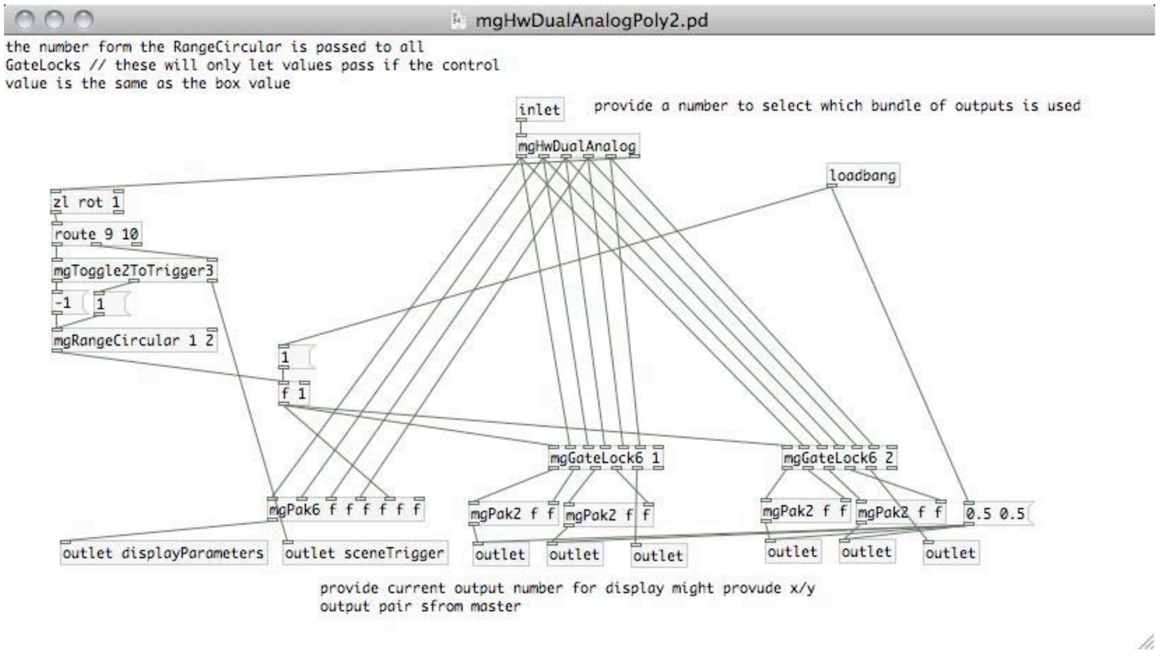
buttons 1-8, 11

list pairs of state, keyId
buttons 1-8 have states of 0, 1

buggon 11 has states of 0, 1, 2, 3, 4

buttons 9-10
buttons 9, 10 have states of 0, 1

- Polymorphic control: [mgHwDualAnalogPoly2]



the number form the RangeCircular is passed to all
GateLocks // these will only let values pass if the control
value is the same as the box value

provide a number to select which bundle of outputs is used

provide current output number for display might provude x/y
output pair sfrom master

- Use the appropriate version of [mgHwDualAnalogPoly*] for your platform/controller combination

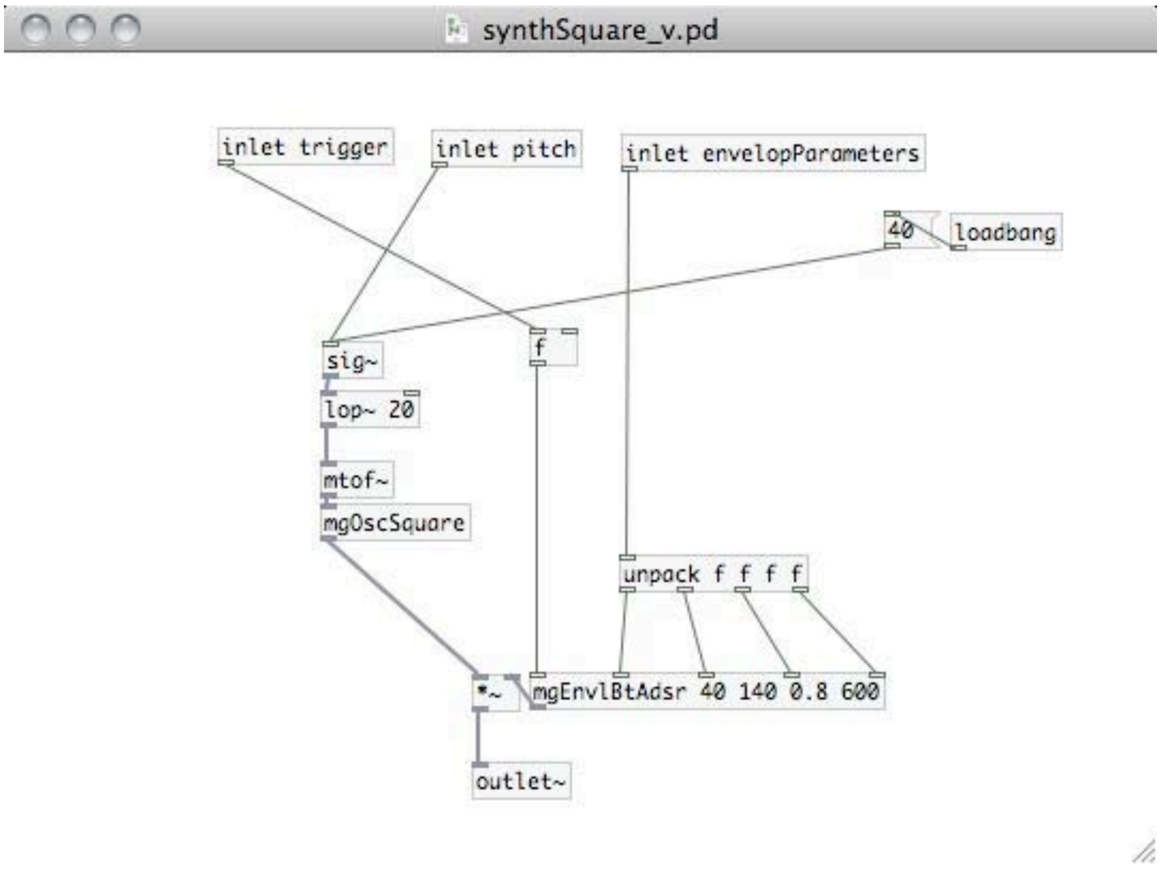
  Examples: [mgHwDualAnalogPoly2], [mgHwDualAnalogPoly2ChillStream], [mgHwDualAnalogPoly2Joystick], [mgHwDualAnalogPoly2Sixaxis]

9.5. VIPP Design Pattern: Synth Square

- Download: http://www.flexatone.net/transport/instrument01.zip

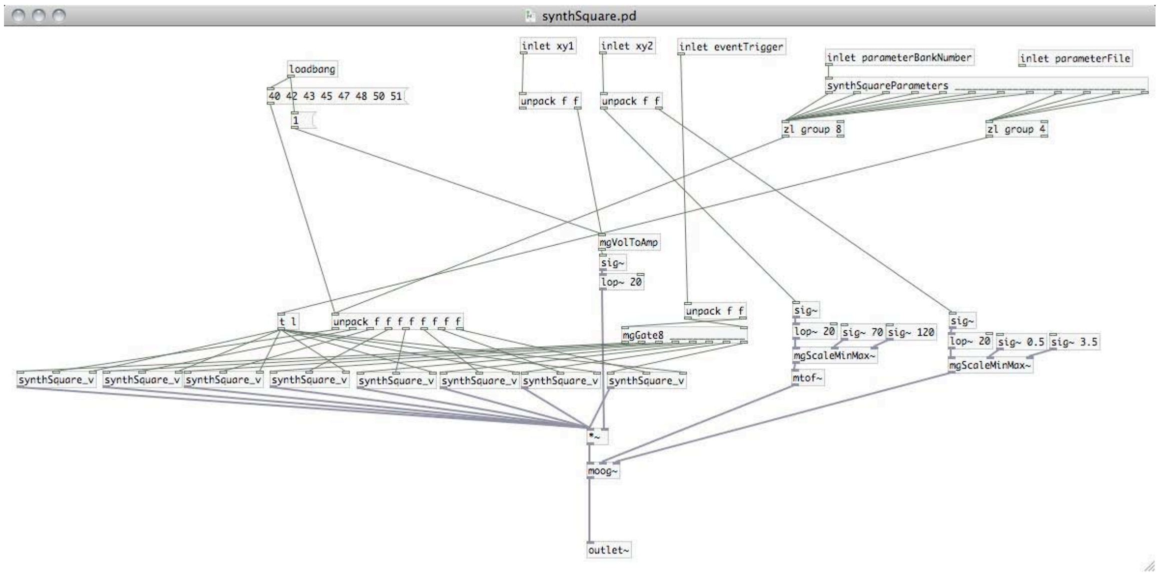
- Voice

  Independent of any implied hardware or control mapping; no stored parameters



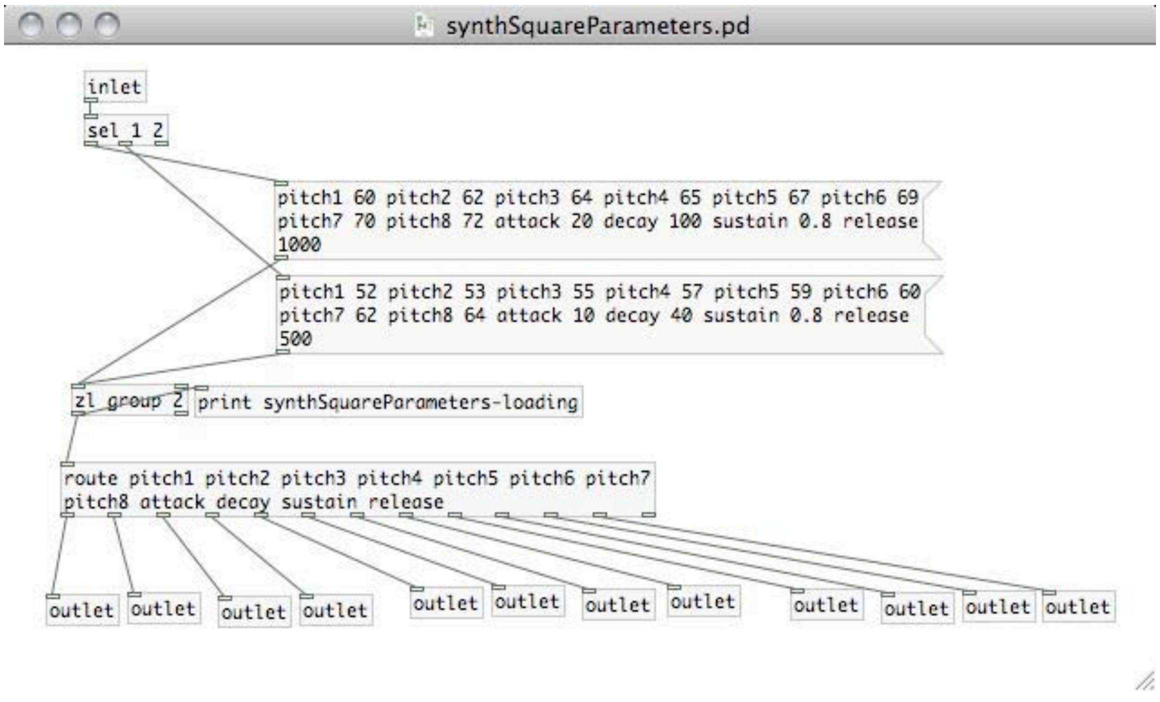
- Instrument

  Assumes basic control data types and ranges; may store parameters, may get parameters from elsewhere



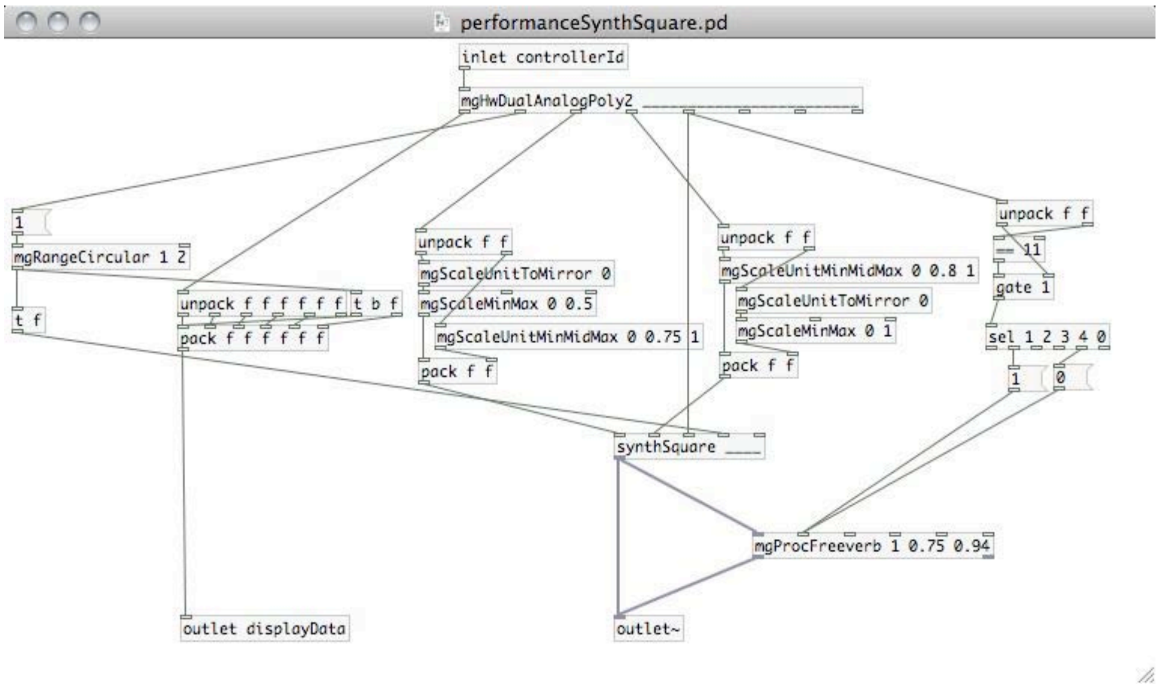
- Parameters

  Parameter storage tied directly to an instrument


synthSquareParameters.pd
inlet
sel 1 2
pitch1 60 pitch2 62 pitch3 64 pitch4 65 pitch5 67 pitch6 69 pitch7 70 pitch8 72 attack 20 decay 100 sustain 0.8 release 1000
pitch1 52 pitch2 53 pitch3 55 pitch4 57 pitch5 59 pitch6 60 pitch7 62 pitch8 64 attack 10 decay 40 sustain 0.8 release 500
zl group 2
print synthSquareParameters-loading
route pitch1 pitch2 pitch3 pitch4 pitch5 pitch6 pitch7 pitch8 attack decay sustain release
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet

- Performance

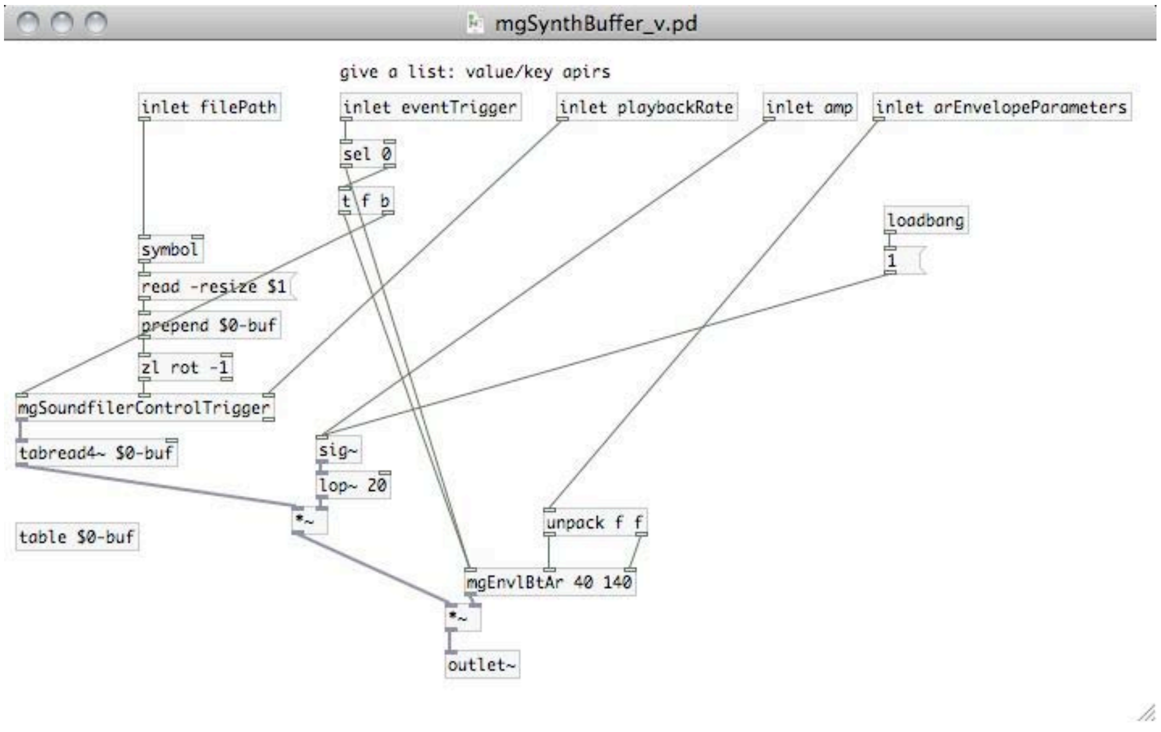
  Linking a specific hardware interface to one or more instruments; may store mapping parameters, my control distribution of parameters to instruments

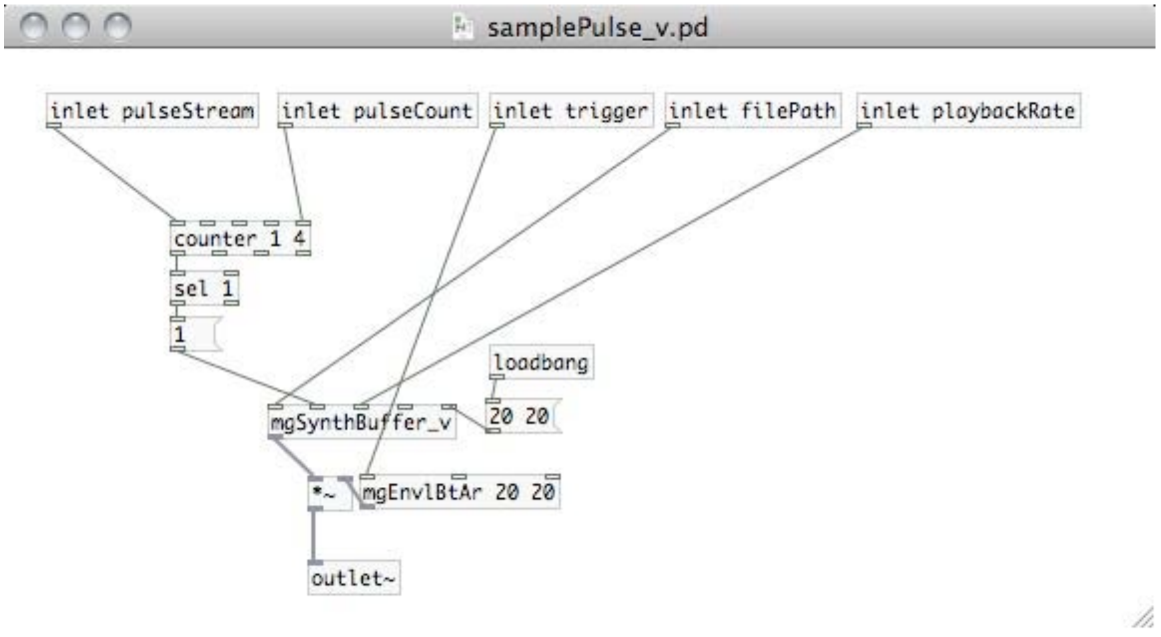


9.6. VIPP Design Pattern: Sample Pulse

- Download: http://www.flexatone.net/transport/instrument02.zip
- Voice

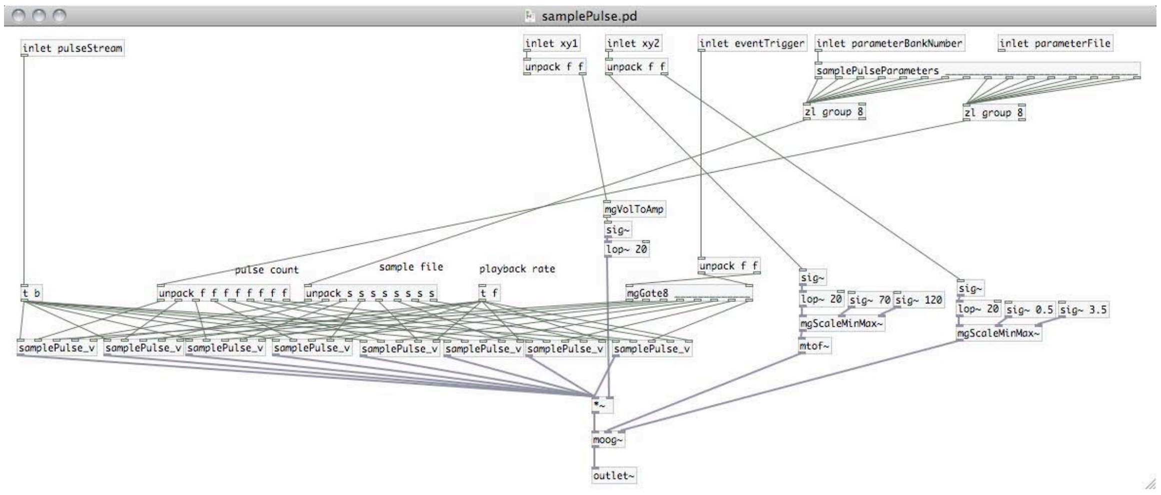
  Independent of any implied hardware or control mapping; no stored parameters





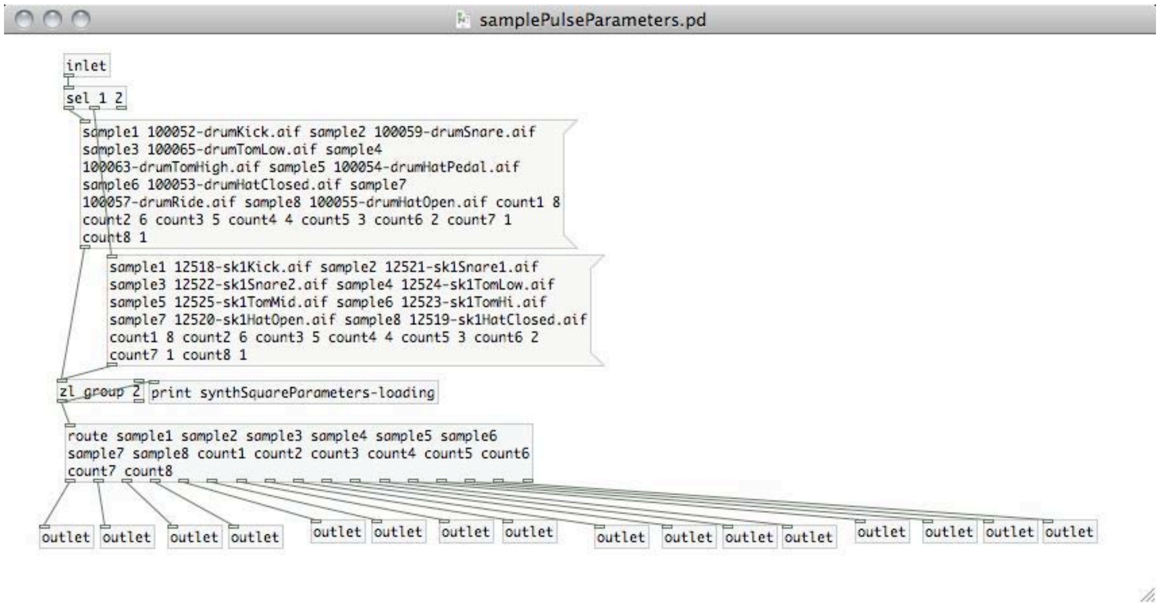
- Instrument

  Assumes basic control data types and ranges; may store parameters, may get parameters from elsewhere



- Parameters

  Parameter storage tied directly to an instrument



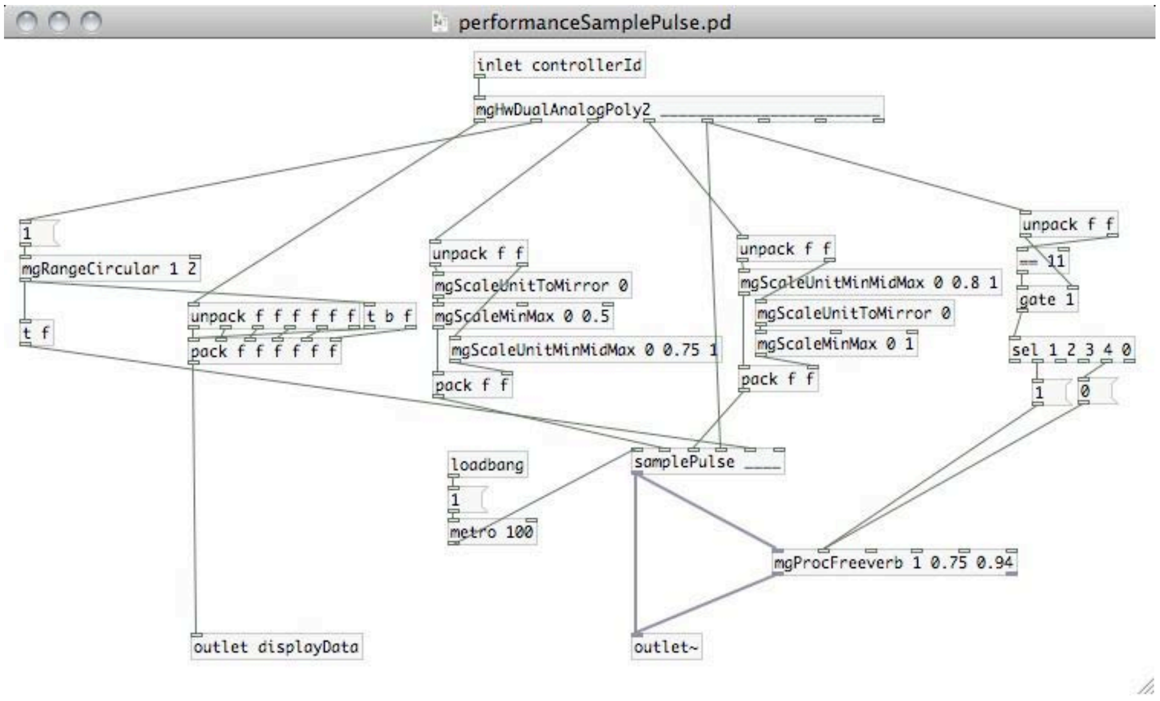
```
sample1 100052-drumKick.aif sample2 100059-drumSnare.aif
sample3 100065-drumTomLow.aif sample4
100063-drumTomHigh.aif sample5 100054-drumHatPedal.aif
sample6 100053-drumHatClosed.aif sample7
100057-drumRide.aif sample8 100055-drumHatOpen.aif count1 8
count2 6 count3 5 count4 4 count5 3 count6 2 count7 1
count8 1
```

```
sample1 12518-sk1Kick.aif sample2 12521-sk1Snare1.aif
sample3 12522-sk1Snare2.aif sample4 12524-sk1TomLow.aif
sample5 12525-sk1TomMid.aif sample6 12523-sk1TomHi.aif
sample7 12520-sk1HatOpen.aif sample8 12519-sk1HatClosed.aif
count1 8 count2 6 count3 5 count4 4 count5 3 count6 2
count7 1 count8 1
```

zl group 2
print synthSquareParameters-loading
route sample1 sample2 sample3 sample4 sample5 sample6 sample7 sample8 count1 count2 count3 count4 count5 count6 count7 count8
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet
outlet

- Performance

  Linking a specific hardware interface to one or more instruments; may store mapping parameters, my control distribution of parameters to instruments



9.7. Controller/Interface/Instrument Design 1 Draft

- Can be a new synthesis instrument of some sort
- Can be an extension, modification, or transformation of existing martingale instruments or sample instruments
- Need a model that makes sound with dual analog control: need not have full control

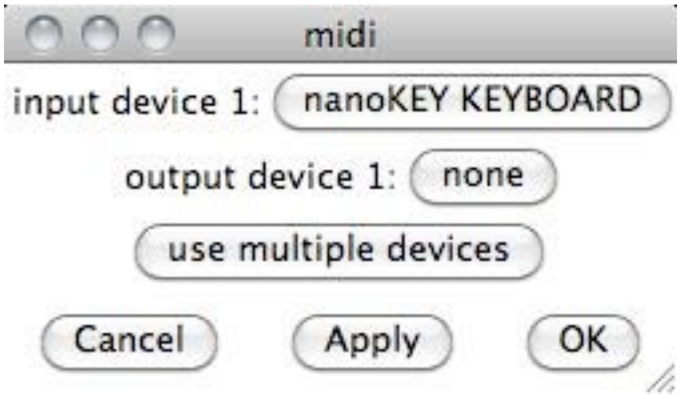
9.8. Common Controllers

- Many common controllers transmit MIDI
- Keyboards
- Combinations of buttons (triggers and toggles) and sliders
- Touchpads, ribbons, and other continuous controllers
- Turntables

9.9. MIDI Messages

- Can be thought of as messages encoded as pairs or triples of data

- Generally, first data element is type of message, second data elements is value
- Operating system is generally responsible for representing MIDI devices to the software
- Most modern MIDI devices communicate MIDI over USB
- Pd MIDI configuration: need to select input device

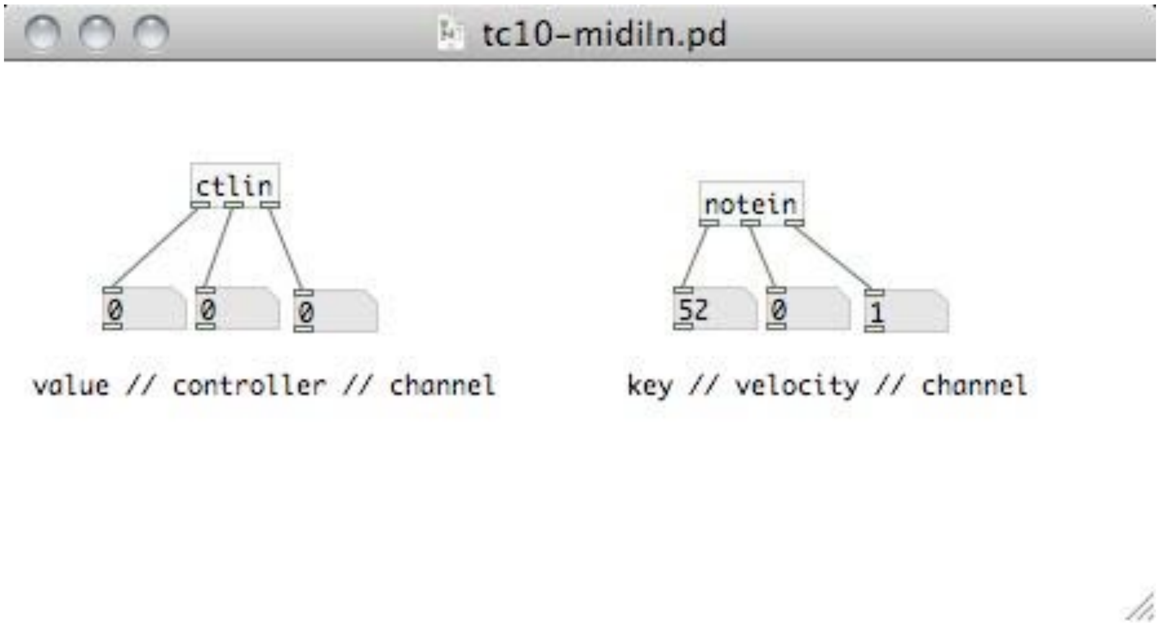


input device 1: nanoKEY KEYBOARD

output device 1: none

use multiple devices

- Basic objects that provide MIDI input: notein [notein] and control in [ctlin]



```
ctlin                              notein
0   0   0                          52   0   1
value // controller // channel     key // velocity // channel
```

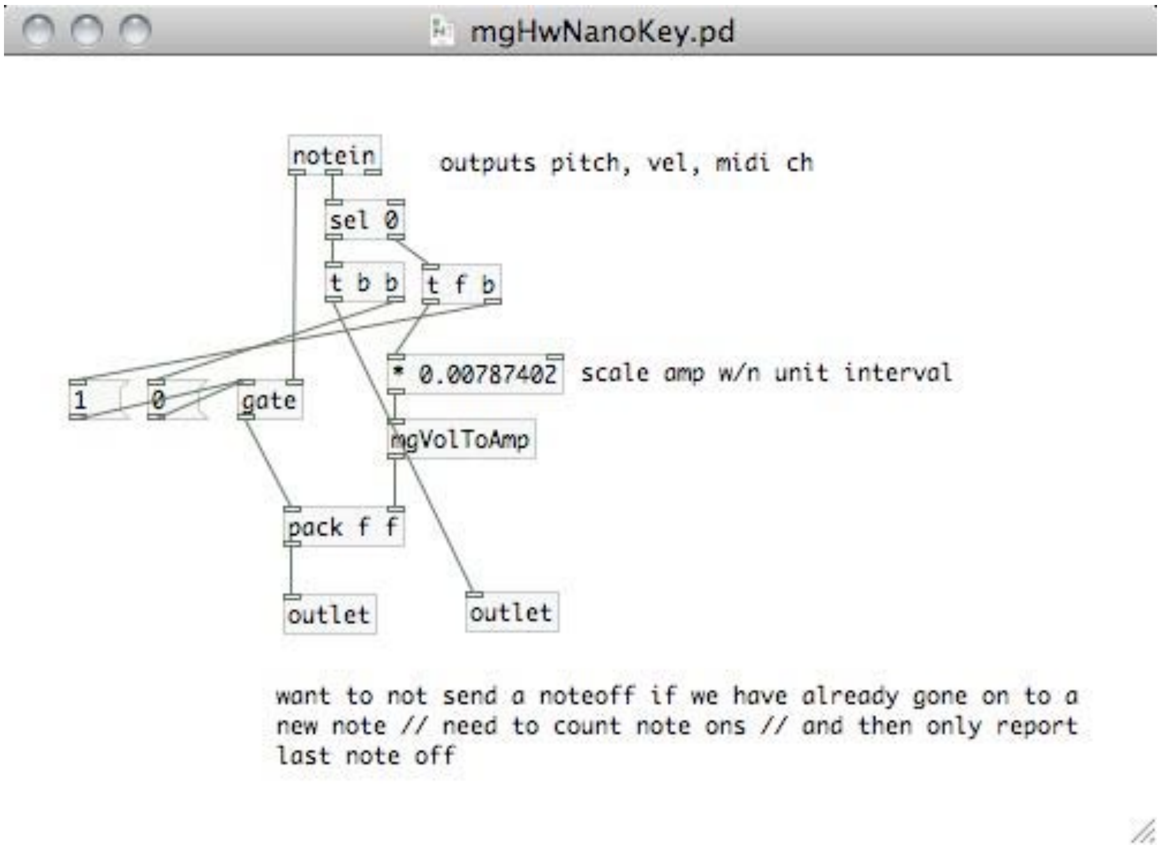
- Values generally in the range of 0 to 127

9.10. MIDI Keyboards

- Transmits note-on messages received from [notein]
- Korg NanoKey

- Akai LPK 25

- [mgHwNanoKey]: abstraction of note-in functionality

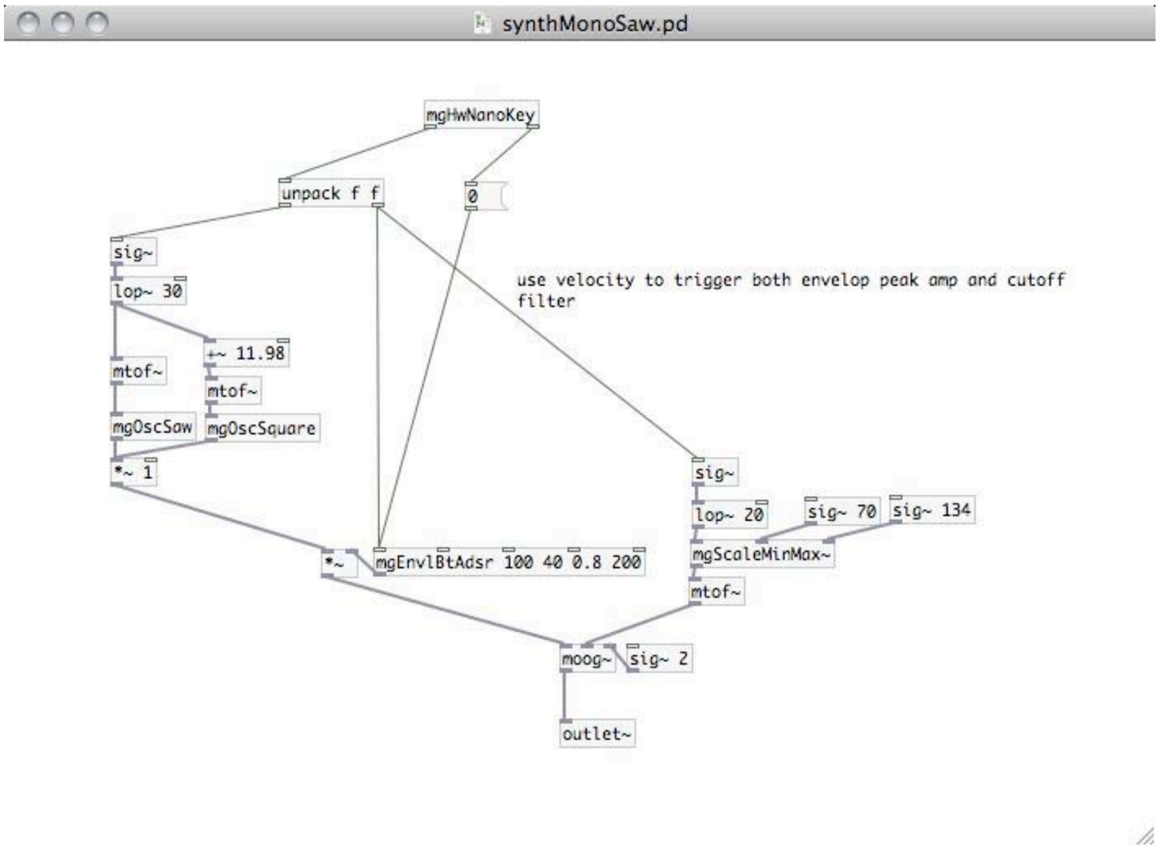


outputs pitch, vel, midi ch

scale amp w/n unit interval

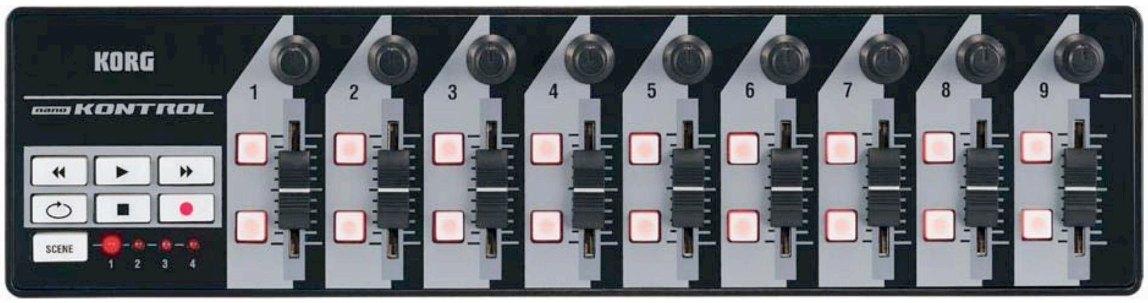
want to not send a noteoff if we have already gone on to a new note // need to count note ons // and then only report last note off

- A simple monophonic synth: martingale/instruments/nanoKey

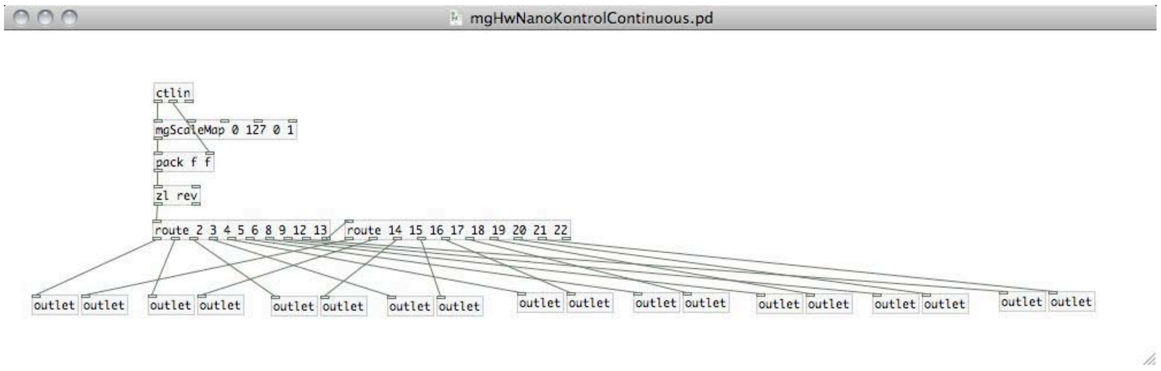


9.11. MIDI Sliders and Knobs

- Up to 128 continuous controllers can be used to send data values between 0 and 127
- Korg NanoKontrol



- [mgHwNanoKontrolContinuous] and [mgHwNanoKontrolDiscrete]


mgHwNanoKontrolContinuous.pd
ctlin
mgScaleMap 0 127 0 1
pack f f
zl rev
route 2 3 4 5 6 8 9 12 13
route 14 15 16 17 18 19 20 21 22
outlet

9.12. MIDI Pads and Touch Controls

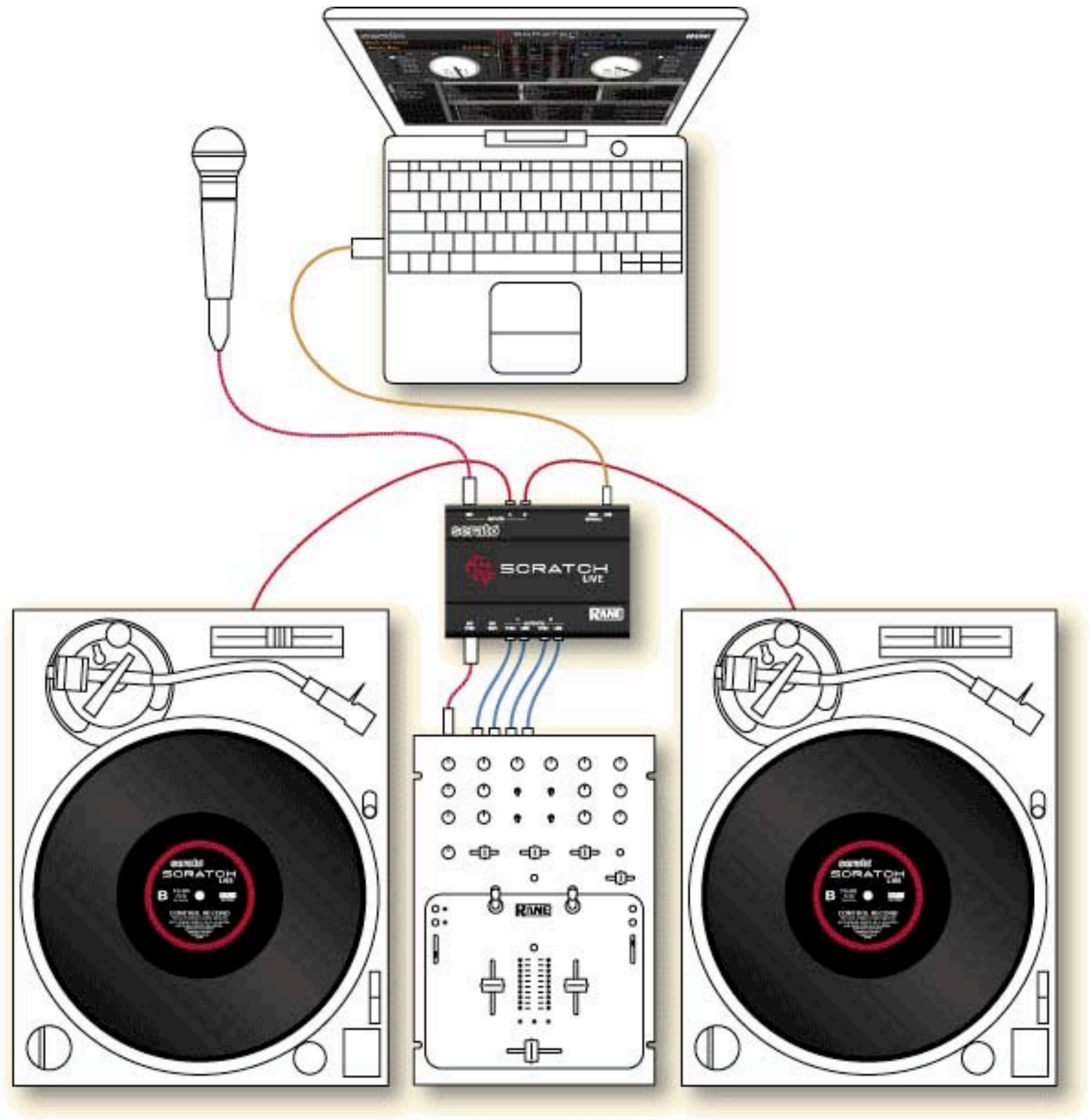
- Pads were made popular by early drum machines and samplers such as the Linn drum and Akai MPC
- Some pad controllers provide aftertouch: dynamic control of pressure on each pad
- Korg NanoPad: provides velocity sensitive pad controls and XY touch pad

- Akai LPD 8

- Akai MPD 18

9.13. Turntables and Other Controllers

- Numerous approaches to making the turntable into a computer controller
- Rane/Serato scratch live



9.14. Listening: D-Styles and Kid Koala

- Creative and musical extensions of the turntable
- D-Styles, “Felonius Funk,” *Scratchology*, 2003
- Kid Koala, “Like irregular Chickens,” *Carpal Tunnel Syndrome*, 2000

9.15. Reading: Perkis, Some Notes on My Electronic Improvisation Practices

- Perkis, T. 2009. "Some Notes on My Electronic Improvisation Practices." In R. T. Dean, ed. *The Oxford Handbook of Computer Music.* Oxford University Press, pp. 161-166.
- Improvising and playing with acoustic musicans led Perkis to adopt a few important strategies: what are they?
- What inhibits many players from really knowing how to play their computer-music instruments? What is gained from really knowing how to play your instrument?
- Does Perkis directly create musical events, or create macro events, or something in between?
- What role does the wah-wah perform in Perkis's setup?

9.16. Reading: Fiebrink, Wang, and Cook, Don't Forget the Laptop

- Fiebrink, R. and G. Wang, P. Cook. 2007. "Don't Forget the Laptop: Using Native Input Capabilities for Expressive Musical Control." *Proceedings of the Conference on New Interfaces for Musical Expression* pp. 164-167.
- What are some novel control options presented in this paper?
- The authors claim that these approach offer portability: but do they?
- What are some possible applications of integrated webcams and microphones for musical control?

Chapter 10. Meeting 10, Workshop: Performance and Improvisation

10.1. Announcements

- Due Wednesday, 16 March: Controller/Interface/Instrument Design 1 Report

  Will accept as late as midnight Friday, 18 March

  Must submit code

  See syllabus for report details

- Martingale poly/performance-d coming soon: please test

- If you want to meet with me to talk about your instrument, or review code, please do so by or before this weekend

10.2. Reading: Kuivila, Open Sources

- Kuivila, R. 2004. "Open Sources: Words, Circuits and the Notation-Realization in the Music of David Tudor." *Leonardo Music Journal* 14: pp. 17-23.

- What precedents do we see in Tudor's work with Cage's *Cartridge Music* (1960) and his own *Bandoneon!* (1966)?

- What does it mean to "to compose notations that circumscribed a field of musical possibility out of which an unrepeatable stream of unique sounds and actions could emerge"

- What led to the Rainforest series of works being so popular and well known?

- Some modern live electronics artists perform on "no-input mixer"; how does this relate practices of Tudor?

- Why did Tudor move away from the piano?

10.3. Reading: Driscoll and Rogalsky, David Tudor's Rainforest

- Driscoll, J. and M. Rogalsky. 2004. "David Tudor's 'Rainforest': An Evolving Exploration of Resonance." *Leonardo Music Journal* 14: pp. 25-30.

- Was Rainforest one work? What does this say about changing ideas of the work concept?

- What was the basic technical mechanism for Rainforest works? How did composers refine their instruments and sculptures?
- What were some of the sources used for resonating the sculptures?
- How big was Rainforest IV?
- How did a work like Rainforest IV support collaborative composition?
- How did a work like Rainforest IV support new experiences for the audience?

10.4. Listening: David Tudor

- Listening: David Tudor, *Rainforest*, 1973
- Listening: David Tudor, *Pulsers*, 1976
- Listening: David Tudor, *Toneburst*, 1975

10.5. Exercise: Smooth Fades

- Load: poly performance-c.test.pd

  martingale/pd/instruments/dualAnalog*/poly/performance-c.test.pd
- Ensemble: instrument 1: fade in over 8 counts, fade out over 8 counts
- Variation: in over 4, out over 4 + 8

Variation: in over 8 + 4, out over 4

10.6. Exercise: Chord Sequence Pulses

- Load: poly performance-c.test.pd

  martingale/pd/instruments/dualAnalog*/poly/performance-c.test.pd

- Ensemble: scene 2, instrument 3: articulate the following button values as a chord sequence, where each chord lasts 4 beats

  Button chords: 1, 5 // 3, 8 // 1, 5 // 2, 7 // 4, 6 // 1, 5 ||

- Variation: two players take noise solos with instrument 1

10.7. Prototype Instruments

- Sonic vision
- Control ambitions

10.8. Instrument C Solos

- ?

Chapter 11. Meeting 11, Practices: Touch Interfaces and OpenSoundControl

11.1. Announcements

- Due Wednesday, 16 March: Controller/Interface/Instrument Design 1 Report

  Will accept as late as midnight Friday, 18 March

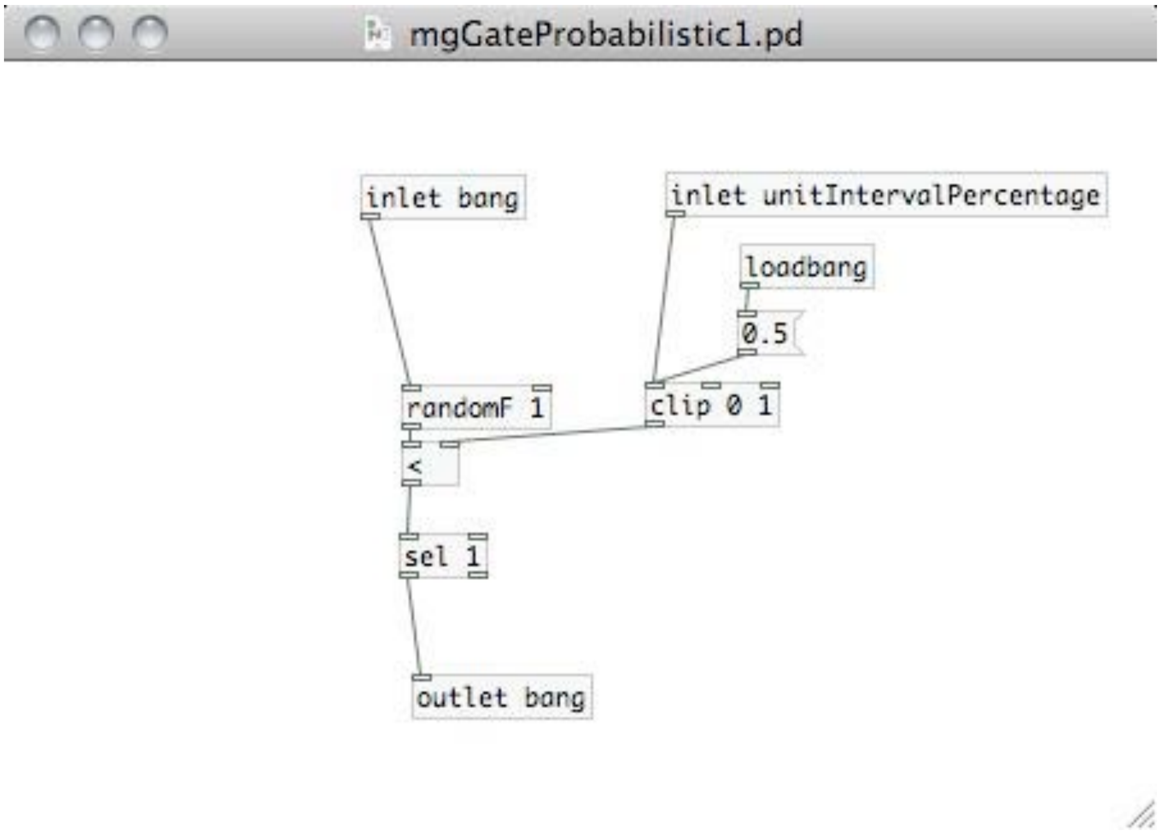
  Must submit code and report

  See syllabus for report details

- Begin exploring instruments in Martingale poly/performance-d

11.2. Using Probabilistic and Random Control

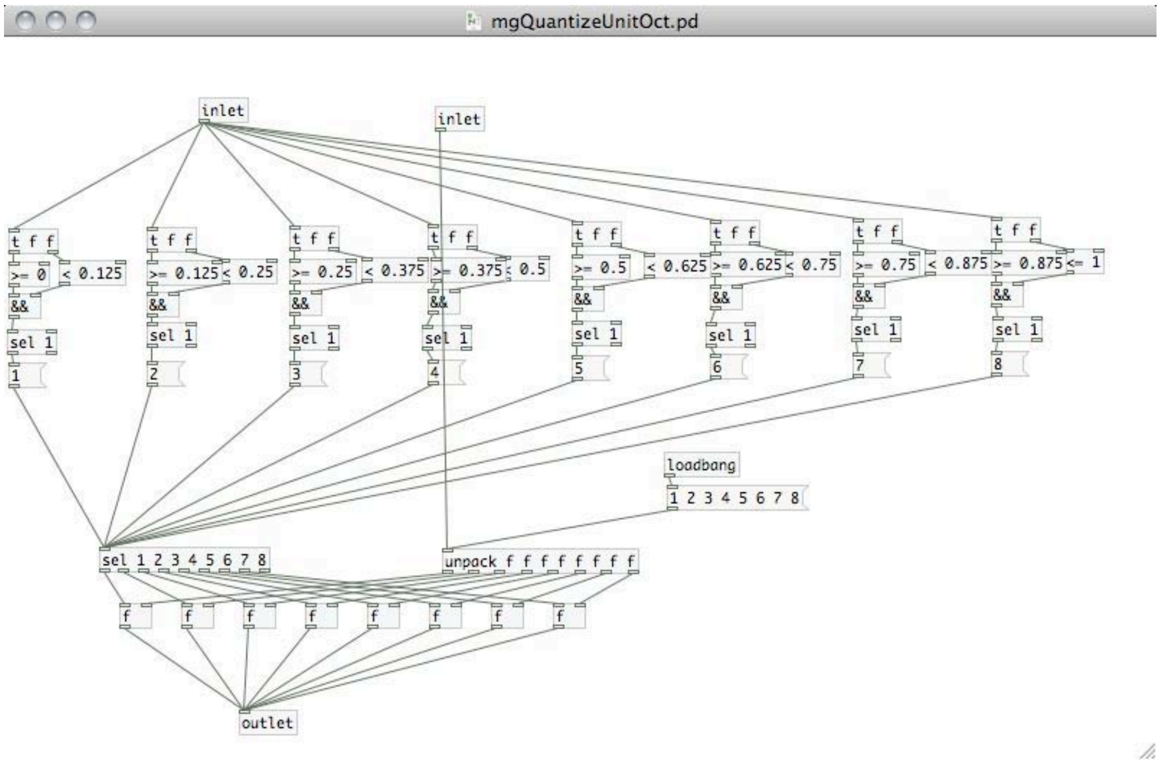
- Can use randomness to select items
- Can use randomness to limit or filter items
- Varying the intensity of random influence becomes an expressive parameter
- [mgGateProbabilistic1]



- What would mgGateProbabilistic3.pd do?

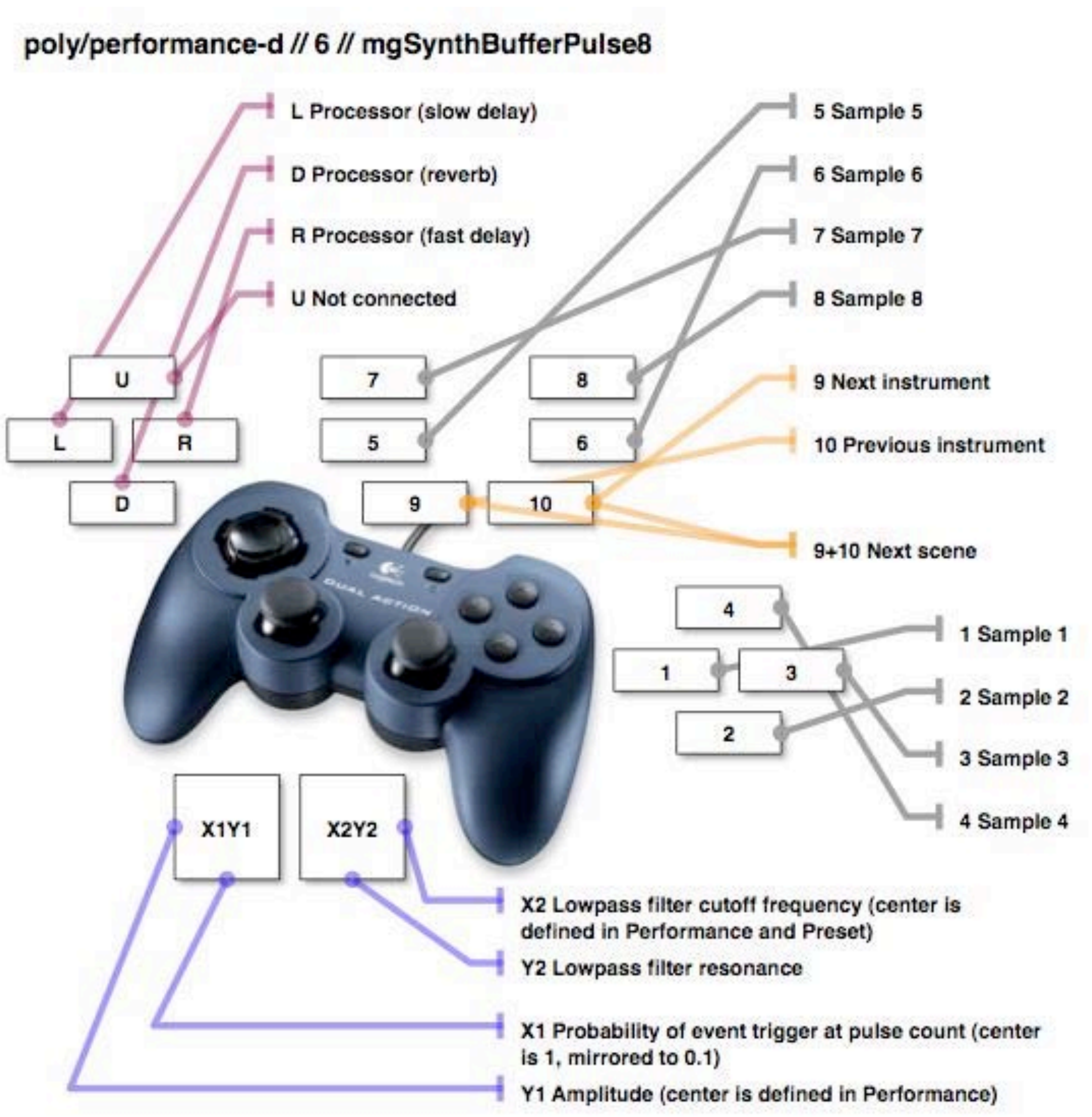
11.3. Making Continuous Values Discrete

- [mgQuantizeUnitOct]

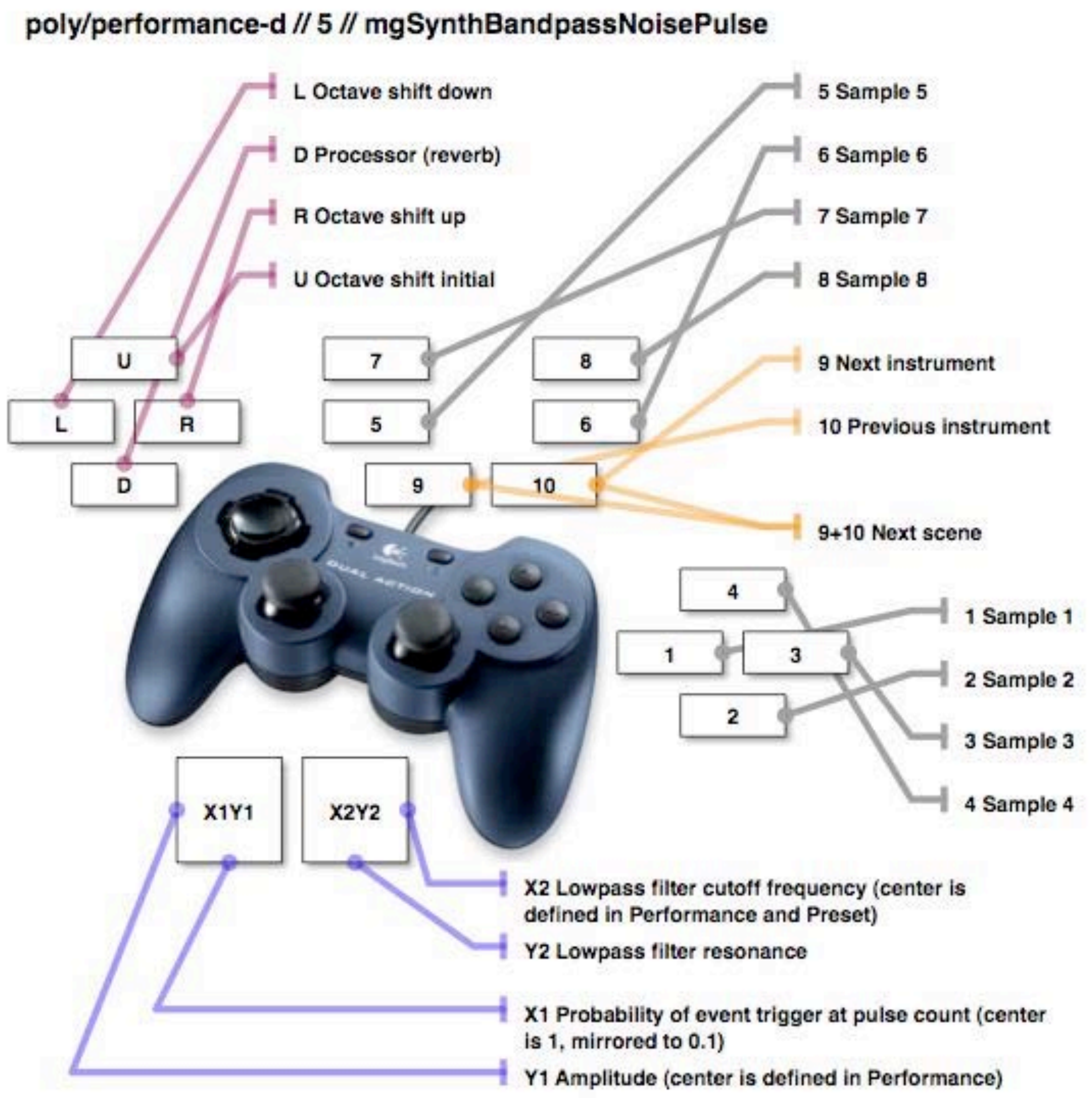


11.4. Poly Performance D: Pulse Driven with Probabilistic Events

- Instrument 6: mgSynthBufferPulse8

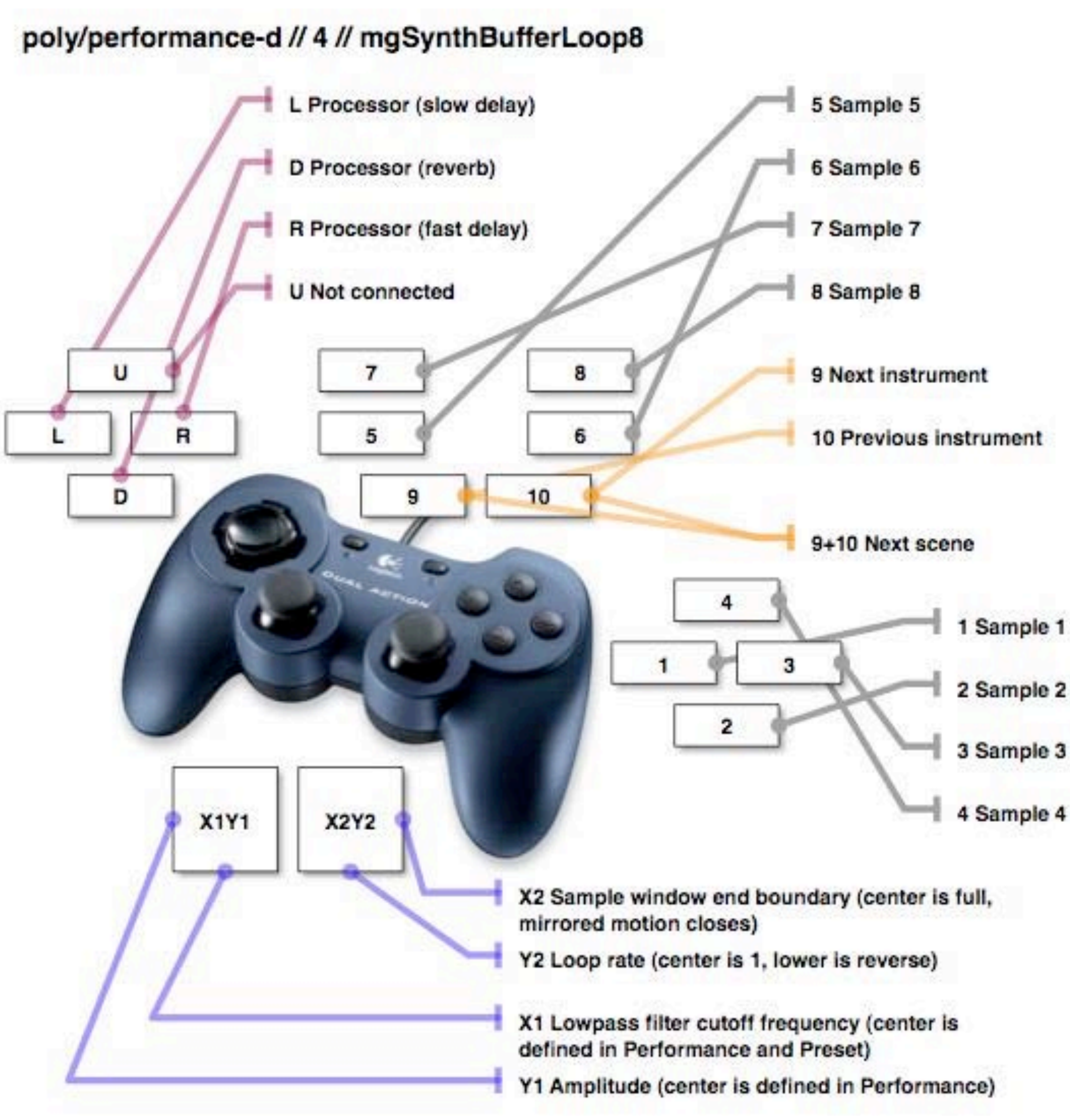

poly/performance-d // 6 // mgSynthBufferPulse8
L Processor (slow delay)
D Processor (reverb)
R Processor (fast delay)
U Not connected
5 Sample 5
6 Sample 6
7 Sample 7
8 Sample 8
U
L
R
D
7
5
8
6
9
10
9 Next instrument
10 Previous instrument
9+10 Next scene
4
1
3
2
1 Sample 1
2 Sample 2
3 Sample 3
4 Sample 4
X1Y1
X2Y2
X2 Lowpass filter cutoff frequency (center is defined in Performance and Preset)
Y2 Lowpass filter resonance
X1 Probability of event trigger at pulse count (center is 1, mirrored to 0.1)
Y1 Amplitude (center is defined in Performance)

- Instrument 5: mgSynthBufferPulse8


poly/performance-d // 5 // mgSynthBandpassNoisePulse
L Octave shift down
D Processor (reverb)
R Octave shift up
U Octave shift initial
5 Sample 5
6 Sample 6
7 Sample 7
8 Sample 8
U
L
R
D
7
8
5
6
9
10
9 Next instrument
10 Previous instrument
9+10 Next scene
4
1
3
2
1 Sample 1
2 Sample 2
3 Sample 3
4 Sample 4
X1Y1
X2Y2
X2 Lowpass filter cutoff frequency (center is defined in Performance and Preset)
Y2 Lowpass filter resonance
X1 Probability of event trigger at pulse count (center is 1, mirrored to 0.1)
Y1 Amplitude (center is defined in Performance)

11.5. Poly Performance D: Looping Buffer with Dynamic Playback Rate and Window Size

- Instrument 4: mgSynthBufferLoop8


poly/performance-d // 4 // mgSynthBufferLoop8
L Processor (slow delay)
D Processor (reverb)
R Processor (fast delay)
U Not connected
5 Sample 5
6 Sample 6
7 Sample 7
8 Sample 8
9 Next instrument
10 Previous instrument
9+10 Next scene
1 Sample 1
2 Sample 2
3 Sample 3
4 Sample 4
U
L
R
D
7
8
5
6
9
10
4
1
3
2
X1Y1
X2Y2
X2 Sample window end boundary (center is full, mirrored motion closes)
Y2 Loop rate (center is 1, lower is reverse)
X1 Lowpass filter cutoff frequency (center is defined in Performance and Preset)
Y1 Amplitude (center is defined in Performance)

11.6. MIDI Devices

- Review Meeting 9

11.7. Reading: Wright, Open Sound Control

- Wright, M. 2005. "Open Sound Control: an enabling technology for musical networking." *Organised Sound* 10(3): pp. 193-200.
- The author criticizes many existing roles of network technology in music: what are his criticisms?
- What are some of the design goals of OSC?

- What advantages does OSC have over MIDI?
- What are the benefits and disadvantages of symbolic parameter names
- What are some shortcomings of OSC?
- What is the difference between jitter and latency? Which is acceptable for musical controllers and why?

11.8. OSC Devices

- Numerous hardware and software devices send and receive OSC
- TouchOSC

  http://hexler.net/software/touchosc
- MRMR

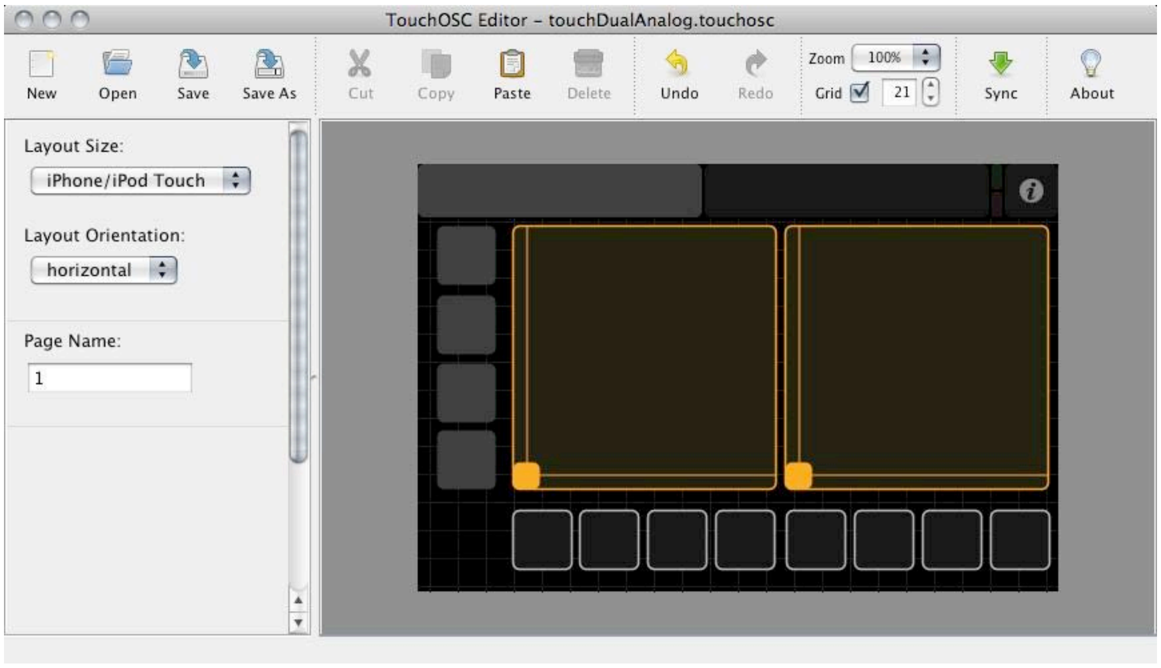
  http://mrmr.noisepages.com

11.9. TouchOSC

- Available for Android and iPhone/iPad

  http://hexler.net/software/touchosc

  http://hexler.net/software/touchosc-android
- Desktop editor permits designing custom interfaces



Layout Size: iPhone/iPod Touch

Layout Orientation: horizontal

Page Name: 1

Courtesy of Hexler. Used with permission.

- Basic interfaces controls:
  - Buttons: push, toggle, multi-toggle
  - Faders: linear, multi-linear, and rotary
  - XY pad
- Basic OSC encoding
  - Each pane defines a top-level OSC division: /1, /2
  - Each control is named and defines a second level: /toggle2, /push3
  - Some controls define a third level: /multifader1/4, /multifader1/6
  - Following the message name is value (unit interval ranges are the best)

11.10. Setting Up TouchOSC

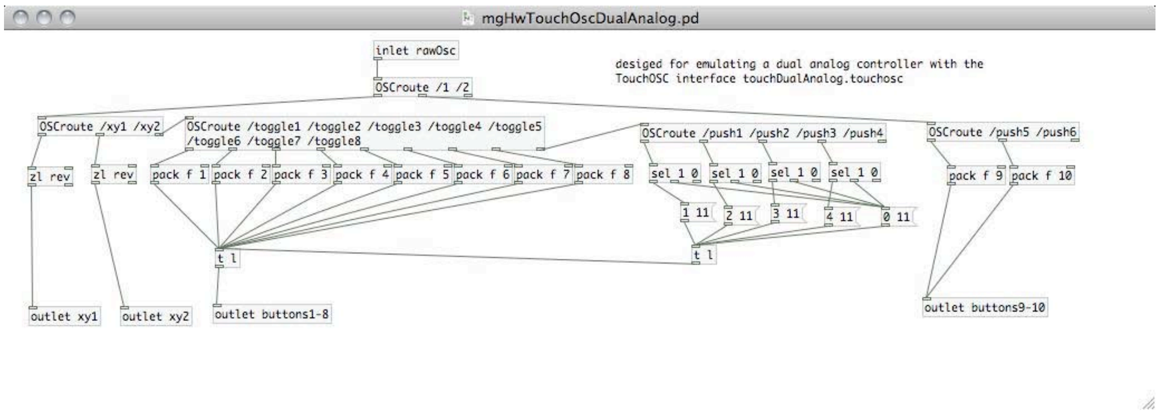
- Create wifi/Airport network with a manual IP address: 192.168.2.1 (alternatives are fine)
- Subnet mask: 255.255.255.0
- Join network with mobile device
- With touchOSC on mobile device, enter IP address as host

- Set outgoing port to 8000 and incoming port to 9000 (alternatives are fine)

11.11. Getting and Processing OSC Values

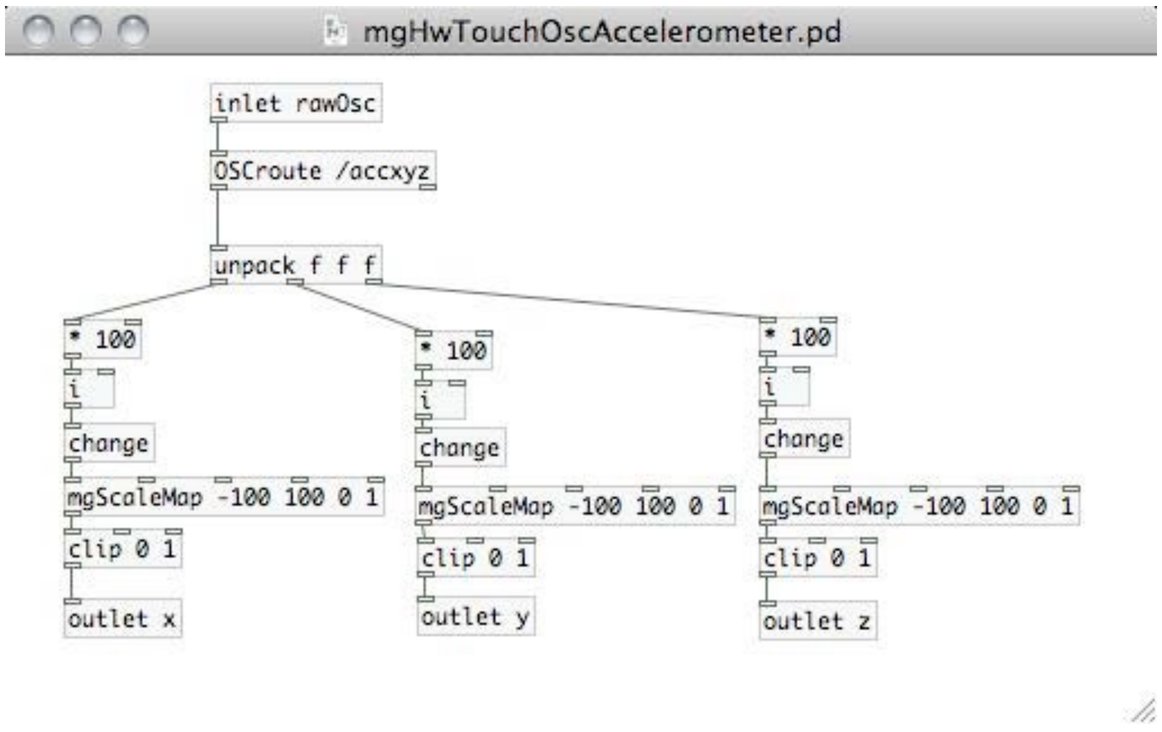
- [dumpOSC 8000]: receives raw OSC values on port 8000
- Can only have one instance of [dumpOSC 8000]; must have argument for port number
- Use [OSCroute] to parse hierarchical structure
- Example: [mgHwTouchOscDualAnalg]

  Emulate dual-analog style control with TouchOSC



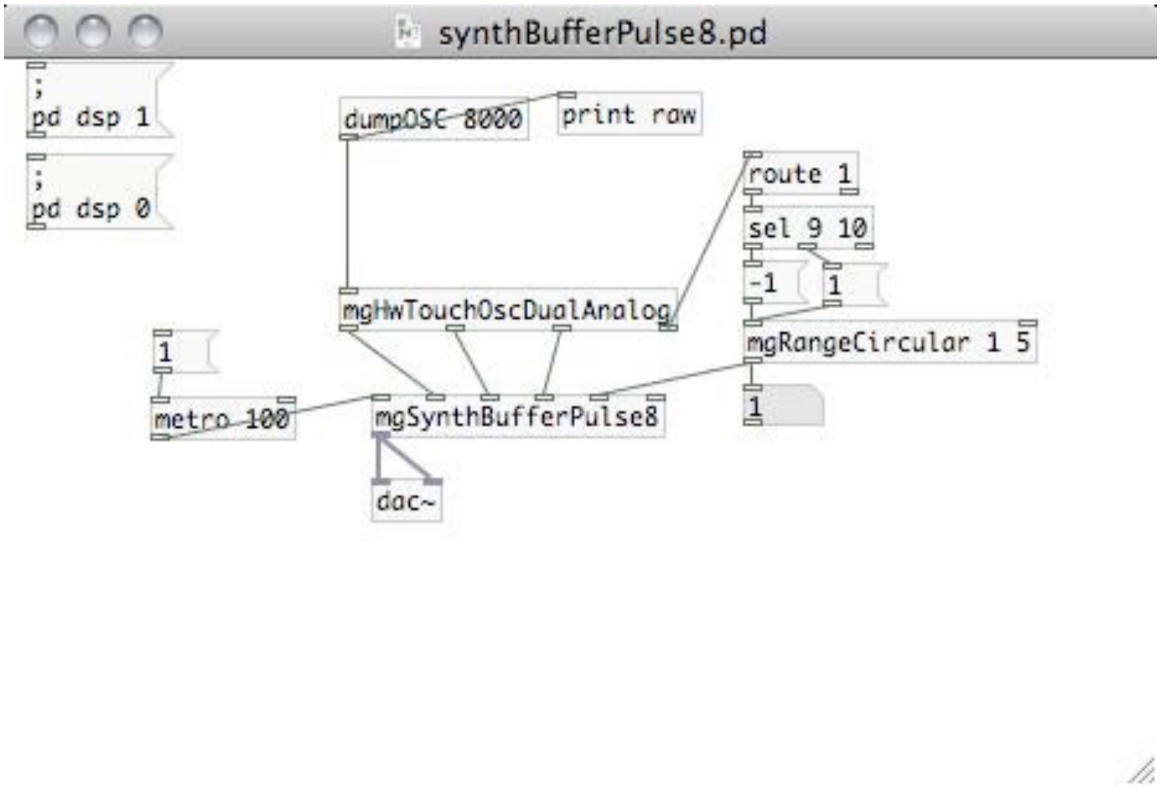
- Example: [mgHwTouchOscAccelerometer]

  Unpack round, filter, scale, and limit accelerometer data



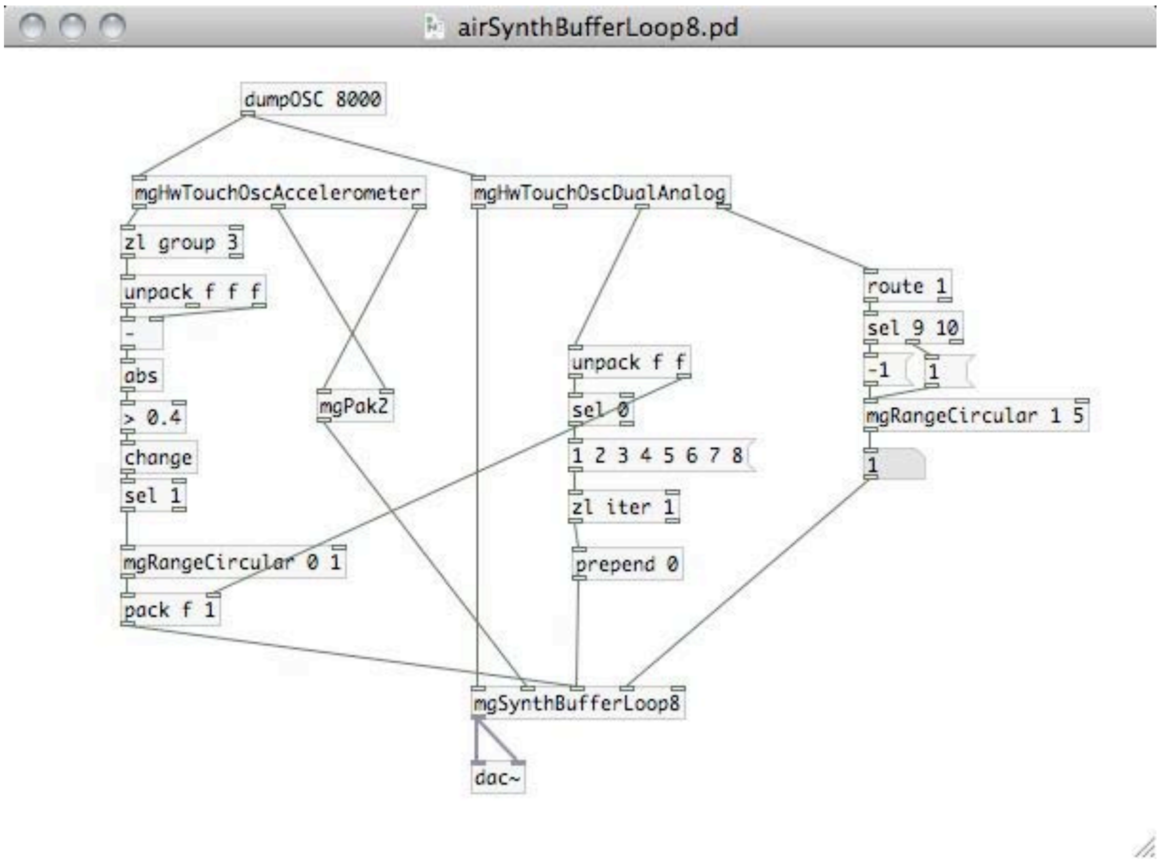
11.12. TouchOSC Instruments Emulating Dual Analog Instruments

- instruments/touchosc/mono/synthNoiseFilter.pd
- instruments/touchosc/mono/synthSaw.pd
- instruments/touchosc/mono/synthBufferPulse8.pd



11.13. TouchOSC Instruments Employing Accelerometer Data

- instruments/touchosc/mono/airSynthBufferLoop8.pd



11.14. Listening: Ryoji Ikeda

- Listening: Ryoji Ikeda, “data.syntax,” Dataplex, 2006

- Listening: Ryoji Ikeda, “data.reflex,” Dataplex, 2006

Chapter 12. Meeting 12, Practices: Laptops and Laptop Orchestras

12.1. Announcements

- Due Wednesday, 16 March: Controller/Interface/Instrument Design 1 Report

  Will accept as late as midnight Friday, 18 March

  Must submit code and report

  See syllabus for report details

- Remember to add Pd tests!
- Upcoming readings in Collins book
- Quiz next class: keywords: VIPPD, Tudor, Rainforest, Perkis, OSC, Plork
- Bring controllers and amps to next class

12.2. Reading: Smallwood, Trueman, Cook, and Wang: Composing for Laptop Orchestra

- Smallwood, S. and D. Trueman, P. R. Cook, G. Wang. 2008. "Composing for Laptop Orchestra." *Computer Music Journal* 32(1): pp. 9-25.
- Does PLork have a specific or singular aesthetic sensibility
- What are some of the practical and technical limitations of this ensemble?
- What performance interfaces are described?
- What are some paradigms of control given to the performers?
- What are some paradigms of control given to the conductors?
- Why do the author's suggest that this ensemble requires more time than a conventional orchestra?
- Does PLork achieve the stated goal of being an open source compositional and technical community?

12.3. Laptop Orchestras

- PlOrk was not the first, but the most well-funded and promoted in the last ten years
- Similar *Ork imitators elsewhere

12.4. Laptop Orchestras: Works

- Smallwood: a breeze brings...

  link (http://plork.cs.princeton.edu/listen/green/breeze.mp3)
- Wang: Clix

  link (http://plork.cs.princeton.edu/listen/green/clix.mp3)
- Oliveros and Polzin: Murphy Mixup: Murphy Intends

  link (http://plork.cs.princeton.edu/listen/green/murphy.mp3)
- Smallwood and Wang: ChucK ChucK Rocket

  link (http://plork.cs.princeton.edu/listen/green/ccr.mp3)
- Smallwood: The Future of Fun

  link (http://plork.cs.princeton.edu/listen/green/fof1983.mp3)
- Documentary on PLOrk: 4:22, 6:04

  link (http://www.youtube.com/watch?v=EO1rA3ewgHY)

12.5. Listening: John Zorn

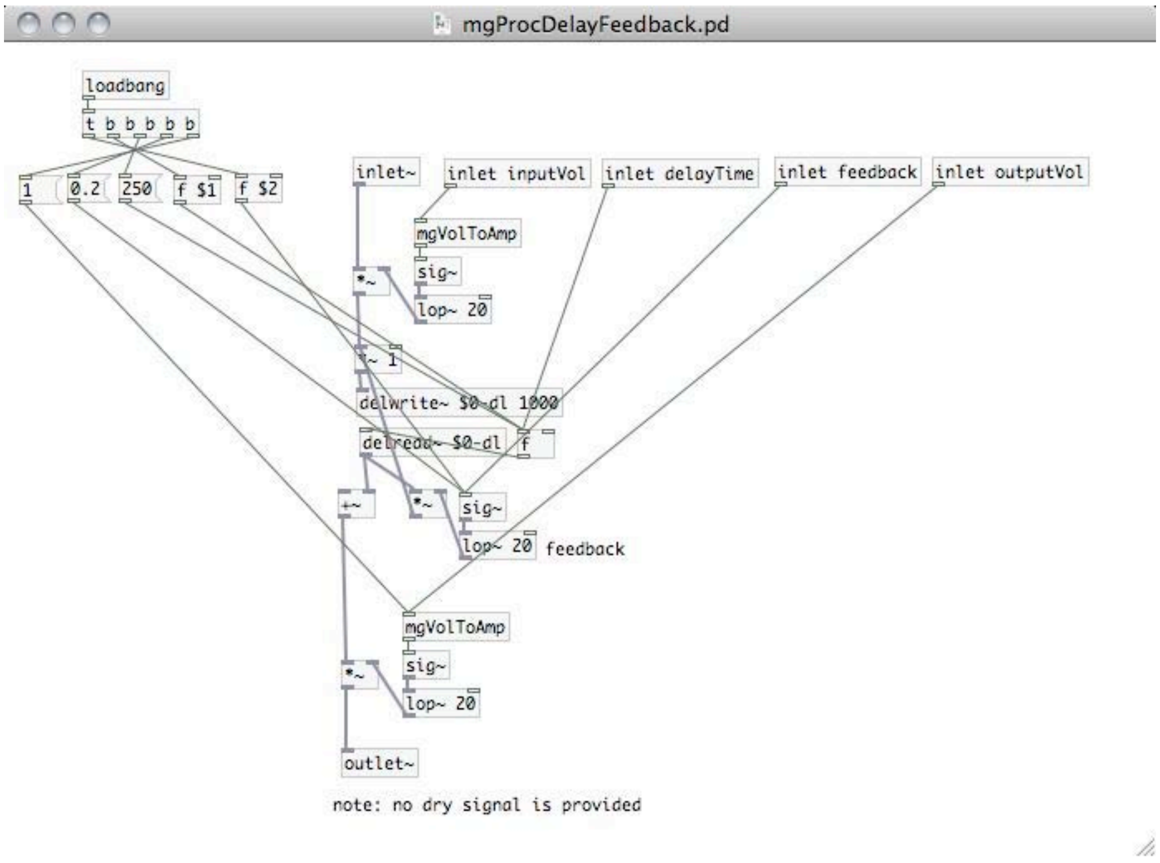
- Listening: John Zorn, “Uluwati,” Cobra: John Zorn’s Game Pieces Vol. 2, 2002
- Listening: John Zorn, “Tamangiri,” Cobra: John Zorn’s Game Pieces Vol. 2, 2002

12.6. Signal Generators and Transformers

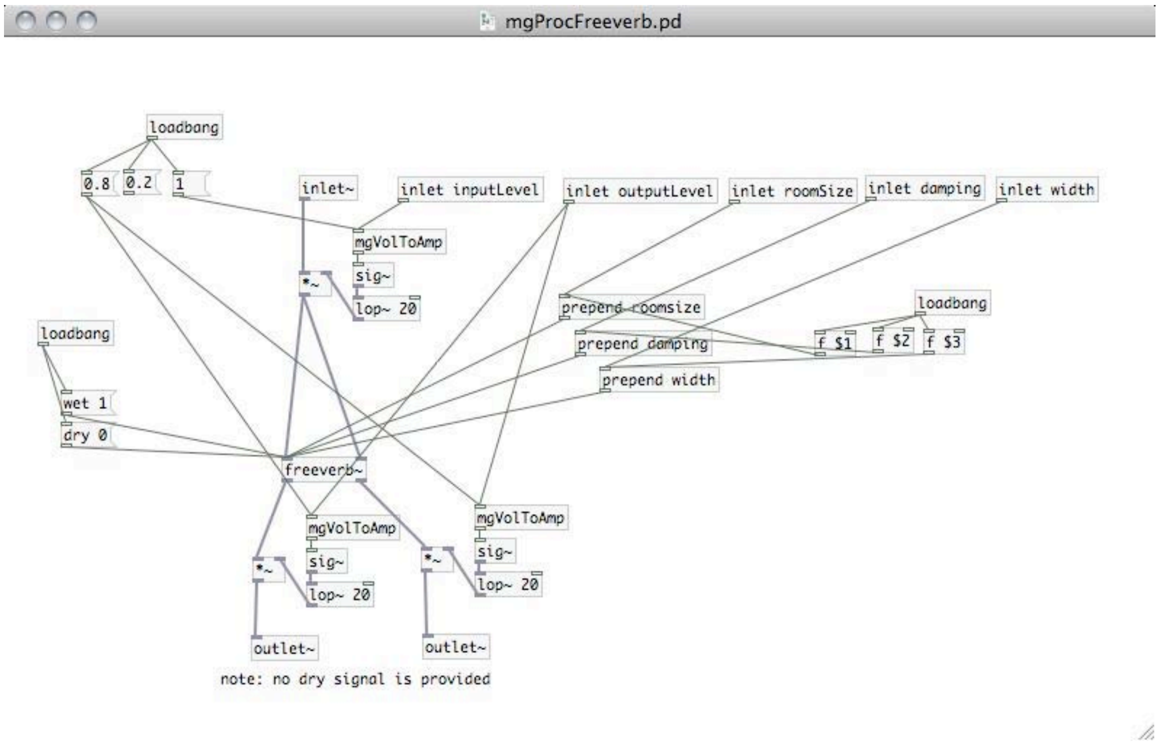
- Two areas of emphasis: signal generation or transformation
- Simple generators can be made powerful with interesting transformers

12.7. General Purpose Signal Processors

- [mgProc...] abstractions analogous to [mgSynth...] abstractions
- Example: [mgProcDelayFeedback]



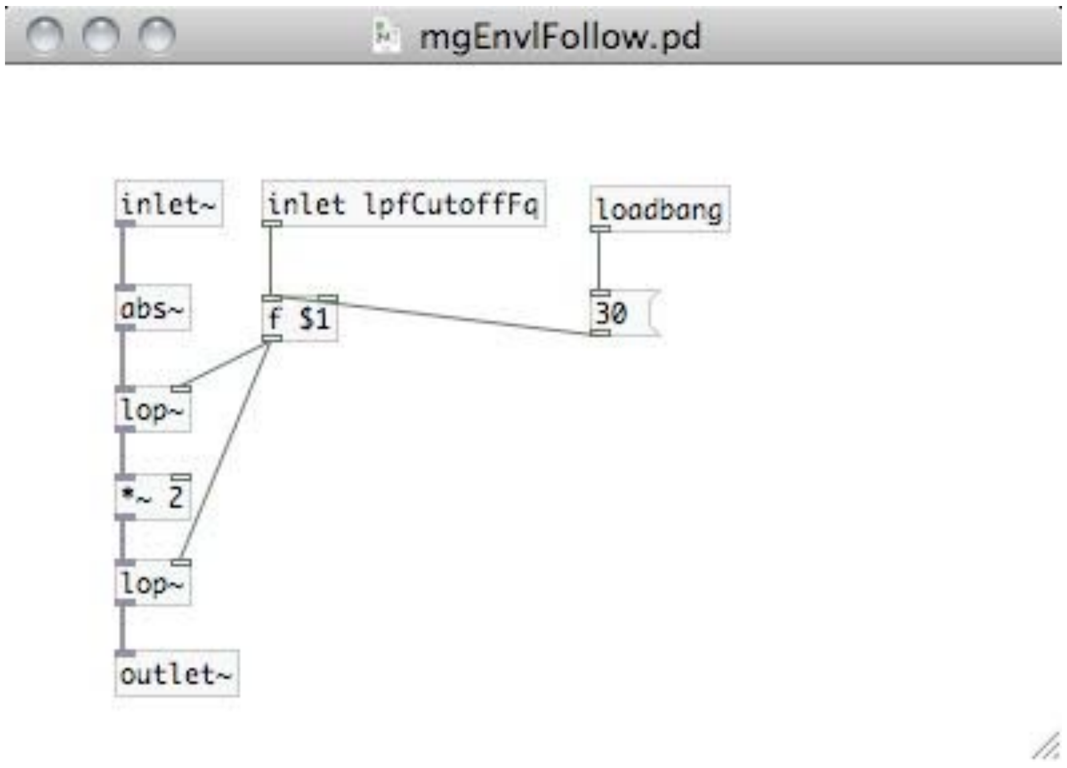
- Example: [mgProcDelayFeedback]



12.8. Using Analog Inputs as a Control Value

- Use [adc~] to get analog input
- Rectify, smooth, and scale into a control signal

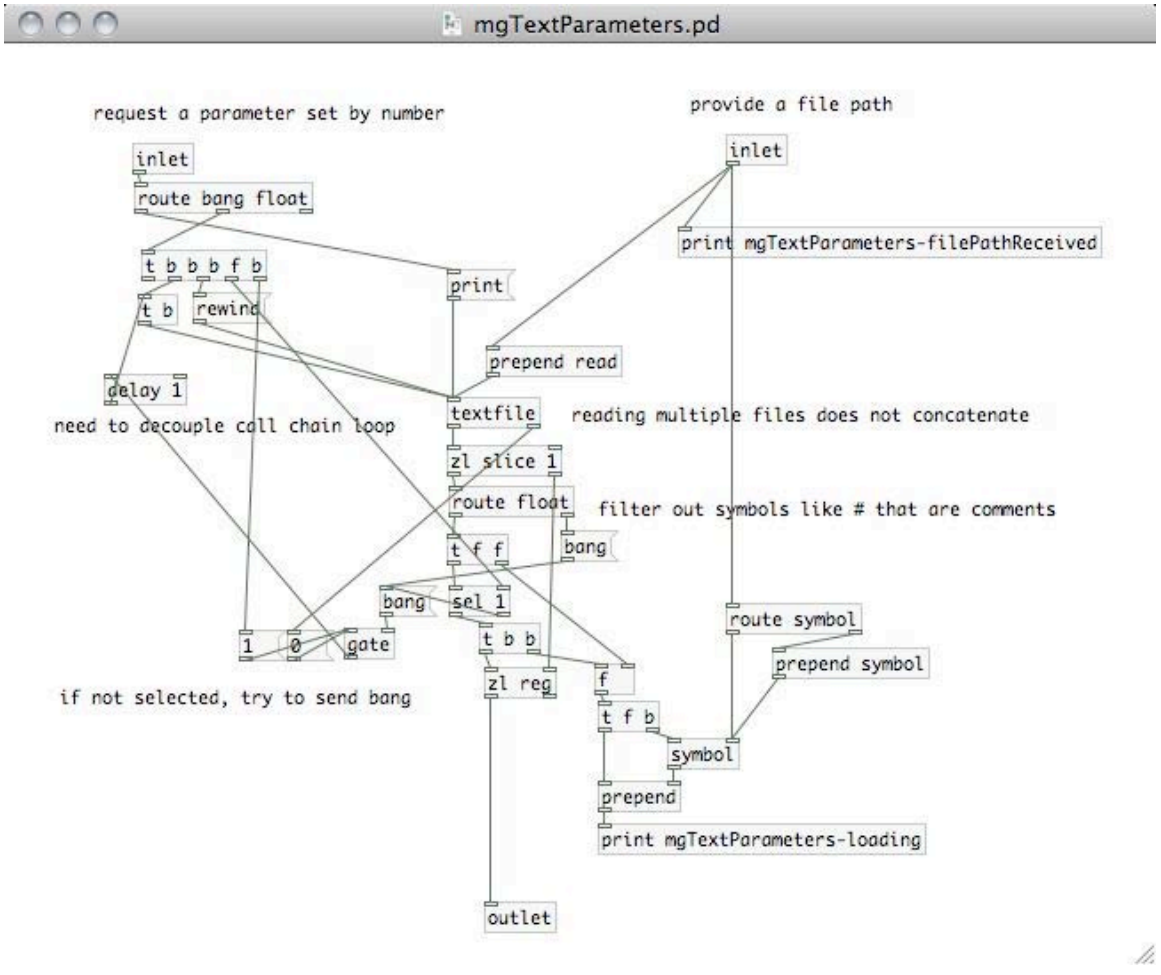
  [mgEnvlFollow], mgEnvlFollow.test



- Combining with a performance instrument
- pd/demos/envelopeFollowingAdc.pd

12.9. Reading Parameter Data from Text Files

- Can use [textfile]: will take in a text file, and will return a line of text for each bang
- Need to identify lines by key numbers, as well as store comments
- [mgTextParameters]

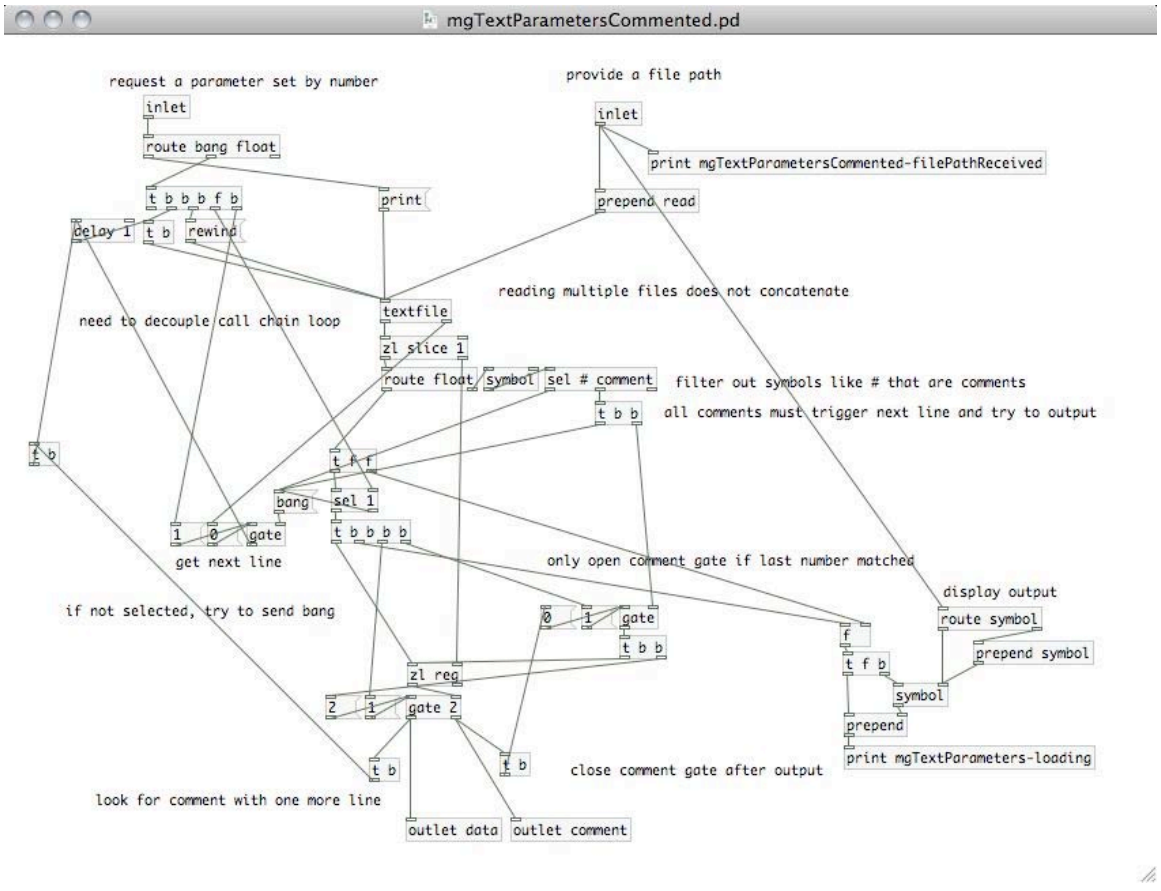

mgTextParameters.pd
request a parameter set by number
inlet
route bang float
t b b b f b
t b
rewind
print
delay 1
need to decouple call chain loop
provide a file path
inlet
print mgTextParameters-filePathReceived
prepend read
textfile
reading multiple files does not concatenate
zl slice 1
route float
filter out symbols like # that are comments
t f f
bang
bang
sel 1
1
0
gate
t b b
if not selected, try to send bang
zl reg
f
route symbol
prepend symbol
t f b
symbol
prepend
print mgTextParameters-loading
outlet

- Sample data file from mgSynthSawParameters.txt

```
# basic ;

# introductory scale;

1 pitch1 60 pitch2 62 pitch3 64 pitch4 65 pitch5 67 pitch6 69 pitch7 70 pitch8 72
attack 20 decay 100 sustain .8 release 1000 fmRate 4 fmDepth .25 lpfMin 70 lpfMax
120 octaveShift 0;

# e phrygian starting below middle c;

2 pitch1 52 pitch2 53 pitch3 55 pitch4 57 pitch5 59 pitch6 60 pitch7 62 pitch8 64
attack 10 decay 40 sustain .8 release 500 fmRate 4 fmDepth .25 lpfMin 70 lpfMax 120
octaveShift 0 ;
```

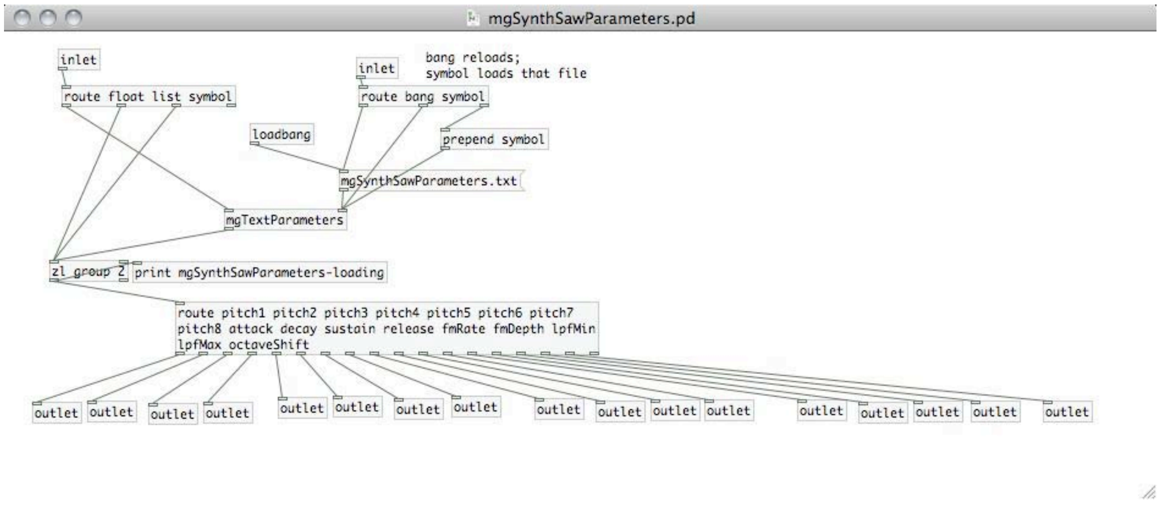
- [mgTextParametersCommented]

  Adds support for associating a comment line of text with each parameter


mgTextParametersCommented.pd
request a parameter set by number
inlet
route bang float
t b b b f b
print
delay 1
t b
rewind
provide a file path
inlet
print mgTextParametersCommented-filePathReceived
prepend read
reading multiple files does not concatenate
need to decouple call chain loop
textfile
zl slice 1
route float
symbol
sel # comment
filter out symbols like # that are comments
t b b
all comments must trigger next line and try to output
t b
t f f
bang
sel 1
1
0
gate
t b b b b
get next line
only open comment gate if last number matched
display output
if not selected, try to send bang
0
1
gate
t b b
route symbol
f
prepend symbol
t f b
zl reg
symbol
2
1
gate 2
prepend
print mgTextParameters-loading
t b
t b
close comment gate after output
look for comment with one more line
outlet data
outlet comment

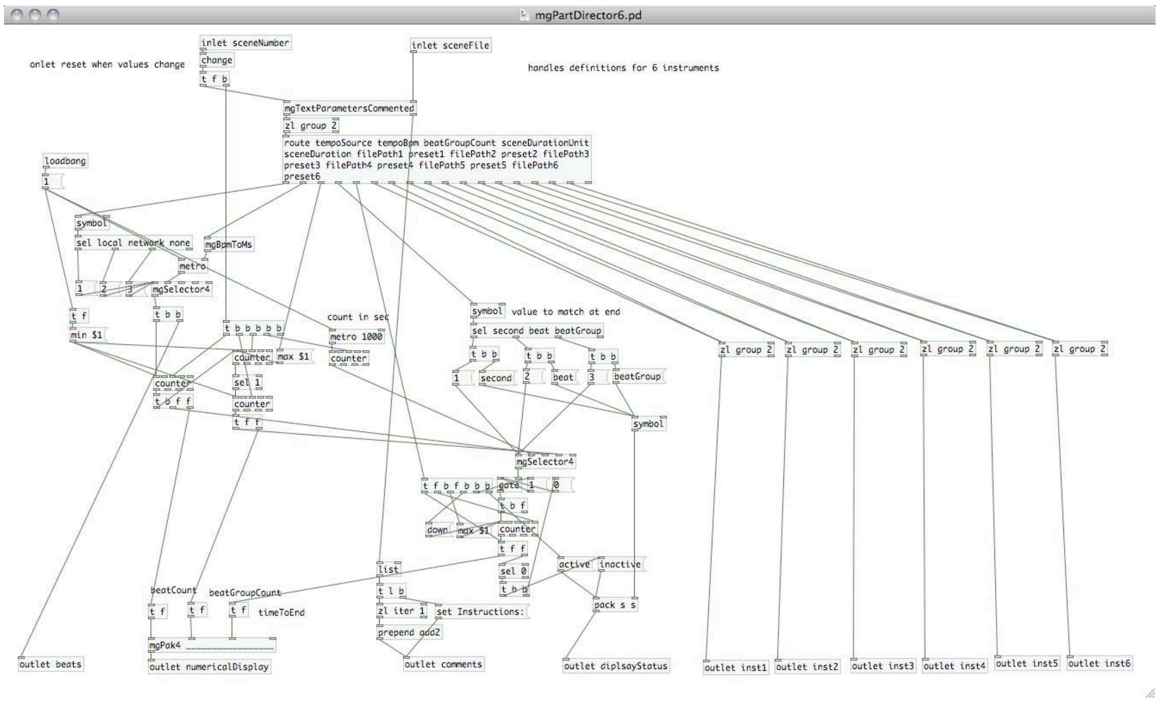
- Parameter abstractions can embed [mgTextParameters]

  [mgSynthSawParameters]

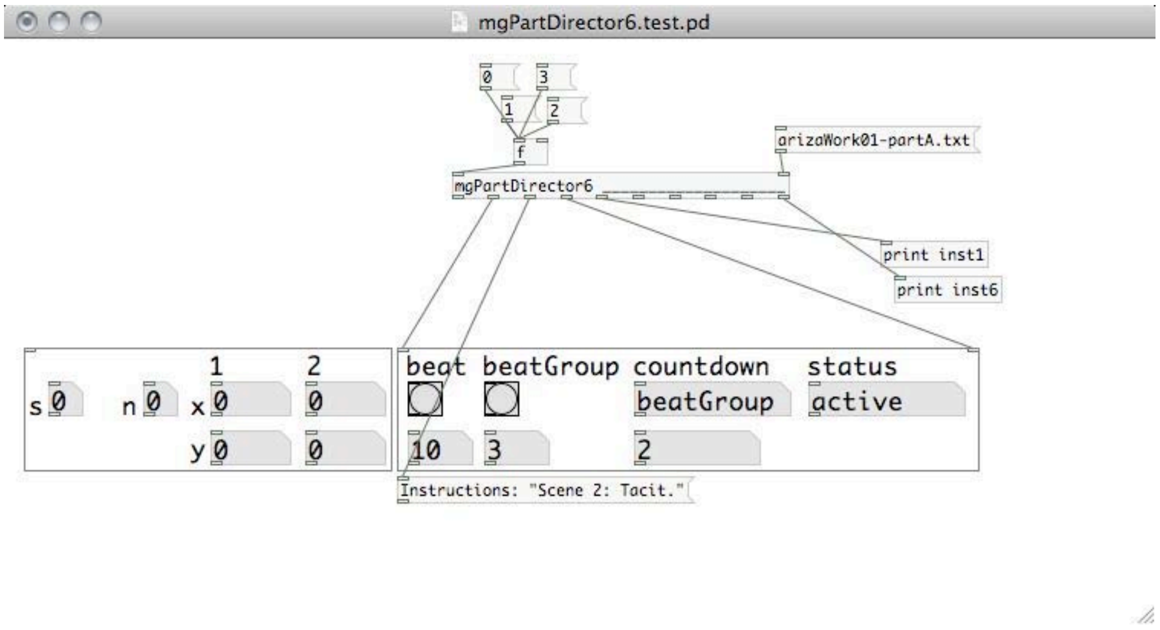


12.10. Controlling VIPP Pattern with a Director

- Director: combination of conductor and director
- Each part (sub-group of ensemble) has its own script
- Each performance embeds a [mgPartDirector6] or similar absraction
- For a given Performance, each scene defines presets (and file source of presets) for each (of fixed number) of instruments
- Each scene defines a beat-source in BPM: local, network, or none
- Each scene defines a scene duration measure in beat, beat groups, or seconds
- Each scene provides extended text instructions to the performer about what to do
- [mgPartDirector6]

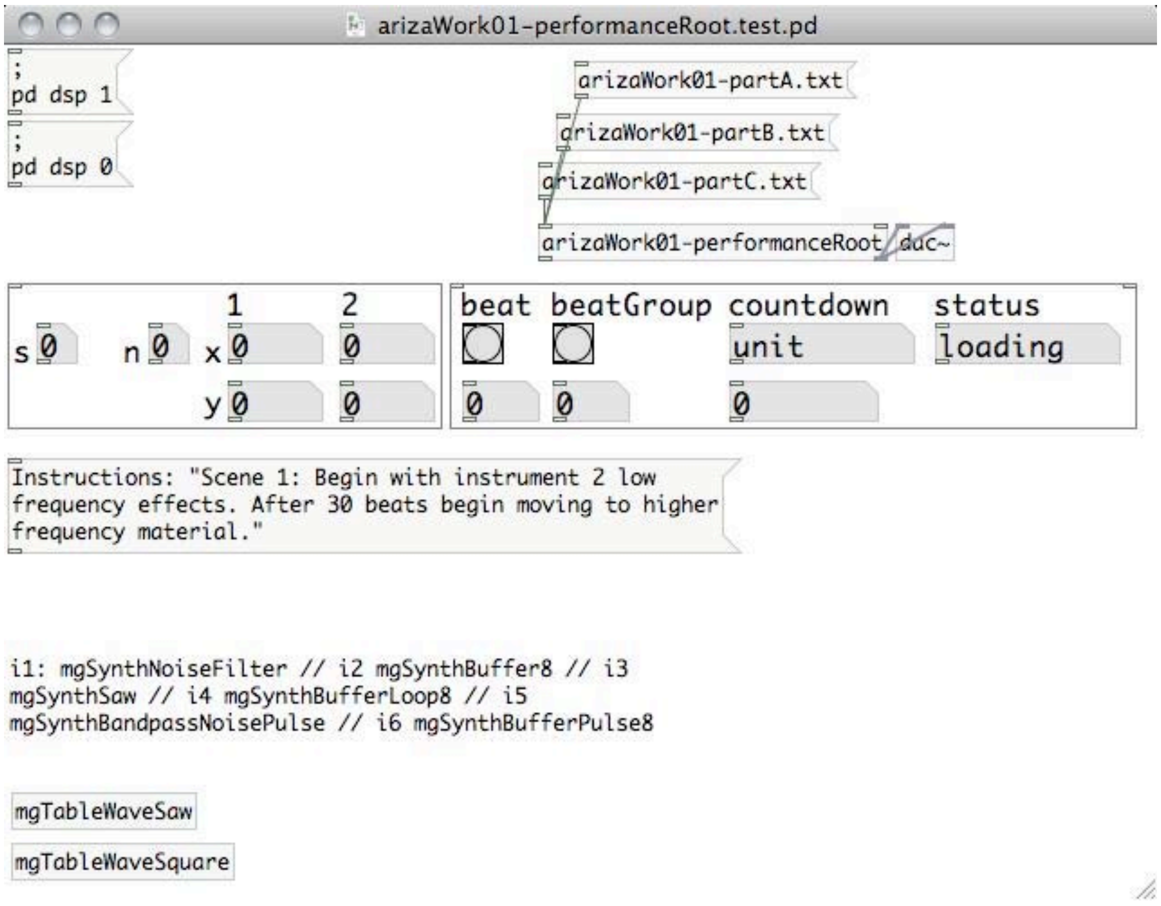


- Countdown refers to how much time is left in scene



12.11. The Work Performance Interface: a Performance with a Director

- Must load script for assigned Part
- Must manually advance scenes when countdown timer has reached zero
- Must then realize instructions
- Performance interface

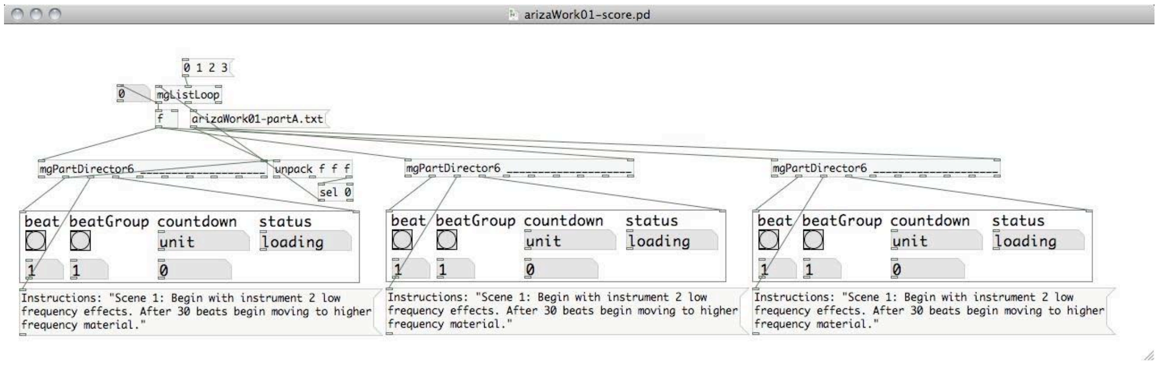


Instructions: "Scene 1: Begin with instrument 2 low frequency effects. After 30 beats begin moving to higher frequency material."

i1: mgSynthNoiseFilter // i2 mgSynthBuffer8 // i3 mgSynthSaw // i4 mgSynthBufferLoop8 // i5 mgSynthBandpassNoisePulse // i6 mgSynthBufferPulse8

12.12. The Score Interface: Simultaneously Viewing all Parts

- For testing and composing, can view all Parts in parallel



Instructions: "Scene 1: Begin with instrument 2 low frequency effects. After 30 beats begin moving to higher frequency material."

Instructions: "Scene 1: Begin with instrument 2 low frequency effects. After 30 beats begin moving to higher frequency material."

Instructions: "Scene 1: Begin with instrument 2 low frequency effects. After 30 beats begin moving to higher frequency material."

Chapter 13. Meeting 13, Workshop: Performance and Improvisation

13.1. Announcements

- Download the most recent martingale
- Due Wednesday, 16 March: Controller/Interface/Instrument Design 1 Report

  Will accept by midnight Friday, 18 March

  Must submit code and report

  See syllabus for report details

13.2. Reading: Collins, Handmade Electronic Music

- Collins, N. 2009. *Handmade Electronic Music: The Art of Hardware Hacking*. 2nd ed. New York: Routledge.
- Electromagnetic induction: using telephone coils, guitar pickups, wrapped wire to find unconventional sounds in the air
- Using radios to pick up noise, or feedback
- Driving a speaker with a direct current; modulating driving current with feedback (The Victorian Synthesizer)
- Speakers as microphones and vice versa
- Motors as oscillators: The Telharmonium
- Reading magnetic data: using tape heads and card readers to sonify data
- Piezoelectric elements as microphones and drivers; Rainforest

13.3. Quiz 2

- ?

13.4. The Performance Interface

- Load: arizaWork02-performance*.test.pd martingale/comopositions/arixaWork02/arizaWork02-performance*.test.pd

- Select and provide controller number to right inlet of [arizaWork02-performance*]
- Enter a part file, e.g., arizaWork02-partA.txt
- Advance to scene 1

13.5. Exercise: Pass-the-Gesture Solo

- Load: arizaWork02-performance*.test.pd

  martingale/comopositions/arixaWork02/arizaWork02-performance*.test.pd

- Performer A, instrument 4: Create a memorable musical gesture, lasting between 5 and 15 seconds
- Performer B, instrument 4: Re-create the musical gesture of Performer A, not necessarily using the same sound source, but maintaining temporal and spectral outline.
- Continue process from performer to performer, moving around in a circle
- Variation: Performer B responds to A not in similarity, but maximal contrast, producing the opposite idea

13.6. Exercise: Pass-the-Gesture Ensemble

- Load: arizaWork02-performance*.test.pd

  martingale/comopositions/arixaWork02/arizaWork02-performance*.test.pd

- Ensemble, instrument 1: Create a subtle, low frequency background texture, contributing a small bit to the total sound

  Performer A: Create a memorable musical gesture, lasting between 5 and 15 seconds; repeat it twice, with variable space in-between gestures.

- Performer B: Re-create the musical gesture of Performer A, not necessarily using the same sound source, but maintaining temporal and spectral outline.

  Ensemble: Continue

- Continue process from performer to performer, moving around in a circle
- Variation: Performers A and B selected in pairs: A and B in a dialog or argument

- Variation: Performer B responds to A not in similarity, but maximal contrast, producing the opposite idea

13.7. Work II

- Three part form
- 18 scenes
- Focus on texture and heterophony

Chapter 14. Meeting 14, Practices: Analog Electronics and Circuit Bending

14.1. Announcements

- Controller/Interface/Instrument Design 1: comments and grades this weekend
- Due Wednesday 6 April: Controller/Interface/Instrument Design 2 Proposal
- Due Wednesday 13 April: Performance Frameworks Draft

  Must email me now with special requests for groups

14.2. Quiz Review

- ?

14.3. Reading: Ghazala, The Folk Music of Chance Electronics: Circuit-Bending the Modern Coconut

- Ghazala, Q. R. 2004. "The Folk Music of Chance Electronics: Circuit-Bending the Modern Coconut." *Leonardo Music Journal* 14(1): pp. 97-104.
- How did Ghazala get started building instruments?
- What are the basic techniques of chance electronics?
- What is Ghazala's notion of the immediate canvas?
- How is the coconut a useful metaphor for building new instruments?
- For Ghazala, what are living instruments?
- Why might Ghazala's approach be seen as a democratization of music making?

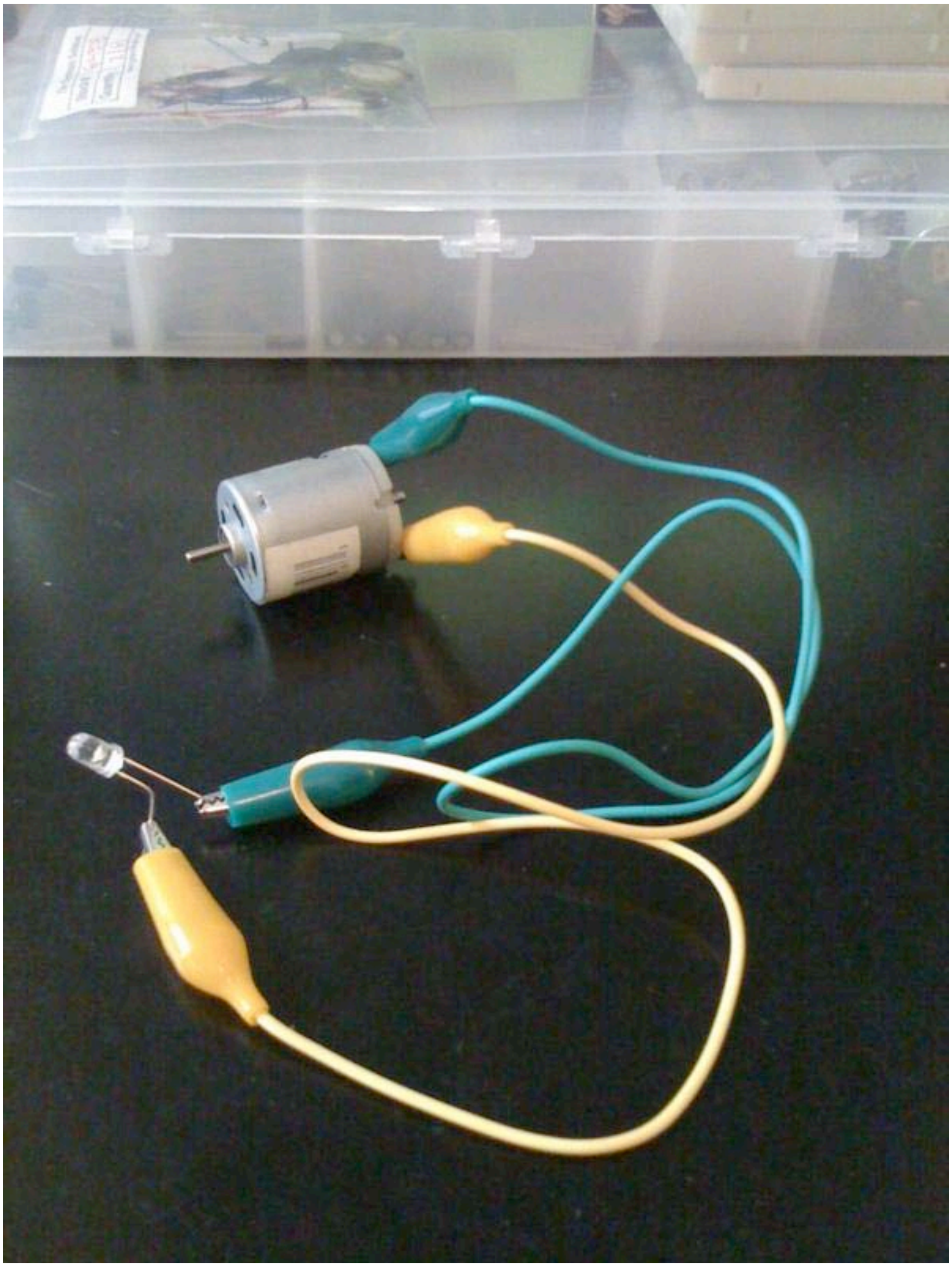
14.4. Reading: Collins, Handmade Electronic Music

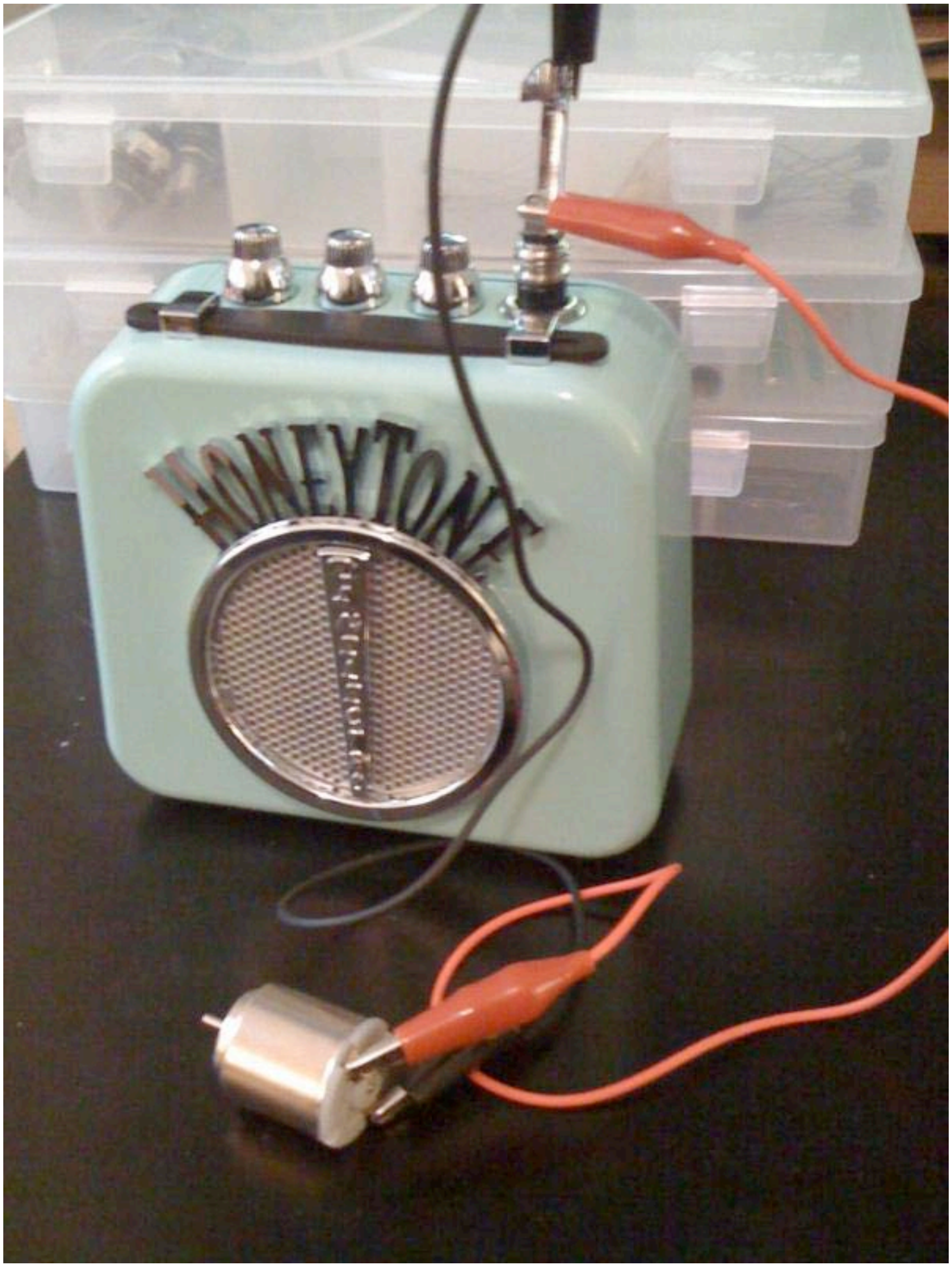
- Collins, N. 2009. *Handmade Electronic Music: The Art of Hardware Hacking*. 2nd ed. New York: Routledge.
- Laying of hands: radios, circuit boards
- The cracklebox

- Adjusting, replacing, and manipulating clocks in toys
- Using potentiometers, photoresistors, and alternative electrodes
- Circuit bending: the difference between bent and hacked?

14.5. Hardware Hacking: Mini Telharmonium

- Materials: DC Motor, amplifier, aligator clips
- Motor creates pulses that, when fast enough, produce a tone

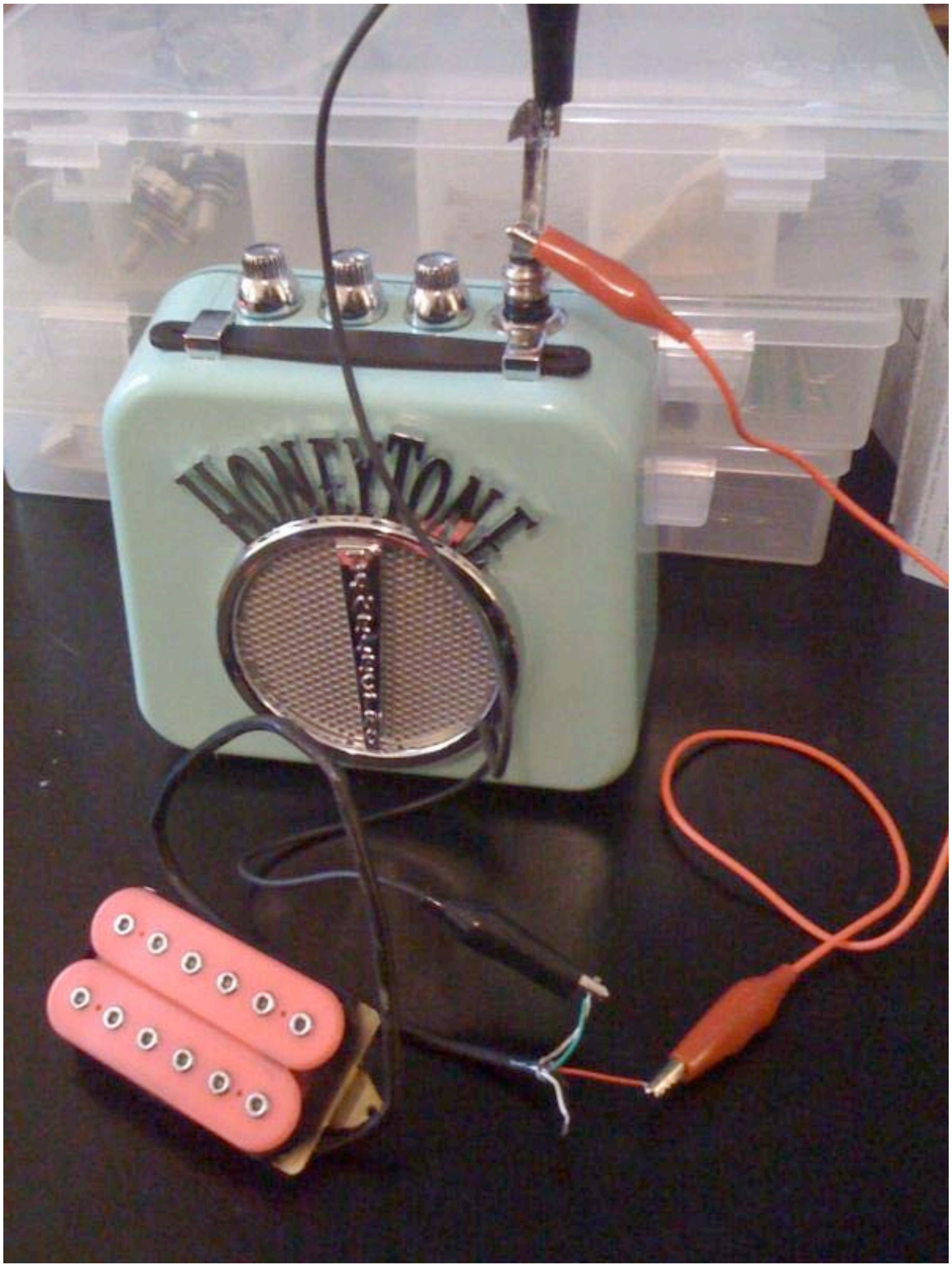




- Variation: jumping speaker; using a speaker as a microphone

14.6. Hardware Hacking: Electromagnetic Transduction

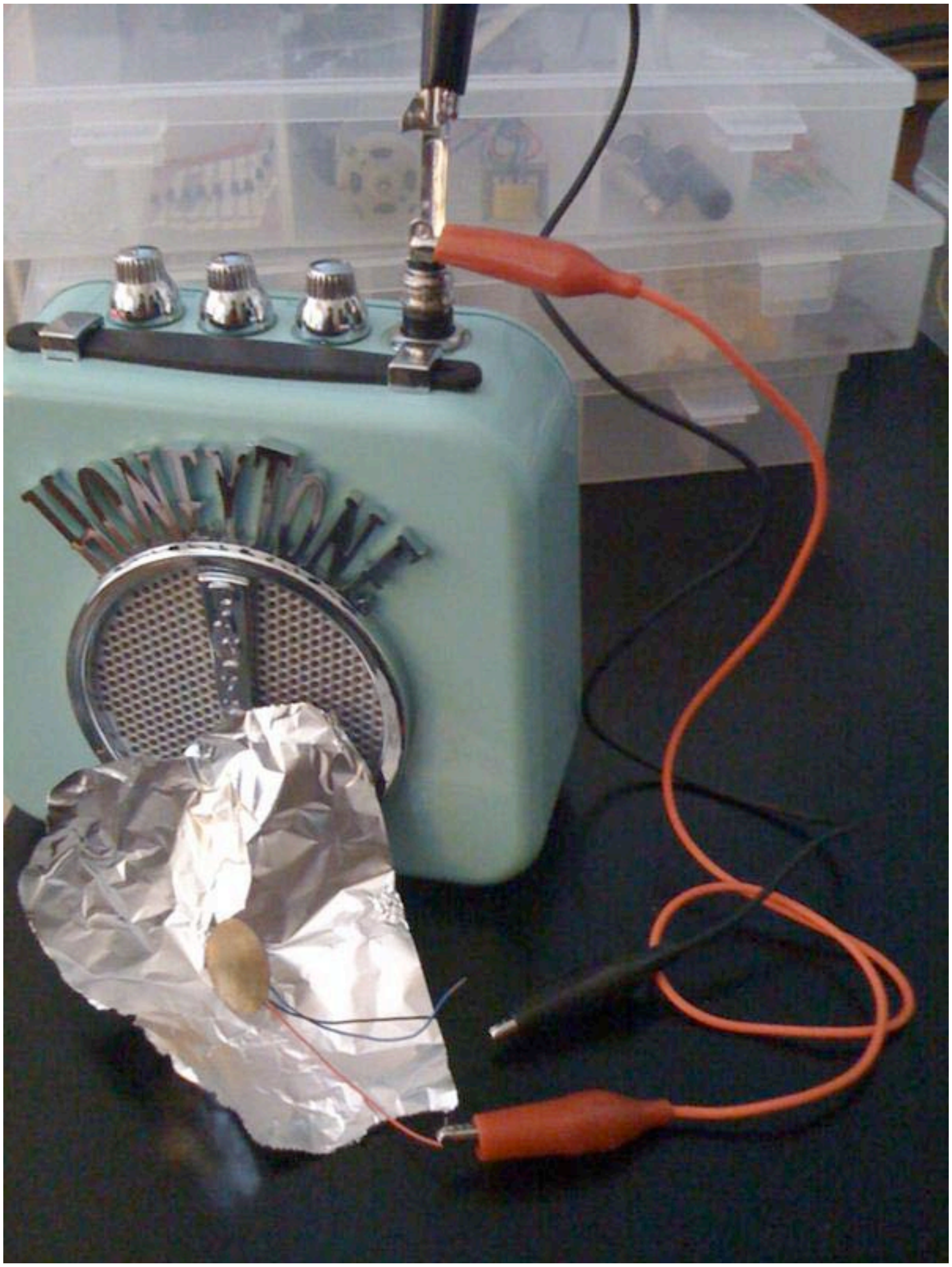
- Materials: Tape head, phonograph cartridge, guitar pickup, amplifier, aligator clips
- Can read magnetic variations and transduce vibrations
- Guitar pickup and iPhone



- Tape heads and ID cards

14.7. Hardware Hacking: Piezo-electric Transduction

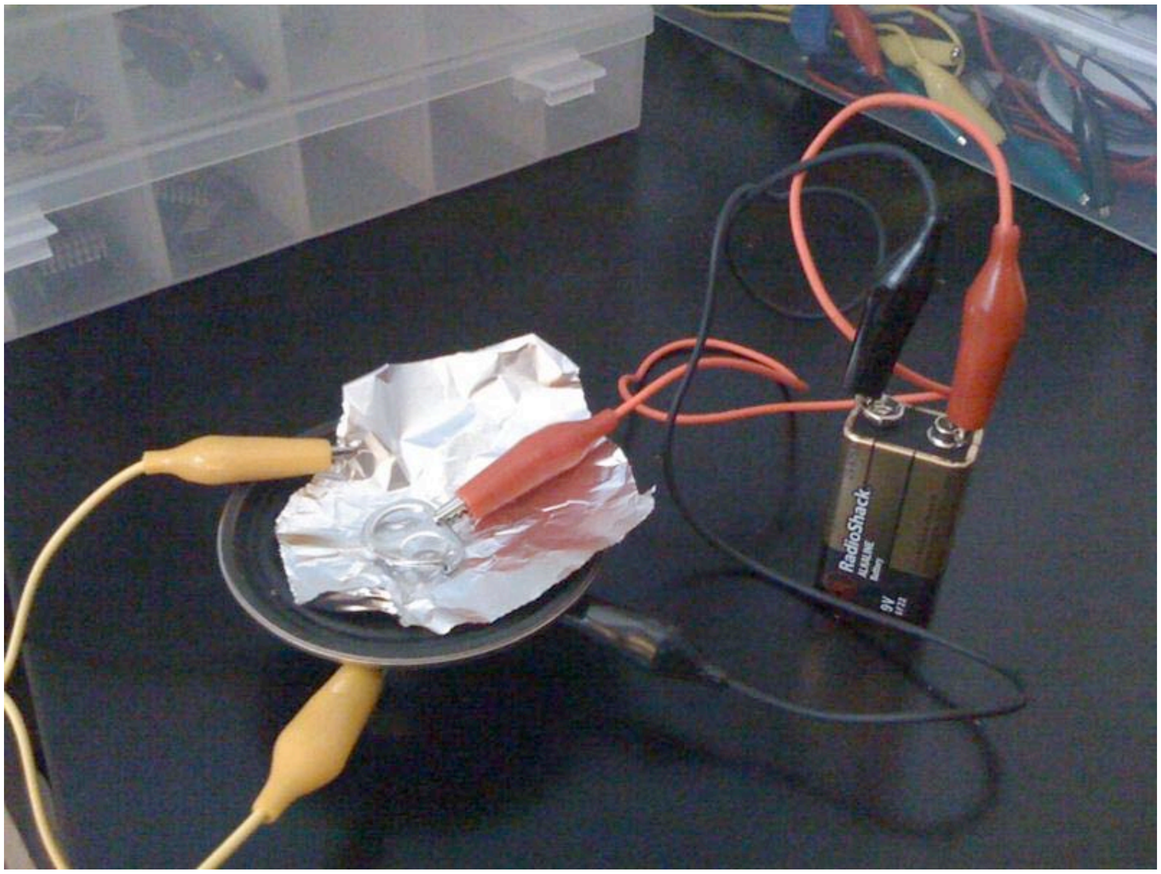
- Materials: piezo microphone, slinky, amplifier, aligator clips



- Can pick up surface vibrations for direct transduction
- Can pick up vibrations travelling through a slinky

14.8. Hardware Hacking: Victorian Oscillator

- Materials: 9v battery, speaker, three aligator clips, metal materials
- Connection to speaker from battery is created and broken through mechanical feedback



14.9. Hardware Hacking: Video Excerpts

- Project tutorials / Laying of Hands, Chapter 11
- Project tutorials / Hack the Clock, Chapter 12-15 (1:07, 1:13, 1:14)
- Circuit bent examples:
  - Emir Bijukic: Solar Cell Calculator
  - Michael T. Bullock: Open-backed radio
  - Chapman & Collinson: The Bent Radio Orchestra
  - Seth Cluett: Open-backed Cassette

    - Stewart Collinson: Hacked Cyberhead
    - David First: The Radiotron
    - Steve Marsh: Bent Stix Drum Machine
    - Dan Wilson: Work Cracklebox
- Project tutorials / Drivers Chapter 8

14.10. Listening: Collins, Behrman, Yasunao Tone

- Listening: Nicolas Collins, "El Loop," *Handmade Electronic Music*

- Listening: David Behrman, Players With Circuits

- Listening: Yasunao Tone, Imperfection Theorem of Silence

Chapter 15. Meeting 15, Practices: Electronics and Sensors

15.1. Announcements

- Workshop on Monday: bring controller and amps
- Controller/Interface/Instrument Design 1: comments and grades this weekend
- Due Wednesday 6 April: Controller/Interface/Instrument Design 2 Proposal
- Due Wednesday 13 April: Performance Frameworks Draft

  Must email me now with special requests for groups

15.2. Reading: Tanaka: Sensor-Based Musical Instruments and Interactice Music

- Tanaka, A. 2009. "Sensor-Based Musical Instruments and Interactive Music." In R. T. Dean, ed. *The Oxford Handbook of Computer Music*. Oxford University Press, pp. 233-257. (243-257)
- What trends are can be seen in the availability and cost of sensors?
- What are examples of biosignal instruments?
- Tanaka writes that: "musicians have the uncanny ability to appreciate and repurpose machinery and technology for expressive musical ends": what are some examples?

15.3. Listening and Viewing: Sensorband and Atau Tanaka

- Viewing: Sensorband Performance (DEAF96): YouTube (http://www.youtube.com/watch?v=XLSoPmY6jGM)
- Viewing: Atau Tanaka performs using bio-metric sensor: YouTube (http://www.youtube.com/watch?v=FB_yE_Y3_8k)
- Listening: Sensorband / Atau Tanaka, Sola Produxies

15.4. Sensor Interfaces: Tactile and Table

- James Patten's AudioPad: http://www.jamespatten.com/audiopad
- Sergi Jorda's ReacTable: http://www.reactable.com/ (http://www.reactable.com)

15.5. Specialized Sensor Hubs

- iCube: http://infusionsystems.com/

  Example setup: USB-microSystem with 8 inputs and 8 outputs with USB computer input ($156)

Courtesy of Infusion Systems Ltd. Used with permission.

- Teabox http://shop.electrotap.com/products/teabox

  Example setup: Teabox Sensor Interface with 8 inputs with neutrik combo jacks and SPDIF computer input ($395)

- BioMuse by BioControl Systems: http://www.biocontrol.com

  BioWave (EEG, EOG, and EMG $505), BioFlex (EMG from arm and leg muscles $403), BioBeat (ECG signals form chest and torso $546)

15.6. General Purpose Microcontrollers

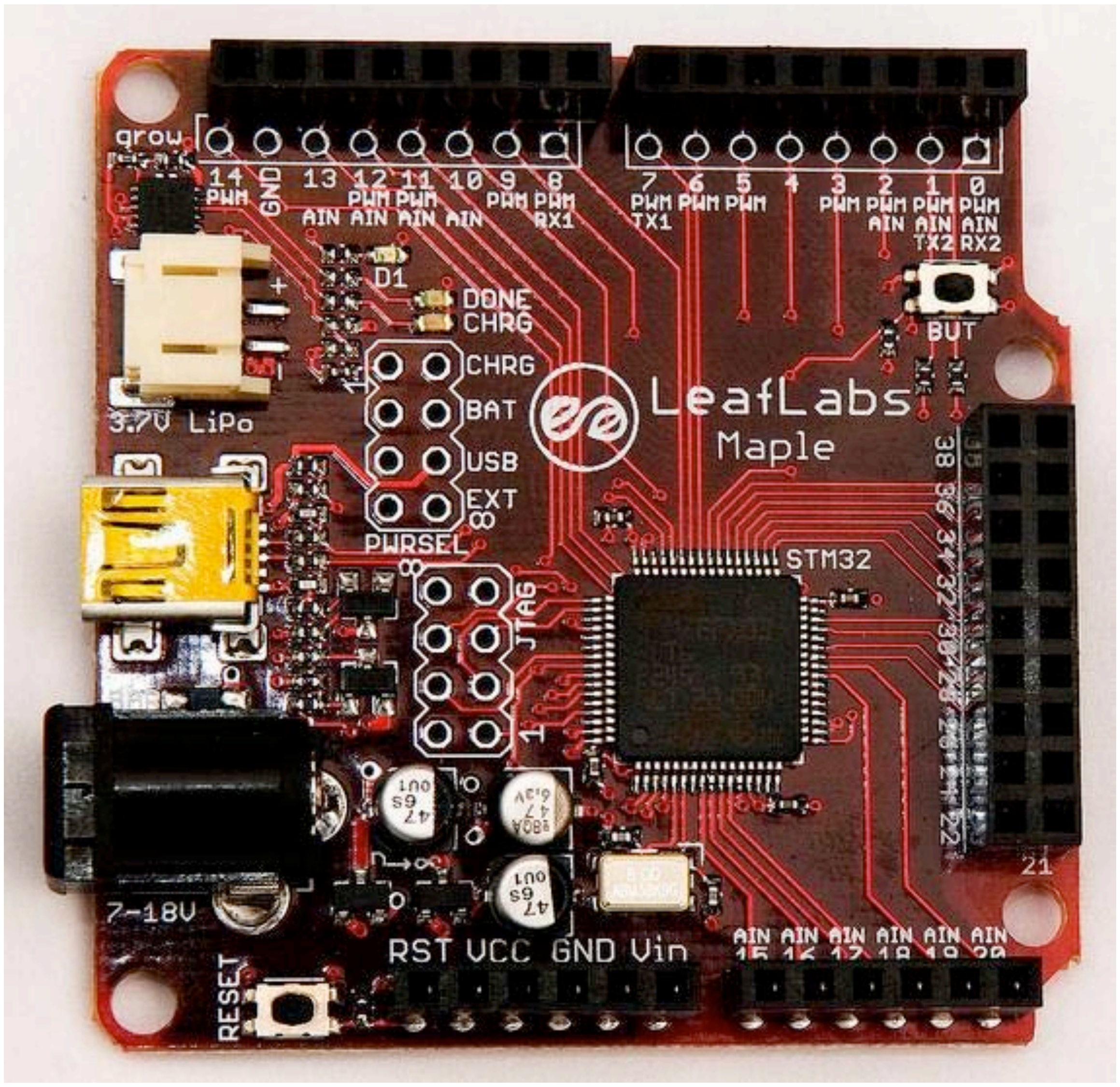
- Arduino: http://www.arduino.cc (http://www.arduino.cc/)

  Example setup: Arduino UNO with 6 analog inputs and 14 digital i/o ($30)

Photo courtesy of SparkFun Electronics.

- The Maple from LeafLabs http://leaflabs.com/devices/maple/

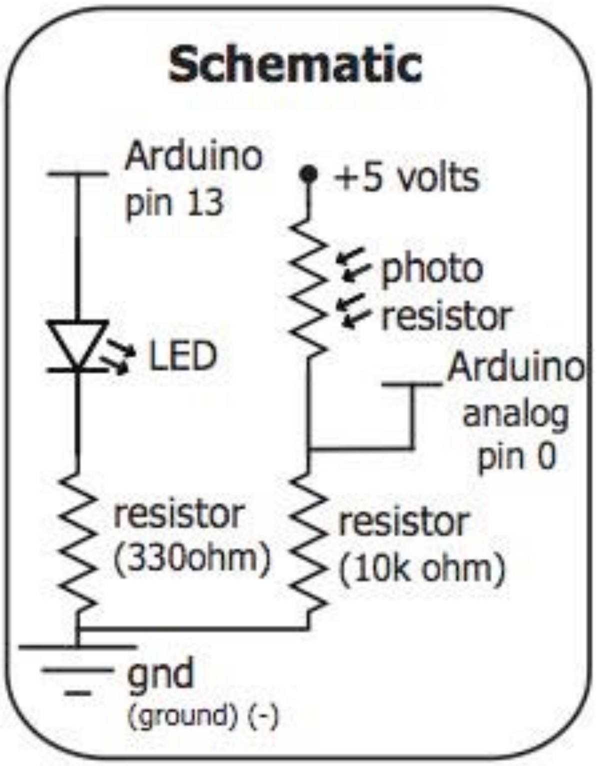
  Example setup: Maple with 16 analog inputs and 39 digital i/o ($50)



Courtesy of LeafLabs LLC.
License CC BY 3.0

15.7. Getting Analog Signals into the Arduino

- Can use a variety of sensors that produce variabel voltage resistance
- Often need to only supply power and ground, as well as a resistor for voltage dividing
- Example of components for reading values from a photoresistor (ignore LED)



Source: Manual for SparkFun Inventor's Kit for Arduino. © .:oomlout:. License CC BY-SA.

15.8. Arduino Code for Processing and Transmitting Analog Input

- The Arduino IDE provides code-editing and uploading to Arduino over USB
- Code is a light-weight version of C++ that is compiled for the ATmega328
- Code divided into two functions: setup() and loop()
- Can read from an analog pin with analogRead()
- Can print to the serial interface (transmit bytes) via Serial.print()
- Delay time in ms with delay()
- Complete code for reading, mapping, and printing values from two analog pins

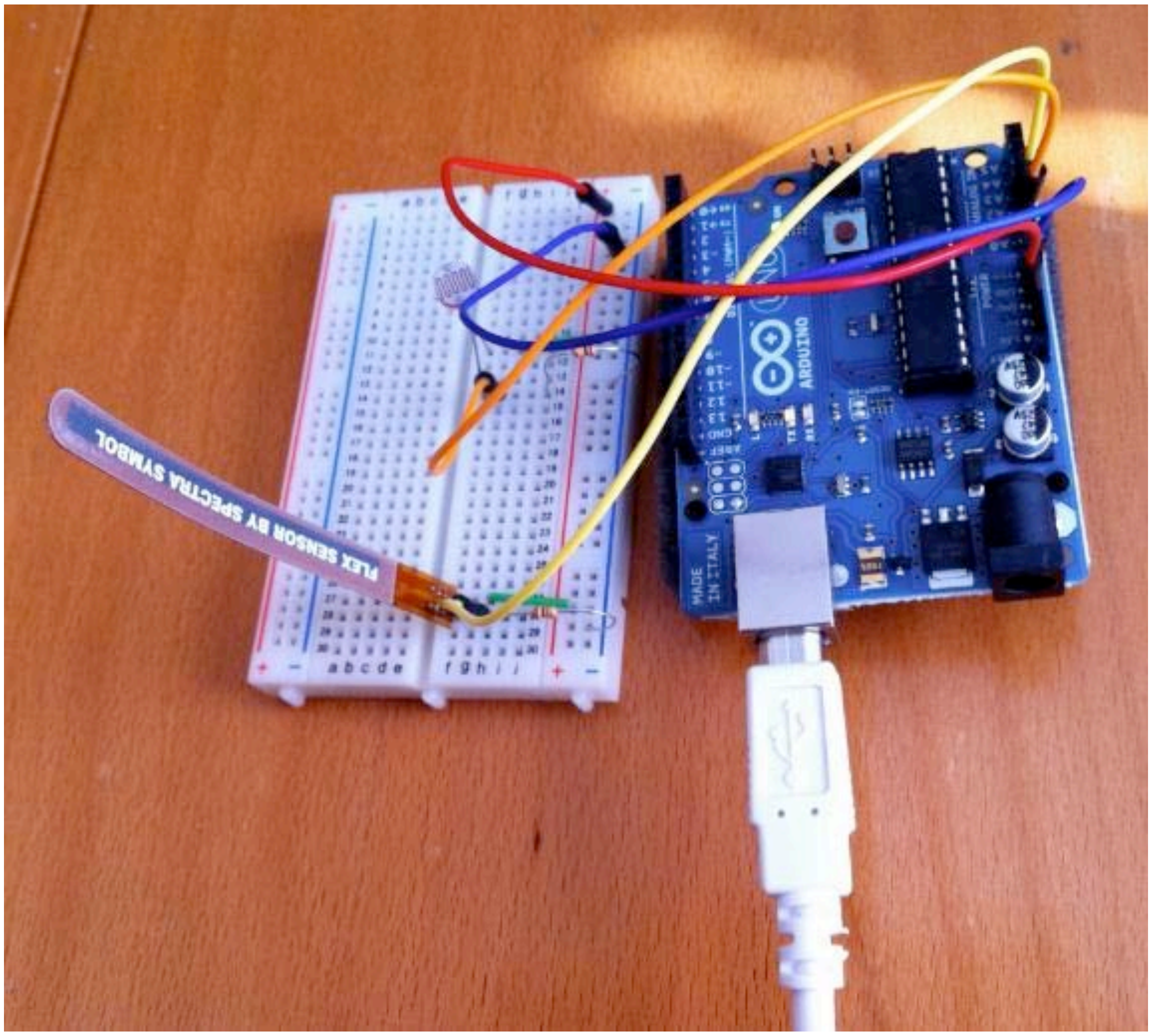
```
int photoPin = 0;
int flexPin = 1;
int x;
int y;

void setup() {
    Serial.begin(9600);
}

void loop () {
    Serial.print('A', BYTE); // char 65
    // practical range from 200 to 900
    x = analogRead(photoPin);
    Serial.print(map(x, 200, 900, 1000, 2000)); // sends 3 bytes
    Serial.print('\n', BYTE); // char 10

    Serial.print('B', BYTE); // char 66
    // raw values are between 47 and 282
    y = analogRead(flexPin);
    Serial.print(map(y, 47, 282, 1000, 2000));
    Serial.print('\n', BYTE); // char 10

    delay(30);
}
```

15.9. Reading Serial Data from Arduino in PD

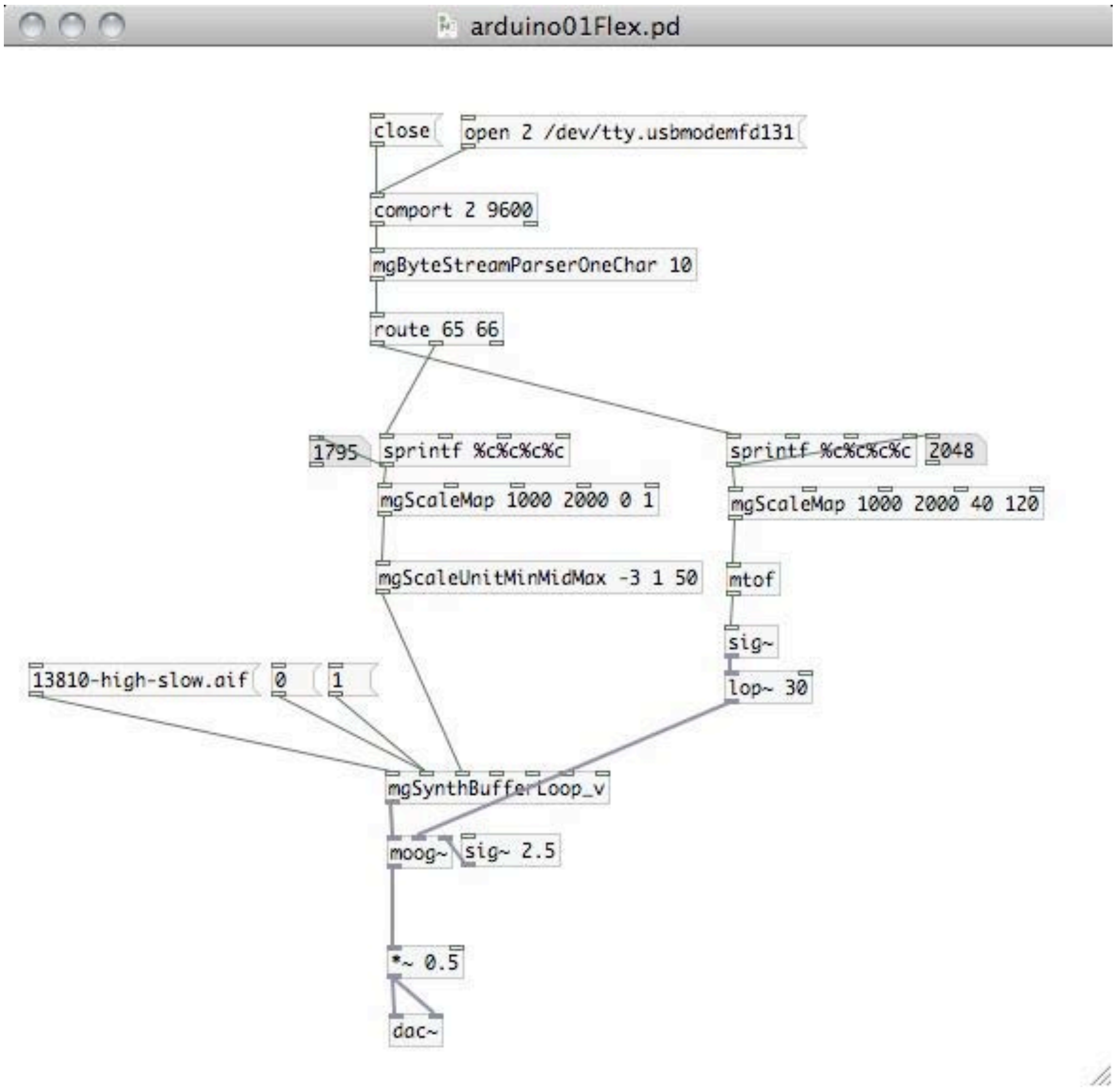
- [comport] object reads bytes from named serial port
- Byte stream needs to be broken into chunks: [mgByteSTreamPaserOneChar] divides byte stream into lists based on a sentinel byte (\n, char 10, is a good choice).
- A header byte (e.g., ASCII A and B) can be used to tag individual message from different inputs
- After breaking into lists, can use [route] to get bytes for each message, and reform three-byte integers into numbers with [sprintf]

15.10. Arduino/Pd Instrument: Flex sensor and photo resistor

- Upload code to Arduino first, then open serial connection in Pd
- Flex sensor controls playback rate of looping sample; photo-resistor controls a low-pass filter
- Arduino and Breadboard



- Pd patch:



15.11. Alternative Sensors and Inputs for Arduino and other Microcontrollers

- Force Sensitive Resistors (FSR) (pressure sensor)

Photo courtesy of SparkFun Electronics.

- Flex sensors ($13 for 4.5 inch)

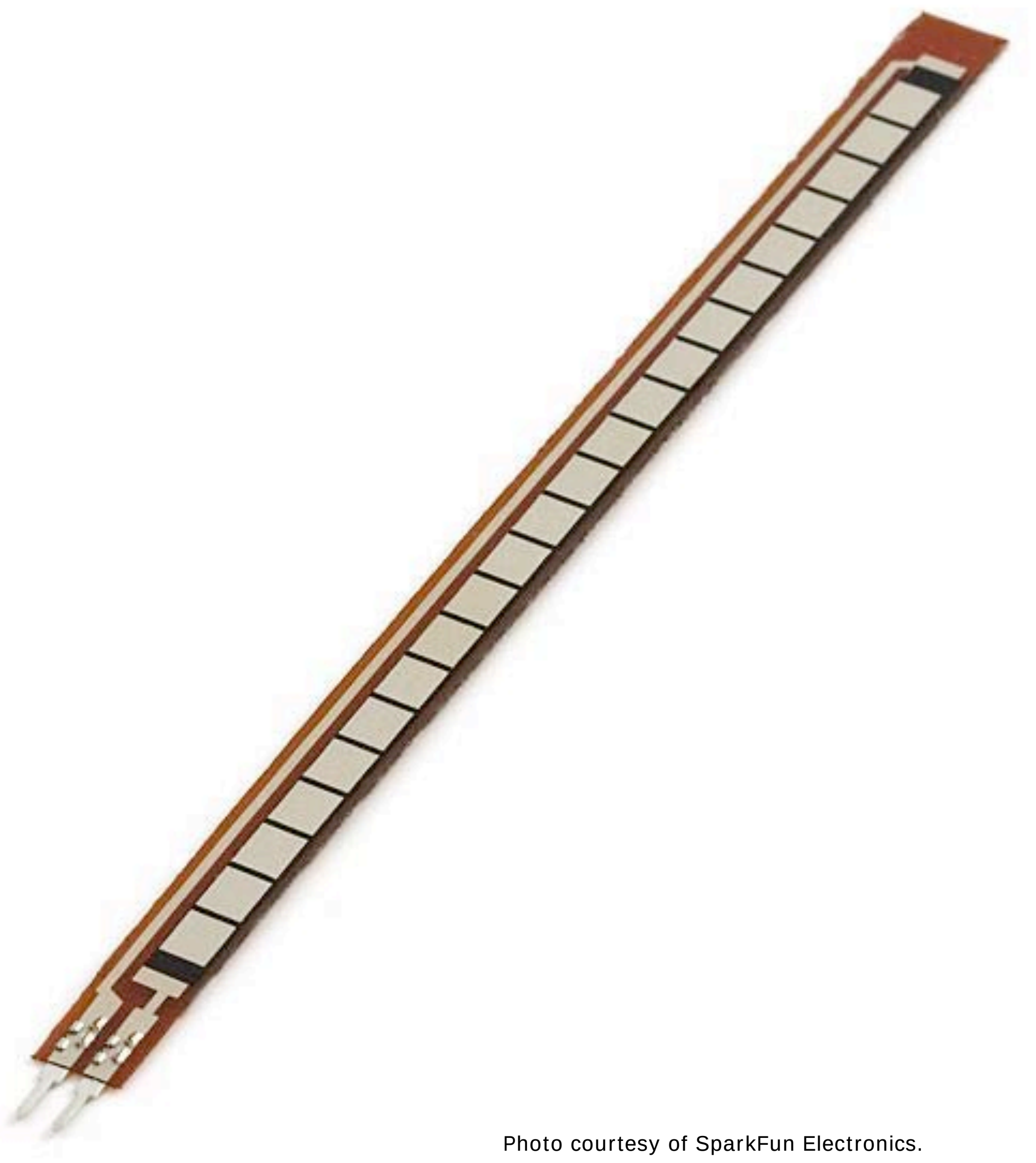


Photo courtesy of SparkFun Electronics.

- SoftPot Membrane Potentiometers ($15)

Photo courtesy of SparkFun Electronics.

- Temperature Sensor: LM335A ($1.50)

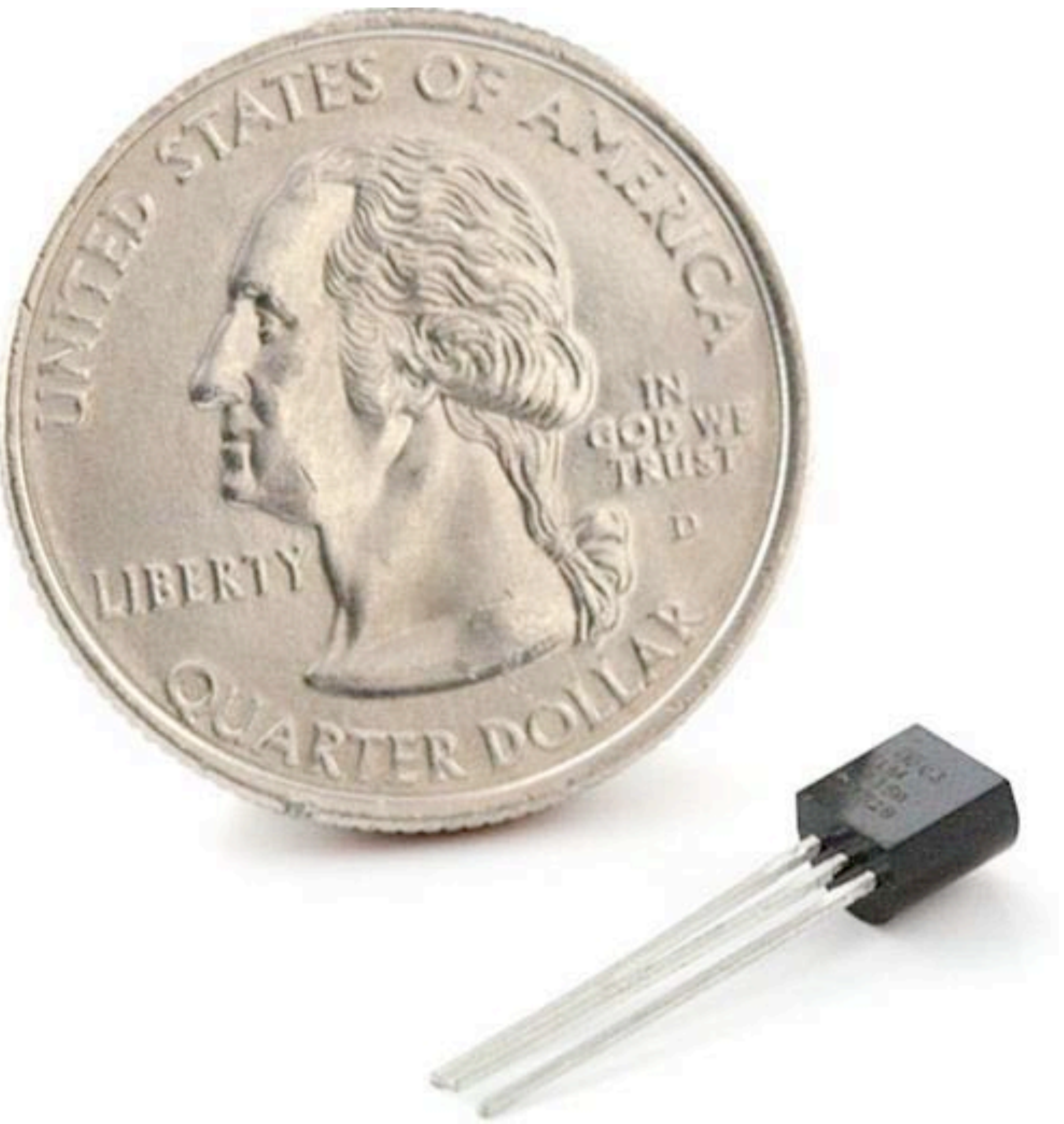


Photo courtesy of SparkFun Electronics.

- Infrared Proximity Sensor: Sharp GP2Y0A21YK ($14)

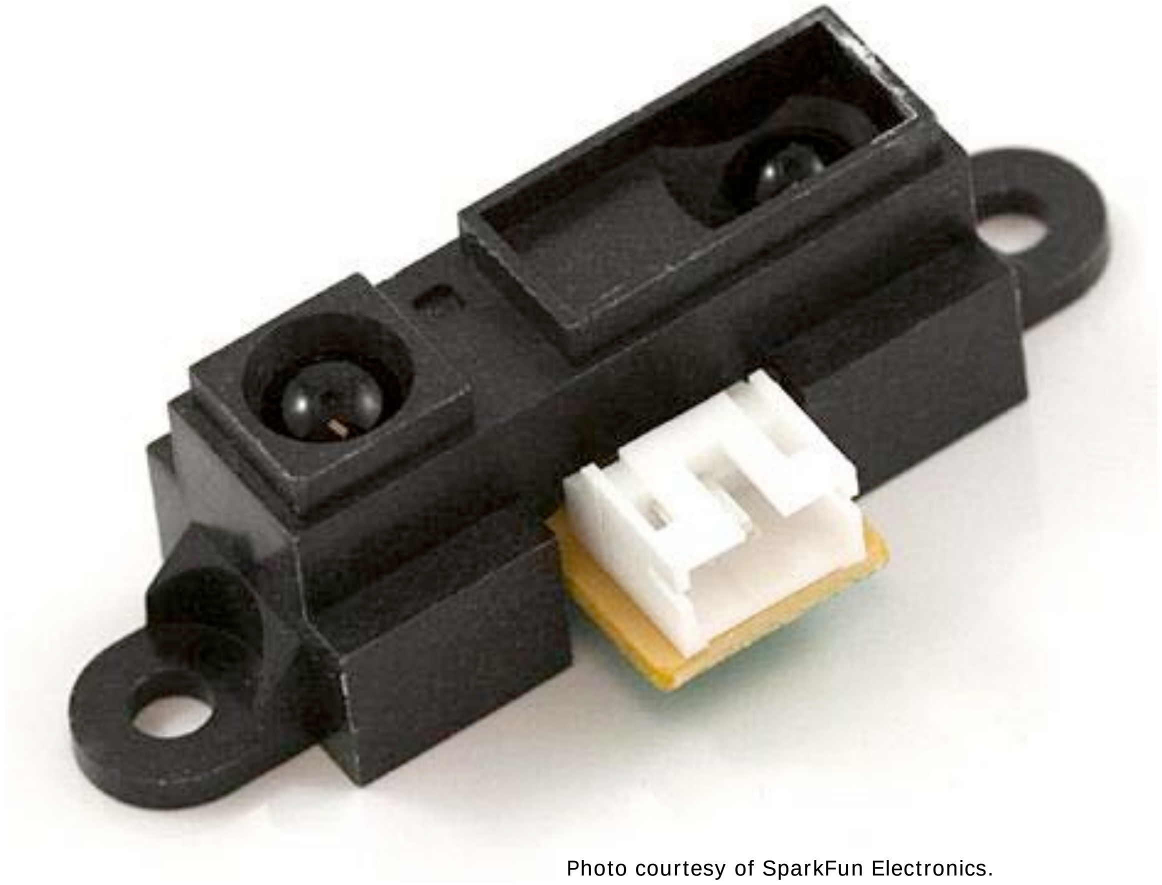


Photo courtesy of SparkFun Electronics.

- Compass Module: HMC6352 ($35)

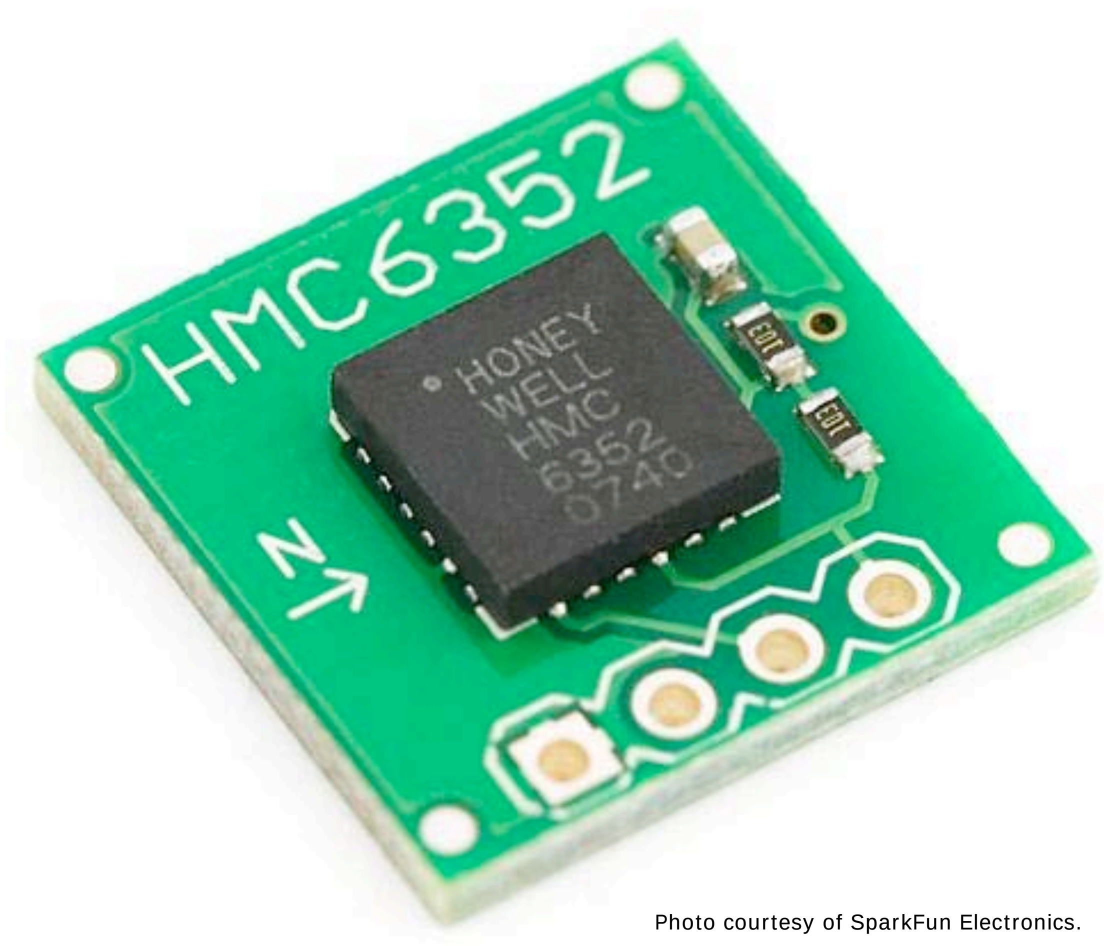


Photo courtesy of SparkFun Electronics.

- Triple Axis Accelerometer: ADXL330 ($25)

  Or, buy a Wiimote

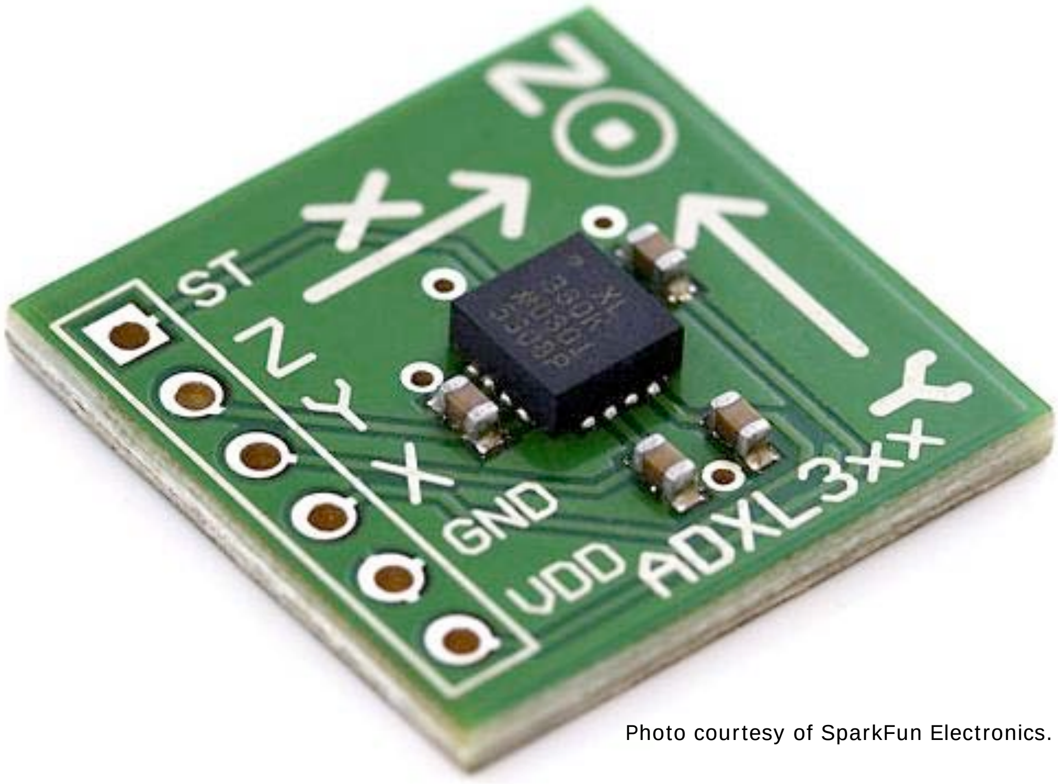


Photo courtesy of SparkFun Electronics.

15.12. Arduino/Pd Instrument: DangerBeat

- Sparkfun Danger Shield ($30)

  Bundles three sliders, three buttons, temperature, light, and knock sensor, plus LEDs, piezo buzzer, and 7-segment LED

- Arduino and DangerShield

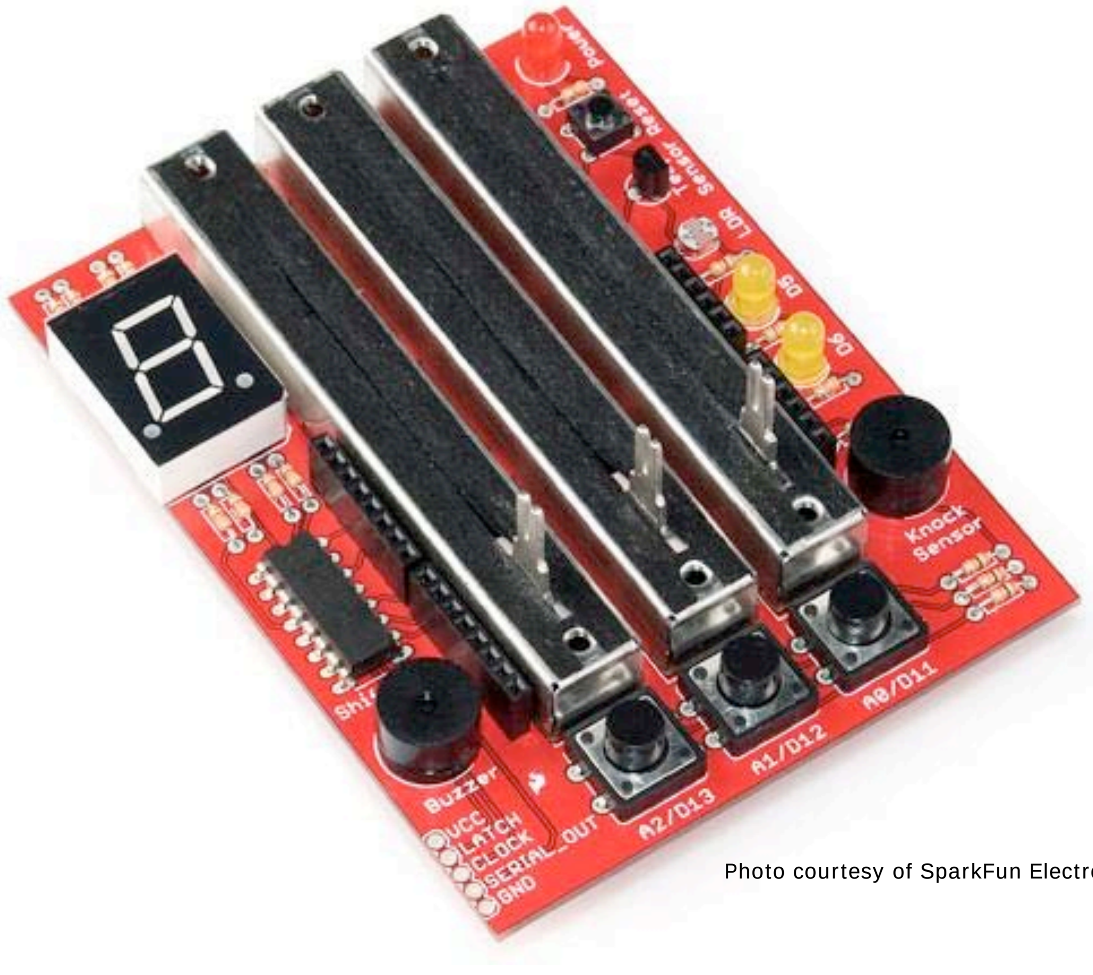
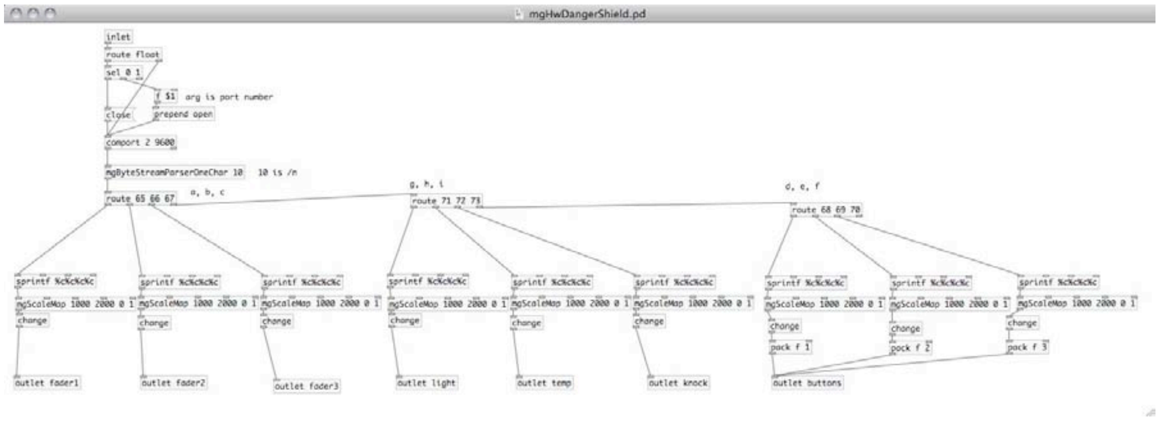
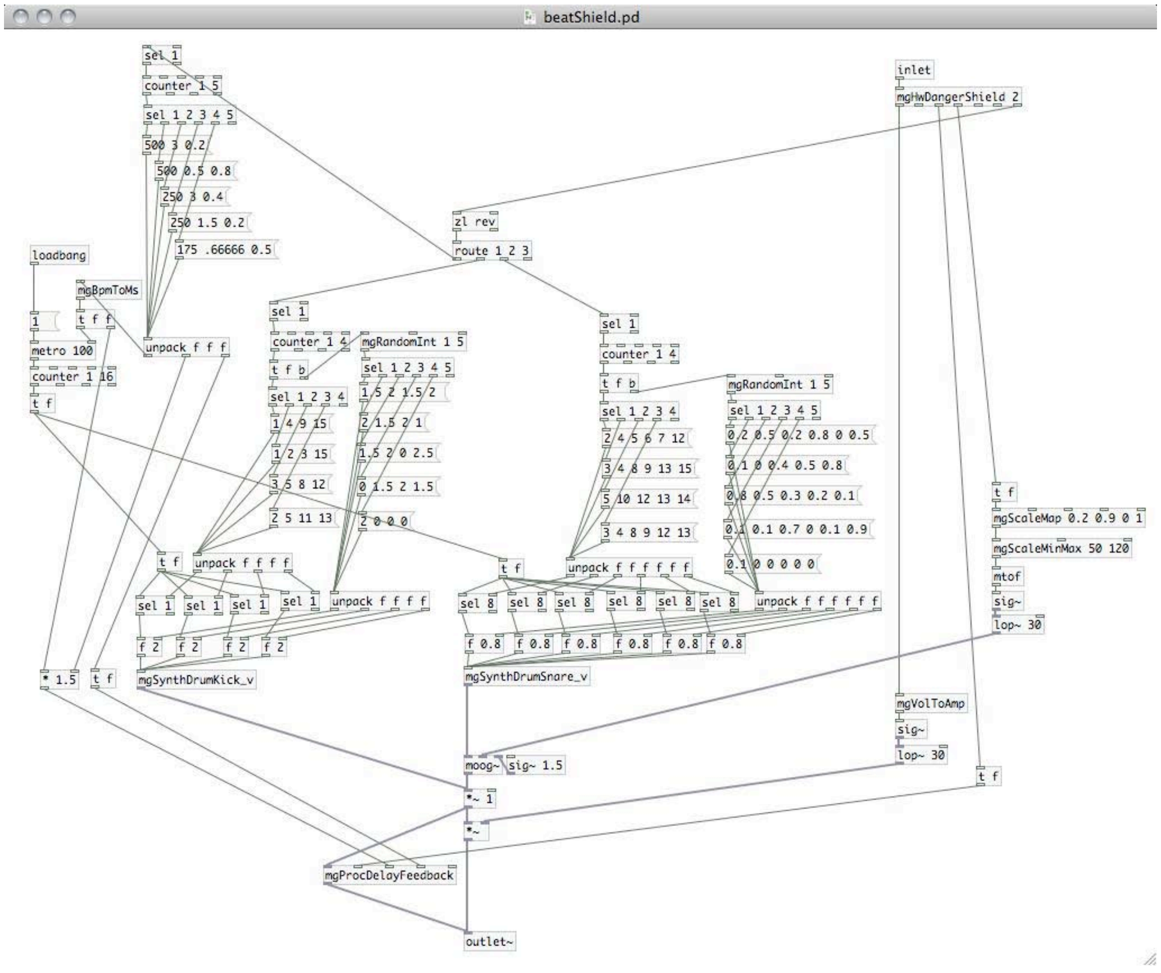


Photo courtesy of SparkFun Electronics.

- Instrument controls beat of two drum synthesis
- martingale/interfaces/dangerShield/dangerShield.pde: outputs data from all sensors mapped between 1000 and 2000
- [mgHwDangerShield]: package and manage data streams


mgHwDangerShield.pd
inlet
route float
sel 0 1
f $1
arg is port number
close
prepend open
comport 2 9600
mgByteStreamParserOneChar 10
10 is /n
route 65 66 67
a, b, c
g, h, i
route 71 72 73
d, e, f
route 68 69 70
sprintf %c%c%c%c
mgScaleMap 1000 2000 0 1
change
pack f 1
pack f 2
pack f 3
outlet fader1
outlet fader2
outlet fader3
outlet light
outlet temp
outlet knock
outlet buttons

- Pd patch:


beatShield.pd
sel 1
counter 1 5
sel 1 2 3 4 5
500 3 0.2
500 0.5 0.8
250 3 0.4
250 1.5 0.2
175 .66666 0.5
loadbang
mgBpmToMs
t f f
unpack f f f
1
metro 100
counter 1 16
t f
inlet
mgHwDangerShield 2
zl rev
route 1 2 3
sel 1
counter 1 4
mgRandomInt 1 5
t f b
sel 1 2 3 4
1 4 9 15
1 2 3 15
3 5 8 12
2 5 11 13
sel 1 2 3 4 5
1 5 2 1.5 2
2 1.5 2 1
1.5 2 0 2.5
0 1.5 2 1.5
2 0 0 0
counter 1 4
sel 1 2 3 4
2 4 5 6 7 12
3 4 8 9 13 15
5 10 12 13 14
3 4 8 9 12 13
0 2 0.5 0.2 0.8 0 0.5
0 1 0 0.4 0.5 0.8
0 8 0.5 0.3 0.2 0.1
0.1 0.1 0.7 0 0.1 0.9
0.1 0 0 0 0 0
t f
unpack f f f f
sel 1
unpack f f f f
f 2
unpack f f f f f f
sel 8
f 0.8
mgSynthDrumKick_v
mgSynthDrumSnare_v
* 1.5
t f
mgScaleMap 0.2 0.9 0 1
mgScaleMinMax 50 120
mtof
sig~
lop~ 30
mgVolToAmp
moog~
sig~ 1.5
*~ 1
*~
mgProcDelayFeedback
outlet~

Chapter 16. Meeting 16, Workshop: Performance and Improvisation

16.1. Announcements

- Due this Wednesday 6 April: Controller/Interface/Instrument Design 2 Proposal
- Performance Frameworks groups will be posted tonight
- Due Wednesday 13 April: Performance Frameworks Draft

  Must email me immediately with special requests for groups (will send out tonight)
- End of semester concert is set for May 4:

  ```
  Sonorous Currents:
  A Concert of 21M.380, Live Electronics Performance Practices

  Wednesday, May 4, from 4 to 5 PM
  Lewis Music Library, MIT

  New works for live electronics with laptops, iPhones, circuits, and other
  sonological mechanisms.

  Free and open to the public.
  ```

16.2. Performance Frameworks Groups

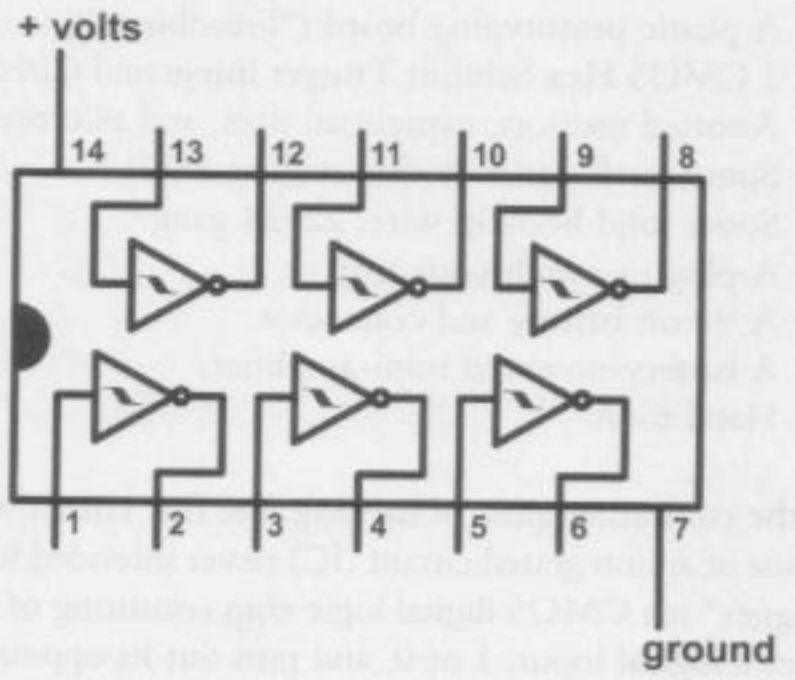
- `Groups of 2 to 4 students - names removed for privacy`

16.3. Reading: Collins, Handmade Electronic Music

- Collins, N. 2009. *Handmade Electronic Music: The Art of Hardware Hacking*. 2nd ed. New York: Routledge.
- The Hex Schmitt Trigger digital logic chip: six inverters on a chip
- Given an input of 1 (9 volts) output 0 (0 volts) and vica versa

- The resistor permits feedback, causing alternation between 9 and 0 volts and producing a squarish sound wave
- 74C14 Pins

Figure 18.1
74C14 pinout.



- Using a resistor to create feedback oscillation

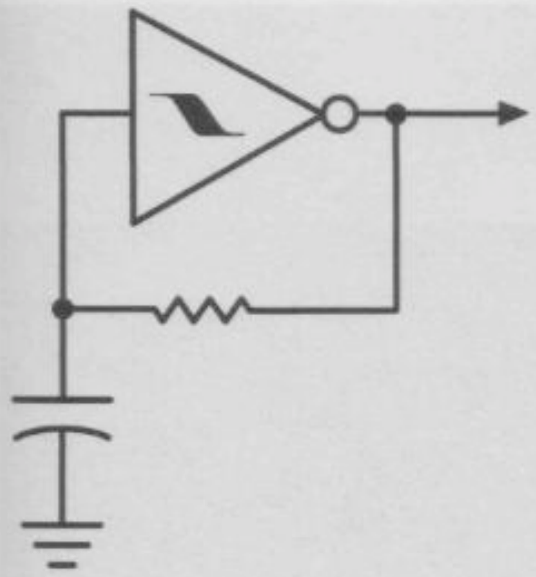
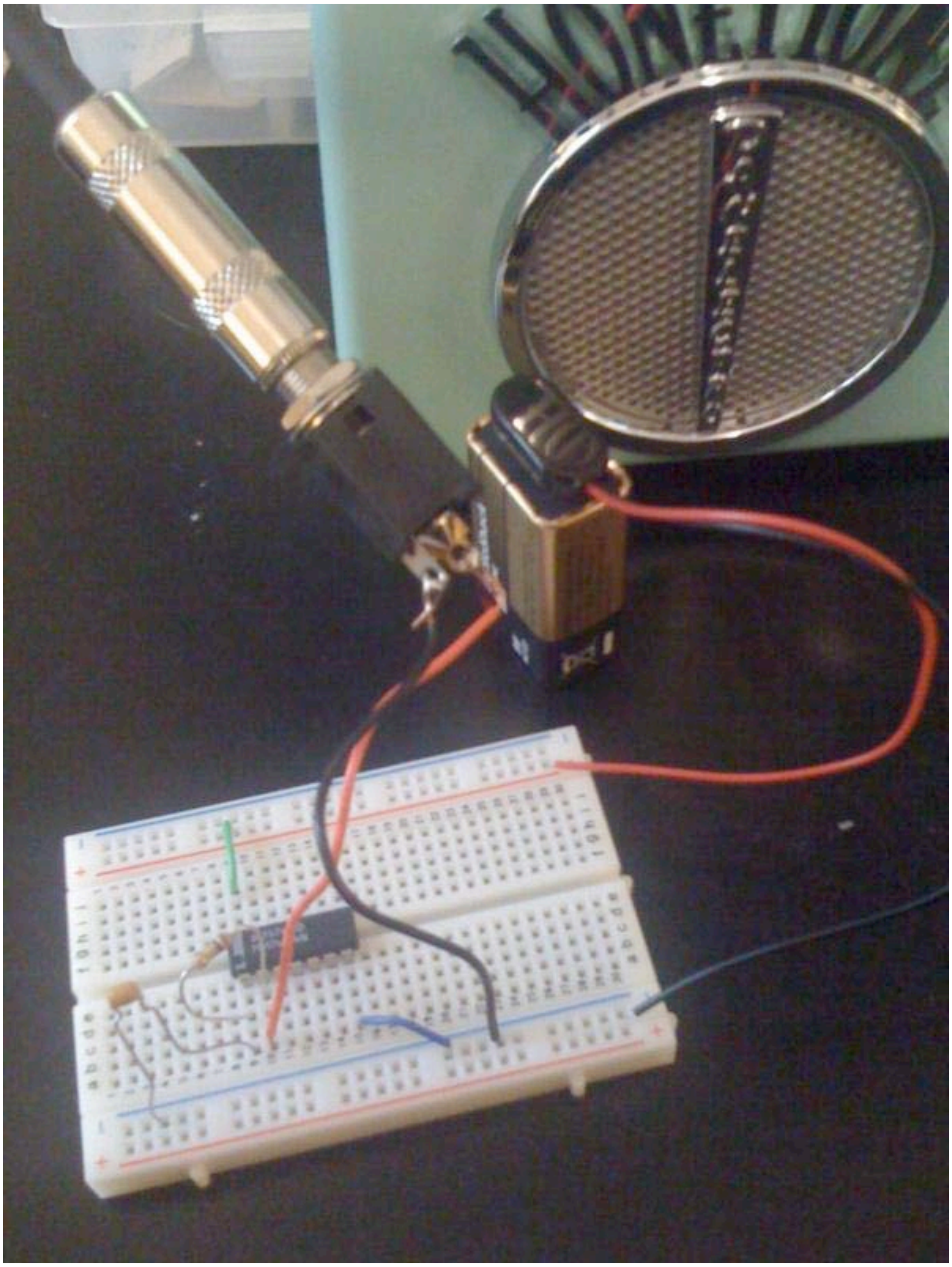


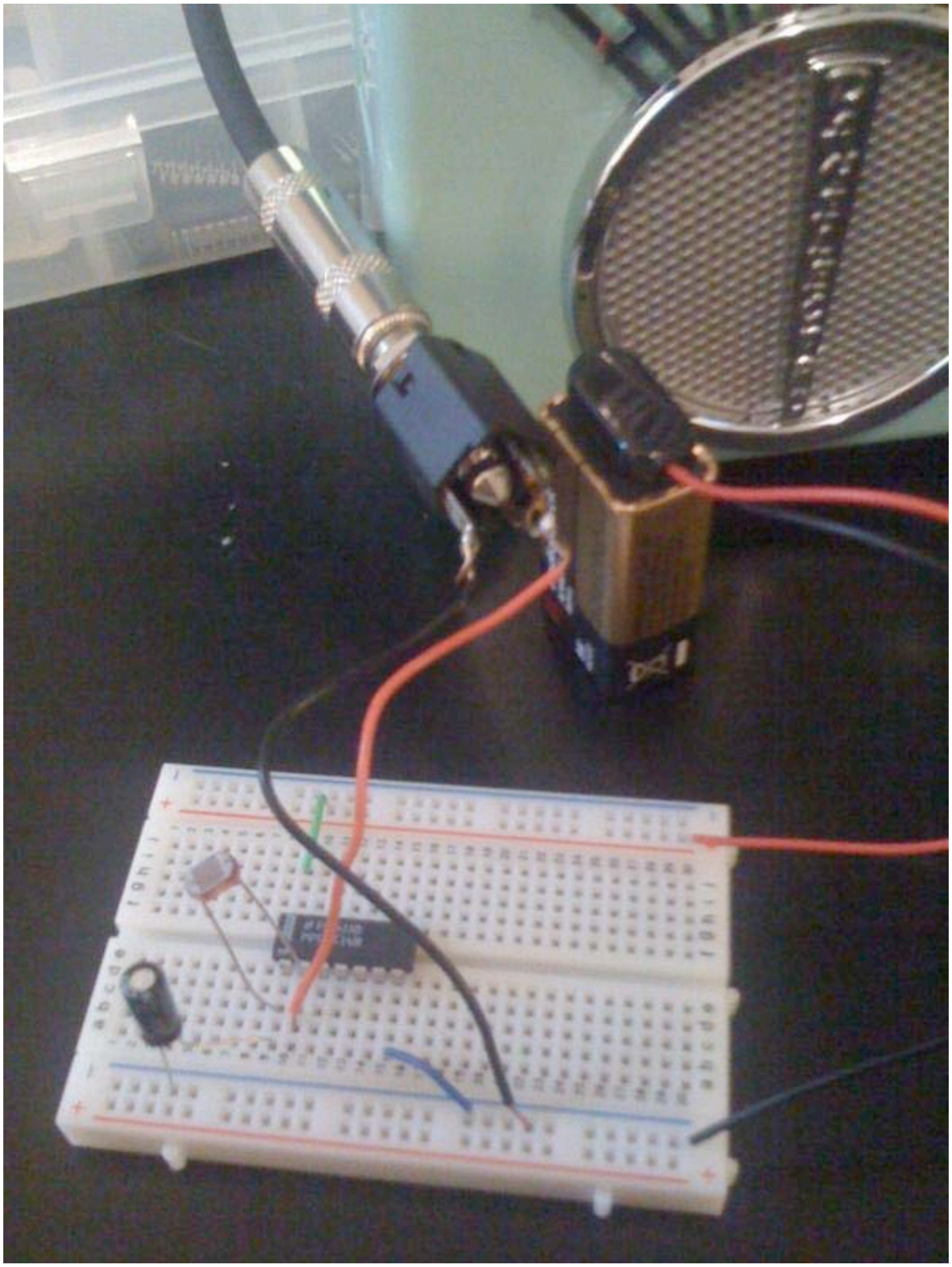
Figure 18.8
Schematic representation of our oscillator.

- Breadboard



- Varying resistance changes the frequency of oscillation
- A photoresistor decreases resistance with more light, increases resistance with less light

- Varying the capacitor sets the range of oscillation
- Breadboard



16.4. Exercise: Improvisation with Controller/Interface/Instrument Design 1

- Load: instruments created for Controller/Interface/Instrument Design 1
- Ensemble: each person enter staggered; do ostinato layers

16.5. Work II

- Load: arizaWork02-performance*.test.pd

  martingale/comopositions/arixaWork02/arizaWork02-performance*.test.pd
- Focus on texture and heterophony

Chapter 17. Meeting 17, Practices: Approaches to Composing Improvisations

17.1. Announcements

- Due today: Controller/Interface/Instrument Design 2 Proposal
- Quiz on Monday
- Due Wednesday 13 April: Performance Frameworks Draft
- Please test and practice Work II for next Wednesday

17.2. Approaches to Composing Improvisations

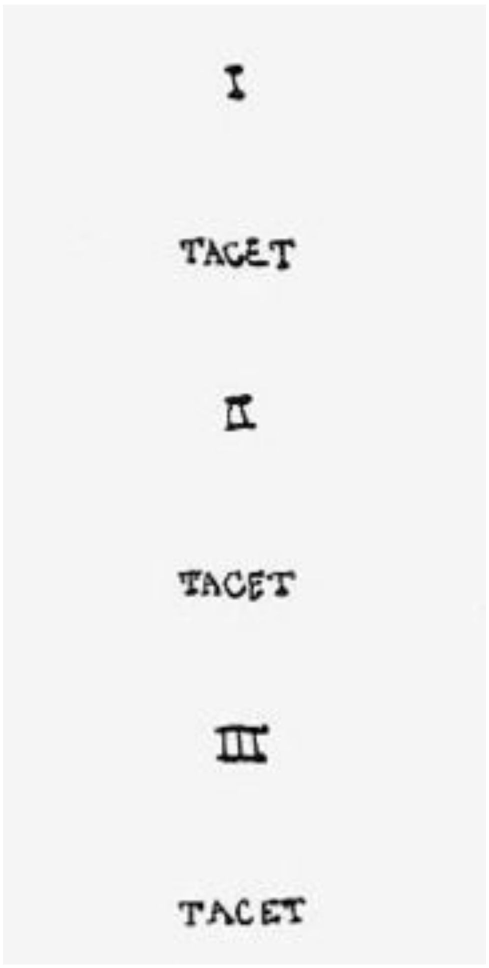
- Scripts: partitioned sections (scenes) with verbal instructions
- Games: encouraging interaction through play, desire to win
- Graphic Scores: directly or indirectly represent or suggest musical parameters

17.3. Approaches to Composing Improvisations: Scripts

- Problem of using natural language
- John Cage: 4'33"

Cover page of John Cage's *4' 33"* (Tacet, any instrument or combination of instruments) removed due to copyright restrictions.

Paragraph of text instructions to performers from John Cage's *4' 33"* (Tacet, any instrument or combination of instruments) removed due to copyright restrictions.



17.4. Approaches to Composing Improvisations: Games

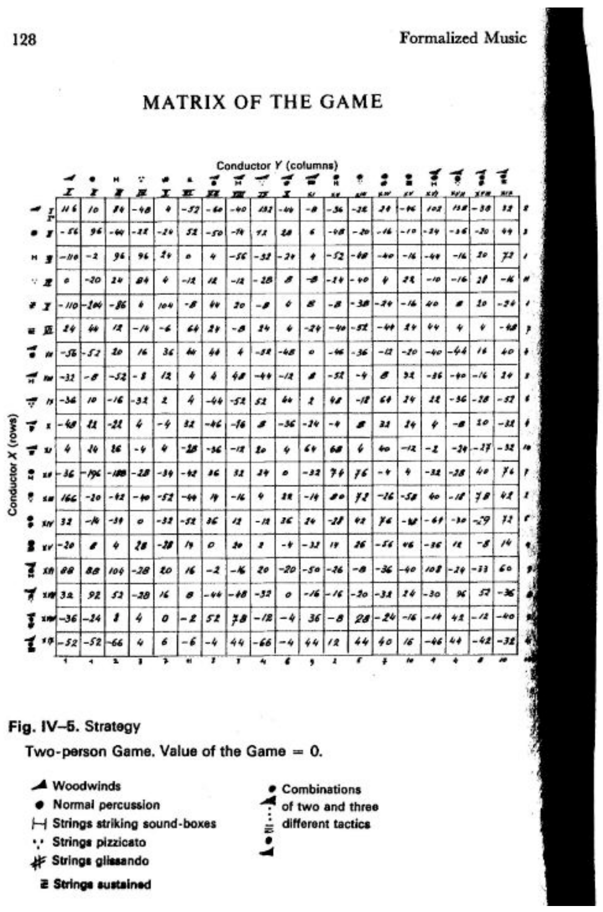
- Iannis Xenakis
    - Works using game theory
    - Duel (1958-59): Two conductors and two orchestras, with each conductors back to the other

      6 categories of events, where certain combinations from each group produce quality scores

      Event types are given to each orchestra without the other’s knowledge

      An orchestra wins by playing the best strategy
    - Strategie (1962): two orchetras with conductors back to back

      6 classes of textures



128 Formalized Music

MATRIX OF THE GAME

Conductor Y (columns)

Conductor X (rows)

	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV	XV	XVI	XVII	XVIII	XIX	
I	116	10	84	-48	4	-52	-60	-40	132	-44	-8	-36	-28	24	-46	102	138	-38	32	2
II	-56	96	-64	-28	-24	52	-50	-74	72	28	6	-48	-20	-16	-10	-24	-36	-20	44	3
III	-110	-2	96	96	24	0	4	-56	-32	-24	4	-52	-68	-40	-16	-44	-16	20	72	1
IV	0	-20	24	84	4	-12	12	-12	-28	8	-8	-24	-40	4	22	-10	-16	28	-16	11
V	-110	-204	-86	6	104	-8	44	20	-8	6	8	-8	-38	-24	-16	40	8	20	-24	1
VI	24	44	12	-14	-6	64	24	-8	24	4	-24	-40	-52	-44	24	44	4	4	-48	7
VII	-56	-52	20	16	36	44	44	4	-52	-48	0	-46	-36	-12	-20	-40	-44	16	40	4
VIII	-32	-8	-52	-8	12	4	4	48	-44	-12	8	-52	-4	8	32	-36	-40	-16	24	3
IX	-36	10	-16	-32	2	4	-44	-52	52	64	2	48	-18	64	24	22	-36	-28	-57	6
X	-48	22	-22	4	-4	32	-46	-56	8	-36	-24	-4	8	32	24	4	-8	20	-32	4
XI	4	24	26	-4	4	-28	-36	-12	20	4	64	68	6	40	-12	-2	-24	-27	-32	10
XII	-36	-196	-188	-28	-34	-42	26	32	24	0	-32	74	76	-4	4	-32	-28	40	74	7
XIII	166	-20	-42	-40	-52	-44	14	-16	4	22	-14	80	72	-26	-58	40	-18	78	42	2
XIV	32	-16	-34	0	-32	-52	36	12	-12	36	24	-28	42	76	-48	-64	-30	-29	72	6
XV	-20	8	4	28	-38	14	0	20	2	-4	-32	14	26	-56	46	-36	12	-8	14	4
XVI	88	88	104	-28	20	16	-2	-16	20	-20	-50	-26	-8	-36	-40	108	-24	-33	60	9
XVII	32	92	52	-28	16	8	-44	-48	-32	0	-16	-16	-20	-32	24	-30	96	57	-36	[illegible]
XVIII	-36	-24	8	4	0	-2	52	78	-12	-4	36	-8	28	-24	-16	-14	42	-12	-40	[illegible]
XIX	-52	-52	-66	4	6	-6	-4	44	-66	-4	44	12	44	40	16	-46	44	-42	-32	[illegible]
	1	4	2	1	7	11	3	1	4	6	5	2	6	7	10	4	4	8	10	[illegible]

Fig. IV–5. Strategy

Two-person Game. Value of the Game = 0.

- Woodwinds
- Normal percussion
- Strings striking sound-boxes
- Strings pizzicato
- Strings glissando
- Strings sustained
- Combinations of two and three different tactics

Strategy, Linear Programming, and Musical Composition 129

Conductor Y (columns) — Strategy for X

	3	0	−1	
	3	−3	3	
	−5	4	−3	
Strategy for Y				

	3	−1	0	
	−5	−1	4	
	3	1	−3	
Strategy for Y				

Fig. IV–6

Two-person Zero-sum Game.
Value of the Game = 1/11.
This game is not fair for Y.

Two-person Zero-sum Game.
Value of the Game = 0.
This game is fair for both conductors.

- Woodwinds
- Normal percussion
- H Strings striking sound-boxes
- Strings pizzicato
- Strings glissando
- Strings sustained
- Combinations of two distinct tactics
- Combinations of three distinct tactics

Simplification of the 19 × 19 Matrix

To make first performances easier, the conductors might use an equivalent 3 × 3 matrix derived from the 19 × 19 matrix in the following manner:

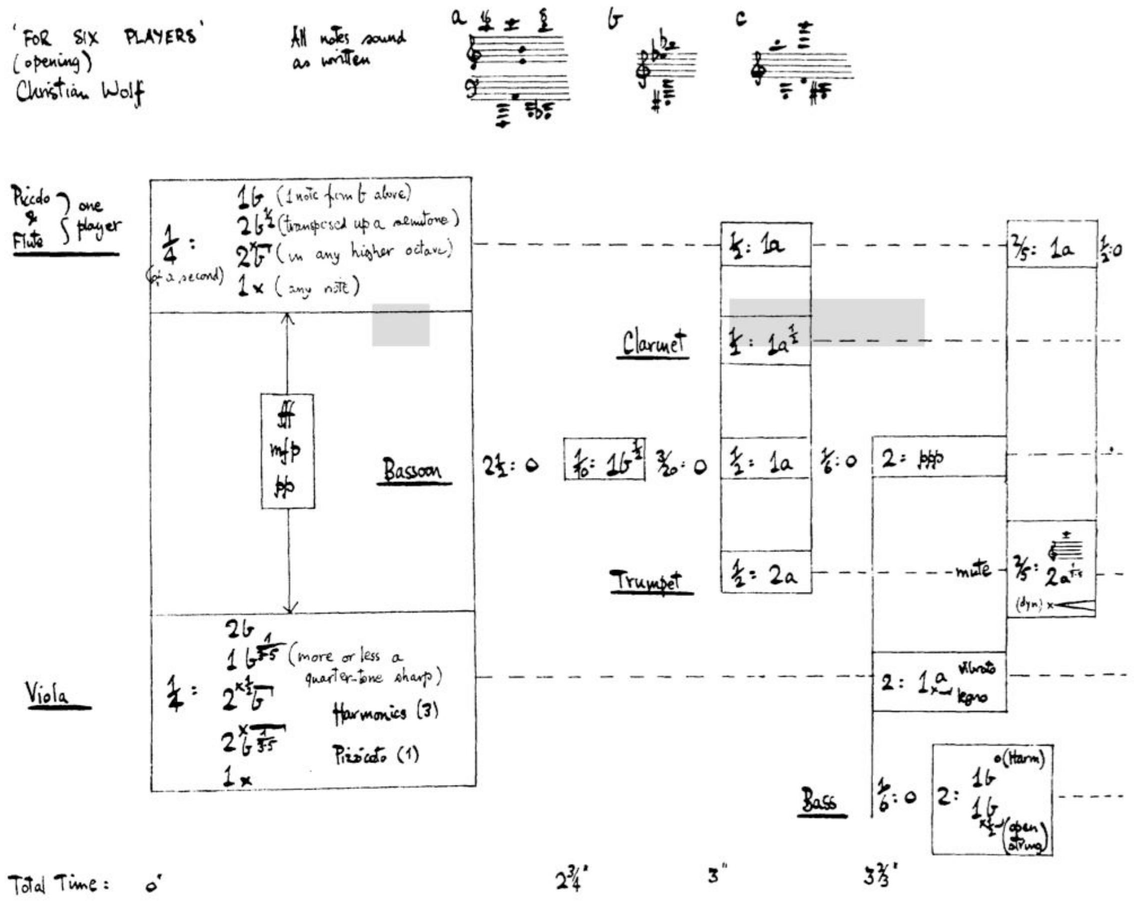
Let there be a fragment of the matrix containing row tactics $r + 1, \ldots, r + m$ and column tactics $s + 1, \ldots, s + n$ with the respective probabilities $q_{r+1}, \ldots, q_{r+m}$ and $k_{s+1}, \ldots, k_{s+n}$.

$$\begin{array}{cccccc} & k_{s+1} & & k_{s+j} & & k_{s+n} \\ q_{r+1} & a_{r+1,s+1} & \cdots & a_{r+1,s+j} & \cdots & a_{r+1,s+n} \\ & \vdots & & \vdots & & \vdots \\ q_{r+i} & a_{r+i,s+1} & \cdots & a_{r+i,s+j} & \cdots & a_{r+i,s+n} \\ & \vdots & & \vdots & & \vdots \\ q_{r+m} & a_{r+m,s+1} & \cdots & a_{r+m,s+j} & \cdots & a_{r+m,s+n} \end{array}$$

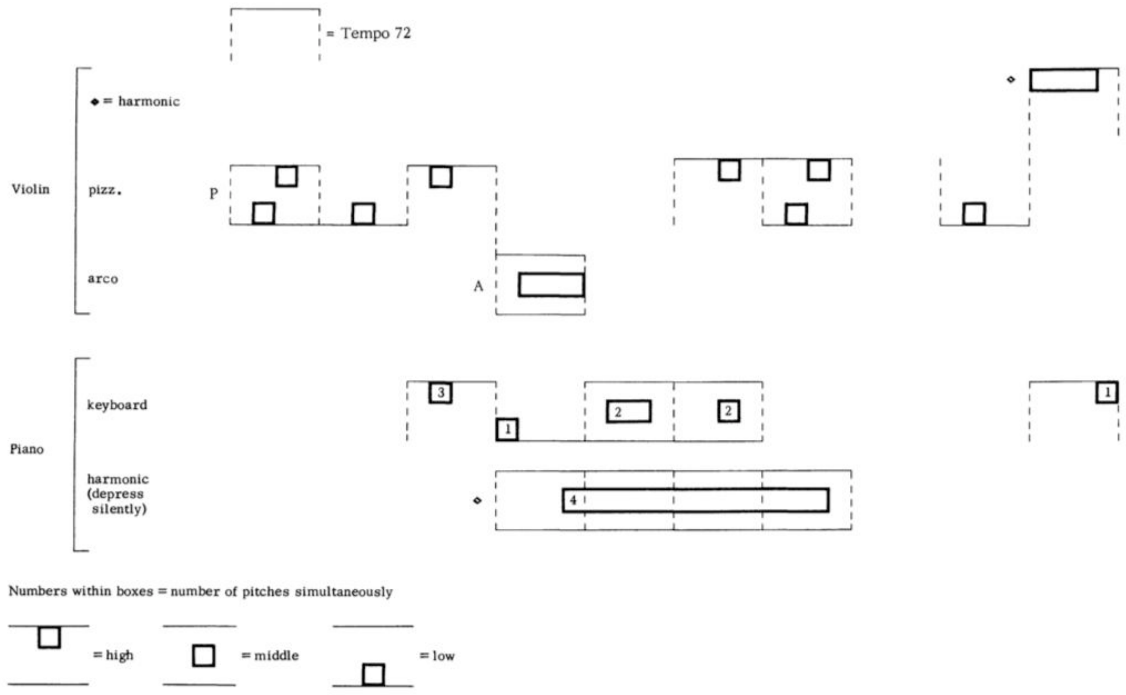
Courtesy of Pendragon Press. Used with permission.

17.5. Approaches to Composing Improvisations: Graphic Scores

- What would a graphic score of the first two sections of Work II look like?
- Common assumptions about time and pitch: horizontal and vertical space
- Christian Wolff: For six players



- Morton Feldman: Projection IV


= Tempo 72
◆ = harmonic
Violin
pizz.
arco
P
A
Piano
keyboard
harmonic (depress silently)
3
1
2
2
4
1
Numbers within boxes = number of pitches simultaneously
= high
= middle
= low

Ex. 4, Feldman, *Projection IV,* first page

- Cardew (1936-81): Treatise (1963-67)

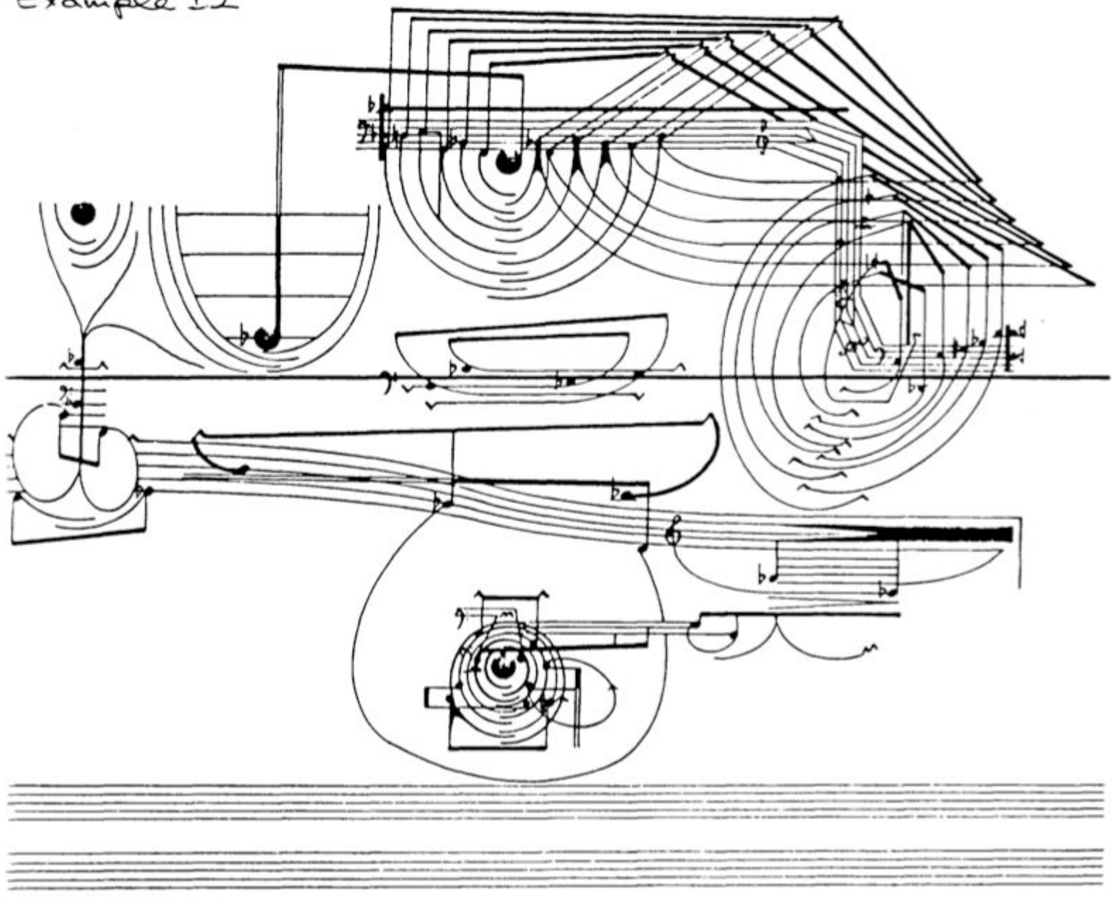
  Assistant to Stockhausen at WDR Cologne from 1958 to 1960

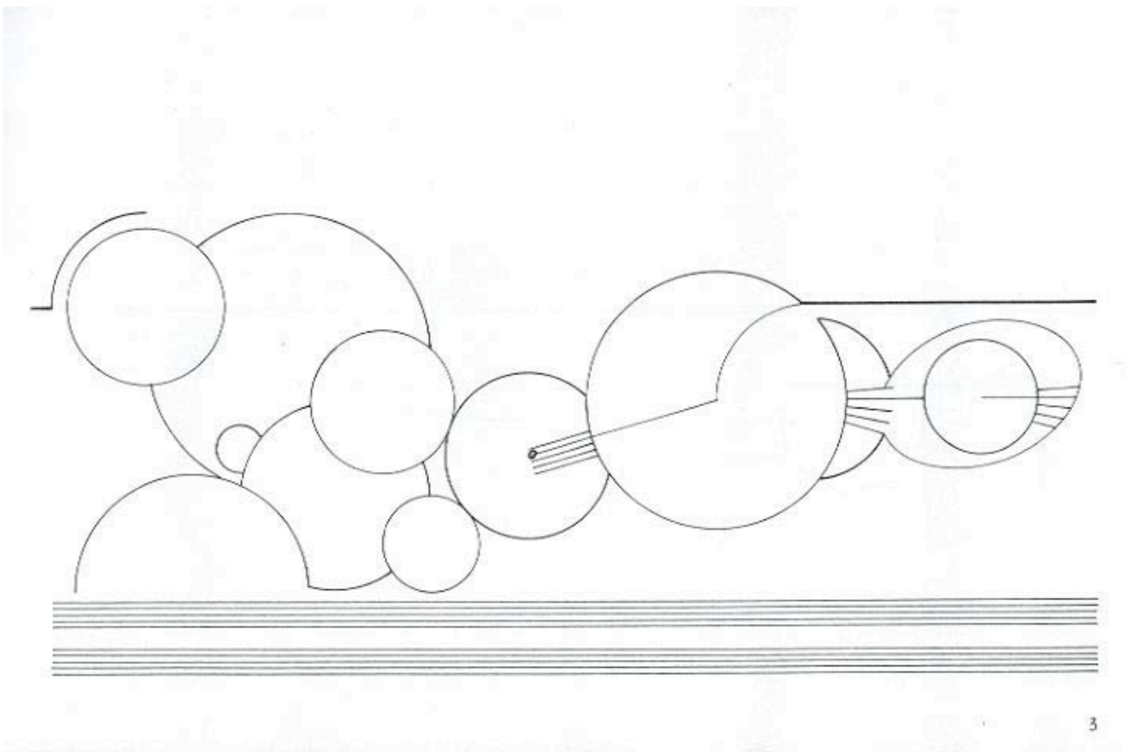
  Joined AMM in 1966

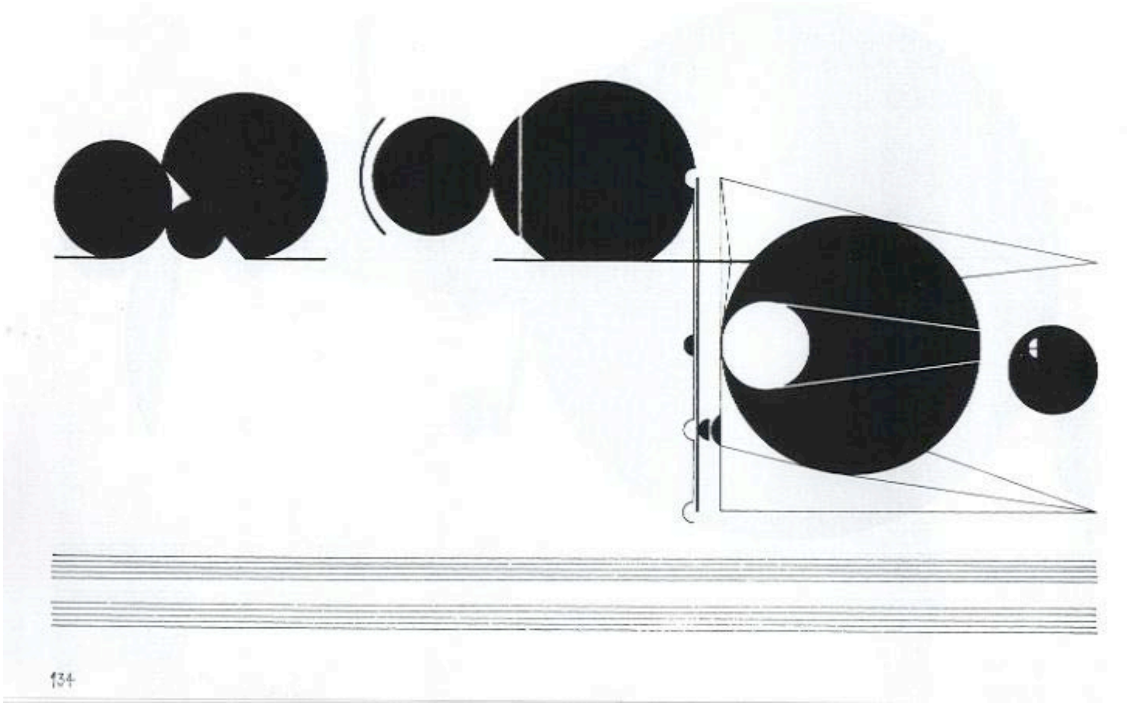
  Stated that space does not correspond to sound; stated that goal was musicalness in the notating

  Score is 193 pages

Example 12







17.6. Reading: Dennis, Cardew's Treatise

- Dennis, B. 1991. "Cardew's 'Treatise' (Mainly the Visual Aspects)." *Tempo* 177: pp. 10-16.

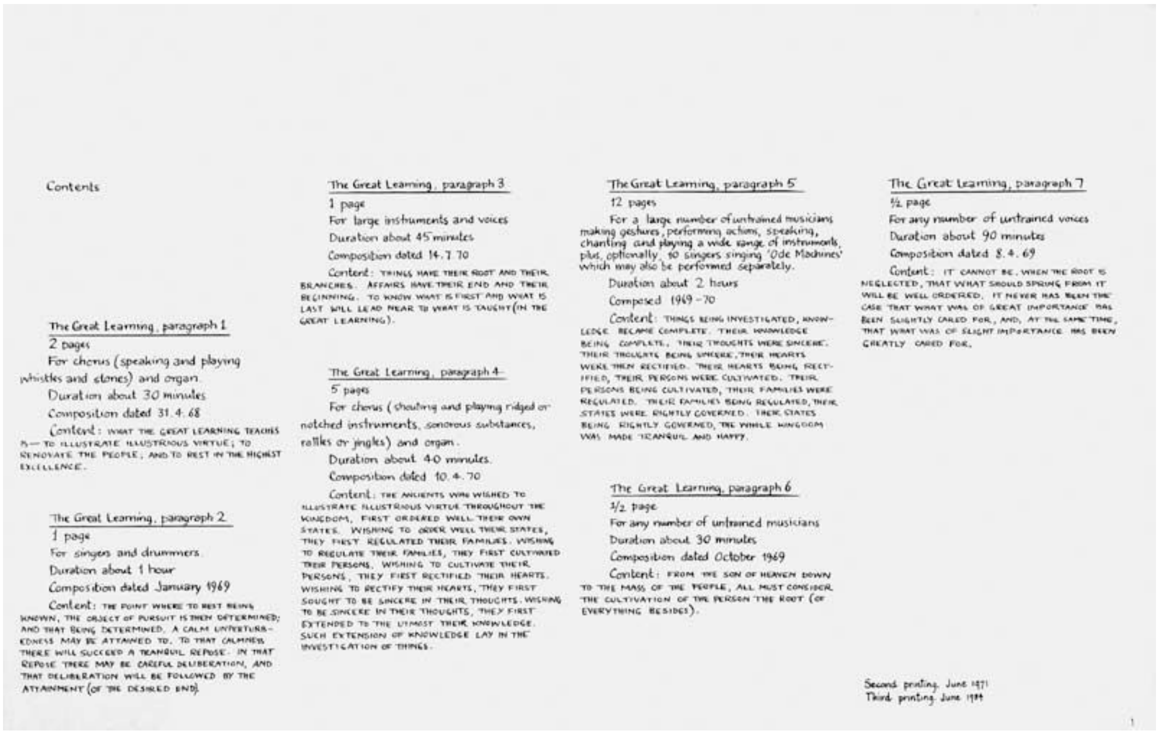
- What are the main elements of the score to Treatise?
- What elements from standard western notation are maintained?
- What criteria did Cardew look for in people performing this work?
- Listening: Cornelius Cardew, Treatise

  Audio: http://ubu.artmob.ca/sound/cardew_cornelius/memorial_concert/Cardew-Cornelius_Memorial-Concert_1-03_Treatise.mp3

17.7. Listening: Cardew

- Listening: Cornelius Cardew, "Paragraph 2," The Great Learning, 2000

- Scratch orchestra formed in 1968: a large, experimental improvisatory ensemble for interpreting The Great Learning
- Great Learning score



Contents

The Great Learning, paragraph 1

2 pages

For chorus (speaking and playing whistles and stones) and organ.

Duration about 30 minutes

Composition dated 31.4.68

Content: WHAT THE GREAT LEARNING TEACHES IS — TO ILLUSTRATE ILLUSTRIOUS VIRTUE; TO RENOVATE THE PEOPLE; AND TO REST IN THE HIGHEST EXCELLENCE.

The Great Learning, paragraph 2

1 page

For singers and drummers

Duration about 1 hour

Composition dated January 1969

Content: THE POINT WHERE TO REST BEING KNOWN, THE OBJECT OF PURSUIT IS THEN DETERMINED; AND THAT BEING DETERMINED, A CALM UNPERTURBEDNESS MAY BE ATTAINED TO. TO THAT CALMNESS THERE WILL SUCCEED A TRANQUIL REPOSE. IN THAT REPOSE THERE MAY BE CAREFUL DELIBERATION, AND THAT DELIBERATION WILL BE FOLLOWED BY THE ATTAINMENT (OF THE DESIRED END).

The Great Learning, paragraph 3

1 page

For large instruments and voices

Duration about 45 minutes

Composition dated 14.7.70

Content: THINGS HAVE THEIR ROOT AND THEIR BRANCHES. AFFAIRS HAVE THEIR END AND THEIR BEGINNING. TO KNOW WHAT IS FIRST AND WHAT IS LAST WILL LEAD NEAR TO WHAT IS TAUGHT (IN THE GREAT LEARNING).

The Great Learning, paragraph 4

5 pages

For chorus (shouting and playing ridged or notched instruments, sonorous substances, rattles or jingles) and organ.

Duration about 40 minutes.

Composition dated 10.4.70

Content: THE ANCIENTS WHO WISHED TO ILLUSTRATE ILLUSTRIOUS VIRTUE THROUGHOUT THE KINGDOM, FIRST ORDERED WELL THEIR OWN STATES. WISHING TO ORDER WELL THEIR STATES, THEY FIRST REGULATED THEIR FAMILIES. WISHING TO REGULATE THEIR FAMILIES, THEY FIRST CULTIVATED THEIR PERSONS. WISHING TO CULTIVATE THEIR PERSONS, THEY FIRST RECTIFIED THEIR HEARTS. WISHING TO RECTIFY THEIR HEARTS, THEY FIRST SOUGHT TO BE SINCERE IN THEIR THOUGHTS. WISHING TO BE SINCERE IN THEIR THOUGHTS, THEY FIRST EXTENDED TO THE UTMOST THEIR KNOWLEDGE. SUCH EXTENSION OF KNOWLEDGE LAY IN THE INVESTIGATION OF THINGS.

The Great Learning, paragraph 5

12 pages

For a large number of untrained musicians making gestures, performing actions, speaking, chanting and playing a wide range of instruments, plus, optionally, 10 singers singing 'Ode Machines' which may also be performed separately.

Duration about 2 hours

Composed 1969–70

Content: THINGS BEING INVESTIGATED, KNOWLEDGE BECAME COMPLETE. THEIR KNOWLEDGE BEING COMPLETE, THEIR THOUGHTS WERE SINCERE. THEIR THOUGHTS BEING SINCERE, THEIR HEARTS WERE THEN RECTIFIED. THEIR HEARTS BEING RECTIFIED, THEIR PERSONS WERE CULTIVATED. THEIR PERSONS BEING CULTIVATED, THEIR FAMILIES WERE REGULATED. THEIR FAMILIES BEING REGULATED, THEIR STATES WERE RIGHTLY GOVERNED. THEIR STATES BEING RIGHTLY GOVERNED, THE WHOLE KINGDOM WAS MADE TRANQUIL AND HAPPY.

The Great Learning, paragraph 6

1/2 page

For any number of untrained musicians

Duration about 30 minutes

Composition dated October 1969

Content: FROM THE SON OF HEAVEN DOWN TO THE MASS OF THE PEOPLE, ALL MUST CONSIDER THE CULTIVATION OF THE PERSON THE ROOT (OF EVERYTHING BESIDES).

The Great Learning, paragraph 7

½ page

For any number of untrained voices

Duration about 90 minutes

Composition dated 8.4.69

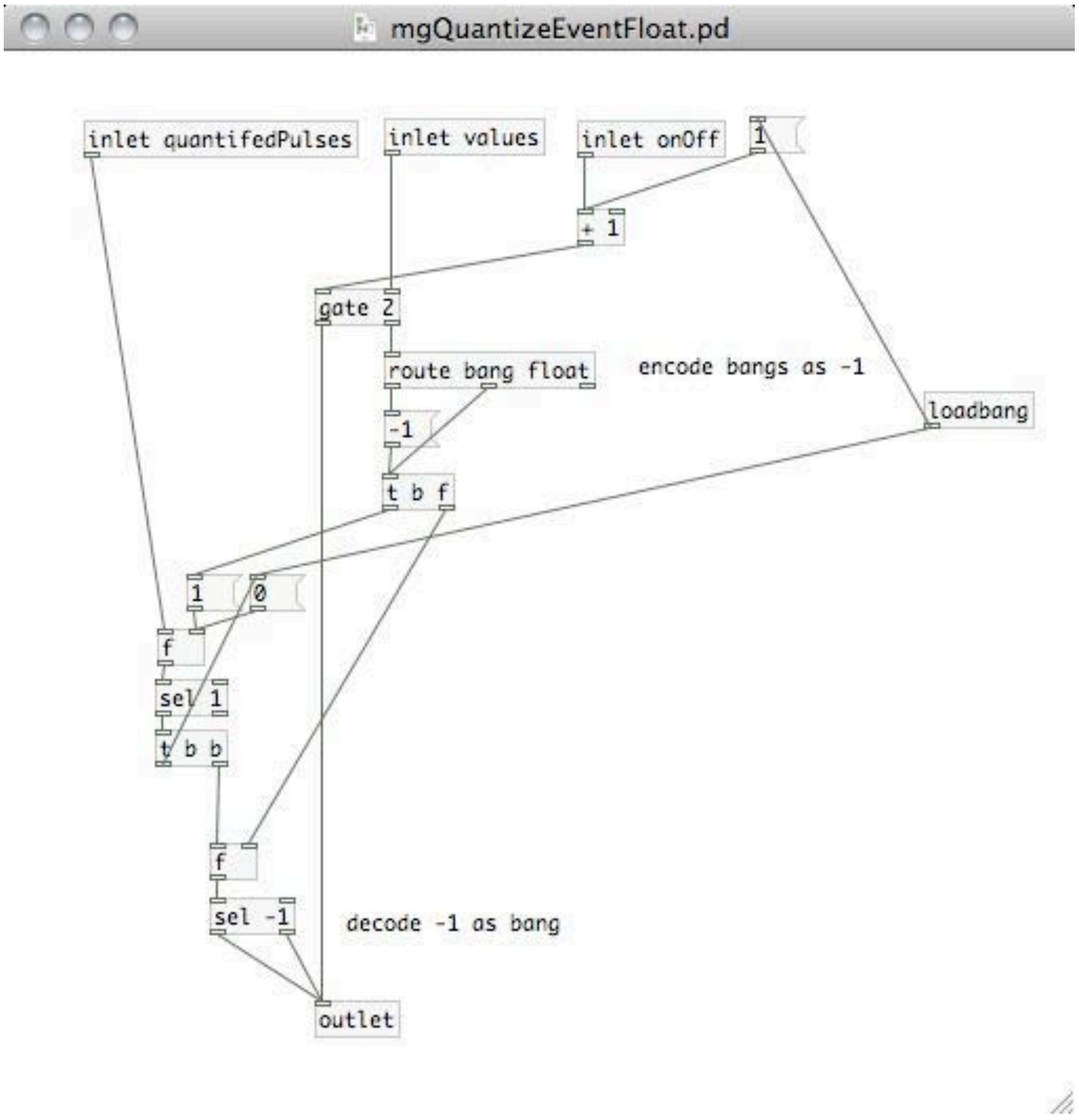
Content: IT CANNOT BE, WHEN THE ROOT IS NEGLECTED, THAT WHAT SHOULD SPRING FROM IT WILL BE WELL ORDERED. IT NEVER HAS BEEN THE CASE THAT WHAT WAS OF GREAT IMPORTANCE HAS BEEN SLIGHTLY CARED FOR, AND, AT THE SAME TIME, THAT WHAT WAS OF SLIGHT IMPORTANCE HAS BEEN GREATLY CARED FOR.

Second printing June 1971
Third printing June 1984

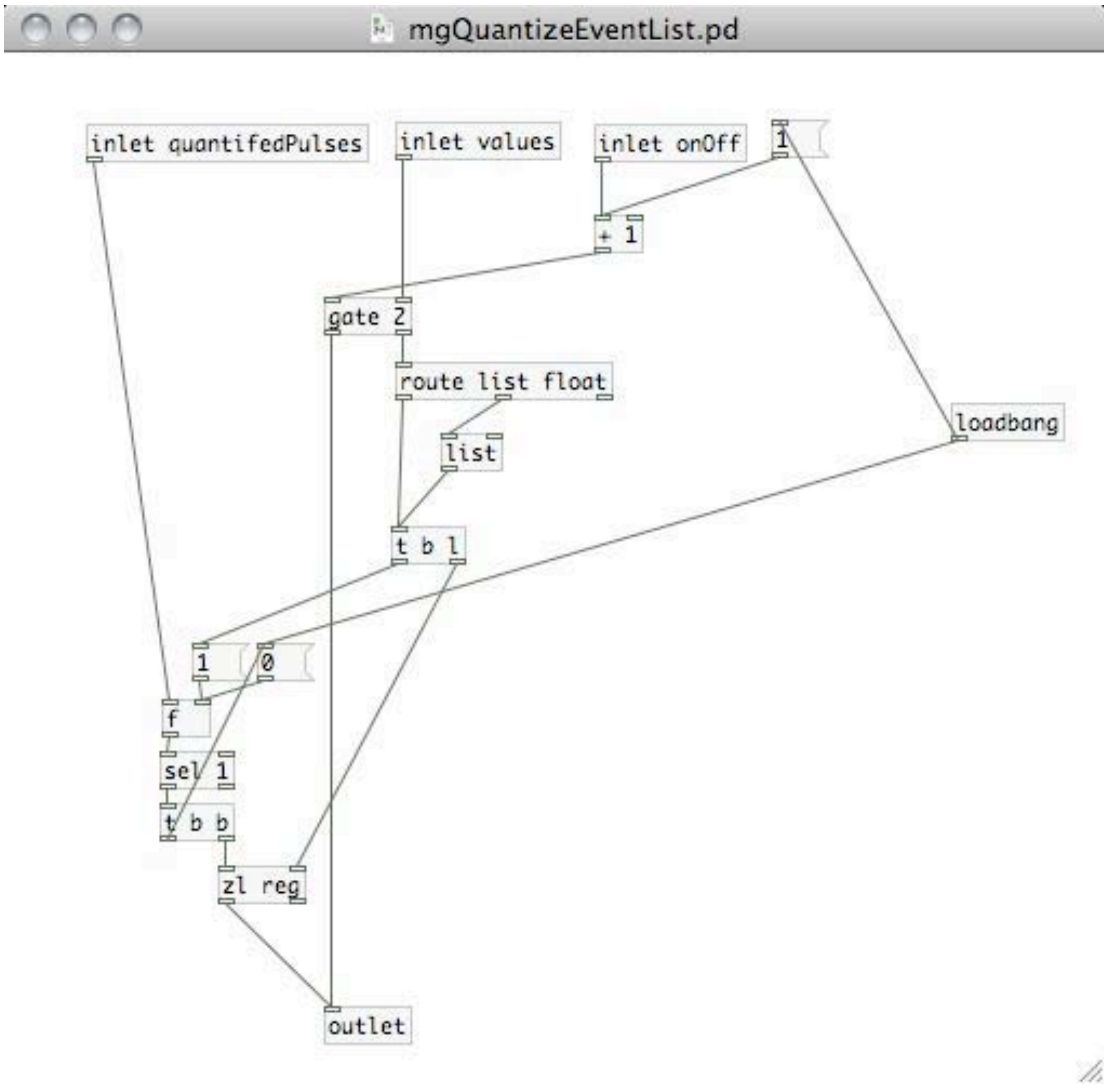
1

17.8. Quantizing a Real-Time Pulse Stream

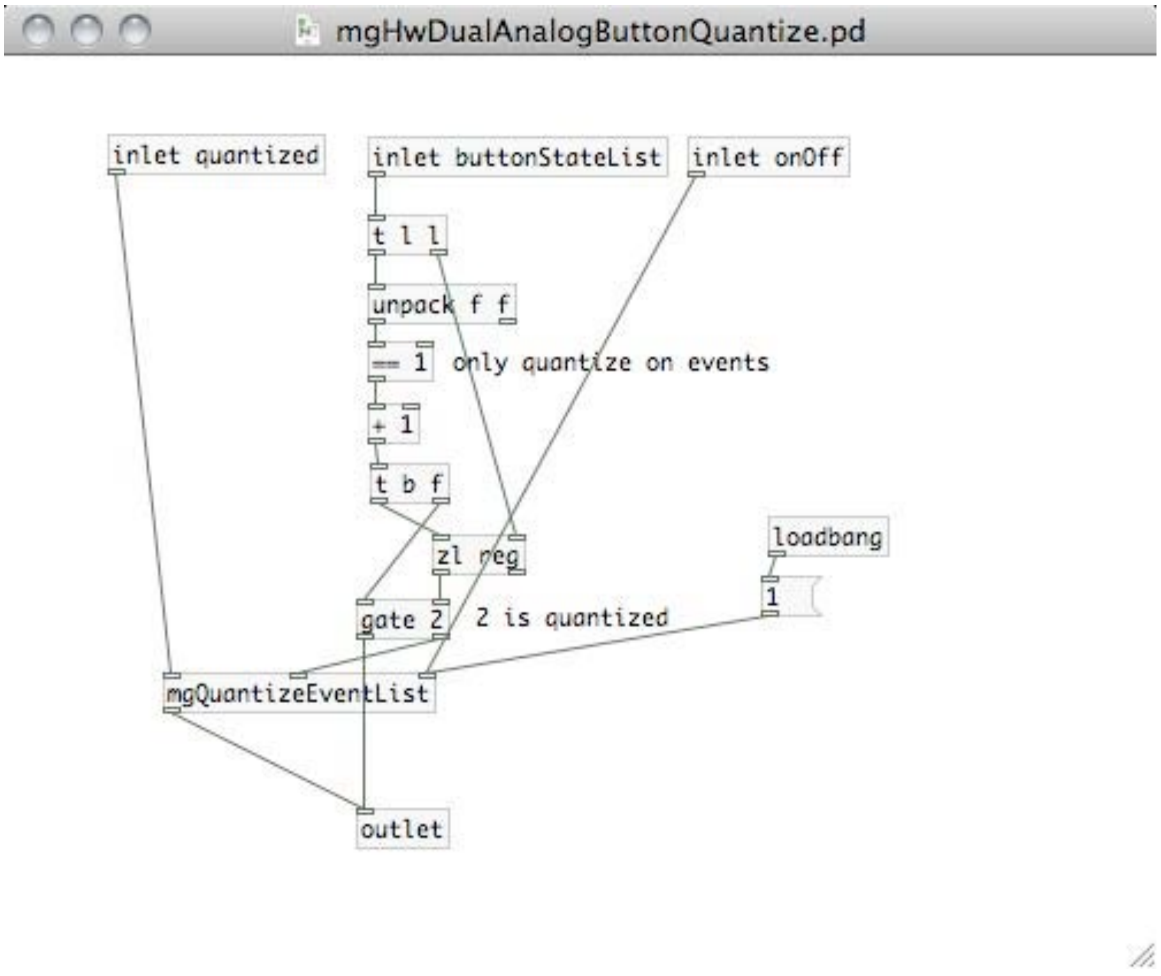
- [mgQuantizeEventFloat]



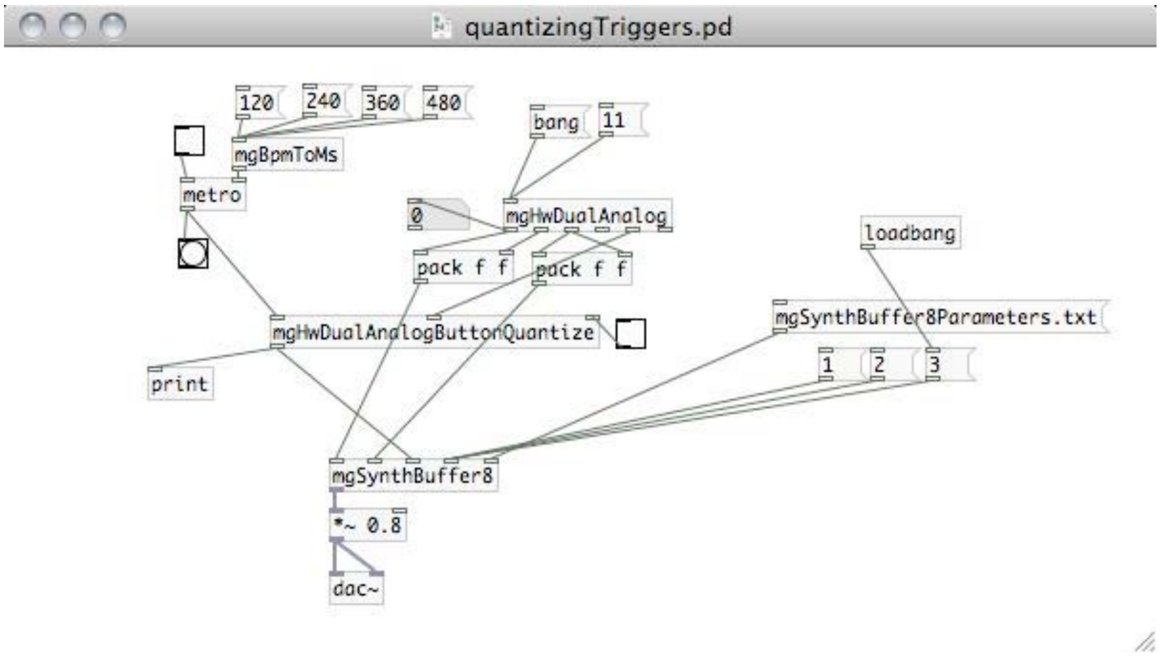
- [mgQuantizeEventList]



- [mgHwDualAnalogButtonQuantize]

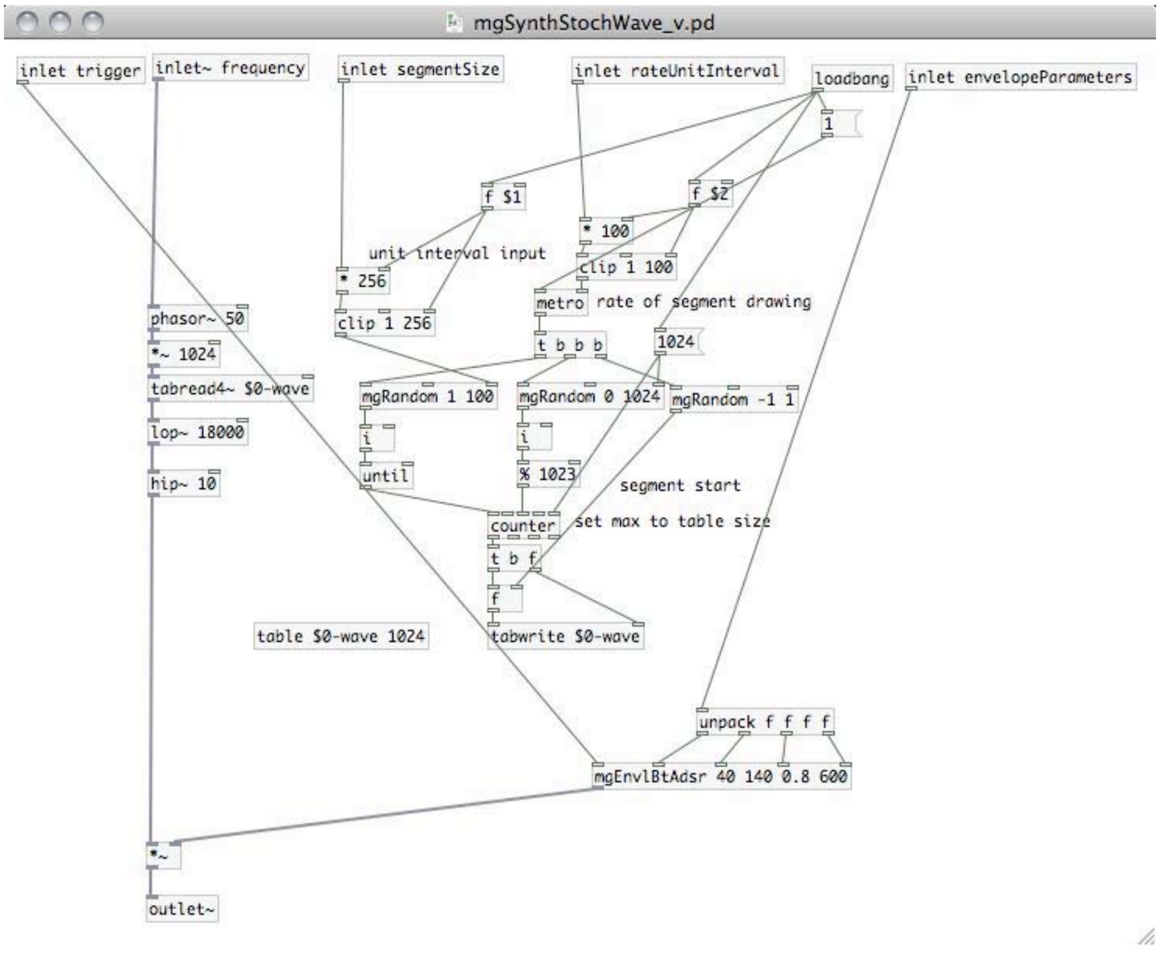


- Example: quantizingTriggers.pd

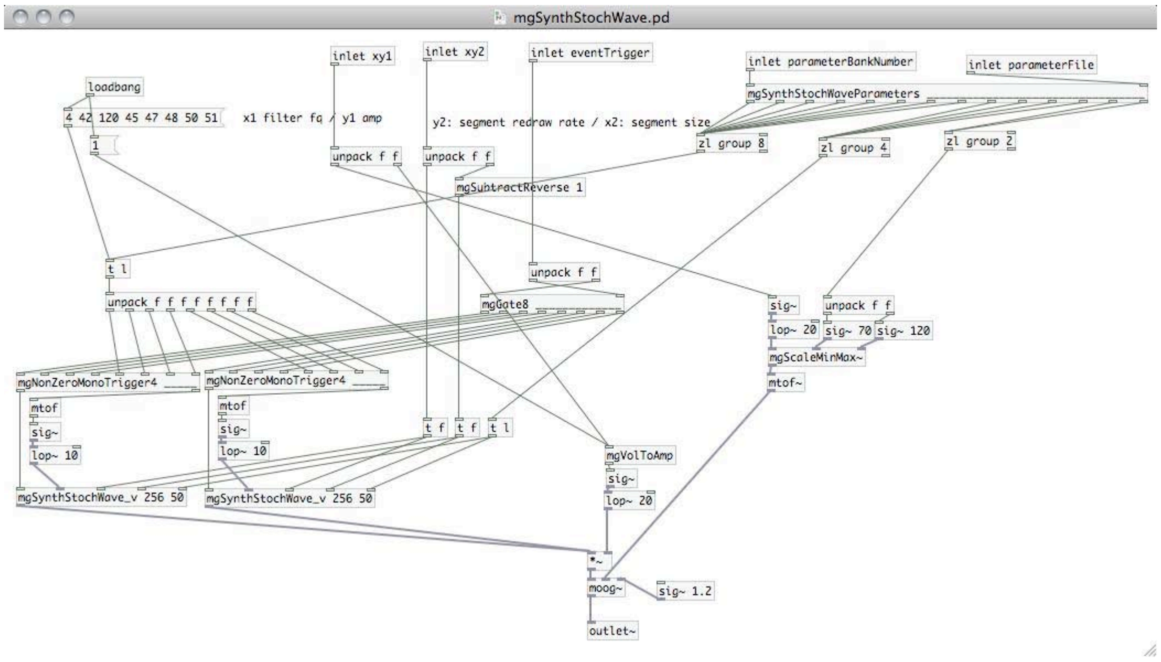


17.9. Dynamic Stochastic Synthesis

- Technique of algorithmic waveform construction developed by Xenakis
- [mgSynthStochWave_v]

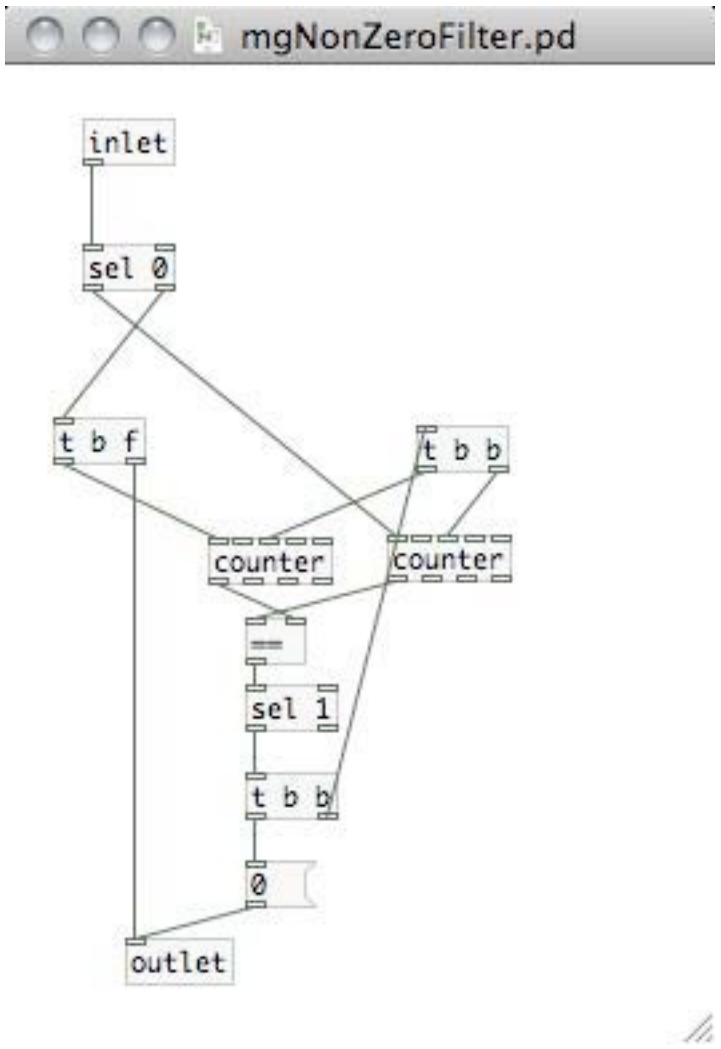


• [mgSynthStochWave]

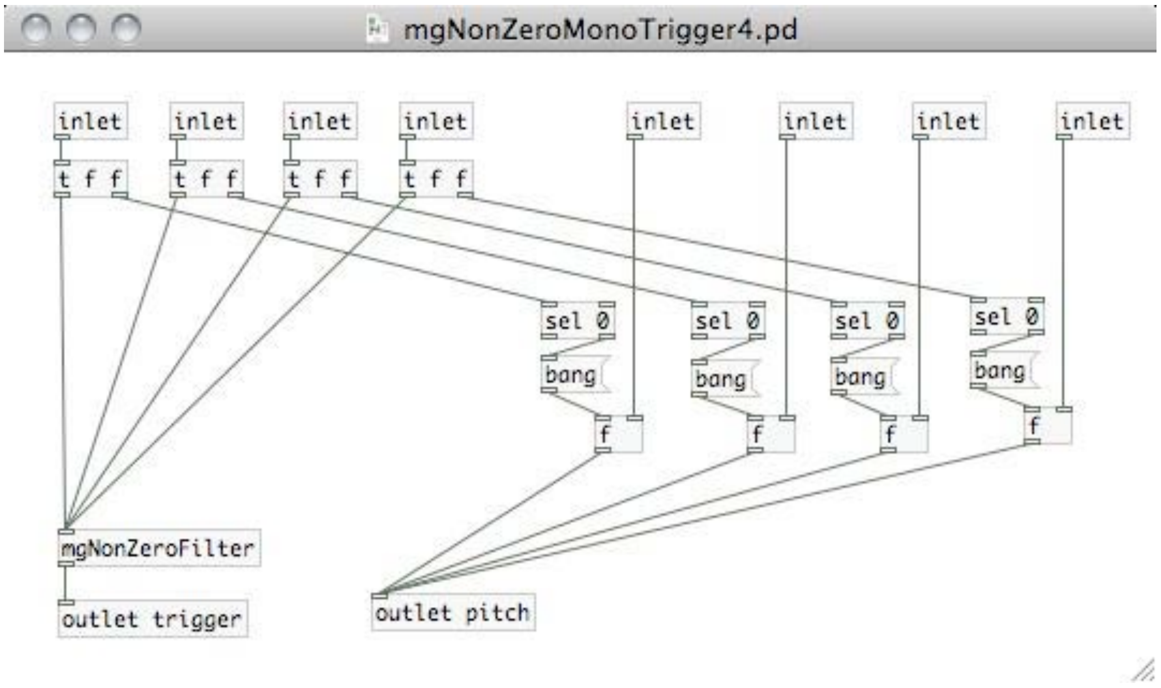


17.10. Producing a Monophonic Instrument

- Efficiency of using a single oscillator
- Challenge of triggering
- [mgNonZeroFilter]



- [mgNonZeroMonoTrigger4]


mgNonZeroMonoTrigger4.pd
inlet
inlet
inlet
inlet
inlet
inlet
inlet
inlet
t f f
t f f
t f f
t f f
sel 0
sel 0
sel 0
sel 0
bang
bang
bang
bang
f
f
f
f
mgNonZeroFilter
outlet trigger
outlet pitch

Chapter 18. Meeting 18, Practices: Timing and Networking

18.1. Announcements

- Due this Wednesday 13 April: Performance Frameworks Draft
- Bring amps and controllers to next class
- Monday 18 April: No class (Patriots day)
- Due next Wednesday 20 April: Instrument 2 Drafts/Prototypes
- Due next Wednesday 20 April: Performance Frameworks

18.2. Quiz 3

- ?

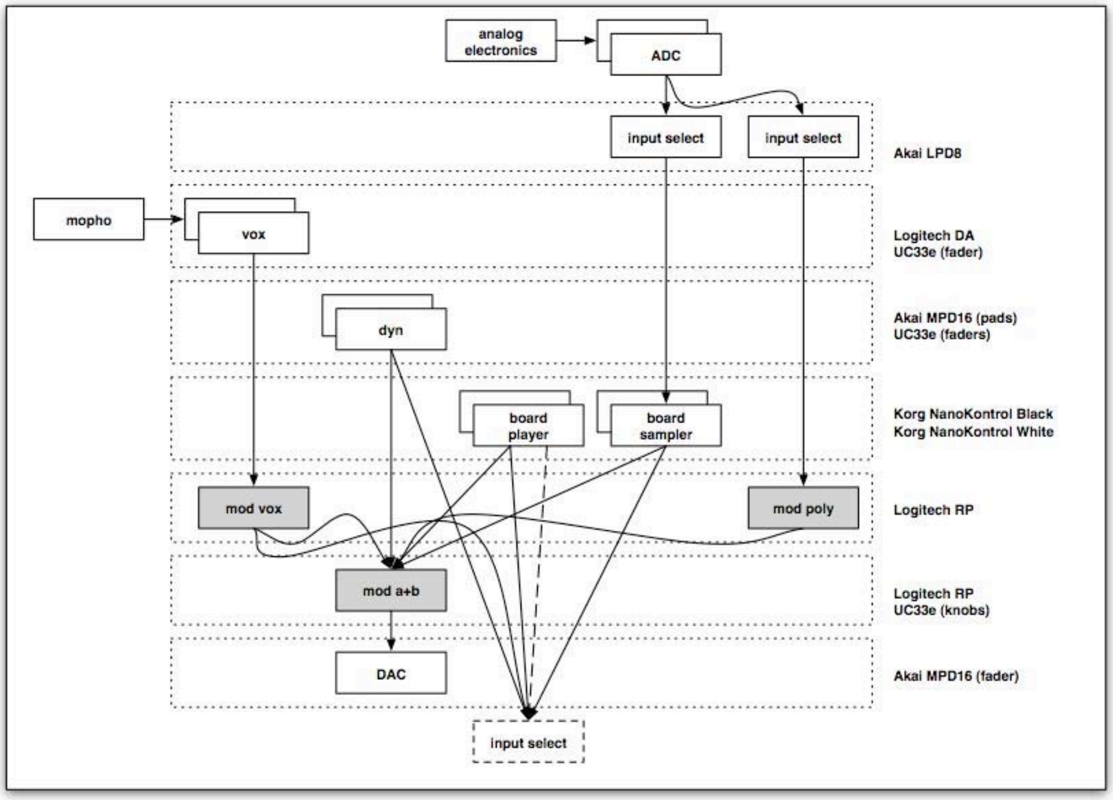
18.3. Listening: KIOKU

- Kioku, “Pinari,” Both far and near, 2007

18.4. A Large Live Electronics System: Pulsefact

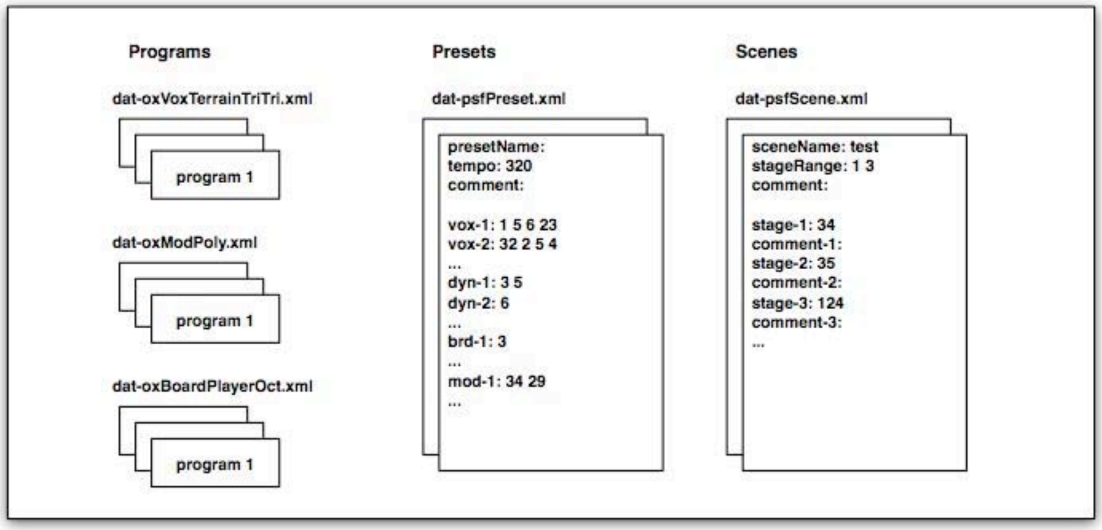
- Developed for KIOKU, expanded for pulsefact project
- Based on over 10 years of development in Max/MSP
- Permits multiple sound sources, self sampling and acoustic sampling, flexible routing, dynamic quantization configuration, data and preset management
- Signal and control flow

Signal Flow and Controller Assignments



- Data management

Parameter Organization



18.5. Pulsefact: System Design Goals

- Polyphony, layering, and simultaneity
- Employ audio fragments composed and produced outside of Max/MSP
- A variety of generators: direct control, indirect control, and looping
- A variety of transformers: effects and processors with variable input selection
- Ability to play in time, and to synchronize internal triggers with real-time quantization
- Ability to sample and process live and/or internal sources
- Huge dynamic range, multiple levels of gain control

18.6. Pulsefact: System Components

- Vox
  - Lead instruments, immediate control
  - Performed on Logitech Dual Actions
- Dyn
  - Background textures and layers
  - Triggered start and end, autonomous
  - Performed on Akai MPD-16
- Boards
  - Foreground/background textures and layers
  - Manual or looped beat triggered start and end, autonomous
  - Performed on Korg NanoKontrol
- Mods
  - Processing and transforming signals with variable input
  - Performed on Logitech Dual Action

18.7. Pulsefact: Dyn: Mapping

- Akai MPD 16 and Evolution UC33
- Top eight pads trigger events, bottom eight stop events
- Aftertouch on top eight can be used for volume modulation
- Eight faders of Evolution UC33 provide levels

18.8. Pulsefact: Vox: Mapping

- Logitech DA and Evolution UC33
- Joysticks provide primary control interface
- Keys select polyphonic voices, pitches, samples
- 9/10 keys rotate around circular array of Vox assignments
- Key-pads select samples, sample ranges, octave ranges, initial envelope amplitude levels
- Evolution UC33 provides master level

18.9. Pulsefact: Board: Mapping

- Korg NanoKontrol (2)
- 8 voices and a master level control
- Top buttons trigger event or open gate to receive pulses
- Bottom buttons send signal to auxiliary output or start/stop sampling
- Knobs select pulse beat multiplier or playback rate scalar

18.10. Pulsefact: Mod: Mapping

- Logitech RP and Evolution UC33
- Joysticks provide primary control interface
- Keys select processing types
- 9/10 keys rotate around circular array of Mod assignments

- Knobs on Evolution UC33 permit setting fixed levels

18.11. Pulsefact: Data Storage: Lowest Level: Programs

- Each component/instrument has an XML file read by [pattrstorage]
- XML files define numerous programs by number

18.12. Pulsefact: Data Storage: Mid Level: Presets

- A system preset assigns a list of programs to each instrument
- The first value is treated as primary; other values are available for browsing

18.13. Pulsefact: Data Storage: Outer Level: Scenes

- A sequence of stages, each defined by a preset and a comment (performative context)
- Can step through scene stages to create a performance set
- Design scenes for specific performance/venue contexts

18.14. Pulsefact: Preset/Quantization Controls: Mapping

- Akai LDP 8
- Upper pads browse programs
- Lower pads turn quantization on/off
- Upper knobs select preset browse focus
- Lower knobs select input for sampler/oxModPoly

18.15. Reading: Rebelo and Renaud, The FrequencyliatorÑDistributing Structures for Networked Laptop Improvisation

- Rebelo, P. and A. Renaud. 2006. "The Frequencyliator—Distributing Structures for Networked Laptop Improvisation." *Proceedings of the Conference on New Interfaces for Musical Expression* pp. 53-56.
- What motivated early attempts at networked music ensembles?
- What are some aspects of musical improvisation with acoustic instruments that authors describe as taken for granted?

- What are the main features of the described networking tool? What types of data is shared? What types of interaction is possible?

18.16. Networking: Device to Device Networks

- Can create a network on a local machine and have other machines join it
- Likely much faster and more bandwidth
- Mac OS X: Create Network..., give a name, and set IP address to something memorable: 192.168.1.100 (for a server)

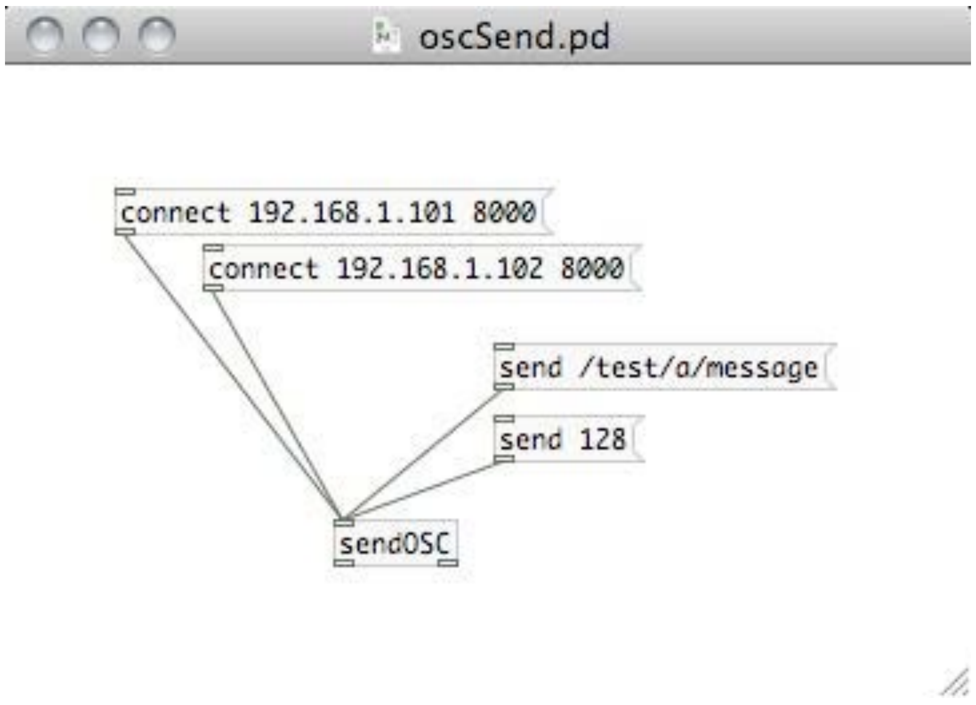
  Do not need to set subnet mask or router IP address
- For clients, set to Using DHCP to get IP address

18.17. Networking in Pd: Device to Device

- Use [sendOSC] to send to specific IP address on the network

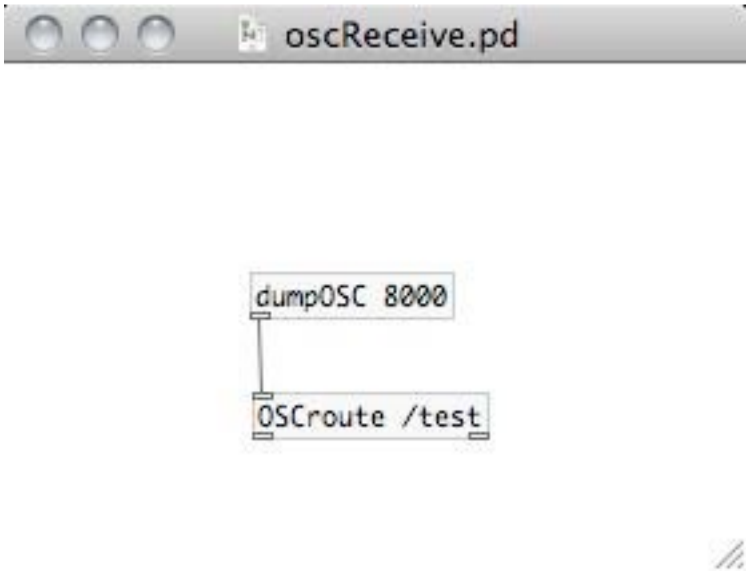
  Must send a connect message with an IP and Port: [connect 192.168.1.100 8000]

  Can then send any message that leads with a [send] message



```
connect 192.168.1.101 8000
connect 192.168.1.102 8000
send /test/a/message
send 128
sendOSC
```

- Can receive that message on another device with [dumpOSC 8000], where port is the same as that specified by [sendOSC]



```
dumpOSC 8000
|
OSCroute /test
```

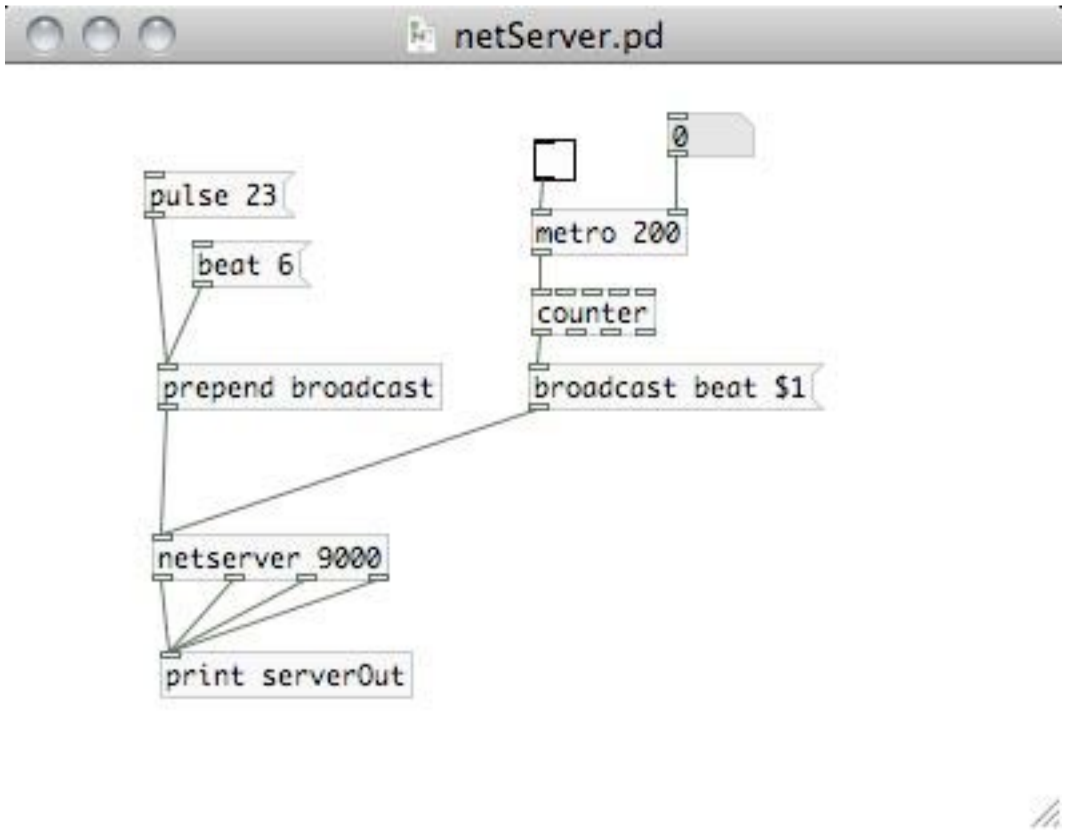
- Use [OSCroute] to parse messages

18.18. Networking in Pd: Server to Clients

- Use [netserver 9000] to create a server at a specific port

  Send messages to all clients by preceding with [broadcast] message

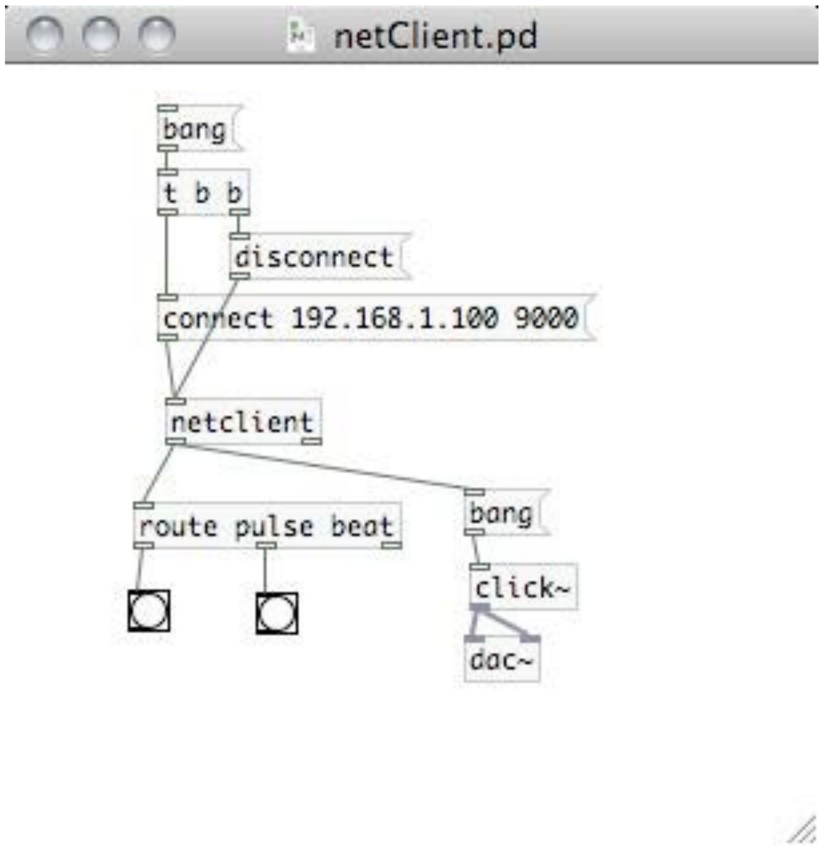
  Warning: can only have one server on one port at a time



- Use [netclient] on each client machine

  Create a connection to the server with a connect message, IP address, and port: [connect 192.168.1.100 9000]

  Warning: sending the connect message more than once without disconnecting may cause Pd to hang



- Message transmitted can be lists, symbols, or floats

Chapter 19. Meeting 19, Workshop: Performance and Improvisation

19.1. Announcements

- Bring amps and controllers to next class
- Monday 18 April: No class (Patriots day)
- Due next Wednesday 20 April: Bring to class Instrument 2 Drafts/Prototypes
- Due next Wednesday 20 April: Performance Frameworks documentation/scores

19.2. Performance Frameworks Drafts

- `[Student names removed for privacy.]`

19.3. Exercise: Improvisation with Controller/Interface/Instrument Design 1

- Load: instruments created for Controller/Interface/Instrument Design 1 or any other instrument
- Ensemble: each person enter staggered; do ostinato layers
- Ensemble: explore dialogs

19.4. Work III

- Add martingale/compositions/arizaWork03 to Pd Paths
- Load: arizaWork03-performance*.test.pd

  martingale/comopositions/arixaWork03/arizaWork03-performance*.test.pd
- Test of performance and new instruments

19.5. Work III (mgSynthStochWave)

- Stochastic wave form synthesis
- A duophonic instrument: voice 1 with keys 1-4, voice 2 with keys 5-8
- Keys select different base-frequency pitches
- Y1: amplitude; X1: low pass filter cutoff frequency
- Y2: segment draw rate (up is faster); X2: segment size (left is smaller)
- D-pad 1 (left): slow echo; D-pad 2 (down): long reverb; D-pad 3 (right): fast echo

19.6. Work III (mgSynthGranularSample)

- Stochastic wave form synthesis
- A duophonic instrument: voice 1 with keys 1-4, voice 2 with keys 5-8; each voice employs a different source sample
- Keys select different regions of audio to grab sound from
- Y1: amplitude; X1: sample playback rate and pitch
- Y2: grain density (up is more dense); X2: grain window duration
- D-pad 1 (left): slow echo; D-pad 2 (down): long reverb; D-pad 3 (right): fast echo

19.7. Work III (mgSynthSawSequenceDuo)

- 2x16 stored sequencer patterns of pitch/amplitude sequences
- Multiple waveforms with FM modulation
- A duophonic instrument: voice 1 with keys 1-4, voice 2 with keys 5-8; each voice employs a oscillator
- Keys select different patterns
- Y1: amplitude; X1: low pass filter cutoff frequency
- Y2: NC; X2: NC
- D-pad 1 (left): slow echo; D-pad 2 (down): long reverb; D-pad 3 (right): fast echo

19.8. Work III (mgSynthPafSequenceDuo)

- 2x16 stored sequencer patterns of pitch/amplitude sequences
- A synthesis method related to phase aligned formant synthesis
- A duophonic instrument: voice 1 with keys 1-4, voice 2 with keys 5-8; each voice employs a oscillator
- Keys select different patterns
- Y1: amplitude; X1: low pass filter cutoff frequency
- Y2: formant center; X2: formant bandwidth
- D-pad 1 (left): slow echo; D-pad 2 (down): long reverb; D-pad 3 (right): fast echo

19.9. Work II

- Load: arizaWork02-performance*.test.pd

  martingale/comopositions/arixaWork02/arizaWork02-performance*.test.pd
- Focus on texture and heterophony

19.10. Reading: Cascone, Grain, Sequence, System [three levels of reception in the performance of laptop music]

- Cascone, K. 2004. "Grain, Sequence, System [three levels of reception in the performance of laptop music]." *Intelligent Agent* 4(1).
- What factors does Cascone suggest has led to the current position of laptop performers?
- How does laptop music disrupt the expectations of both acoustmatic music and pop/electronic music?
- How does Cascone suggest is necessary for the continued growth of electronic music?

19.11. Listening: Kim Cascone, Dust Theories 2: Alchemical Residue, Musicworks.82

- Listening: Kim Cascone, "Dust Theories 2: Alchemical Residue," Musicworks.82

[Page intentionally left blank]

Chapter 20. Meeting 20, Workshop: Performance and Improvisation

20.1. Announcements

- All remaining class through Meeting 24 (Wednesday) will be rehearsals
- Concert on Wednesday May 4
- Final Instrument 2 reports are due on 9 May
- Final Instrument 2 presentations are on 9 and 11 May

20.2. Controller/Interface/Instrument Design 2

- Drafts and prototypes

20.3. Work II

- Load: arizaWork02-performance*.test.pd

  martingale/comopositions/arixaWork02/arizaWork02-performance*.test.pd

- Focus on texture and heterophony

20.4. Work III

- Load: arizaWork03-performance*.test.pd

  martingale/comopositions/arixaWork03/arizaWork03-performance*.test.pd

- Focus on texture and heterophony

Chapter 21. Meeting 21, Practices: Novel and Custom Interfaces

21.1. Announcements

- Concert on Wednesday May 4
- Final Instrument 2 reports are due on 9 May
- Final Instrument 2 presentations are on 9 and 11 May
- Full rehearsal on Monday May 2

21.2. Reading: Weinberg, Playpens, Fireflies, and Squeezables

- Weinberg, G. 2002. "Playpens, Fireflies, and Squeezables: New Musical Instruments for Bridging the Thoughtful and the Joyful." *Leonardo Music Journal* 12: pp. 43-51.
- Video of beatbugs performance:

  link (http://web.media.mit.edu/~roberto/beatbug/beatbug_video/Glasgow%20-%20Concert.mov)
- What motivates Weinberg's new instruments? Are his assumptions about "high art" and novices well founded?
- Why does Weinberg think that his instruments may offer useful approaches for early music education? Do his instruments achieve his goals?
- What is the interface and mapping of the Musical Playpen?
- What is the interface and mapping of the Musical Fireflies?
- What is the interface and mapping of the Squeezables?

  link (http://xenia.media.mit.edu/~gan/Gan/Education/MIT/MediaLab/Research/Squeezables)

21.3. Reading: Collins, Handmade Electronic Music

- Collins, N. 2009. *Handmade Electronic Music: The Art of Hardware Hacking*. 2nd ed. New York: Routledge.
- "If there is any identifying trait, it is the desire to disrupt technology's seemingly perfect inviolability"

21.4. Work II

- Load: arizaWork02-performance*.test.pd

  martingale/comopositions/arixaWork02/arizaWork02-performance*.test.pd

- Focus on texture and heterophony

21.5. Work III

- Load: arizaWork03-performance*.test.pd

  martingale/comopositions/arixaWork03/arizaWork03-performance*.test.pd

- Focus on timing and space

Chapter 22. Meeting 22, Workshop: Performance and Improvisation

22.1. Announcements

- Concert on Wednesday May 4: arrive before 3:30
- Final Instrument 2 reports are due on 9 May
- Final Instrument 2 presentations are on 9 and 11 May
- Full rehearsal on Monday May 2

22.2. Work II

- Load: arizaWork02-performance*.test.pd

  martingale/comopositions/arixaWork02/arizaWork02-performance*.test.pd
- Focus on texture and heterophony

22.3. Work III

- Load: arizaWork03-performance*.test.pd

  martingale/comopositions/arixaWork03/arizaWork03-performance*.test.pd
- Focus on timing and space

Chapter 23. Meeting 23, Practices: Live Coding, Live Algorithms, and Generative Techniques

23.1. Announcements

- Concert on Wednesday May 4: arrive before 3:30
- Final Instrument 2 reports are due on 9 May
- Final Instrument 2 presentations are on 9 and 11 May

23.2. Reading: Collins et al: Live Coding in Laptop Performance

- Collins, N. 2003. "Live Coding in Laptop Performance." *Organised Sound* 8(3): pp. 321-330.
- What are the attributes of a live-coding performance?
- What features of live coding do the author's find desirable?
- Is live coding new?
- What are some software systems and interfaces used for live coding? Do they have common features?

23.3. Reading: Brown and Sorensen: Interacting with Generative Music through Live Coding.

- Brown, A. R. and A. Sorensen. 2009. "Interacting with Generative Music through Live Coding." *Contemporary Music Review* 28(1): pp. 17-29.
- What about live coding suggests generative approaches?
- What are the criteria for generative processes in live coding, as suggested by the authors?
- The author's describe multiple levels of musical control and generation; what levels are given to algorithms, and what levels are reserved for humans?

23.4. Rehearsal

- Full set

Chapter 24. Meeting 24, Workshop: Performance and Improvisation

24.1. Announcements

- Concert on Wednesday May 4: arrive before 3:30
- Final Instrument 2 reports are due on 9 May
- Final Instrument 2 presentations are on 9 and 11 May

References

Ariza, C. 2009. "Pure Data Object Glossary." Available online at http://flexatone.net/docs/pdg.

Ballou, D. 2009. "Towards a Pedagogy for the Improvisation Ensemble." *Third Annual Conference of the International Society of Improvised Music (ISIP).*

Brown, A. R. and A. Sorensen. 2009. "Interacting with Generative Music through Live Coding." *Contemporary Music Review* 28(1): pp. 17-29.

Cascone, K. 2004. "Grain, Sequence, System [three levels of reception in the performance of laptop music]." *Intelligent Agent* 4(1).

Collins, N. 2003. "Live Coding in Laptop Performance." *Organised Sound* 8(3): pp. 321-330.

Collins, N. 2003. "Generative Music and Laptop Performance." *Contemporary Music Review* 22(4): pp. 67-79.

Cook, P. 2001. "Principles for designing computer music controllers." *Proceedings of the Conference on New Interfaces for Musical Expression.*

Cook, P. R. 2004. "Remutualizing the Musical Instrument: Co-Design of Synthesis Algorithms and Controllers." *Journal of New Music Research* 33(3): pp. 315-320.

Dannenberg, R. B. and S. Cavaco, E. Ang, I. Avramovic, B. Aygun, J. Baek, E. Barndollar, D. Duterte, J. Grafton, R. Hunter, C. Jackson, U. Kurokawa, D. Makuck, T. Mierzejewski, M. Rivera, D. Torres, A. Yu. 2007. "The Carnegie Mellon Laptop Orchestra." *Carnegie Mellon Computer Science Department, Paper 513.*

Dennis, B. 1991. "Cardew's 'Treatise' (Mainly the Visual Aspects)." *Tempo* 177: pp. 10-16.

Driscoll, J. and M. Rogalsky. 2004. "David Tudor's 'Rainforest': An Evolving Exploration of Resonance." *Leonardo Music Journal* 14: pp. 25-30.

Fiebrink, R. and G. Wang, P. Cook. 2007. "Don't Forget the Laptop: Using Native Input Capabilities for Expressive Musical Control." *Proceedings of the Conference on New Interfaces for Musical Expression* pp. 164-167.

Ghazala, Q. R. 2004. "The Folk Music of Chance Electronics: Circuit-Bending the Modern Coconut." *Leonardo Music Journal* 14(1): pp. 97-104.

Gresham-Lancaster, S. 1998. "The Aesthetics and History of the Hub: The Effects of Changing Technology on Network Computer Music." *Leonardo Music Journal* 8: pp. 39-44.

Holmes, T. 2008. "Live Electronic Music and Ambient Music." In T. Holmes, ed. *Electronic and Experimental Music.* Third ed. New York: Routledge, pp. 376-406.

Jaeger, T. 2003. "The (Anti-)Laptop Aesthetic." *Contemporary Music Review* 22(4): pp. 53-57.

Kahn, D. 2004. "A Musical Technography of John Bischoff." 14 ed. *Leonardo Music Journal* 14: pp. 74-79.

Kiefer, C. and N. Collins, G. Fitzpatrick. 2008. "HCI Methodology For Evaluating Musical Controllers: A Case Study." *Proceedings of the Conference on New Interfaces for Musical Expression*.

Kreidler, J. 2009. "Programming Electronic Music in Pd." Wolke Publishing House. Available online at http://www.pd-tutorial.com.

Kuivila, R. 2004. "Open Sources: Words, Circuits and the Notation-Realization in the Music of David Tudor." *Leonardo Music Journal* 14: pp. 17-23.

McCartney, J. 2002. "Rethinking the Computer Music Language: SuperCollider." *Computer Music Journal* 26(4): pp. 61-68.

Monroe, A. 2003. "Ice on the Circuits/Coldness as Crisis: The Re-subordination of Laptop Sound." *Contemporary Music Review* 22(4): pp. 35-43.

Paradiso, J. 1997. "New Ways to Play: Electronic Music Interfaces." *IEEE Spectrum* 34(12): pp. 18-30.

Perkis, T. 2009. "Some Notes on My Electronic Improvisation Practices." In R. T. Dean, ed. *The Oxford Handbook of Computer Music*. Oxford University Press, pp. 161-166.

Puckette, M. 2002. "Max at 17." *Computer Music Journal* 26(4): pp. 31-43.

Rebelo, P. and A. Renaud. 2006. "The Frequencyliator—Distributing Structures for Networked Laptop Improvisation." *Proceedings of the Conference on New Interfaces for Musical Expression* pp. 53-56.

Roads, C. 1980. "Interview with Max Mathews." *Computer Music Journal* 4(4): pp. 15-22.

Ryan, J. 1991. "Some Remarks on Musical Instrument Design at STEIM." *Contemporary Music Review* 6(1): pp. 3-17.

Smallwood, S. and D. Trueman, P. R. Cook, G. Wang. 2008. "Composing for Laptop Orchestra." *Computer Music Journal* 32(1): pp. 9-25.

Sorensen, A. and A. R. Brown. 2007. "aa-cell in Practice: An Approach to Musical Live Coding." *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association, 2: pp. 292-299.

Stuart, C. 2003. "The Object of Performance: Aural Performativity in Contemporary Laptop Music." *Contemporary Music Review* pp. 59-65.

Tanaka, A. 2009. "Sensor-Based Musical Instruments and Interactive Music." In R. T. Dean, ed. *The Oxford Handbook of Computer Music*. Oxford University Press, pp. 233-257.

Trueman, D. 2007. "Why a Laptop Orchestra?." *Organised Sound* 12(2): pp. 171-179.

Wanderley, M. M. and N. Orio. 2002. "Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI." *Computer Music Journal* 26(3): pp. 62-76.

Wang, G. 2007. "A History of Programming and Music." In N. Collins and J. d'Escriván, eds. *The Cambridge Companion to Electronic Music*. Cambridge: Cambridge University Press, pp. 55-71.

Wang, G. and D. Trueman, S. Samllwood, P. R. Cook. 2008. "The Laptop Orchestra as Classroom." *Computer Music Journal* 32(1): pp. 26-37.

Weinberg, G. 2002. "Playpens, Fireflies, and Squeezables: New Musical Instruments for Bridging the Thoughtful and the Joyful." *Leonardo Music Journal* 12: pp. 43-51.

Weinberg, G. 2005. "Local Performance Networks: musical interdependency through gestures and controllers." *Organised Sound* 10(3): pp. 255-266.

Wessel, D. and M. Wright. 2002. "Problems and Prospects for Intimate Musical Control of Computers." *Computer Music Journal* 26(3): pp. 11-22.

Wright, M. 2005. "Open Sound Control: an enabling technology for musical networking." *Organised Sound* 10(3): pp. 193-200.