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

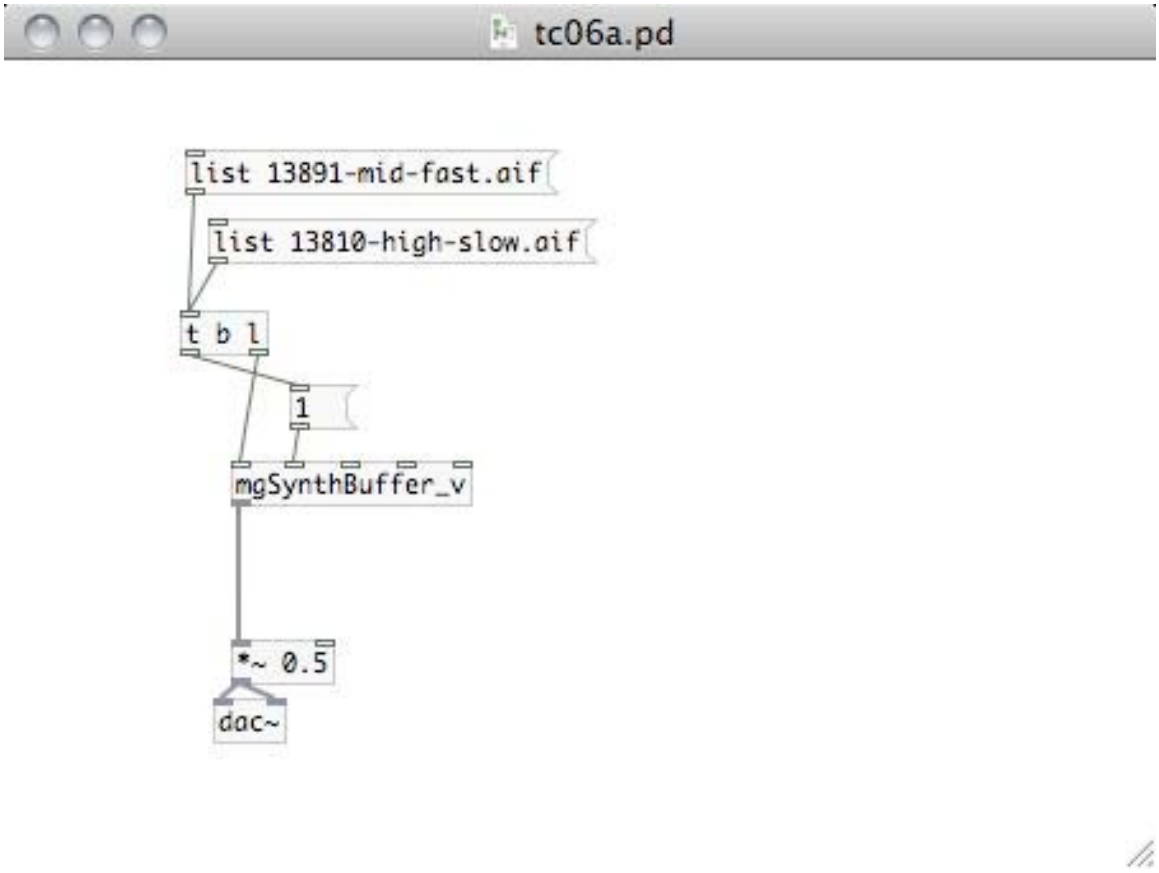
- A number of math objects for data and signals: [+], [-]
- Always favor multiplication over division: [* .01] is better than [/ 100]
- For multiple multiple computations, the [expr] object is conventient
- Operator precedence is multiplication or division, then addition or substraction; Values can be 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]

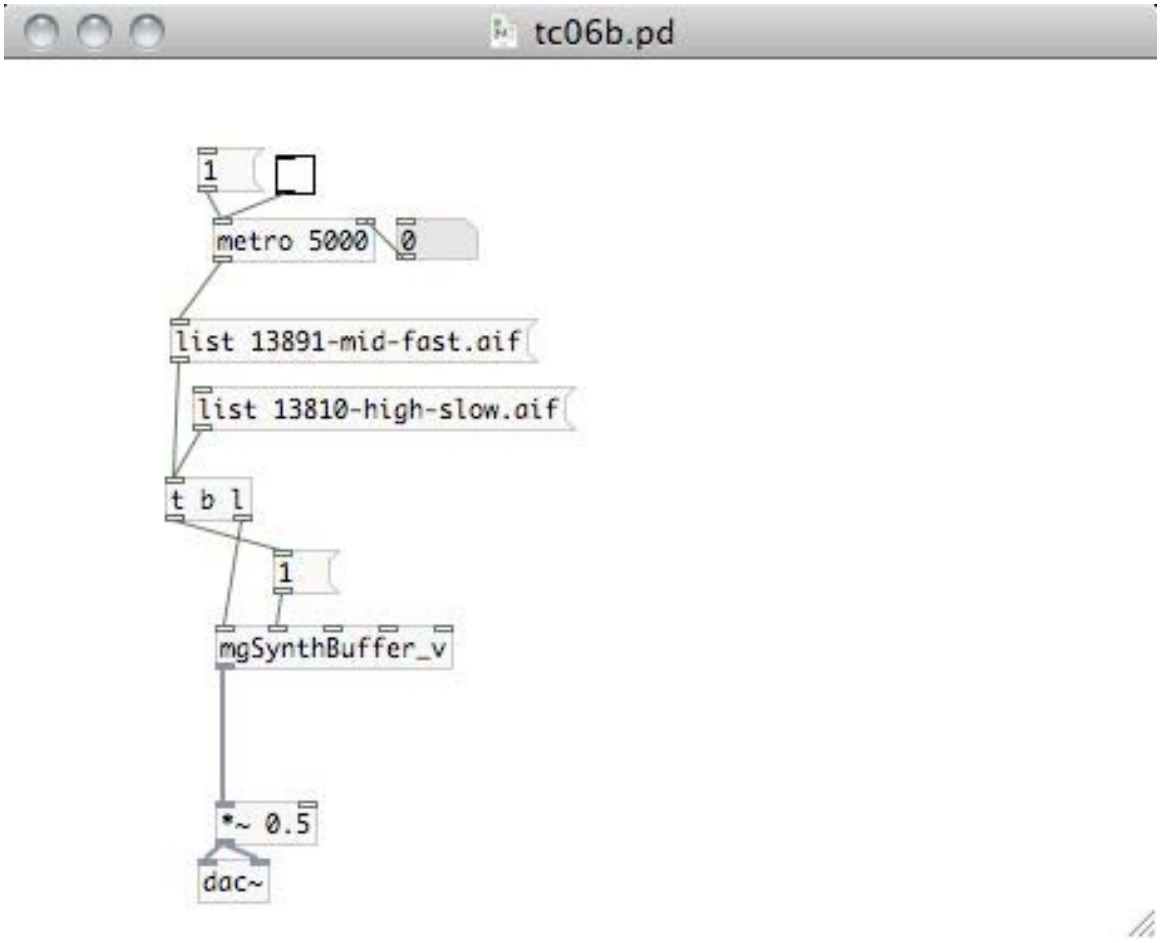
4.13. The Trigger Object

- 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

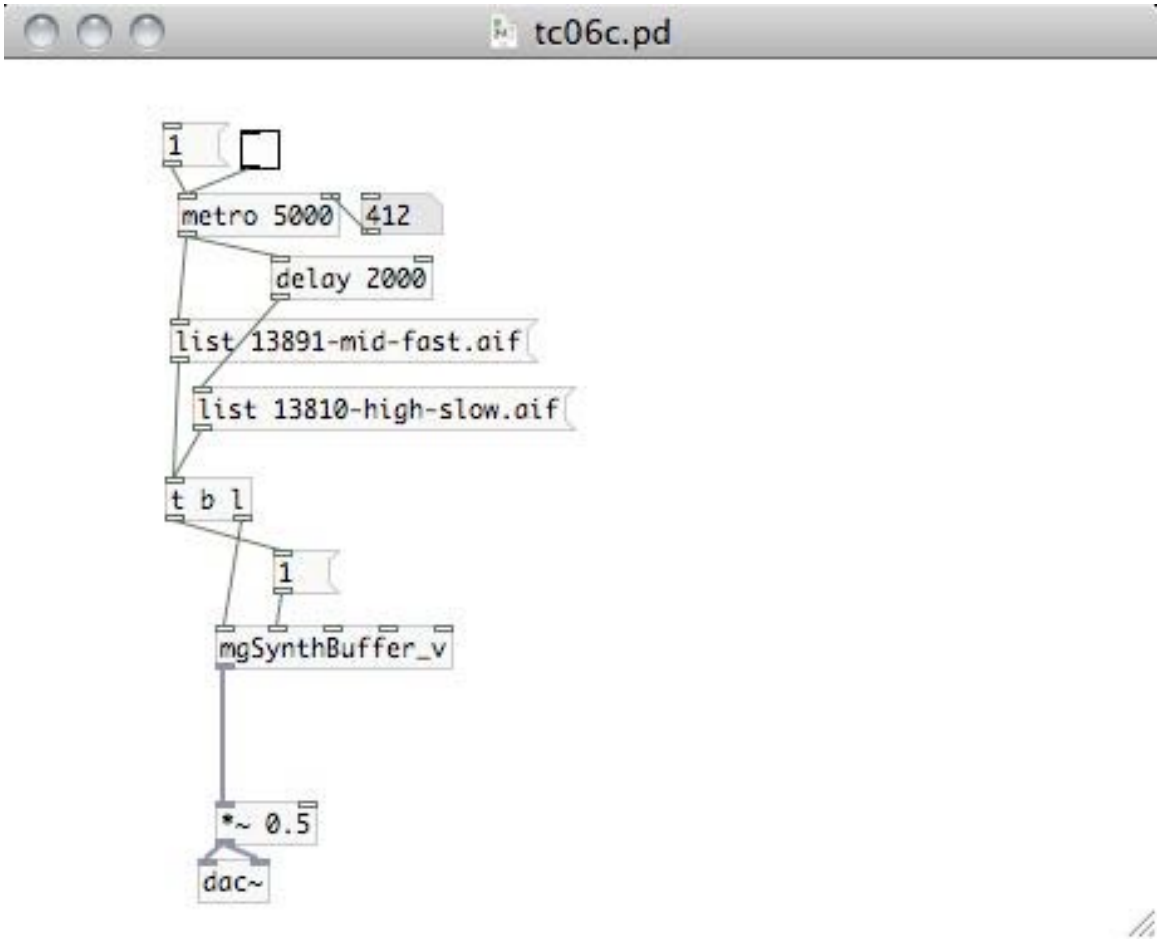


4.14. A Source of Triggers: [metro]

- 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

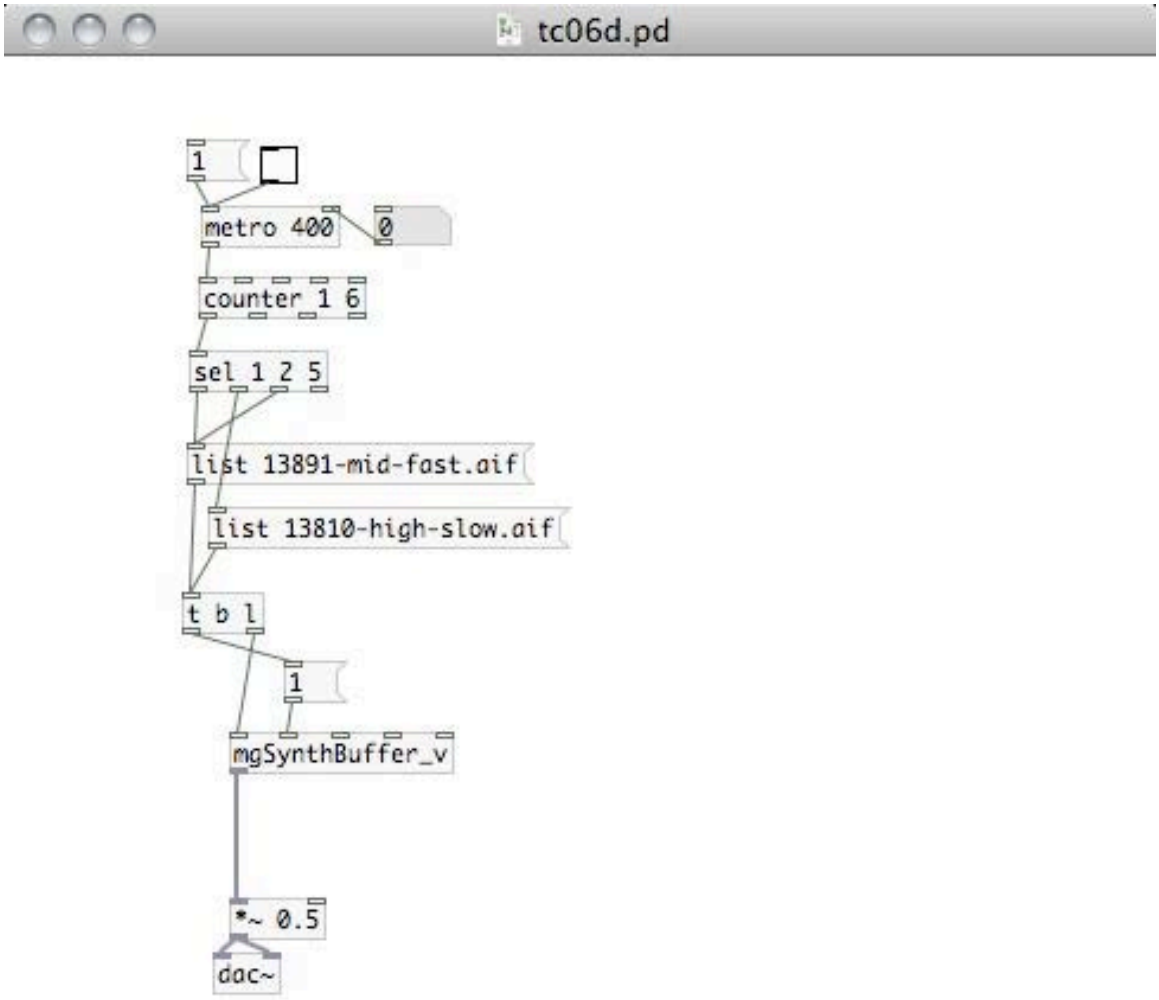


- Bangs can be delayed with [delay]; messages can be delayed withÊ[pipe]
- Example: using [delay] to trigger a second sample after the first



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

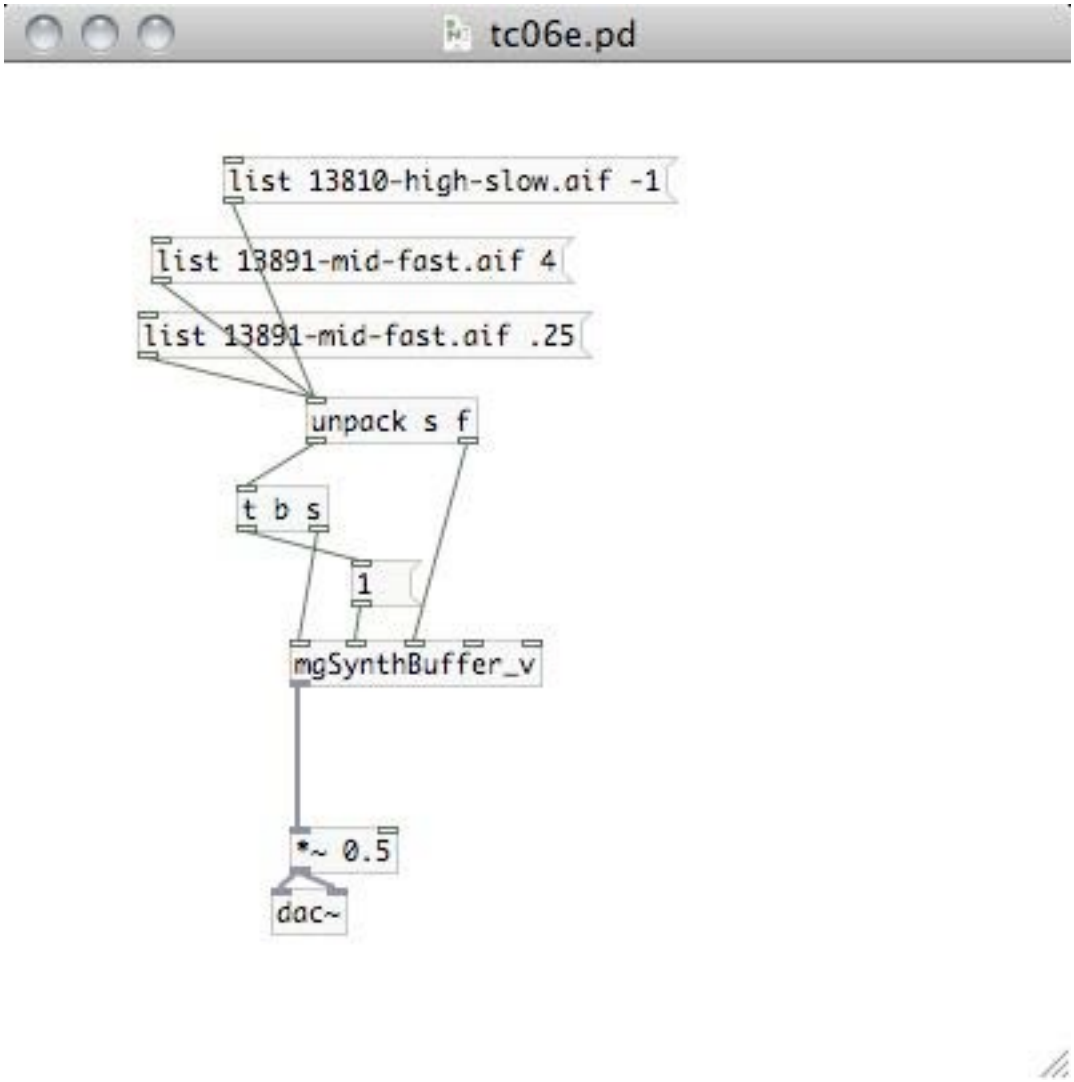
- Can be all numbers in a message box
- Can be all symbols in a message box preceded by "list"
- [zl] tools (there are many), [pack], [unpack] are main processors
- Length: [zl len]

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

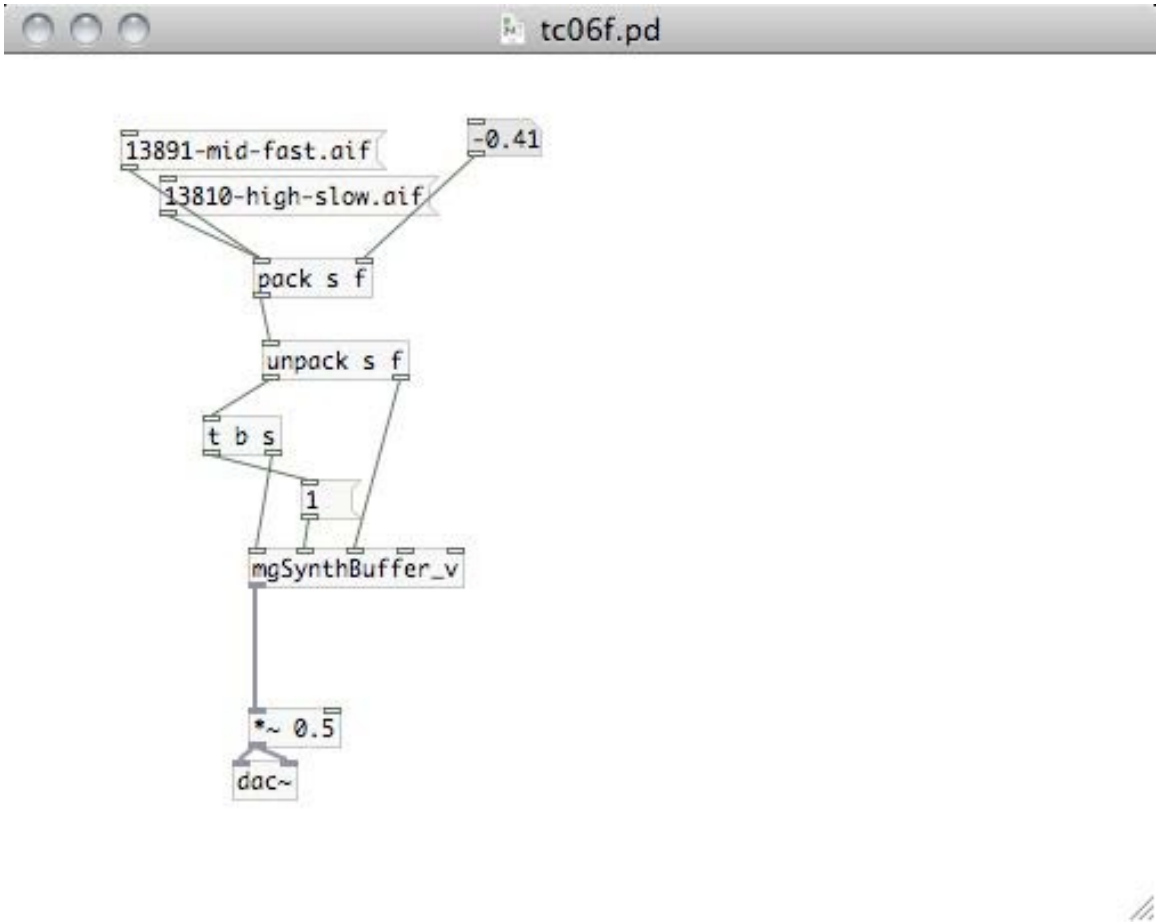
4.20. Breaking Lists

- [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



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

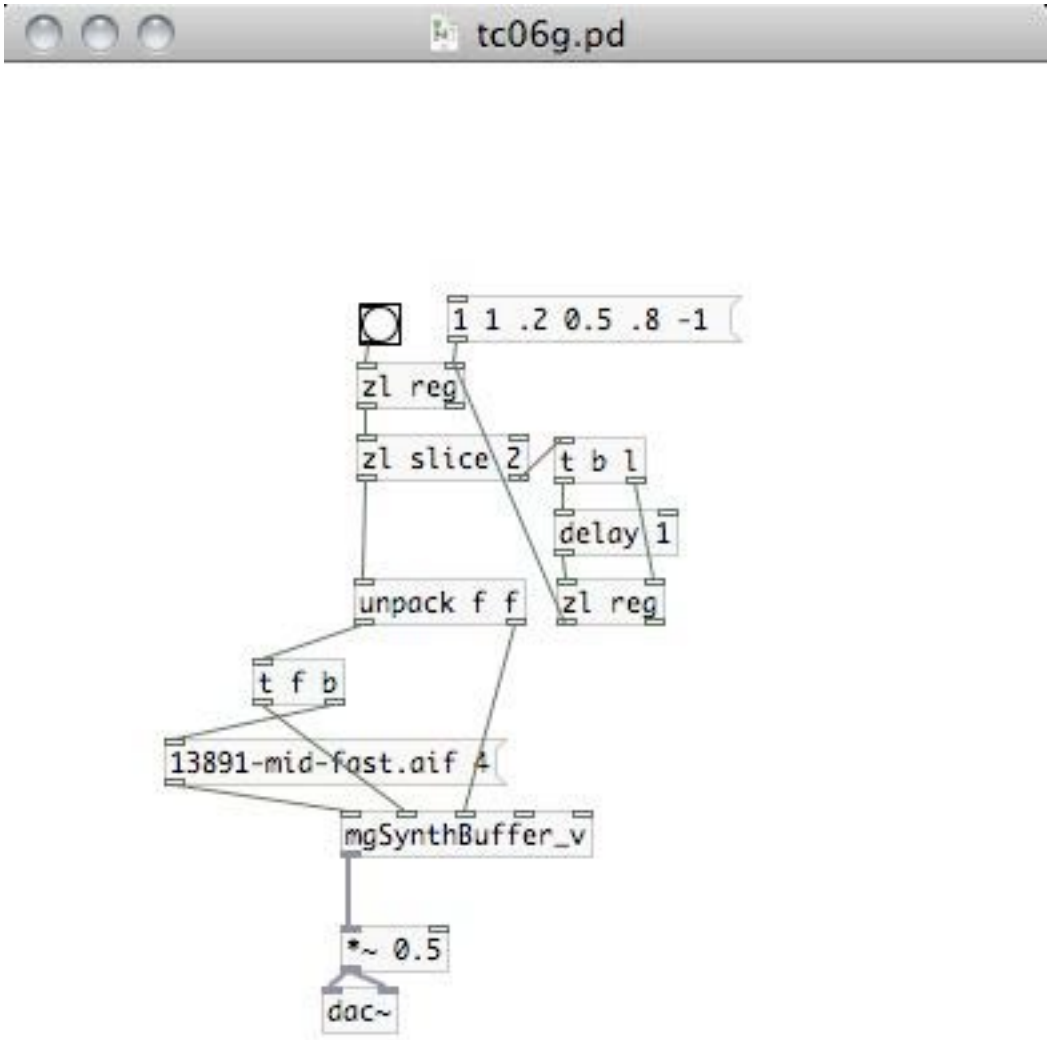


4.22. Lists: Iterating, Grouping, Joining

- 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]

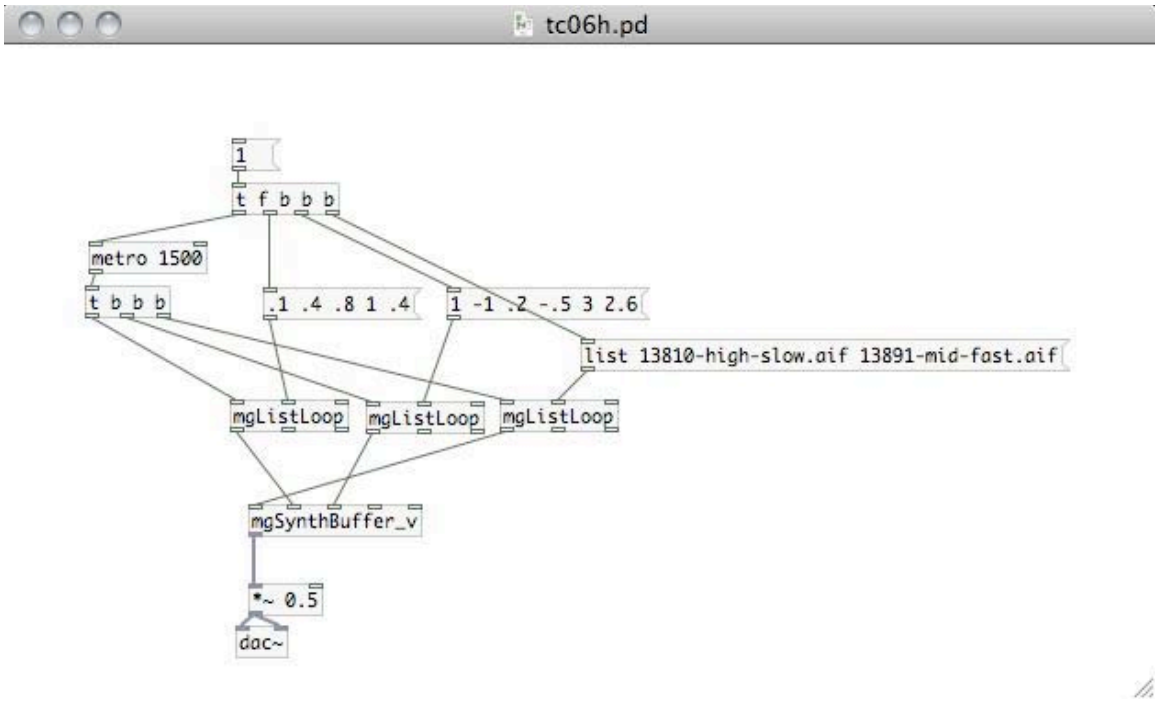
4.23. Lists: Slicing, Rotating, and Reversing

- 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

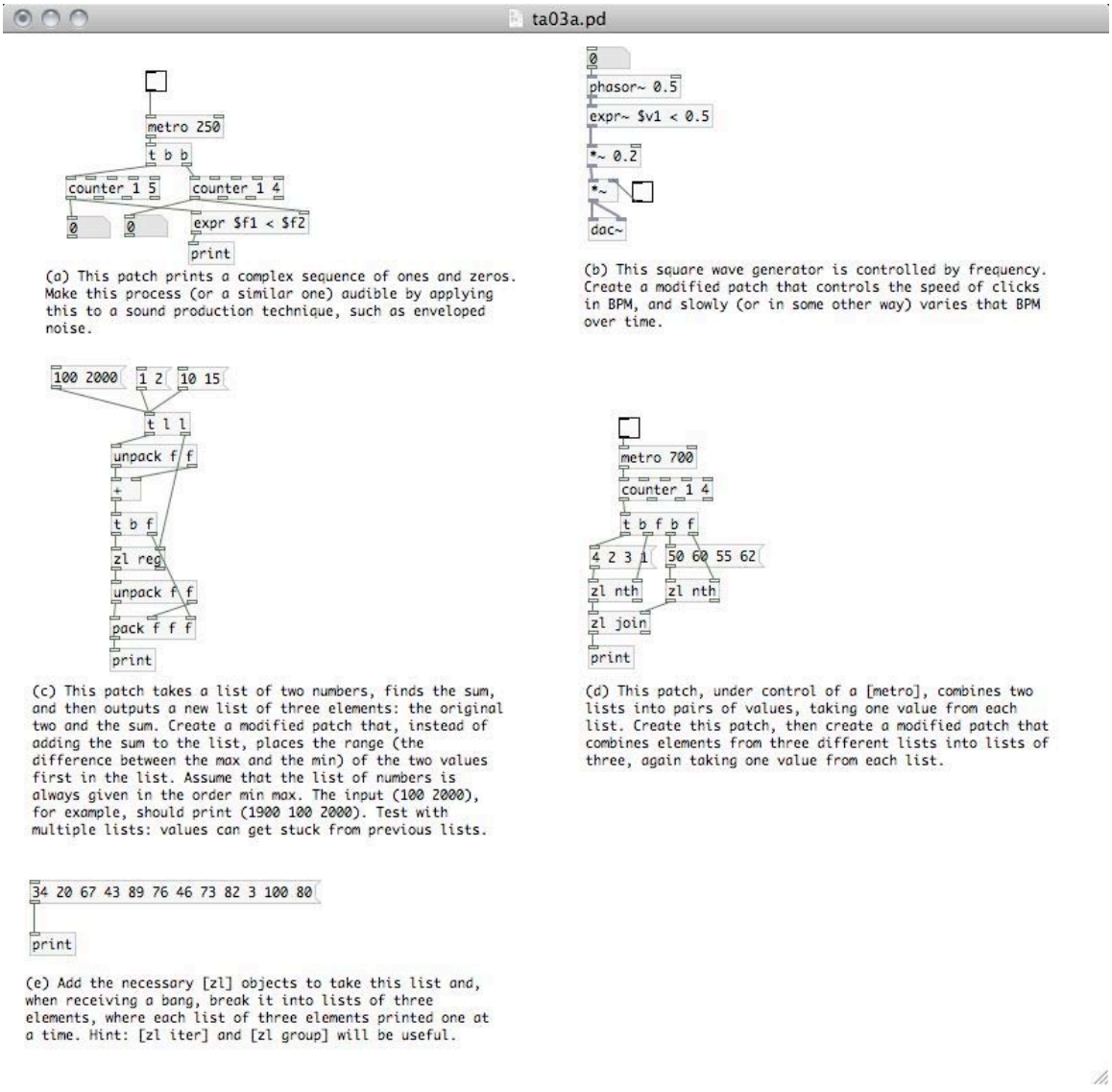
- Listening: Ikue Mori, "The Pit & The Pendulum", Garden, 1996

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

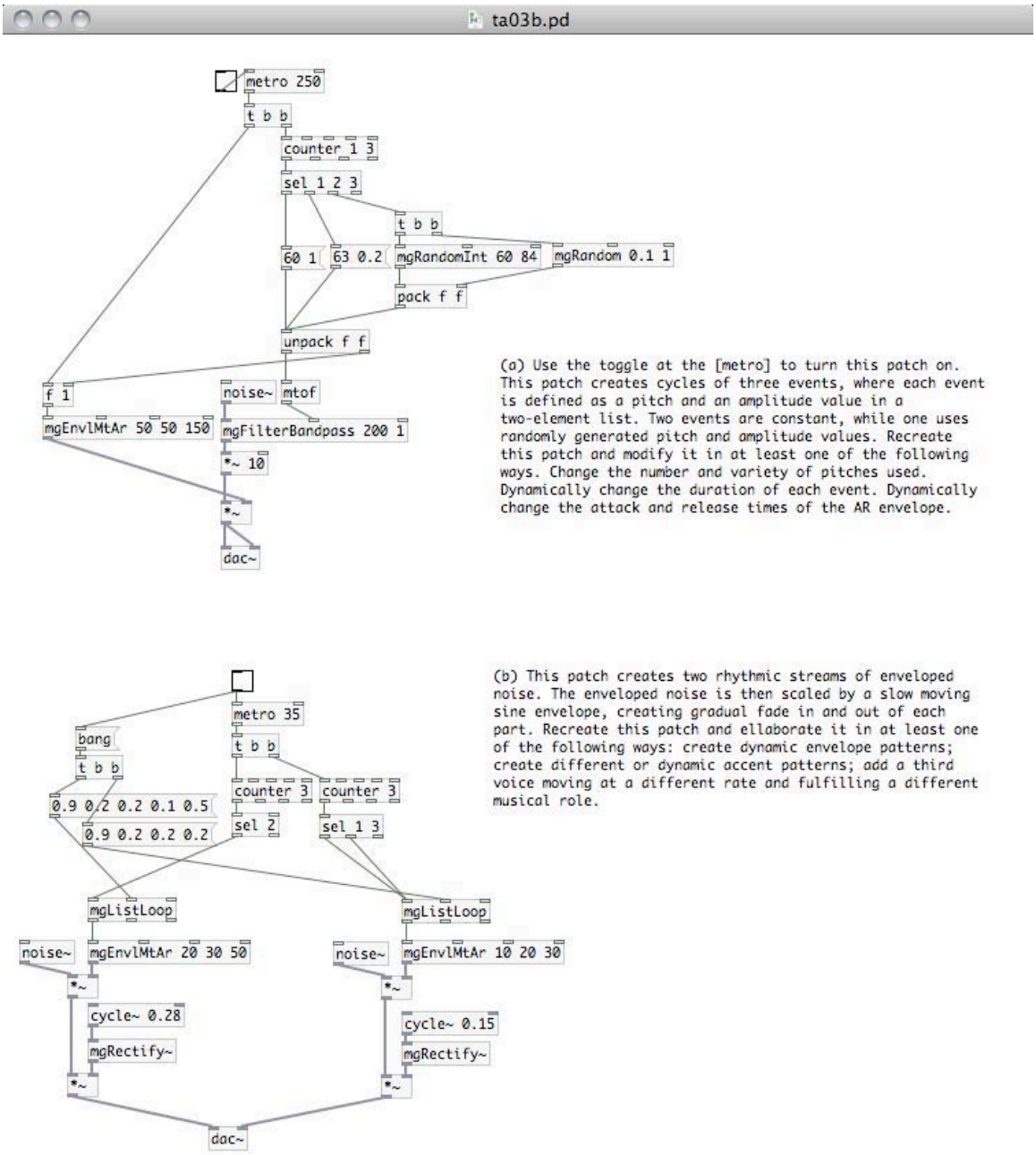
- Listening: Ikue Mori, “Day of Locusts,” Labyrinth, 2001

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.



2. Create and extend the following patch.


ta03b.pd
metro 250
t b b
counter 1 3
sel 1 2 3
t b b
60 1
63 0.2
mgRandomInt 60 84
mgRandom 0.1 1
pack f f
unpack f f
f 1
noise~
mtof
mgEnvlMtAr 50 50 150
mgFilterBandpass 200 1
*~ 10
*~
dac~
(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.
metro 35
bang
t b b
t b b
counter 3
counter 3
0.9 0.2 0.2 0.1 0.5
0.9 0.2 0.2 0.2
sel 2
sel 1 3
mgListLoop
mgListLoop
noise~
mgEnvlMtAr 20 30 50
noise~
mgEnvlMtAr 10 20 30
*~
*~
cycle~ 0.28
cycle~ 0.15
mgRectify~
mgRectify~
*~
*~
dac~
(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.

21M.380 Music and Technology: Live Electronics Performance Practices
Spring 2011