21M.361 Composing with Computers I (Electronic Music Composition)
Spring 2008

21M.361: Composing with Computers I (Electronic Music Composition)

Peter Whincop

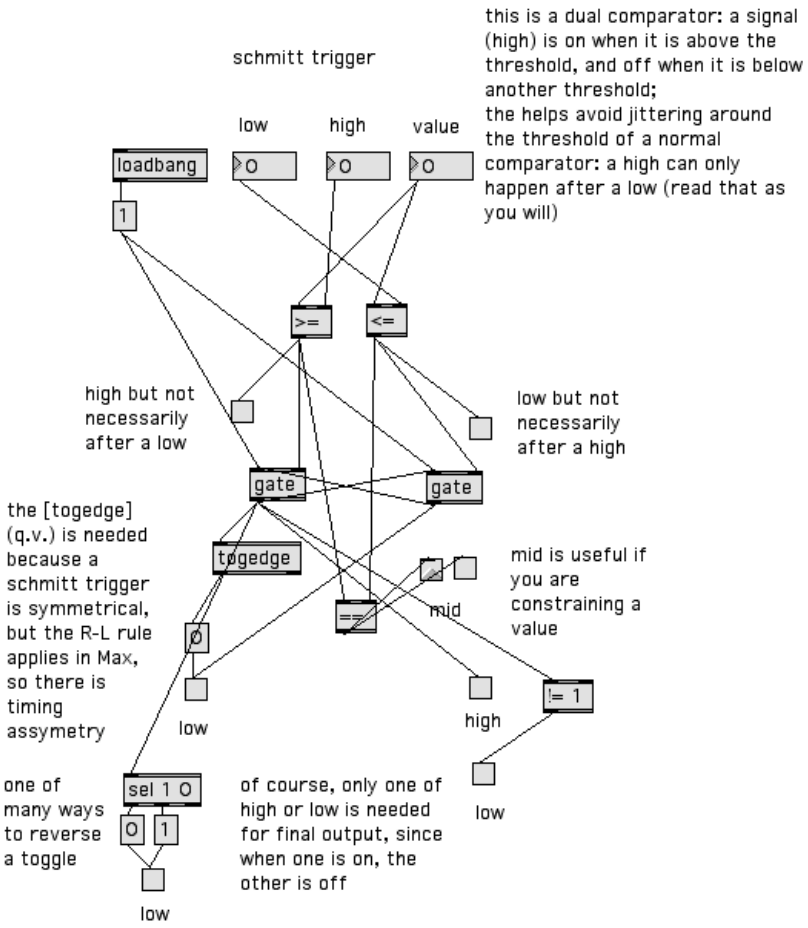
Spring 2008 OCW

[All lab notes are being significantly revised for next term to reflect upgrades in software, different organization, and the incorporation of my own notes on DSP.]

Labs 4.3/4.4: More Max/MSP

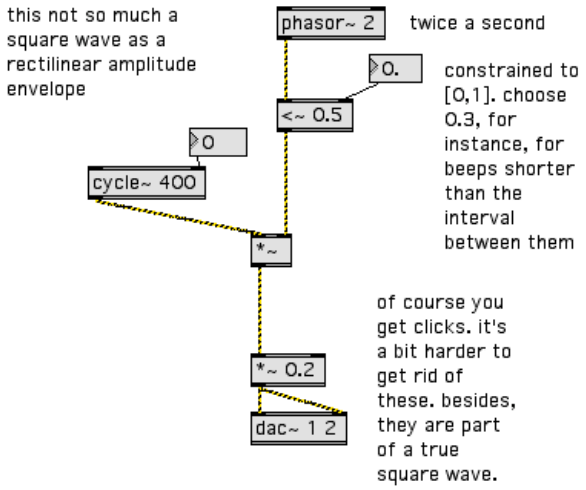
Max Things

1. **Conditionals**. Things like [==] and [<]. The output is Boolean: 1 for true, 0 for false. A bang is sent no matter what, so if you want a bang for true only, connect the conditional to [sel 1] to a bang. ([sel] is short for [select].) Of course, for [==] you could simply use [sel], but for [<] etc. you can't. The complete set of conditions are **[>], [>=], [==] (equal to), [!=] (not equal to), [<=], and [<]**. Here's a not so simple example:

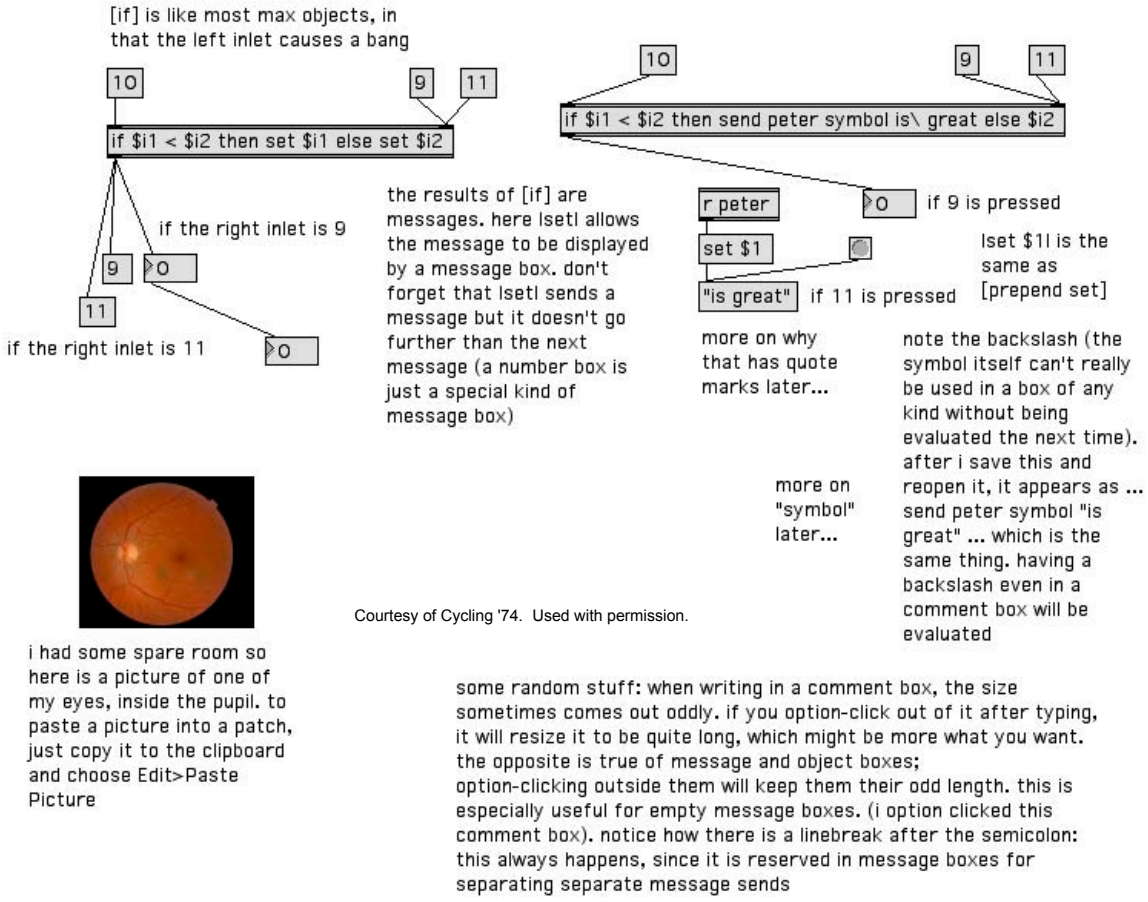


Courtesy of Cycling '74. Used with permission.

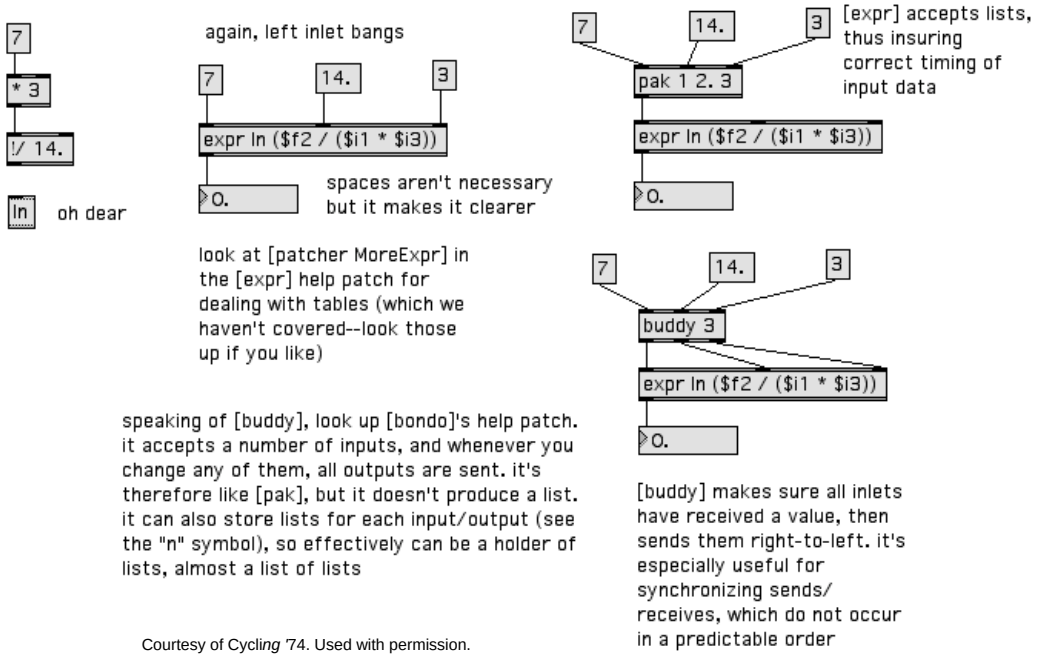
2. **Conditionals work at the audio rate too**. A way of making a pulse, say, a beep, you need an envelope that resembles a square wave, and you might also need it at the audio rate. Having a [phasor~] at whatever frequency you want the beeps to be at insures sample-precise transitions. Don't forget [phasor~] is just a ramp, (like a sawtooth function), but in the range [0,1]:



3. **[if]**, plus a couple of other things (**pictures, backslashes, quotes, symbols, box resizing**):



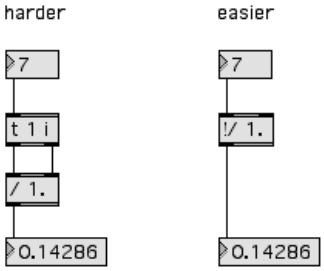
4. **[expr]** allows you to put a whole lot of math into one object, rather than having chains of calculations. There is a subpatch in the [expr] help patch enumerating the permissible functions. I've also shown **[buddy]** and mentioned **[bondo]**.



Courtesy of Cycling '74. Used with permission.

5. **[togedge]** shows 0 to 1 transitions, and vice versa. **[change]** does the same, but sends values. See the Schmitt trigger patch above for an example of [togedge]. [change] is good for filtering out repetitions (wise words). (See [urn] below.)

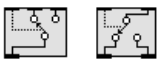
6. **Arithmetic operators beginning with !**. This means reverse the operands. It was used above in the [expr] example.



Courtesy of Cycling '74. Used with permission.

8. **[onebang]**, once the right inlet has been reset, allows only one bang (surprise, surprise) from the left inlet to the left outlet. All subsequent bangs are sent to the right outlet. Very very useful.

9. **Ggate and Gswitch**: go figure.

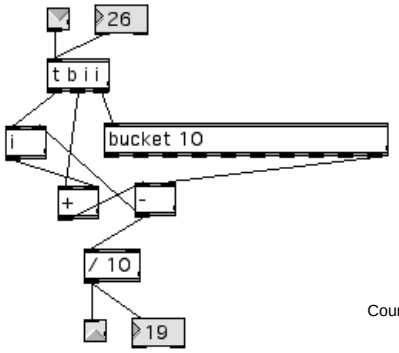


10. [**zmap** 0 127 12 0] will map [0,127] to [12,0]. Saves on a lot of needless math. **[scale]** does the same, but includes an option exponential factor.

11. **[kslider]** is a keyboard that outputs MIDI values. It doesn't have to be used for MIDI, of course. Numbers are just numbers, after all (be quiet, you philosophers). And **[rslider]** is a slider that visually shows a range of values, and those values can be changed graphically (think of the 1960s).

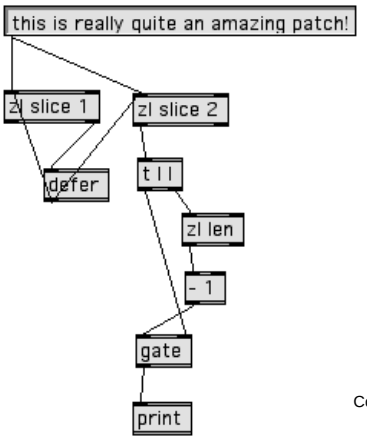
12. **[bucket]** is a shift register. Check out the help patch. [bucket 10] has ten registers/outlets, for instance. An input to the left inlet shifts all values to the right, loses the rightmost one, and the leftmost one takes on the new value. Everything is banged. See this example of a low pass filter for a

stream of integers.



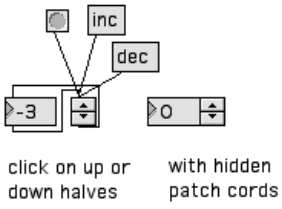
Courtesy of Cycling '74. Used with permission.

13. **[decide]** is the same as [random 2], in other words, it outputs random bits. **[urn]** is like random, but insures that all numbers are used before there is a repetition. **[uzi]** fires out bangs. **[zl]** is an all-purpose list thingy. Look up the help patch; it splits lists, finds the nth element, rotates a list, etc. Look at this obfuscated patch—please don't ask me about **[defer]**:



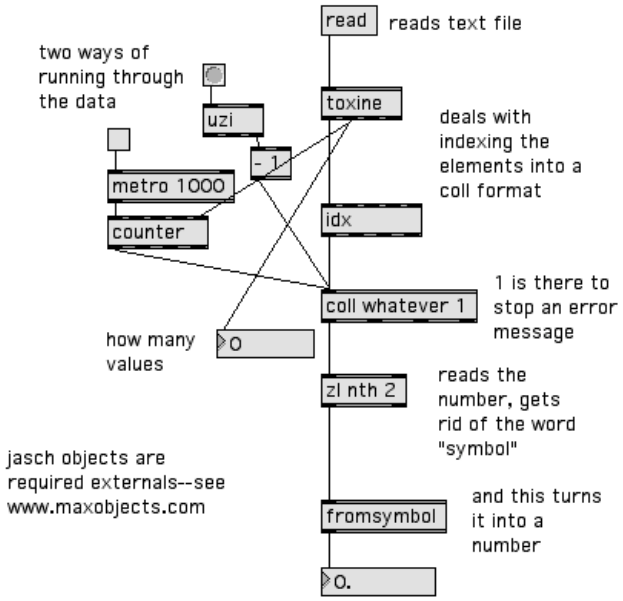
Courtesy of Cycling '74. Used with permission.

14. **[counter]** is a counter. **[incdec]** is a simple graphic increment/decrement object. Here it is in two configurations:



Courtesy of Cycling '74. Used with permission.

14. **[iter]** unpacks elements of a list one at a time, in other words, as successive messages. [cycle] (no ~) cycles through numbers. **[listfunnel]** indexes numbers. To do the latter for the **[coll]** object (which we don't look at this semester), the following patch can be used. It requires the Jasch Object externals:



Courtesy of Cycli*ng* '74. Used with permission.

15. To convert a signal to a float, for example to see what's happen with audio, or to grab a value, **[snapshot~]** can be used, as can the number box on the toolbar with a green wave in its triangle, called **[number~]**. Its right output is a float, polled at the rate specified by its inspector (control-i).

16. Look up the help patches on **[key]** and **[mousestate]** for simple means of interaction.

MSP Stuff

1. **Filters**: **[svf~]** simultaneously outputs highpass, lowpass, bandpass, and notch (band reject) responses. You can control f and Q, but not gain. Note in the help patch how the various outputs are gated. You could use **[gate~]**—see the help patch for that. **[reson~]** is a simple filter, but this time you can control all three: f, Q, and gain. **[lores~]** is a cheap lowpass filter, with all available parameters. **[biquad~]** is the most powerful, but is hard to use since you feed it five rather obscure coefficients, not f, Q, and gain. So we have **[filtergraph~]** which, if there is one filter, goes to the second inlet of **[biquad~]**, and to **[cascade~]** (a cascade of [biquad~]s) if there is more than one filter. The help patch explains the various messages; in particular filter type (see the **[umenu]**, the drop-down menu with the various types, which you can copy from the help patch) and lnfilterl. There is a lot to [filtergraph~].

2. **Reverb**: **[yafr2]** should work; if it doesn't, it can be found in the MaxMsp folder>examples>effects>reverb>lib; copy into your current directory. The parameters take arbitrary values (0–127); experiment. If you have TapTools (you can purchase it from electrotap.com, though I have a back-up copy for my own purposes stored in my Public/21m.361 (NB. case sensitive) directory), you can use **[tap.verb~]**, which is very nice, and not too CPU expensive. Look into **[yafr2]** to see how reverbs are constructed: chains of **[allpass~]** filters (which affect phase, not frequency, and are used for subtle frequency-dependent delays).

3. **Preset**. Eight is the default, resize for more or less. [preset] not connected to anything will store and recall all changeable elements of a patch. See the help patch. If you connect the outlet of [preset] to inlets of changeable elements, only those elements will be stored and recalled. Otherwise, everything will be. To store, shift-click on a box; to recall, just click. The current present is highlighted purple-blue; any preset that has values has a square dot in it. If you are really mad about Max, check out **[mtr]**. And even madder, the **[pattr...]** set of attribute objects—we look at them in 21M.540.

4. **[fiddle~] is a pitch follower**. It is extremely effective. See the help patch. It can track many frequencies; it can also serve as a frequency-dependent amplitude envelope follower. The left output is the predominant frequency, expressed as a MIDI note number. 60 is

middle C. Use **[mtof]** to convert MIDI note numbers to frequency (and **[ftom]** for the opposite). Note that integer boxes can be set to show MIDI or C4 note names. **[bonk~]** is a percussion detector, which would, for instance, pick up on fricatives, but not so much on sibilants. Threshold can be set. It works by detecting amplitude envelopes that pass above a high threshold and then below the low threshold; sensitivity can be set this way. Both [bonk~] and [fiddle~] have to be downloaded, free: http://crca.ucsd.edu/~tapel/software.html for Mac versions, and http://www.akustische-kunst.org/maxmsp/other.html for Windoze. (If these don't work, google "fiddle puckette."

5. More TapTools and other fun things. **[tap.shift~]** is a pitch shifter (transposer), **[tap.spectra~]** screws around in the frequency domain, mapping input 'bins' with output 'bins.' You don't need to understand what that means, just play with it. **[tap.avg~]** and **[tap.average~]** are amplitude envelope following tools (so is **[average~]**). We look at envelope following in detail in 21M.540, as well as how [tap.spectra~] works. If you are delving into the frequency domain, play with **[gizmo~]**. And **[freqshift~]** is a time-domain frequency shifter—NB. that frequency shifting is different from pitch shifting. Try to figure out why.

6. **[delay~]** delays a signal. Very simple. Don't overlook the fact that it measures delay in samples, not milliseconds, and that you have to specify a maximum number of samples. The smallest delay time is determined by the signal vector, one of those things you don't need to know about yet. **[tapin~]** and **[tapout~]** are powerful multi-tap delays: [tapin~ *x*] describes the maximum delay, i.e., it establishes a buffer that is ever-changing; [tapout~ *a b c d*] describes the ms of as many delays as you want. They have to be connected, with nothing intervening; one [tapin~] can have many [tapout~]s, not that you'd really need that. Nothing can come between these two objects, which technically are not connected by an audio patch cord, and the data is not audio.

7. **[kink~]** and **[clip~]** are fun distortion objects. [kink~] explains how [cycle~] is actually a 512-sample floating point wavetable look-up, with a default cosine wave. But you can specify a buffer in [cycle~]. If you don't understand this, don't worry; this is a little advanced, but some of you might like to know about it. **[tap.overdrive~]** is also nice.

Other Junk

1. **MIDI**. We haven't really looked at MIDI (Musical Instrument Digital Interface) this semester. Your computer probably has a basic MIDI synthesizer. MIDI is just numbers, not sound. Use **[noteout]** for raw MIDI information. To turn a note off, you must send it a velocity of 0. The 0 will apply only to the last MIDI note number. **[midiflush]** zeroes all notes. Easier: connect **[makenote]** to [noteout]; [makenote] allows you to set duration, rather than needing to zero every note. [buddy] is useful for synchronizing MIDI data.

2. **Panic**. Don't forget to have a way of stopping all processing, not just switching off audio. [gate] will be especially useful.

3. **Trace**: good for stepping through non-clocked non-signal patches (though it works with clocks and signals sometimes). Other **debugging**: see if a signal is getting through by using [meter~] or [number~]. Check normal numbers all over the place with number boxes, and toggles. See if things bang.