Max & Time

Detecting the rhythm patterns of a performance requires measuring the time between on an off of ecah note (duration) and/or the time between a note on and the on of the following note (rhythm). This can be more complex than you might expect.

The main problem is that note streams from human performers are not always well behaved. A well behaved note stream would have criply timed durations, all notes would end before the next began, and notes in a chord would begin and end absolutely together. Reality is messy. Notes in a melodic line overlap, superfulous note on or off messages come in, chords stagger, and timings are, from the computer's perspective, all over the place.

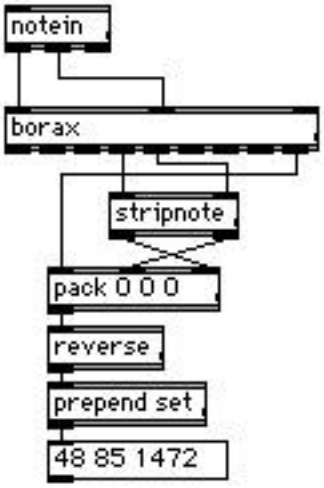
Using Borax

Borax is the basic note timing tool. According to its author, Chris Muir, it was named Borax because it "scrubs note streams sparkling clean" fixing a lot of common problems like duplicate noteons. It gives both duration of each pitch played and delta time, the time from one noteon to the next. Borax has an impressive number of outlets. Here is what's available:

1 Note number since reset (incremented with each noteon, noteoffs report the number assigned at noteon).
2 Voice number for overlapping notes
3 Number of notes now playing
4 Pitch of most recent note message
5 Velocity of most recent note message
6 Note number of noteoff
7 Duration of note that just went off
8 Note number of noteon that began delta time
9 Delta time

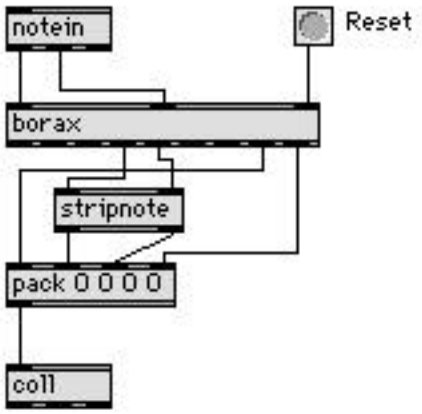
Outputs 9 and 8 are sent only with noteon, 7 and 6 are sent only with noteoff. Everything happens from right to left, of course.

Here is a patch that uses borax to show rhythm values of notes:



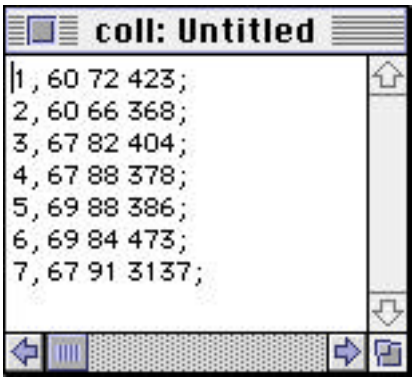
Here, the pitch 48 was played with a velocity of 85, and the next note occured 1472 ms later. The patch looks twisted, because the time between notes is not known until the following note is played. So the pitch and velocity of a noteon are loaded (backwards) into the pack, where they will stay until the next note. The delta time will eject the list, which is reversed and displayed. The pitch and velocity for the next note are immediately loaded for the next cycle.

In this next patch, a series of notes are captured into a coll:

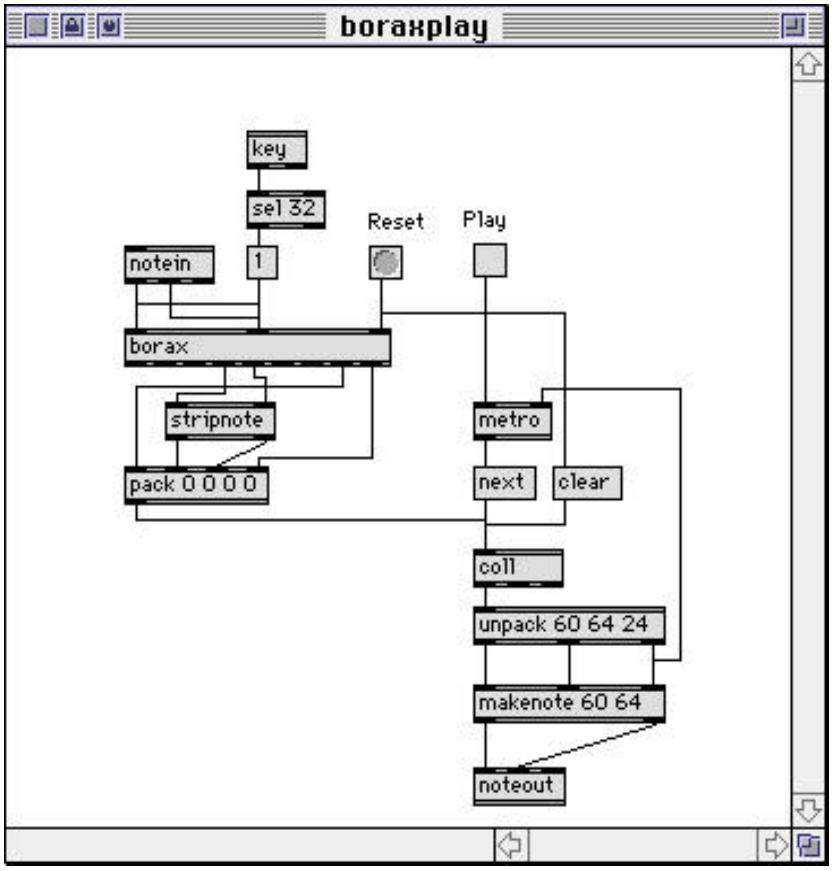


Pitch and velocity are loaded (forwards this time) into the pack, and then with the next note, delta time and the note number complete the list. The note number, as the first item of the list, becomes that address in the coll at which the pitch, velocity and rhythm value are stored.

Here's what winds up in the coll:

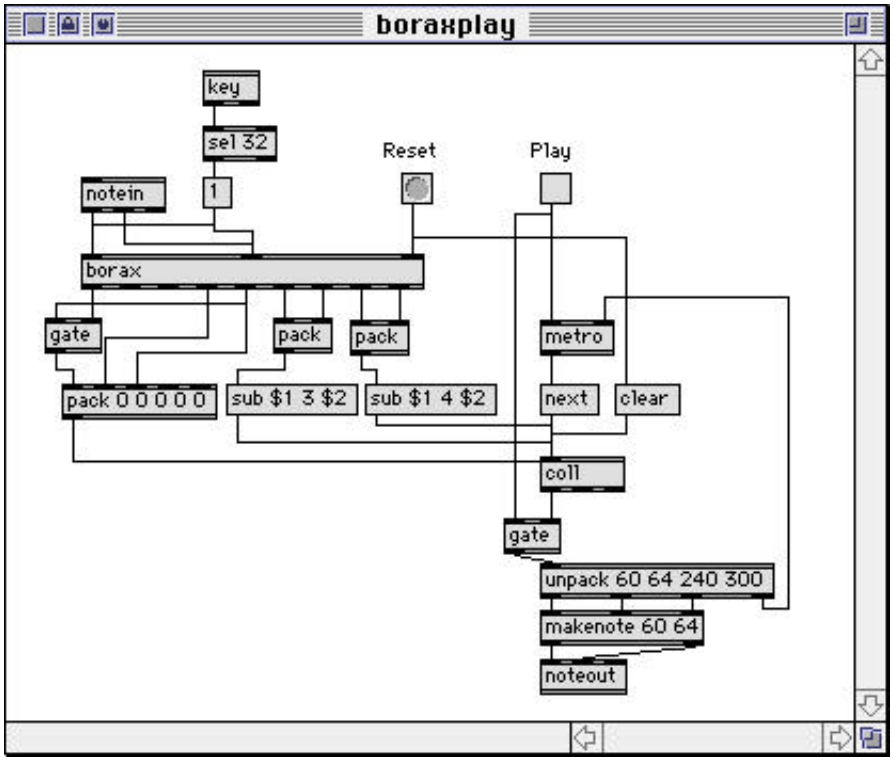


We can play this with a metro as described in the section on Max & Rhythm:



Here the delta time between notes in milliseconds is fed directly to metro to control the rhythm. The reset and clear function are there to clean out the coll and set borax back to note 1 before recording. Notice the key and sel 32 objects. They send a dummy noteon to borax so you can record a duration of the final note by hitting the spacebar.

This form of the patcher only works correctly if the notes are played carefully, one at a time with no overlaps. To deal with a more realistic situation, we have to use both duration and delta time.



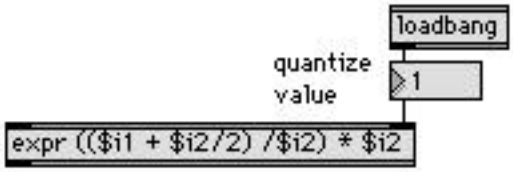
In this version, each line of the coll is constructed in three steps:

When the noteon arrives, a new entry with pitch, velocity, and 0s for duration and delta time is created. (The gate object prevents noteoffs from changing that line.)

When a noteoff arrives, a sub message sets the duration of the appropriate line of the coll. A second gate object must be added in the playback path because the sub message causes the modified line to be output from the coll.

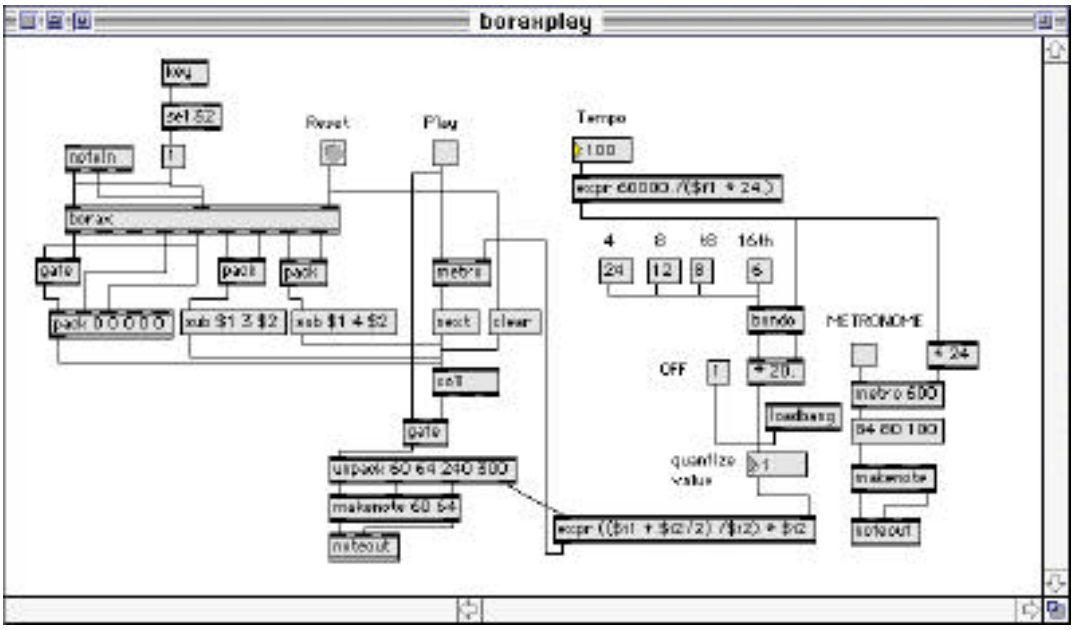
A noteon likewise sets the delta time. The noteoffs and ons may overlap, as borax keeps the times and note numbers together. (You can even play chords.)

This patch will record 250 notes. If you need more, the patch could easily be modified to use Larray instead of coll. What is the advantage of this technique over seq? Here you have direct access to the timing data. You could, for instance, quantize the delta times with a expr between the coll outlet and the metro:



This will round the delta time to the nearest multiple of the quantize value. A quantize value of 1 as shown is no quantizing, a quantize value of 300 would be equivalent to eighth notes at mm = 100. The number box lower range should be set to 1 and the laodbang included to avoid a divide by zero error (and runaway) if the user negelects to set a quantize value before playing the patch. This type of quantizing is rather brutal, but it proves the concept.

Of course, setting quantize times in millseconds is not the way musicians are used to doing things, so we provide conversion from tempo and note value:



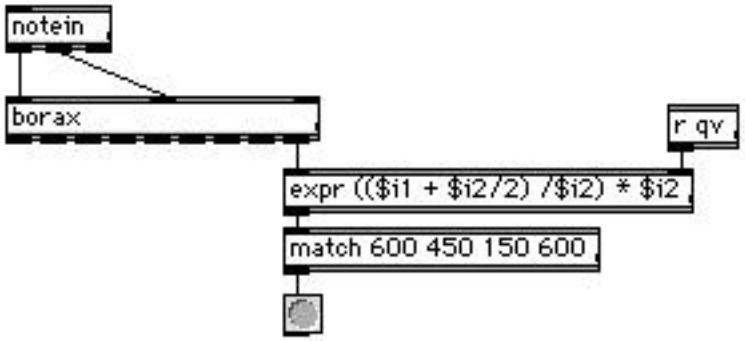
The expression

60000/(tempo * 24)

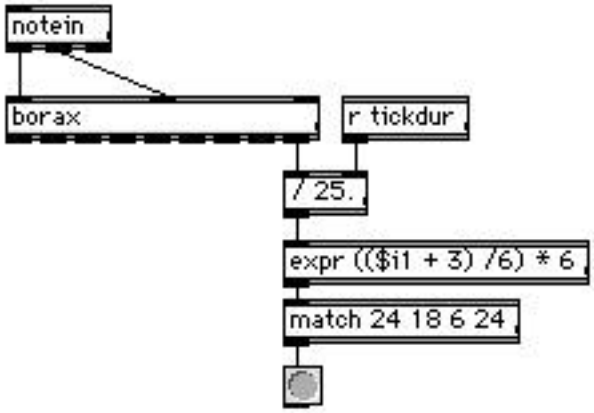
gives tick duration (tickdur) which is then multiplied by the number of ticks in the desired note to give the quantize value. This conversion also provides the correct timing for a metronome.

Rhythm Matching

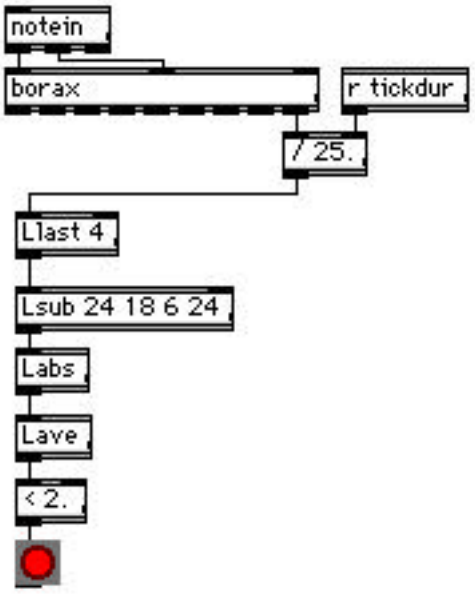
Suppose we wanted to watch an input MIDI stream and trigger an event off a particular rhythm pattern. Borax with quantizing can do this:



In this fragment of a patch, the delta times are quantized and matched. (The quantize value qv is calculated elsewhere.) When a rhythm of quarter, dotted-eighth, sixteenth, quarter, is played, match will respond. Of course, it only works at one tempo, in this case mm = 100. To make it tempo independent, the delta times must be converted to ticks:



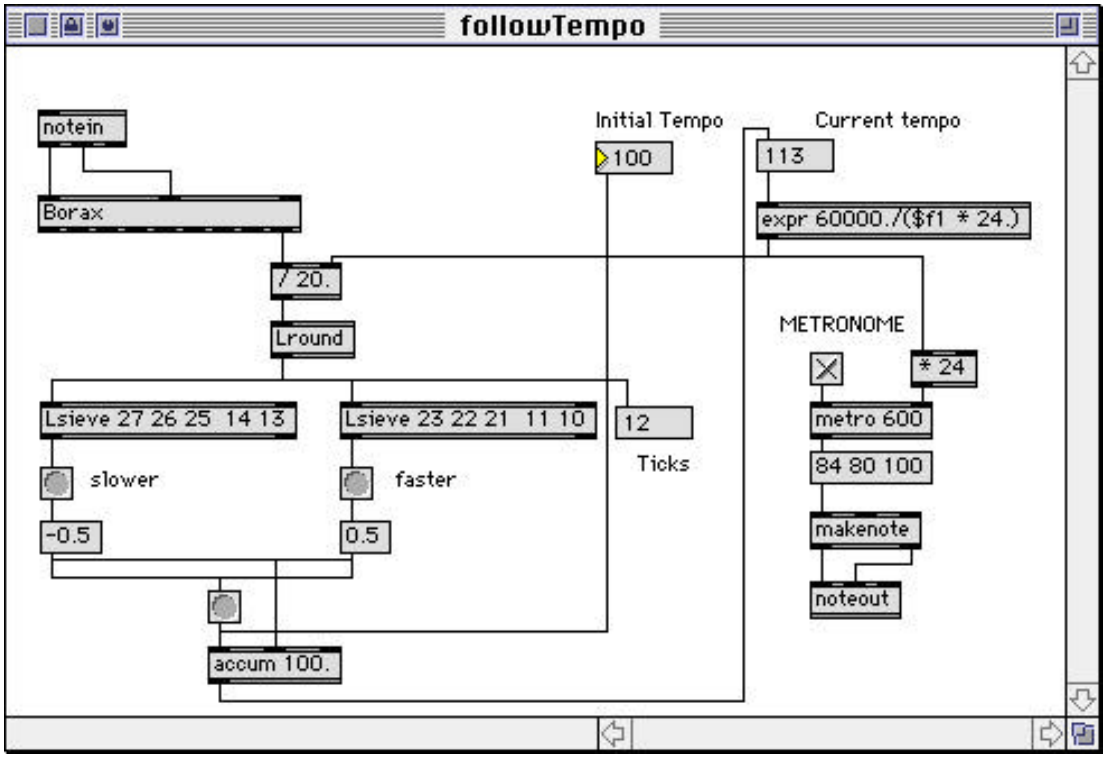
Delta time is converted to ticks by a simple division. (Tickdur is calculated as above). At 24 ticks per quarter note, the pattern to match becomes 24 18 6 24. The patch works as desired, but is pretty fussy about accurate performance. We can create a more forgiving system using some Lobjects:



Llast sends out the 4 most recent delta times. The average difference between these and the target values is computed, and anything under two ticks is accepted. This has the additional advantage of being changable, as the list in Lsub can be replaced.

Tempo Following

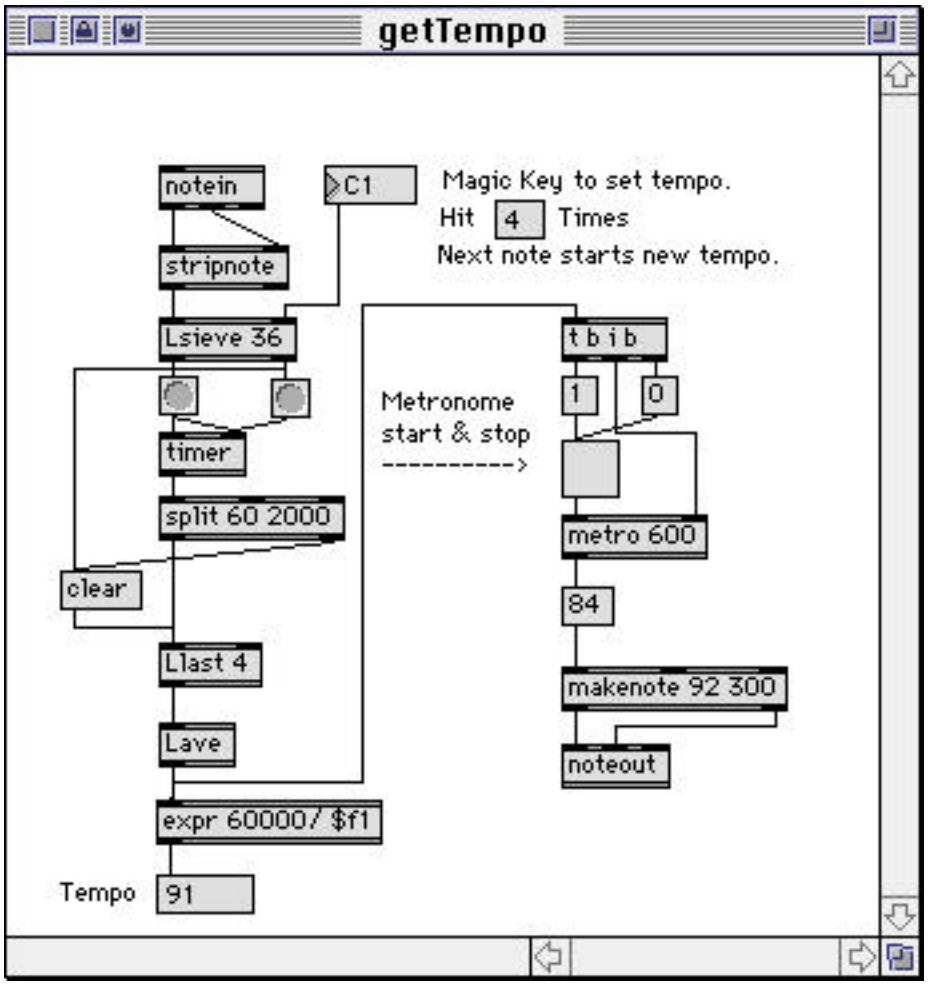
Here is a rough tempo follower:



It works by watching for quarter notes (24 ticks) or eighth notes (12 ticks) with a pair of Lsieves. If the notes are accurately timed, nothing happens. If they are a bit too long, the left Lsieve attempts to reduce the tempo. It takes two tries to actually make a change with the constants shown, since only 0.5 is subtracted from the accumulator each time. Short notes get the same treatment, speeding up the tempo.

This is only suitable for certain types of music. It responds rather slowly, and then only to quarters and eighths. It also will only track slight variations in tempo. There is an insoluble dilemma in tempo tracking- the ability to recognize a new tempo is limited by the expected accuracy of performance and finest division of rhythm used. You don't want the tempo to change just because a note was 5% early, so you ignore small variations. At the other extreme, a sixteenth note is 6 ticks and a triplet is 8 ticks, so you can't tell from the timing alone if a 25% change implies a new tempo or a different rhythm.

It may be possible to create a rhythmic analysis tool that would consider all of the cues we use to determine tempo: accented notes, melodic and harmonic patterns, but the task is daunting. (This is a job for Fuzzy Logic - see that essay.) As a practical solution, we ask the performer to explicitly tell Max what the tempo is. This can be done by assigning a special key or control to the job.



We are looking for note ons of a particular note. Lsieve lets the user choose which one. We average 4 timings with Llast and Lave. The 5th note, which sets and starts the metronome, can be anything. If it's not the magic one, the timer is stopped and Llast is cleared. If magic notes continue, the metro will follow along, but if there is a long pause before a magic note occurs again, the split will clear Llast. Very short timings will also clear Llast- this prevents the metronome from running too fast if there is a miskey. There is a hidden patch connecting the number box in the instruction to Llast. This allows the user to set the number of magic notes needed.