Max and Rhythm

Generating Rhythms

The motive force in Max is the metro object. This is an object that simply emits bangs at a steady rate. Metro can easily be turned on or off by sending a 1 or 0 into the left inlet.

The way to control rhythms in Max is to control the argument to the metro object. To do this properly, we must first understand the relationships between Period, Tempo, Note Value, and Note Duration.

The argument in the metro object is a period[1], the number of milliseconds between bangs at the metro outlet. Tempo is the number of beats per minute.

Note Value is the kind of note: half, whole, thirtysecond triplet, whatever. In Max, these must be represented by a number of some sort. There are two approaches:

> In Denominator Notation, a whole is 1, a half note is 2, a half triplet is 3, and so forth (a 32nd triplet is 48, and a dotted half is 0.75.)
>
> In Tickcount notation, the smallest value you want to represent is assigned two ticks, and all the others are represented by the number of ticks they contain. In this system, a 32nd triplet would be a 2, a plain 32nd a 3, a quarter note 24, and a whole 96.

Each system has its advantages- a list of values in denominator notation is easy to read, but tickcount calculations are usually simpler.

Duration is the amount of time the note will actually last. We have to combine Note Value and tempo to find it. If you are using denominator notation, you first find the duration of a whole note

WholeDur = 60 * 1000 * number of beats in a whole / Tempo

This only has to be done when the tempo changes. For instance, a tempo of 90 would have a WholeDur of:

240000/90 = 2666.6 ms

To find the duration of a note, you divide WholeDur by the denominator value. A quarter note at a tempo of 90 will have a duration of 666 ms.

---

[1]Of course you remember that period is the inverse of frequency. A 500 hz signal has a period of 2 milliseconds.

When using the tickcount system, the first step is to find the duration of one tick in milliseconds. The formula is

TickDur = 60* 1000/ (Tempo * tickcount of a beat)

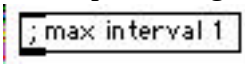
For example, again at tempo = 90, if the quarter note gets the beat

TickDur = 60* 1000/ (90 * 24) = 27.7

TickDur will be multiplied by the tickcount for each note. Again, a quarter note comes out as 666 ms.

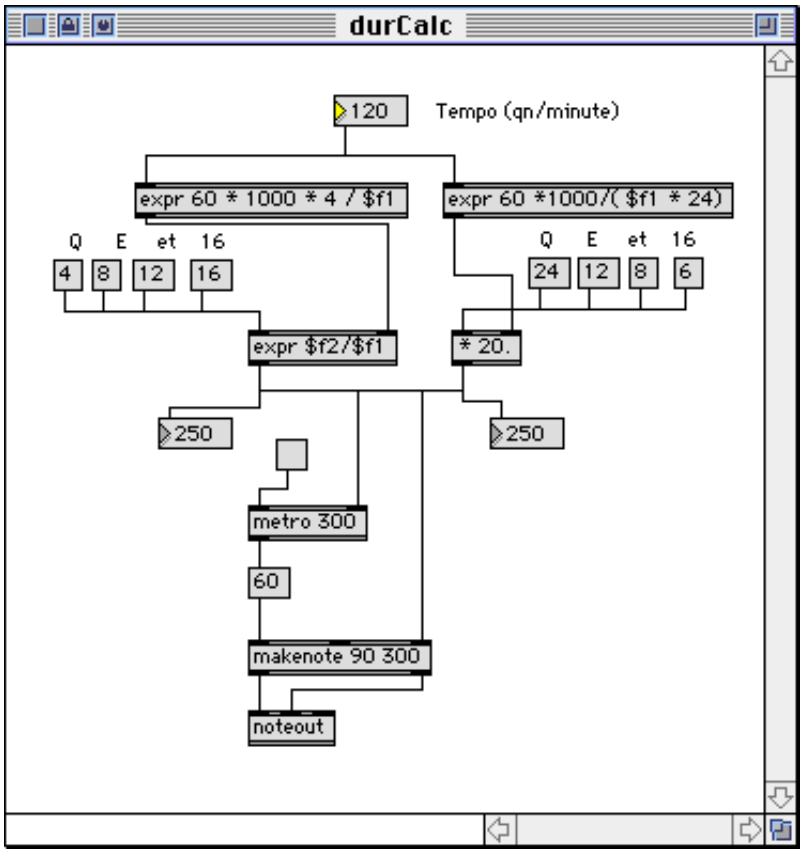
These durations can be accurately calculated, but there may be tiny errors in timing when the results are put to use. There are seveal reasons for this:

1. The metro object is only accurate to the millsecond. If you use 666 as the argument to a metro object, hoping to get a tempo of 90, it will run a mite fast. (The period of a quarter note is really 666.6 ms.) This usually not important, but it can cause occasional problems.
2. Unless you do something about it, Max really only tries to process most events once every 5 milliseconds. To speed things up, include a message box that says this:



3. Other processes in the computer can interfere with Max.
4. Macintosh timing is sort of flakey anyway.

Here is a patcher to illustrate these calculations:



This just plays Middle C at the tempo and value chosen. Turn on the metro object to hear, then click on the Q,E,et or 16 boxes to test various durations.

Notice that the operations are carried out as floats to maintain accuracy.

There are a couple of things I should point out about generating rhythms in general. First, the duration has to be sent to makenote before the note is played.

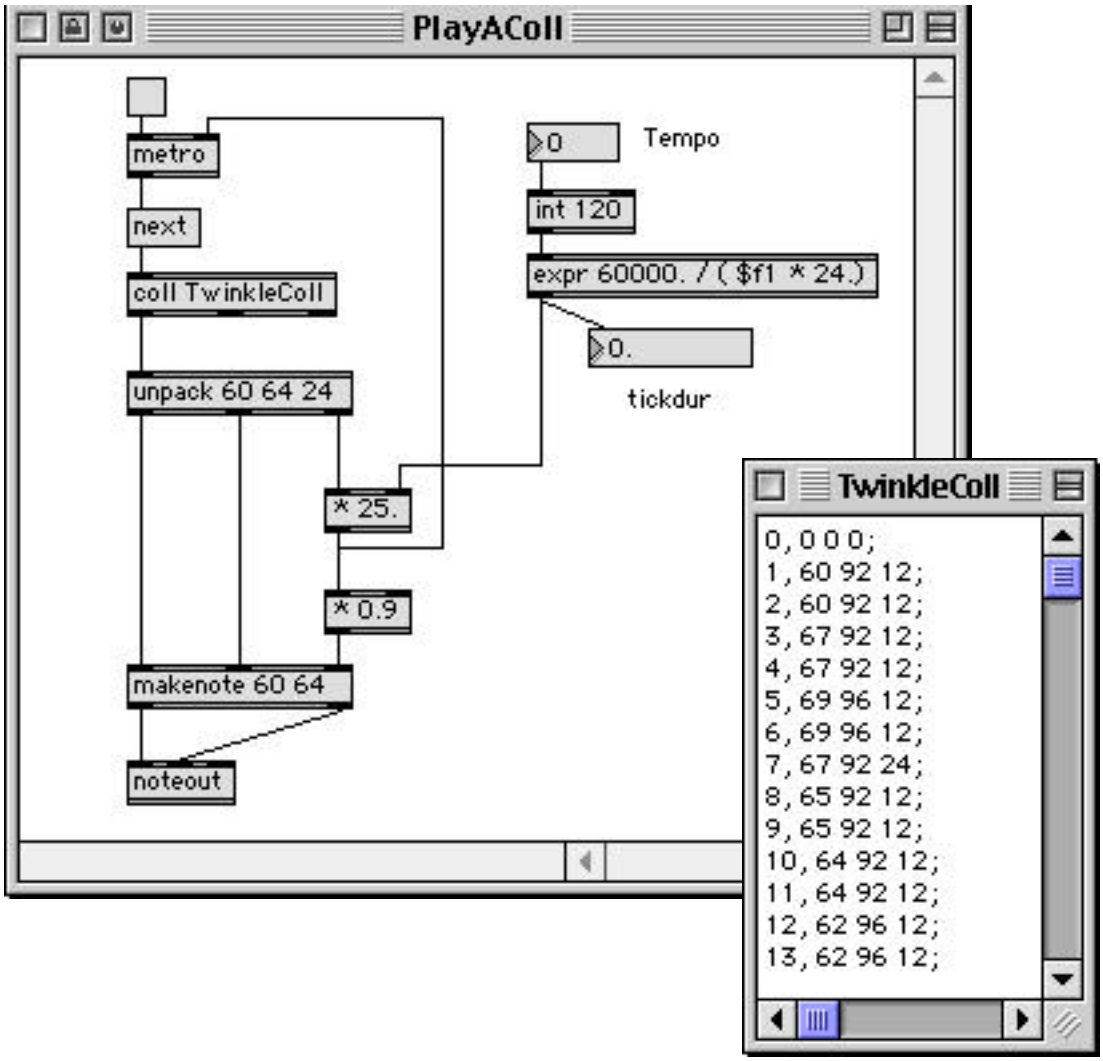
Second, the duration at makenote does not have to be the same as the duration at metro. However, when you play repeated notes like this, if makenote has a longer duration than metro, what is heard with many synthesizers will be the difference between the two durations, because of the way the note ons and offs will overlap. The messages will come out as on, on, off, on, off, on, off, on, off, off. So, it's a good idea to subtract a bit from the duration on the way into makenote. (If you use Lnote instead of makenote, this problem is handled for you. Repeated notes are always turned off before being played again.)

Third, in complex patchers, you may find yourself trying to generate a lot of notes at once for big fat chords. The system (both Max and MIDI) will tend to clog up if you

send the noteoffs for the old chord at the same time as the new notes. Again, it's a good idea to shorten the durations sent to makenote.

Patterns

Steady streams of notes eventually become boring. Most music includes a variety of durations, often in recognizable groups or patterns. Many composers generate a duration with each pitch, and then handle their notes as lists of pitch, velocity, duration. Here is how a series of note lists (stored in a coll) might be played.



Here's what's in the coll: line number, pitch, velocity, duration.

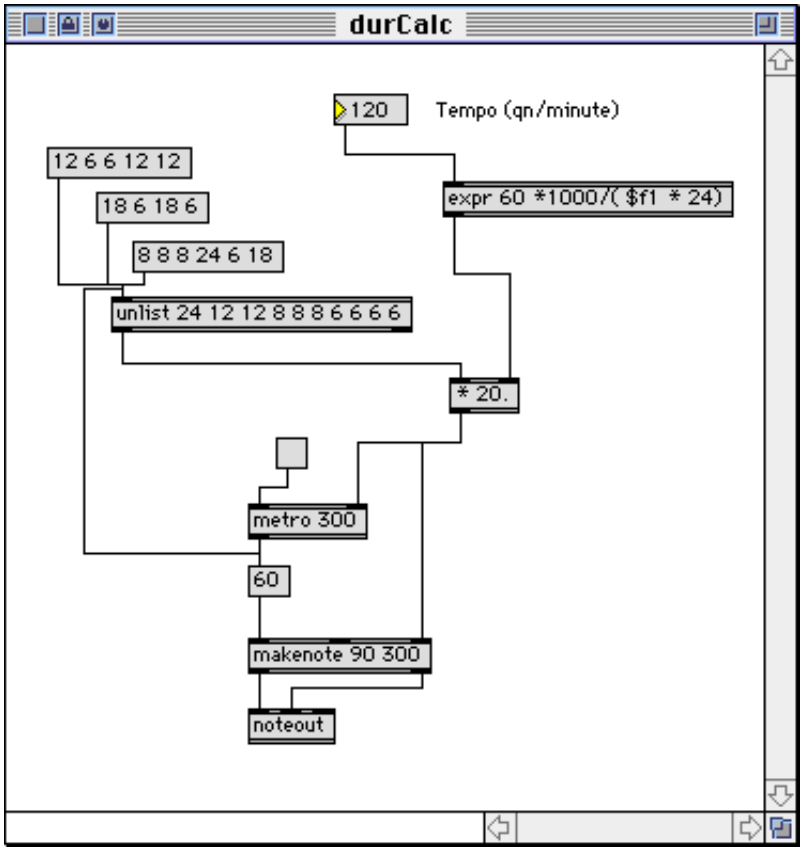
The durations are stored as tickcounts, which are multiplied by tickdur and applied to the metro. (Normally, a change in the rate of metro does not take effect until after the next

bang. However, if the change is a direct effect of a metro banging, it is effective immediately.)

unlist

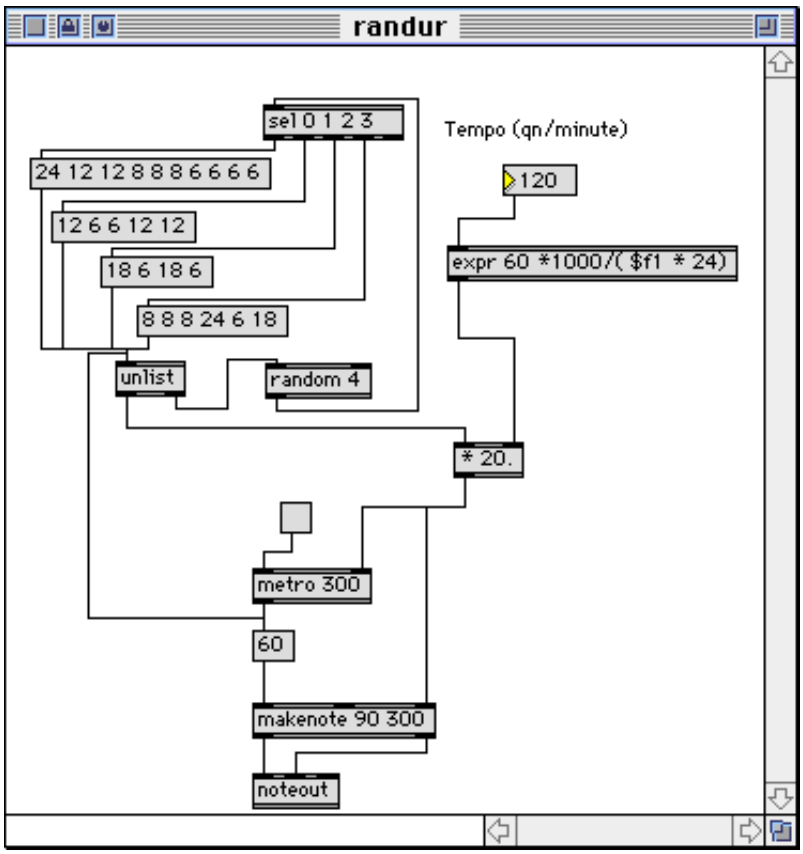
Another approach is to think of rhythm patterns as basic units that get filled up with pitches.
The problem presented to Max is to choose patterns at appropriate times. To facilitate this, I wrote an object called unlist.

The rhythm patterns, in either denominator or TickDur notation, are kept in lists: this patcher illustrates using unlist to manage them



When used with an argument list like this, unlist steps through the list over and over as the inlet is banged. If a new list is received, it replaces the stored list.

With no arguments, unlist behaves differently. When you send a list into the inlet, the first member of the list is sent out immediately. Bangs then step you through the list. When the list is used up, a bang is sent out the right outlet. If this bang triggers some process that applies a new list to the inlet, the first member is sent out and the cycle begins again, with no real break in the output stream. This patcher illustrates:



This way you can have some process that gives changing rhythm patterns.