MetaSyntheisis with Max and Jitter

Note: MetaSynth is a program by Eric Wenger, sold through UI software, www.uisoftware.com. It's a brilliant application that allows manipulation and sonification of images in a variety of ways. These patches give a similar effect, but are no substitute for the real thing.

I am using the term metasynthesis to describe the procedure of using an image to control synthesis of sound. The topic of sonification is primarily investigated by physicists and psychologists who are investigating it as a means of making patterns discernable in complex sets of data. The results are occasionally musically interesting (Some nice examples are at www.tomdukich.com). Musicians, looking for the interesting, sonify all kinds of things, because you never know what will work out. Various composers have been generating music from images since the 1920's, mostly working from drawings specifically created to control synthesis (as in Xenakis' UPIC system), but occasionally working with found images.

In actual application, found images seldom produce entire compositions, but they can generate useful source material or gestures. The simplest sonification scheme is to scan from left to right, assigning low pitch to low pixels and interpreting color as amplitude. In such a scheme, simple images are usually most successful. Image 1 one for example generates a dense tone followed by a nice melodic pattern. Image 2 is primarily a sustained chord with some melodic detail, although it can be processed to provide a set of contrapuntal lines. Image 3 has some interesting rising lines over a sustained chord.


Image 1


Image 2


Image 3

Black and white images often give the clearest translation. Image 4 produces clear rhythms, where image 5 has swoops of continuous tones.

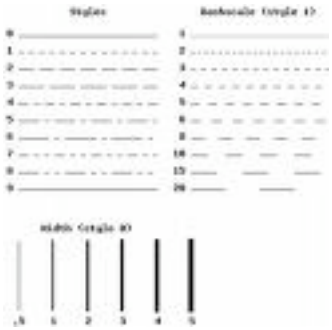


Image 4

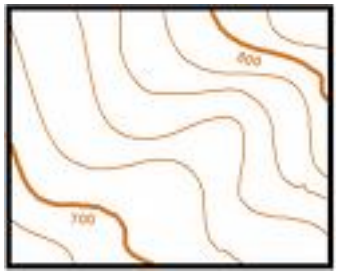


Image 5

Source

The first step to sonification of an image is to bring the image into the patch. The setup in figure 1 allows a lot of flexibility:

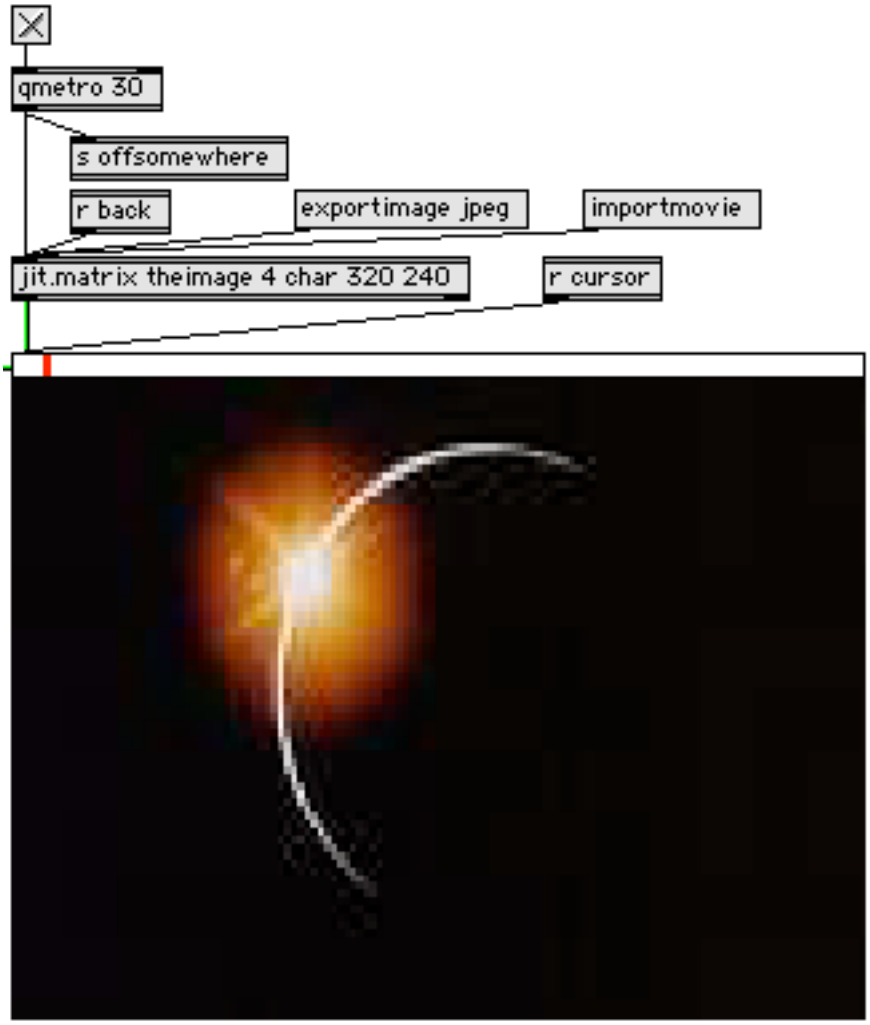


Figure 1.

The image to be scanned will reside in the matrix named "theimage". Stills can be imported directly via importmovie, and interesting images can be saved with the exportimage option. Bangs from the qmetro object are sent to "offsomewhere", and jitter messages can be brought to "back". This mechanism allows various image generators to be plugged into the patch.

The thin white space with the red mark is an lcd a few pixels tall stretched the width of the jit.pwindow. This provides a cursor display. The receive cursor will connect this to the column selection mechanism.

Process

Figure 2 shows the part of the patch that extracts information for sonification.

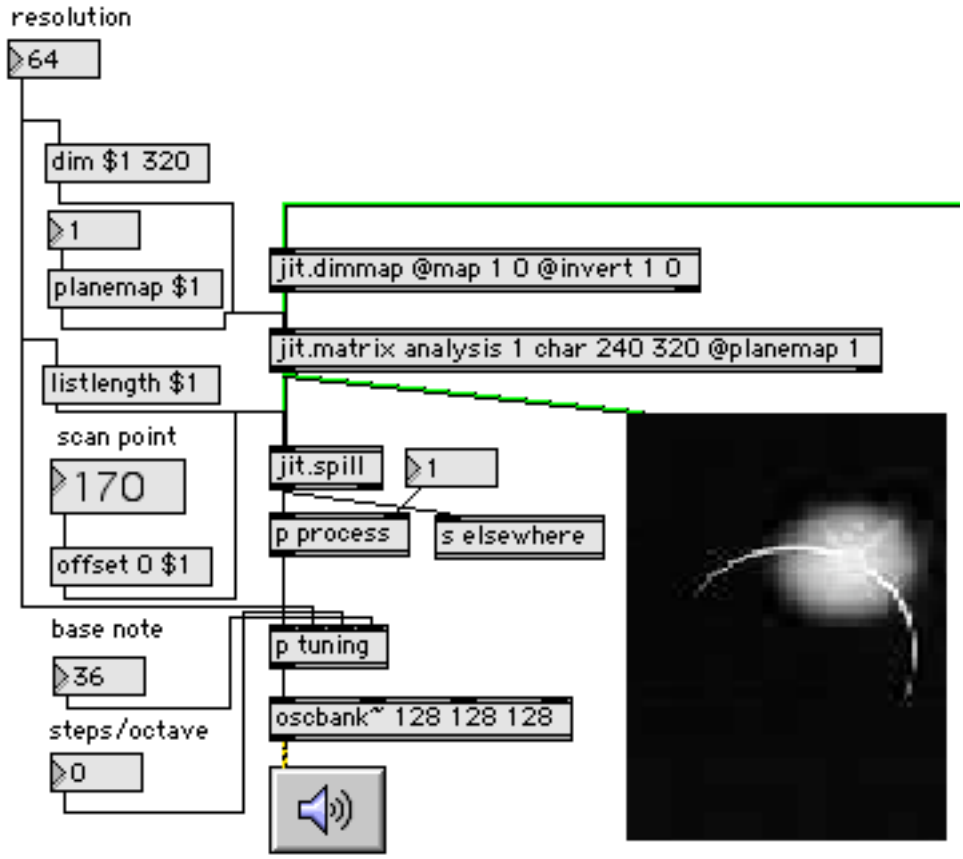


Figure 2.

The jitter martix messages are first fed to jit.dimmap with a set of arguments that turn the image on its side. The analysis matrix allows the selection of one plane of the incoming video (1=red), and reduces the vertical (now horizontal) resolution as desired. The resolution setting will be critical to the type of sound that results. The resolution control also sets the listlength of the jit.spill object which pulls one horizontal row of pixels out of the image as a list of values ranging from 0 to 255. The offset message determines which row is pulled out. Later we will automate the offset for constant scanning.

The lists produced by jit.spill are passed through a box labeled process. This is the place to experiment with modifications of the output of spill. Simple processing is best done at this stage, since only the list of current values need be manipulated. Figure 3 shows a couple of useful processes. The Lcomp subtracts all values from 255, equivalent to producing a negative of the image. This is especially useful for black on white drawings. The lexpr code is a simple edge detector, finding the difference in value between pixels. The formula always finds the bottom edge of the picture, so an lswap removes the first value from the output.

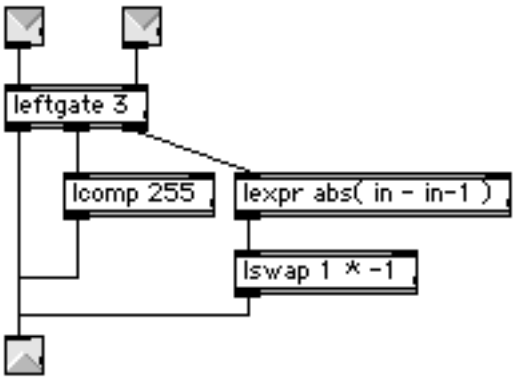


Figure 3.

Tuning

The tuning subpatch shown in figure 4 contains the tricky bits.

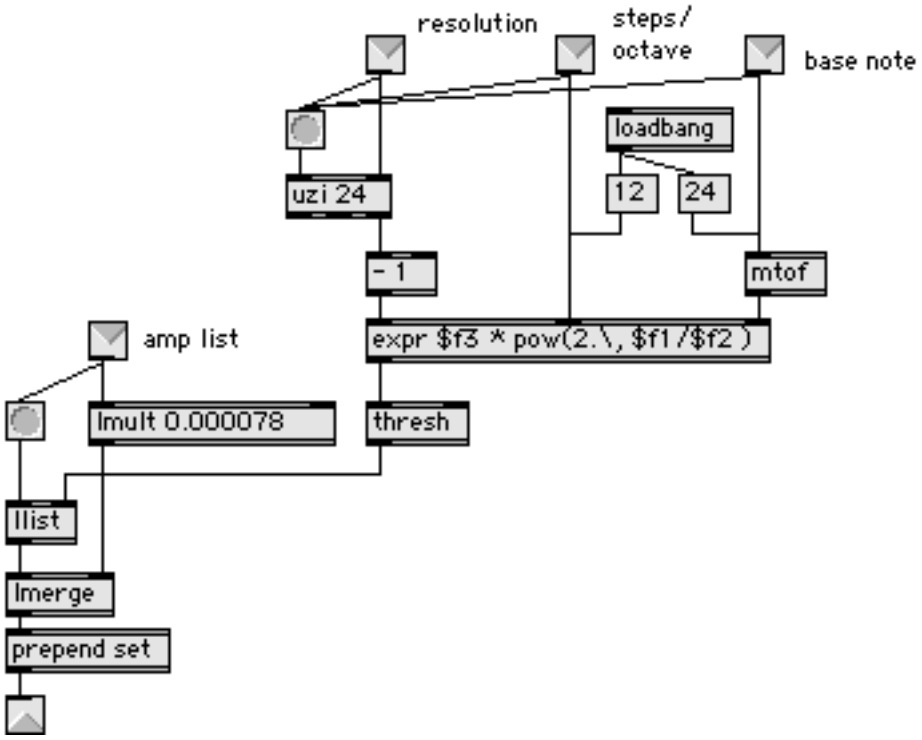


Figure 4.

The three right inlets are used to set up the details of the sonification. Basically, each pixel coming in will create a pitch in an oscbank~ object. These settings determine what those pitches are. The formula in the expr object is the standard for creating equal tempered tuning at arbitrary steps per octave. The formula is better expressed as:

$$F * 2^{n/s}$$

Where f = base pitch n = note number and s = steps per octave.

The resolution input sets the number of bangs an uzi will produce, generating a stream of frequencies that are captured by the thresh object. These are stored in the llist object. When a list comes in from jit.spill it is scaled down to something that will play without distortion and merged in to the list of frequencies. This produces the combined list of alternating frequencies and amplitudes that set the oscbank~. (see fig.2)

The mechanism in figure 5 drives the scanning.

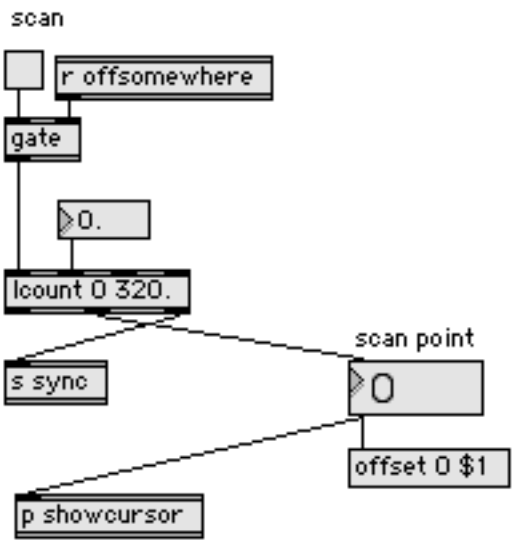
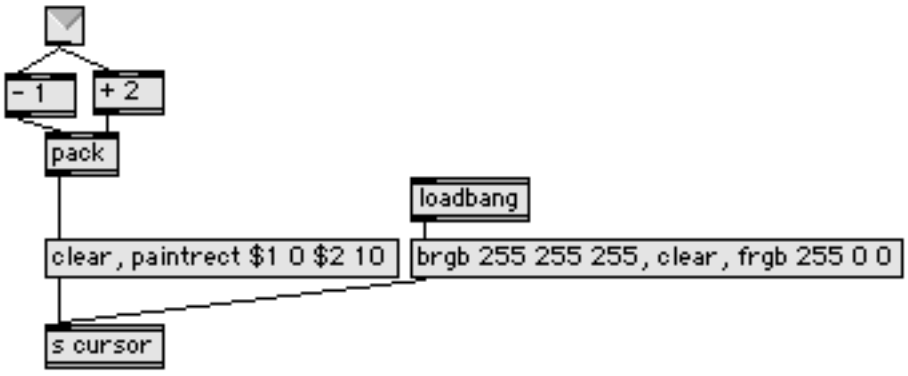


Figure 5.

Lcount used in this way advances the vertical offset to jit.spill. The second inlet to lcount is the counting increment. A fractional increment will slow scanning down from the default one step per frame. The count is also sent to the cursor lcd with a simple drawing mechanism. The send sync will transmit a bang at the end of a scan.



The complete patch is shown in figure 7.

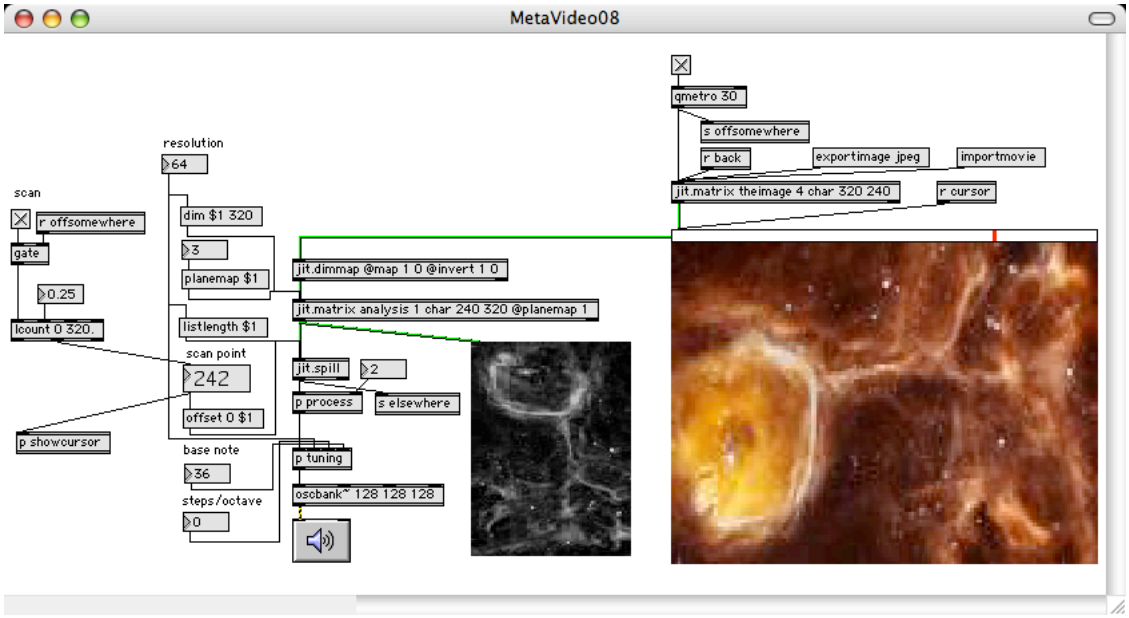


Figure 7.

Image Sources

This patch will work with more than static images. Anything jit.qt.movie can open is available with the patch in figure 8. It's very simple, just a wrapper that synchronizes a jit.qt.movie object with the scanning. If the movie is stopped, the message jump 1 advances the image a single frame at the end of each scan.

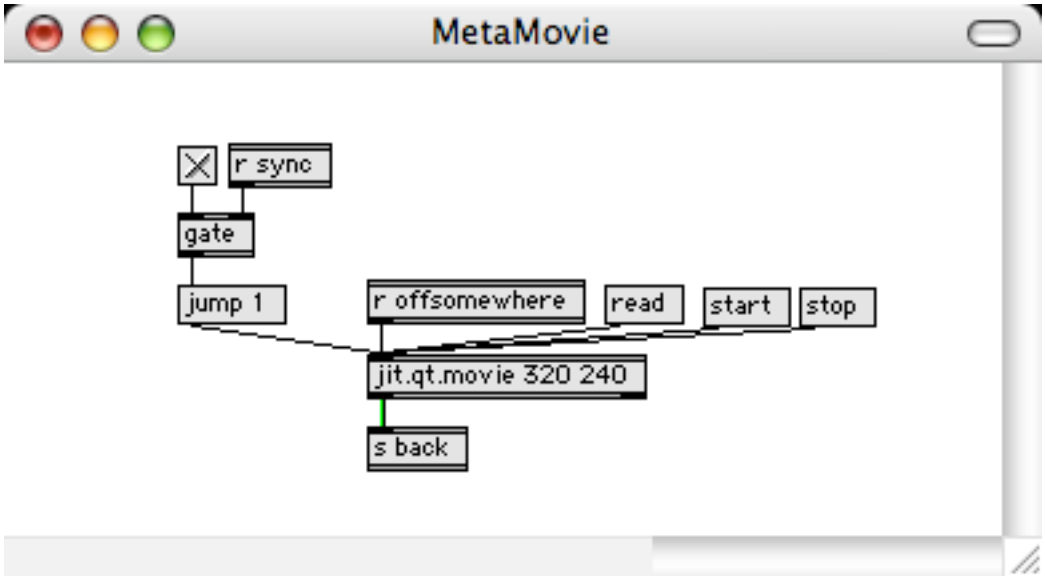


Figure 8.

Similar sources can be constructed using jit.qt.grab for live camera input, or jit.bfg for obscure math patterns.