# The ol' feedback trick.

Feedback was probably the first special effect for video. I remember seeing it on local TV in the 50's. (Usually a mistake, or an April fool's joke.) It's responsible for the opening credits in Dr. Who, the wormhole effects of Star Trek, the textures of iTunes.

To make feedback in jitter, all you need is a named matrix and a way of mixing two images.

With Feedback   Original
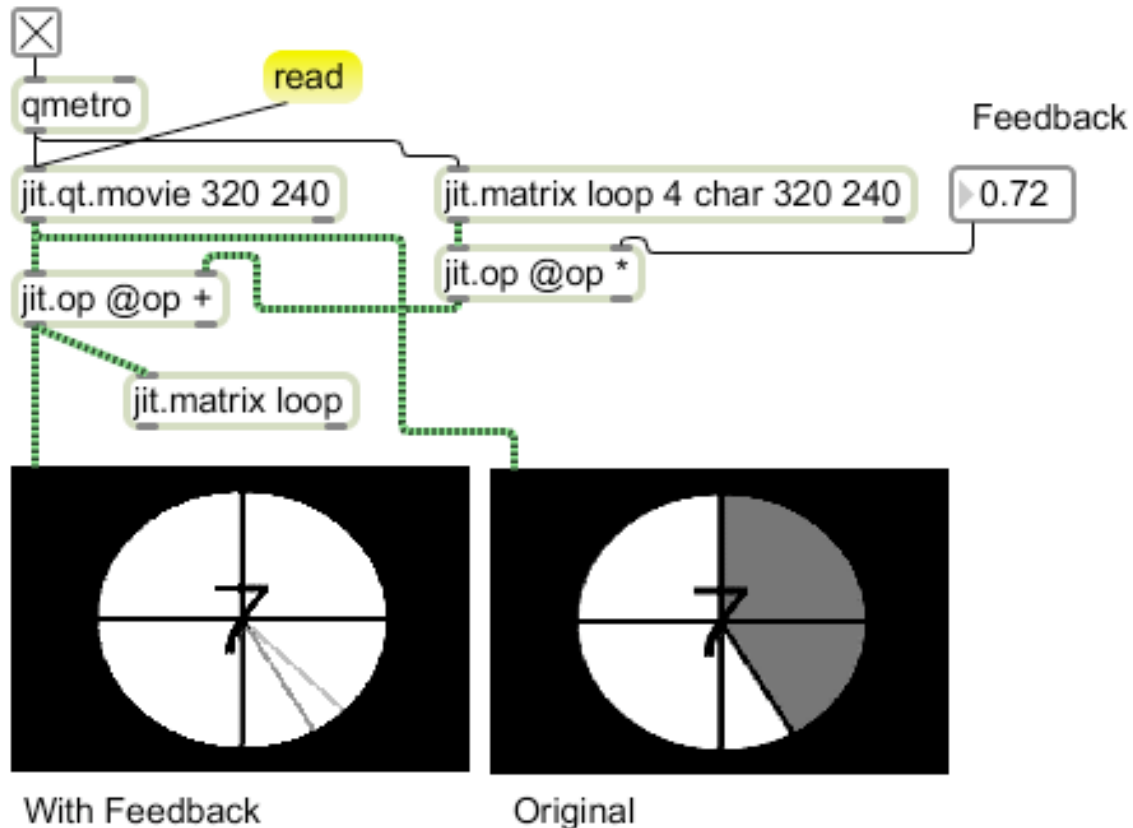
You can see that the matrix called loop appears two places in this patch. The object by the jit.qt.movie has no video input connected, but does have a source of bangs. Video is inserted into the loop by the matrix object. The same video comes out of the top matrix object 1 frame later. This is combined with the original video through a simple jit.op adder. The amount added is controlled by a jit.op multiplier. The main effect of simple feedback is to give the screen a memory. Note that the pointer is echoed a couple of times. The gray shading in the circle is lightened, and as the feedback is increased becomes totally white.
There are other ways to combine the original and fed back images. Each will produce a different effect. Simple addition is effective if the image background is black.

Processed Feedback

Here's another example. In figure 2 a subpatcher draws a group of ovals in jit.lcd on each frame with randomly changing colors. The feedback through the loop matrix simply makes the colors brighter as the succeeding frames are added together. The background of the jit.lcd is set to black so a simple addition can be used to mix the images. You can use jit.lumakey or jit.chromakey to mix the images, but jit.op gives fastest performance.
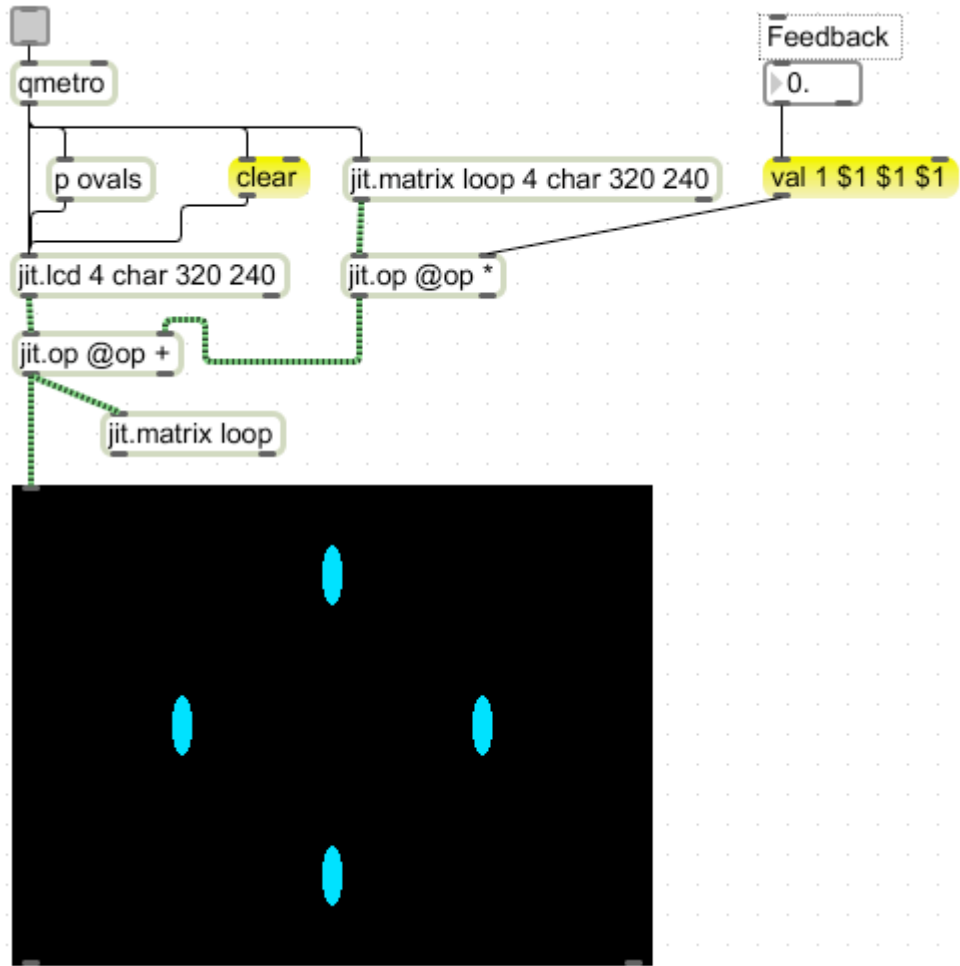

Figure 2.

In figure 3 a jit.rota is added to the loop path. The anchor attributes of jot.rota have been set to the center of the screen. This is equivalent to aiming a camera at a TV screen and turning it on its side. The feedback image is rotated a bit each time through, so objects are progressively twisted and move radially about the center.

qmetro

Feedback
0.66

p ovals   clear   jit.matrix loop 4 char 320 240   val 1 $1 $1 $1

angle
0.15

jit.lcd 4 char 320 240   jit.op @op *

jit.op @op +   theta $1

jit.rota @anchor_x 160 @anchor_y 120
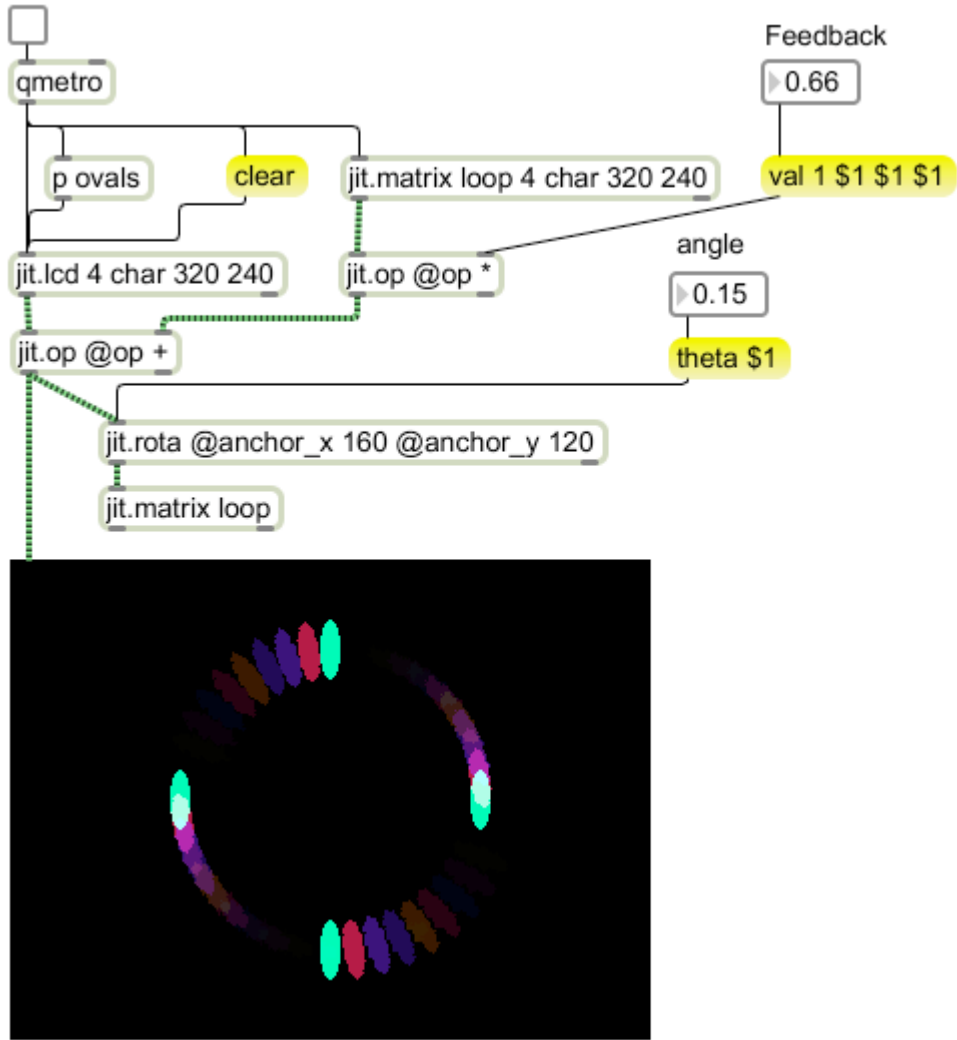
jit.matrix loop

Figure 3.
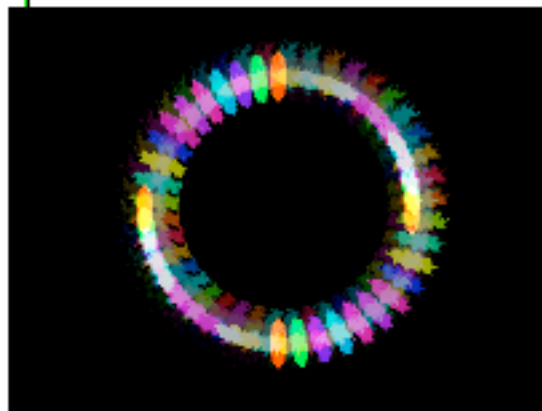As feedback is increased, the trail of fading objects gets longer and there is an impression of motion.

Figure 4.

White areas will appear wherever the color sums exceed 255 255 255. If the angle of rotation is changed, the motion stabilizes at angles that are exact subdivisions of the circle.
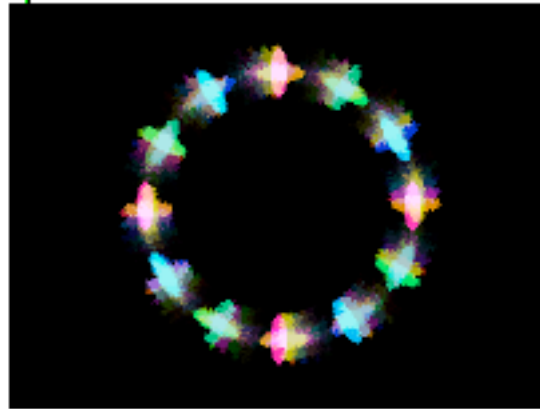
Figure 5.
In figure 5, the rotation is approximately one quarter circle. Multiple overlays create new shapes.

Jit.rota is also capable of zooming the image. Zoom_x and zoom_y can be set independently, but in figure 6A they are both set to 0.84 and in 6B they are set to 1.1 (zoom value of 1.0 does not change the image.)
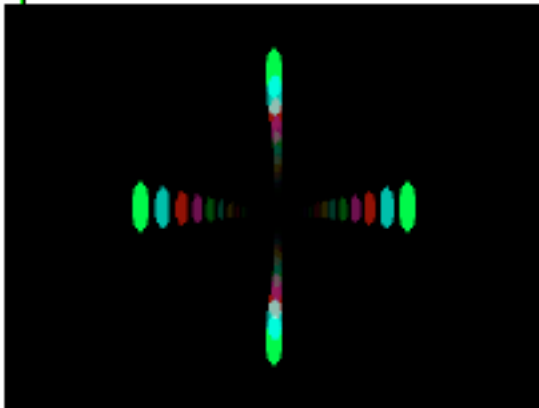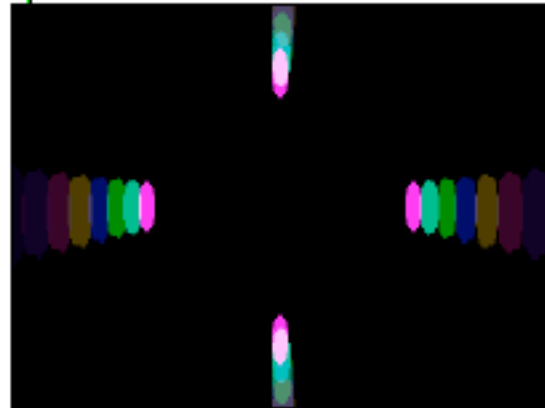

Figure 6A   Figure 6B.

Combining zoom and rotation produces falling in and rushing out effects as shown in figure 7.
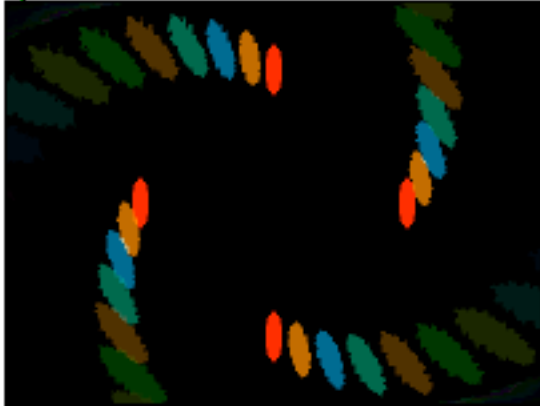
Figure 7A

Figure 7B.

Zooming and rotation can produce some artifacts. In particular, tiny or extreme values of rotation and sudden changes in zoom can fracture and distort the image:
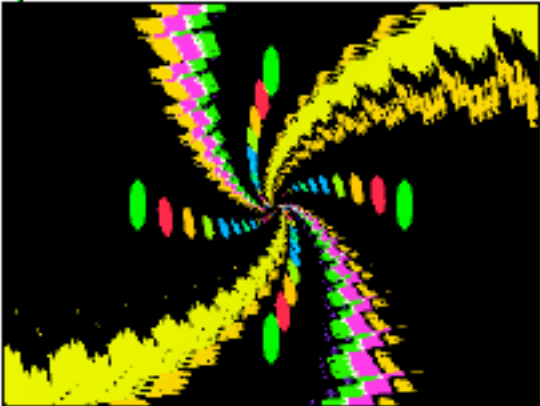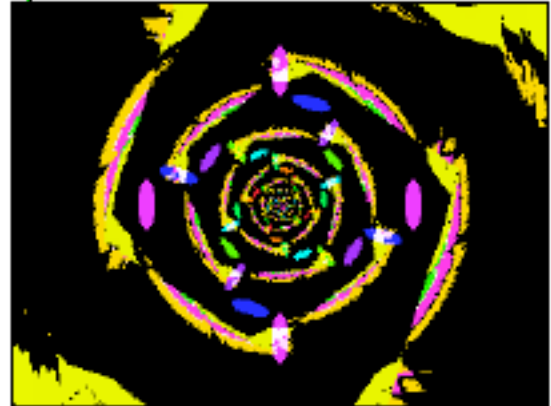

Figure 8A

Figure 8B

The left and right ends of the image require special treatment in rotation. Since the image is wider than it is high there is nothing to put out there on a 45 degree rotation. The boundmode attribute to jit.rota gives you a choice of five options: ignore the space and leave it as it is (default); clear it (turns it black), wrap (copy the image over), clip (which copies the last valid pixel out to the edge) and fold (a reflection of the image). Figure 9 shows what happens when the image is zoomed out to fill the corners, then quickly zoomed in.
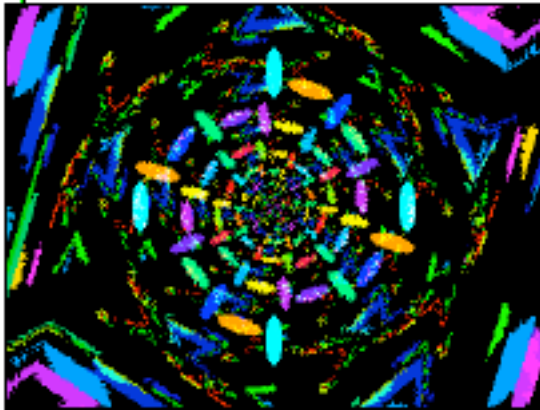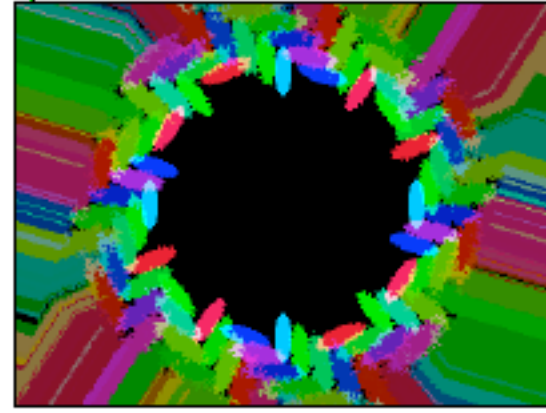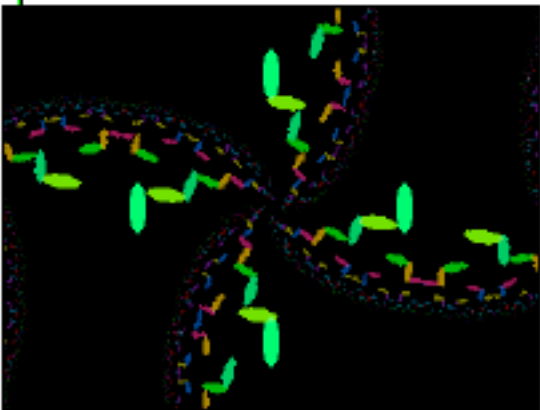
9A Ignore ends
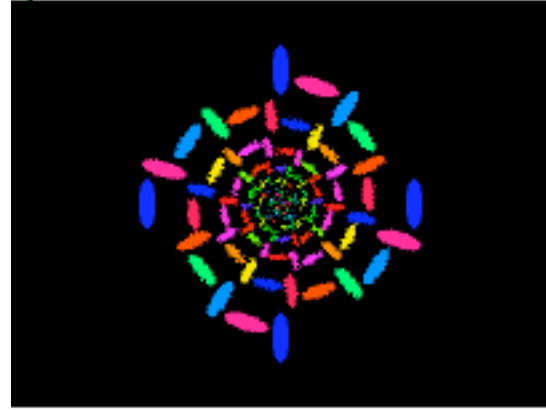
9B Clear ends
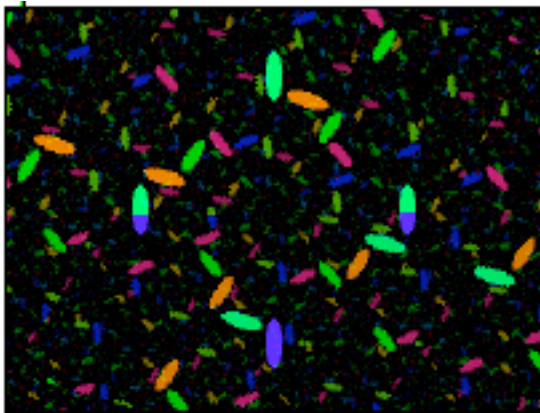
9C Wrap image

9D Clip image

9E fold image.

Of course, once any wrapping or folding occurs, the pixels are recycled through the feedback, giving chaotic results.

Rota has an offset feature, which is like aiming a camera at different parts of the screen. Rota also has many tricks that were never possible with cameras and monitors. For instance, x and y zoom are independent, and a series of processes vary with theta which make some interesting modulations possible. Figure 10 shows a few of these effects.
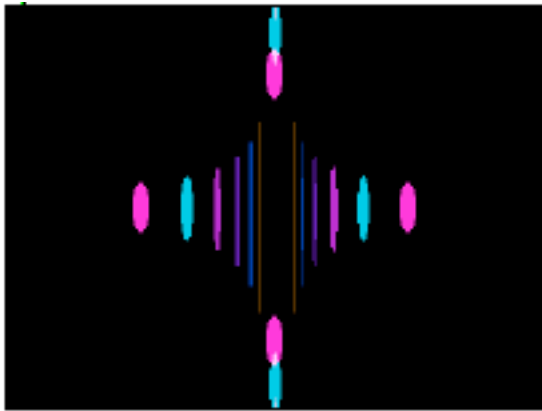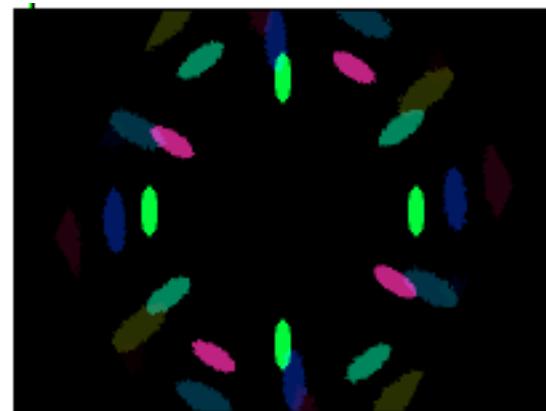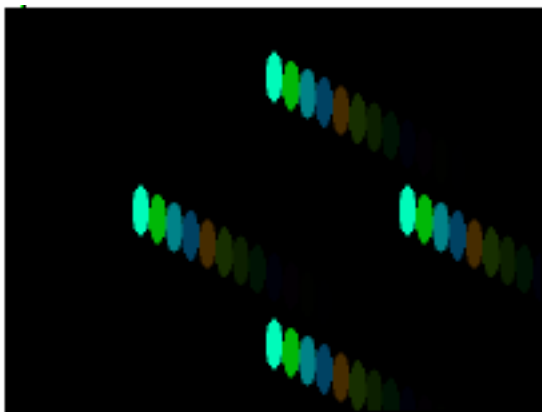
10A Asymmetrical zoom
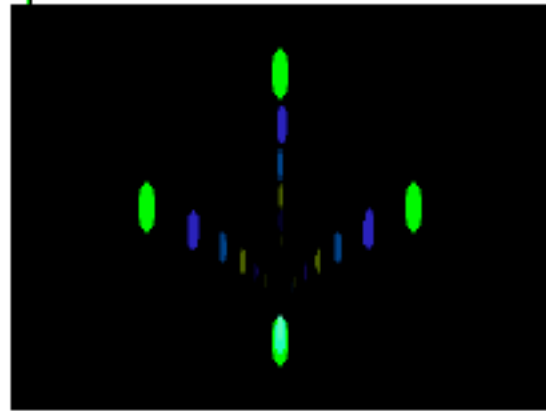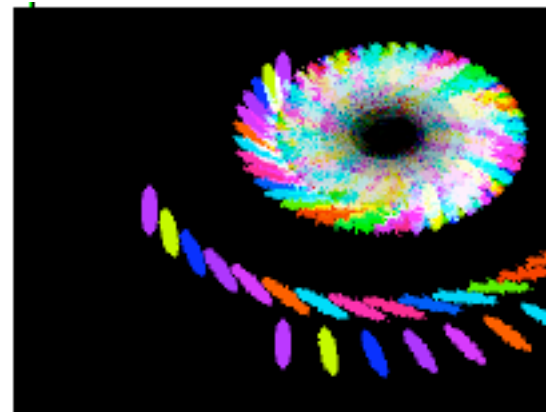

10B Asymmetrical scale with y offset
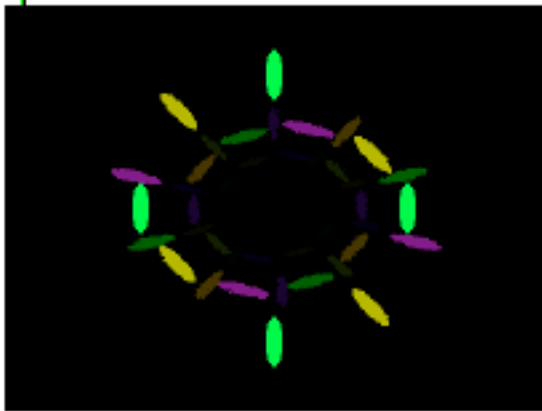
10C Theta dependent scaling theta =0.49

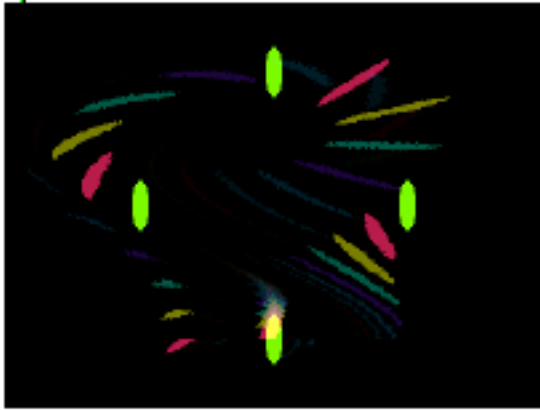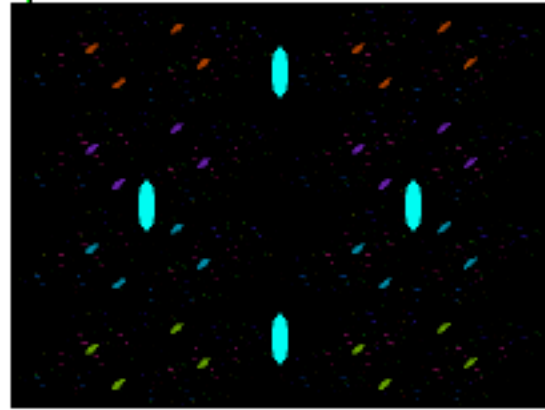10D Theta dependent scaling theta = 1.11
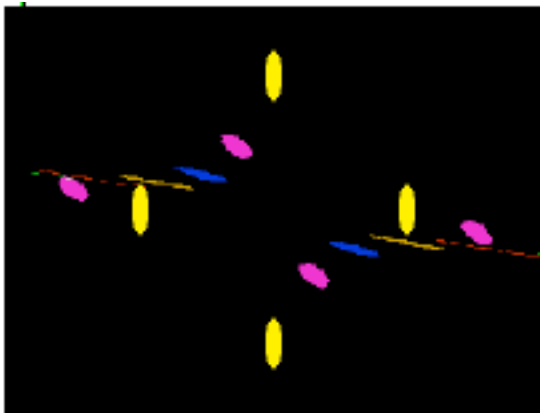

10E Offset

10F Offset and rotation

If the processing power is available other objects can be added to the feed back path:
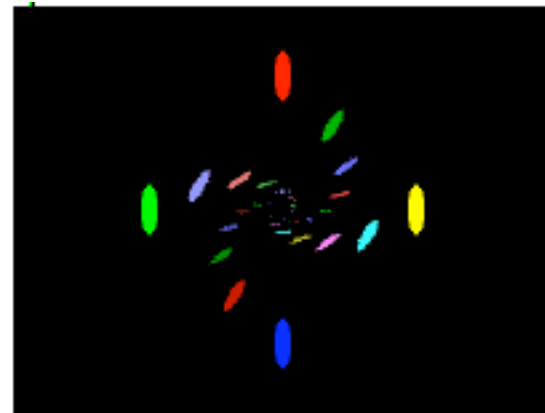
11A jit.scanoffset

11B jit.scanwrap

11C jit.transpose

11D jit.hue (colors fixed at R G B Y)

Video feedback can be a cliché, but there are plenty of unexplored variations and dynamic feedback with modulation of the control parameters is virtually unexplored territory. It remains one of the most flexible and powerful processes available.

Feedback with jit.lcd

The patch of figure 1 works with any video source. If the image originates in jit.lcd, there is a shortcut that can be useful and improve performance. In Max version 5, jit.lcd can accept a matrix input for a base image instead of a clear-- any subsequent drawing will appear on top of this image. If the matrix has a copy of the last frame, feedback will occur in a manner similar to that already explored. The patch of figure 12 is designed to take advantage of this.

Figure 12 produces a cluster of spots with each incoming bang. A new color is chosen randomly each time. Also the locus of the cluster will change according to the operation of the two drunk objects. The lmult objects will cause these clusters to be mirrored in each quadrant of the image. The result is a Rorschach-like shape that eventually fills the display.
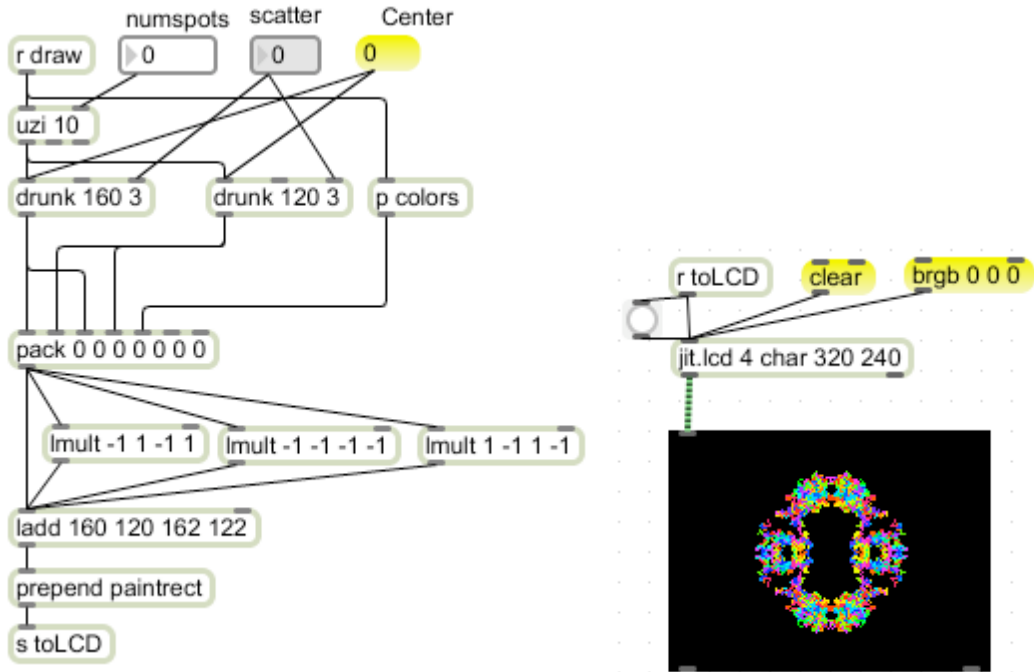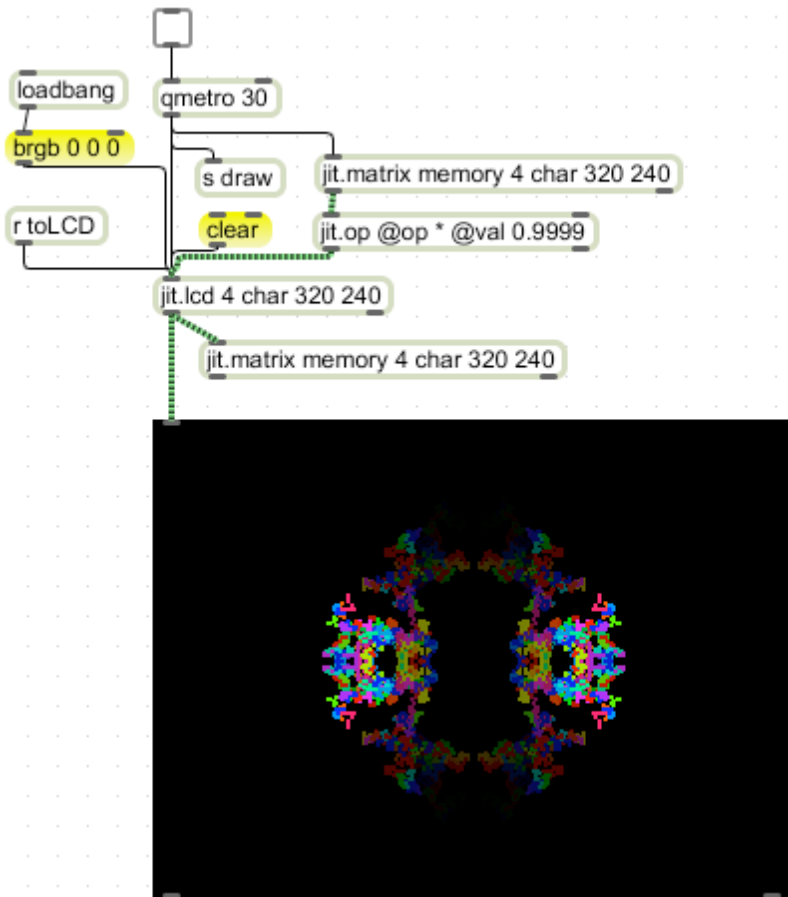
Figure 12.

Figure 13.

Feedback is applied to the jit.lcd as shown in figure 13. This creates a memory effect, where older spots gradually fade away. The timing of the feedback can be adjusted for a stately effect. Figure 14 shows a typical evolution. The controls of figure 12 produce various effects, including moving the drawing back to the center of the image.
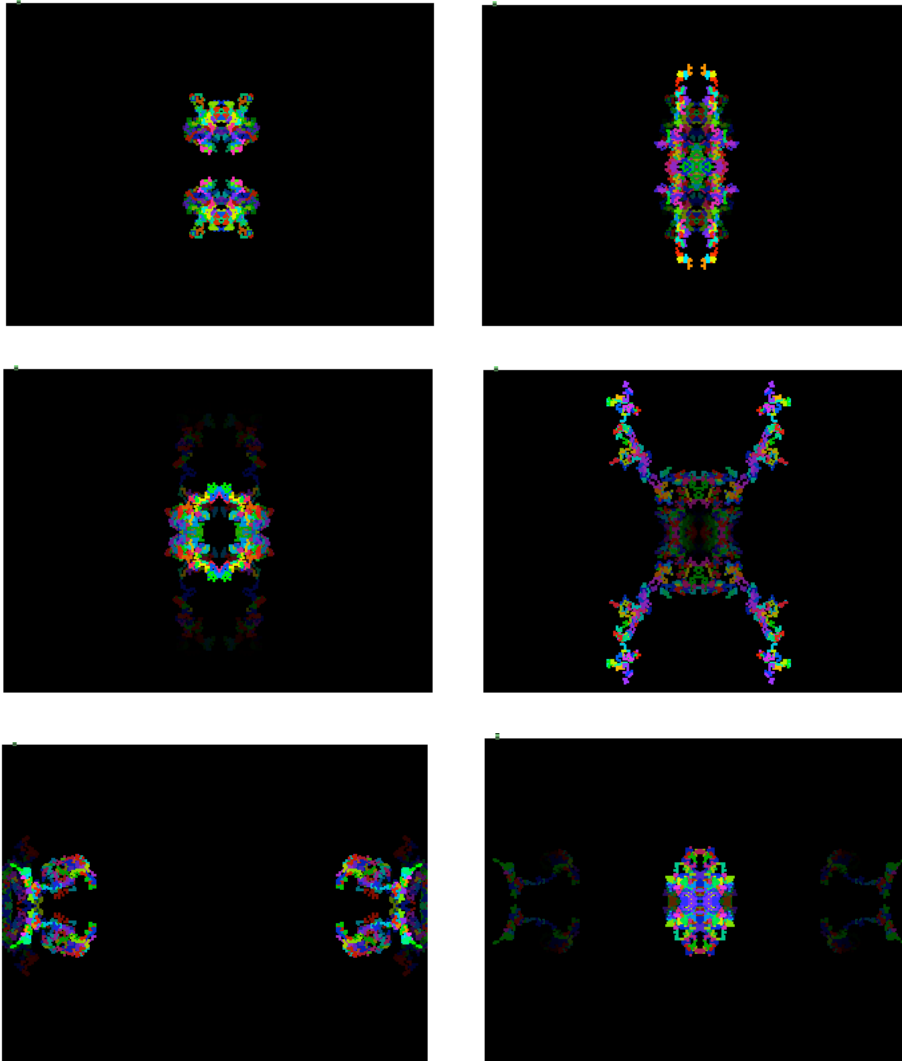

Figure 14.