File: 33cells.doc

The three files 33cellsGS.ps, 33cellsDT.ps and 33cellsLT.ps demonstrate certain characteristics of the LaserWriter's halftoning process.
They are all PostScript files which can be downloaded directly to a LaserWriter. While the code can be downloaded to any PostScript output device, results are not guaranteed to be accurate on any device other than an Apple LaserWriter. I simply haven't had access to any others to test it out. On other 300 dpi printers, the numbers may be wrong. If you output it to a device (such as a Linotronic) with other than 300 dpi resolution, some of the numbers will definitely come out wrong.

What follows is a very condensed explanation of what's going on. A fuller explanation will appear in a 3-part series on the subject in MACazine, beginning with the November issue. I'll be glad to answer any questions if you want to post them on the board to Henry Bortman.

General Explanation

The LaserWriter simulates photographic halftoning with a process known as digital halftoning. It does this by clustering groups of pixels into halftone "cells." The PostScript operator setscreen determines three things about these cells: (1) their size (how many pixels they contain), the sqaure root of which is inversely proportional their frequency (how many cells per inch); (2) their angle of orientation relative to the square output grid of pixels on the LaserWriter are determined by the PostScript operator setscreen; and (3) the order in which the pixels in a cell are turned on for various shades of gray.

The default setting for the LaserWriter is a nominal frequency of 60 and an angle of 45 degrees. You can set these values to anything you like with the setscreen operator, but no matter what you set them to, if you use a frequency between 33 and 150 (there's not much point setting it outside this range), the internals of the PostScript controller in the LaserWriter will select one of the 33 frequency/angle combinations shown on the output of these files. This is due to (1) the fact that you can't have a cell with partial pixels, so things are constrained to integer math; and (2) the necessity of the cells to neatly fit together in an interlocking pattern.

I said the default frequency is nominally 60 (60 cells per inch along both the x and y axes of the cell grid). But since it's rotated to 45 degrees, it actually becomes a frequency of 53. Here's why.

You can get an idea of what's going on by taking a 5 x 5 square piece of graph paper, and rotating on top of a larger piece of the same graph paper. The 5 x 5 square, oriented straight up and down, represents the frequency 60 angle 0 cell. (300 dpi, divided by 60 cells per inch, gives you 5 pixels per cell, along both the x and y axes. Hence, a 5 x 5 square. Remember, the frequency is inversely proportional to the square root of the number of pixels in the cell.)

As you rotate your 5 x 5 square through different angles, see which little squares on the bigger piece of graph paper get covered up. The little squares that you consider covered represent pixels in the cell. See how many different cell patterns you can come up with that will interlock without gaps or overlaps. The patterns need not be symmetrical in all four directions. Try to stay as close to 25 as you can.

All of the available cells on a LaserWriter are based on square cells that get unsquare and change the number of pixels they contain as they get rotated. The range used in these samples is from 9 x 9 (81 pixels, frequency 33) to 2 x 2 (4 pixels, frequency 150).

Each of the three files generates three pages of output. On each page are 11 samples, each representing one of the 33 cells available on the LaserWriter. The headline on each page

represents the range of nominal frequencies for the cells on that page. The actual frequencies may fall outside the range shown in the headline, due to the effects of rotation. The cells with angle 0 are the square ones.

33cellsGS.ps

This file generates a set of gray scale strips, showing all of the available gray levels for each cell (except white). As the cells get larger (lower frequencies), the number of available gray scale levels increases, and vice versa. The number of available gray levels (including white and black) is equal to the number of pixels in the cell plus 1. You'll notice that the in the first strip, frequency/angle 32/18, you can't even see the separate gray levels. In the last strip, 212/45, there are only 3 gray levels available. This is the big tradeoff, gray scale vs. resolution.

Next to each strip is a caption giving the actual frequency and angle for the corresponding cell, the number of pixels in it, and the percentage increment for each successive gray scale level (based on a scale of white = 0% to black = 100%).

33cellsDT.ps

The same set of cells, but this time showing: (a) white text dropped out of a 30% gray background screen; (b) 30% gray text; and (c) black text printed over a 30% gray background screen. To the left of each strip is the actual frequency/angle combination for the corresponding cell. Significantly different results can be achieved using different gray scale percentages. 30% was chosen because it works well for both white dropouts and black overprints.

All of these samples were created using the default LaserWriter spot function. The spot function is the procedure that controls the order in which the pixels in a halftone cell are turned on (made black). The default spot function turns on the pixels in the middle of the cell first and then moves outward. So a 30% gray would result in only the innermost 30% of the pixels being turned on. This simulates a standard halftone dot screen.

33cellLT.ps

This is the same file as the previous one, except that the spot function has been changed. Instead of simulating a dot screen, this spot function simulates a line screen. The dots closest to the x axis of the cell are turned on first, then those farther from the x axis. The angle of the lines reflects the angle of the cell's orientation.

Although I haven't fully explained it - it would take too long - there's a lot of useful information in these routines for PostScript hackers.