Grymoire Navigation

- Unix / Linux
  - Quotes
  - Bourne Shell
  - C Shell
  - File Permissions
  - Regular Expressions
  - grep
  - awk UPDATED
  - sed UPDATED
  - find
  - tar
  - inodes
- Security
  - IPv6
  - Wireless
    - Hardware
  - spam
- Deception
- PostScript
  - Halftones
- Privacy
  - Bill of Rights
- References
  - Top 10 reasons to avoid CSH
  - sed Chart PDF
  - awk Reference HTML
- Magic
- Search
- About
- Donate NEW

Google+: Bruce Barnett

Twitter: @grymoire

Part of my tutorials on **computers**

Note - this is based on an article I wrote for Flash magazine. The drawings are currently missing.

Halftones on PostScript printers

I've read about halftones for years, yet always felt uncomfortable the more I read. Halftone screens were device dependent. Mortals were not suppose to play with them. Leave that to the experts - say the manuals. Some ignored the advice and experimented anyway. Don Lancaster said in the July 1990 issue of Byte magazine that there were 22 different halftone frequency/angle screens available on a 300 DPI PostScript printer, and most people are only aware of a few. He mentioned that an angle of 35 degrees, and frequency of 85 was a good combination for reproduction. And that the 45 degree, 106 LPI was the best gray, if you wanted a single gray value. Where do they get these numbers? Out of thin air, I thought. Suppose I have a 600 or 1200 DPI printer. Which halftone screen will allow me to reproduce the best Black and White photograph? Which combination should I use?

To aggravate the situation, some printers, like the Apple LaserWriters, have a PhotoGrade feature, which produces more gray values than a regular printer, or so they say. Other printers, such as the HP, have different features that may also improve print quality.

If that's not enough, there are several different spot functions that can be used: spot, ellipse, rhomboid, line. You can also use FM and and stochastic screening. How can one choose from all of these combinations?

Don't let me scare you away. I can help.

I can't tell you what is the best halftone screen. There really are a lot of variables, and no single value is best for everybody. So I can't answer all of your questions. I can, however, answer some questions. I can also provide you with some tools you can use to experiment, and find the best halftone screen for your own use. This does requiring knowing PostScript. But before I do, I have to give you some background, which I hope you will find useful even if you never intend to use the techniques I am going to discuss.

How can mere mortals change halftone screens?

Using programs like Photoshop, you can define custom halftones. Adobe Illustrator allows you to create a "riders" file, with an optional plug-in supplied with the program. Adobe Separator Utility allows you to specify custom halftone screenings. You can also edit the PPD file for a printer, which will change the default screening for any application. And if all else fails, you can send your PostScript output to a file, instead of the printer. Then you can edit this file, and send the file directly to the printer. Before I describe this, you need some background in halftone theory.

What are Halftones?

Someone once described stochastic or FM screening as digital, and Halftone screening as analog. Originally, halftone screening is an analog technique, which really means there are large number of values between black and white. But on a PostScript printer, all halftones are digital. Some just look more digital than others.

In a PostScript Level 1 printer, there is just one type of halftone or screen function. There are three parts that must be specified:

Frequency
Angle
Spot Function

Frequency

Let's explore these in more detail. Since each black and white printer only has two colors, all shades of gray are approximated. This is done by creating a cell larger than a single pixel. A two-by-two cell has four spaces. The number of spots in this cell is zero through four, or five different combinations of black and white. There are more, as a single dot can be placed in four different positions in a two-by-two cell. However, each variation has the same approximate color: 1/4th black. These four variations of a cell with a single dot are not very useful: the level of gray remains the same.

The bigger a cell is, the larger the number of combinations. A three-by-three cell has nine pixels, so therefore has ten different shades of gray. A four-by-four cell has 17 shades of grade, five-by-five has 26 shades. In general, the number of shades of gray in a square cell is n*n+1, where n is the width of a cell. This diagram shows some cells with various shades of gray. (See diagram #1)

Note that the cell is typically filled first in the middle, and the number of cells increase as a darker shade of gray is desired. The frequency, or lines per inch, is the number of cells per inch. Increase the frequency, and the cells get smaller, allowing more detail, and less shades of gray. If the cell is aligned with the grid of the printer (i.e. the angle is 0), the relationship between cell size and frequency is easy to understand.

```
Cell width * frequency = printer resolution
```

or

```
Printer resolution/frequency = cell width
```

or

```
Printer resolution/cell width = frequency
```

A cell that is one-by-one (admittedly not very useful) allows 300 cells per inch on a 300 DPI printer. A two-by-two cell is twice the size, and half the frequency (150). A cell that is five pixels wide on a 300 DPI printer is five times wider, and therefore has a frequency of 300/5 or 60 cells per inch, or as they typically say, 60 lines per inch. These tables might make things clearer:

300 DPI Printer, angle = 0 degrees		
Lines per inch	Cell Width	Shades of gray
(Frequency)	(300/frequency)	(cell width * cell width +1)
300	1	2
150	2	5
100	3	10
75	4	17
60	5	26
50	6	37
42.86	7	50
30	10	101
19.75	16	>256

600 DPI Printer, angle = 0 degrees		
Lines per inch	Cell Width	Shades of gray
(Frequency)	(600/Frequency)	(cell width * cell width +1)
600	1	2
300	2	5
200	3	10
150	4	17
120	5	26
100	6	37
75	8	65
60	10	101
37.5	16	>256

It should be clear that the more lines per inch you have, the greater the resolution, but fewer shades of gray. In general, you would like to select the finest resolution needed for the number of shades of gray necessary. If you only have 3 shades to worry about (black, white, 50% gray) you can use a large frequency without losing shades of gray. If you need 256 shades of gray, use a lower frequency. Note that the frequency does not have to be an integer. However, each printer has a specific resolution, and selecting resolutions in between may not have any effect.

If you want to determine the needed resolution, you can think of a cell that is 16 by 16 pixels in size, This is large enough to describe 16*16 or 256 different gray levels - the maximum number of grays in a Level 1 PostScript printer. Steve Roth calls this the Rule of Sixteen. Why sixteen? Since 256 gray values require a 16 by 16 cell, divide the DPI by 16 to get the frequency needed to obtain 256 different gray values. Another way to view this is with the following table:

Maximum frequency needed to obtain 256 different gray values

DPI of printer	Frequency
	(DPI/16)
300	18.75
600	37.5
900	56.25
1000	62.5
1200	75
2400	150

So if you want a frequency of 100, and need 256 values of gray, a 1200 dpi imagesetter can't generate a sufficient number of grays with an angle of 0. You need a higher resolution imagesetter.

Angles

Did you notice I added "with an angle of 0" above? This is important, as you will see. I didn't include it to prevent lawsuits, and nasty letters to the editor. Using angles different than zero changes the rules a bit. Quite a bit, as it turns out. This particular concept puzzled me, and I investigated this further. For instance, Apple's Laser Writer Utility allows you to select the following combinations:

Frequency	53	75	83	106	150
Angle	45	0	56	45	0

The combinations to the left gave better values of gray, while the combinations on the right gave better resolution. These numbers are most confusing. Why were these particular angles selected? And how many shades of gray does each one generate? To explain, I have to show some pictures and even discuss (shudder) trigonometry. Don't fret. I include lots of pictures and tables for those who wish to avoid a repetition of high-school tramas.

A better angle on Halftones?

Some halftone screens have an angle of 0. This simply means the cell that is used to construct a halftone done is oriented the same way as the dots on the printer. The trouble is, halftone screens with this alignment generates artifacts that are very noticeable. The human eye tends to see the linear arrangement of dots. A higher resolution hides this, of course, but so does changing the angle.

Some consider 45 degrees as the best angle to use. The halftone dots are drawn on the diagonal, which many feel hides the digital patterns best. What about the other angles?

There are speed and efficiency advantages to using angles that are easy to calculate. There are also only a few fixed choices, as you cannot move the pixels around on the printer. Also remember that rotating a square 90 degrees has no effect. These facts explain why certain angles are more popular than others. 30 degrees and approximately 15 degrees are two angles that are efficiently calculated. (I'll explain more about this later). Since a one-quarter turn has no effect or a square grid, rotating a square 120 degrees is the same as rotating it 30 degrees.

If an angle is efficiently calculated, the mirror image is just as easy to calculate. This just changes the horizontal and vertical axis. A mirror image angle is found by subtracting the angle from 90. In the case of 30, this is 60, and in the case of 15, this is 75. Therefore the following angles are equivalent:

Equivalent angles

Root angle	Equivalent angles		
0	90	180	270
15	105	195	285
30	120	210	300
45	135	225	315
60	150	240	330
75	165	255	345

Because 90 minus 75 is 15, 75 and 15 produce the same frequencies when used as halftones. Therefore any frequency that works for 75 will also work for 15, 105, 195 285, 165 255 and 345. The same goes for 30 degrees, and 120, 210, 300, 60, 150, 240, and 330. So while the above table lists 24 different angles, there are only 4 that are different for halftones usage (i.e. 0, 15, 30 and 45).

Why are there so many combinations? Who cares, you might ask? Well, you should, especially if you print in color. One important use of halftone screens is dividing a document into 4 separate colors or screens for printing purposes. Adobe Separator does this, and generates transparencies for the four basic colors: Black, Magenta, Cyan and Yellow. Each color is printed as a single step, and four colors simulate the infinite combinations in reality. Proper halftones screens are a critical step in this process, and bad halftones can ruin a print job.

Moirés

If you have ever placed two identical screens on top of one another, you would notice patterns appear when you twist one of the screens. These patterns are moirés. If you had two screens, one 30 LPI and one 31 LPI, the difference would be 1 LPI. That is, every inch you would see a new "beat" frequency appear. Theoretically, each color could have the same halftone screen angle. However, this is unstable. The alignment has to be perfect for this to work. A slight registration error, 1 degree or smaller, can cause moiré patterns to form. Therefore using the same halftone screen for each of the four colors is a bad idea. The larger the angle between screens, the harder it is to see these moiré patterns. The best pattern for printing is called the "rosette." This generates a moiré pattern, but it is hard to see. This rosette pattern can be accomplished by using 3 screens 30 degrees apart. But what about the fourth color? If you used angles 0, 30, and 60, the next angle would be 90, which is the same as 0. Therefore using angles 0 and 90 in the same print job produces undesirable moiré patterns.

Why not make the angles more than thirty degrees? Why not use 45 degrees? If the first color has at 0 degrees, and the second was at 45 degrees, then the third would be at 90 degrees and the fourth at 135 degrees. Do you see the problem? Remember that rotating a square grid 90 degrees has no effect. Therefore 90 is the same as 0, and 135 has the same as 45. Therefore (0, 45, 90, 135) is the same as (0, 45, 0, 45). In other words, there are two unstable combinations that will produce moiré patterns.

If you can't pick larger angles, then smaller angles should solve the problem. Buy here's the catch: the smaller the angle, the more likely a moiré pattern will occur. So how are these angles chosen?

Adobe's Separator program (packaged with or part of Illustrator) selects a "proper" halftone screen for each color. In their Technote #4221, Adobe suggests the following angles:

Black	45 degrees
Cyan	15 (or 105) degrees
Magenta	75 (or 165) degrees
Yellow	0 (or 90) degrees

Let's see how this works. Black is given 45 degrees because it is considered the best angle, and most of the time black is the most important color. If you have an important process color, it could be at 45 degrees instead. Cyan and Magenta are 30 degrees away from black, to reduce the risk of moiré patterns. Yellow is 15 degrees away from Magenta and Cyan, so some moiré patterns might be noticed. Yes, 15 degrees difference is risky, as this is detectable. Yellow was chosen for this reason, as it is less noticeable than the other colors in many print jobs. In addition, Adobe suggests you change the frequency of yellow, making it 8 percent larger or smaller, to help further eliminate moiré patterns.

The other reason for different frequencies besides 45 is the impact on number of pixels in a cell. A diagram will help. Let me show you an array of pixels, and how a 45 degree angle fits on this array: (see diagram #2)

Notice the number of pixels in this strange shaped cell. You have to imagine a theoretical cell, placed over the grid of the printer. Only those cells that fall inside the theoretical cell are used to construct the halftone cell. As you can see, the number of pixels in a 45 degree cell varies, depending on the size of the cell. The table below lists these values:

45 degree cells				
x	y	Cell Width	Pixels in cell	Number of gray values
1	1	1.4142	2	3
2	2	2.8284	8	9
3	3	4.2426	18	19
4	4	5.6569	32	33

5	5	7.0711	50	51
6	6	8.4853	72	73
7	7	9.8995	98	99
8	8	11.3137	128	129

What about other angles?

Earlier I mentioned angles of 60, 15, etc. While you can specify any angle you want, you only get those angles that are rational. That is, the grid on the printer is fixed. If you specify something that doesn't exactly line up to the grid, the software adjusts the values so the corners of the halftone cell aligns precisely on the grid.

To put it another way, if you picture a triangle, the angle is always 90 degrees, and the width and height are always integer values on the grid. A 45 degree triangle has the height and width the same integer. If the height is twice the width, (or vice versa) the angle is 26.5651 or 63.4349. Where did this number come from, and how does it relate to the frequency? Well, it's trigonometry. Don't desert me now, after all we've been through. It's really simple. I will use some formulas, but all you need to know is how to plug the numbers into a calculator. First, let me show you a right angle, and how the angles and sides relate:

(see diagram #3)

Assume a right angle triangle, with height "x" and width "y", then the tangent of the angles are x/y and y/x. Why? Because that is how tangents were defined. If you ask a calculator for the tangent of an angle, it is equal to the height of the triangle divided by the width. But forget the tangent, because you don't need it. What you really want is the inverse function, called the arctangent. Many calculators have this function, usually called "ATAN". This is a very popular function, as it lets you calculate the angle from the height and width. Simply put,

```
ATAN(h/w) = angle1
ATAN(w/h) = angle2
```

Given a triangle that is two pixels wide and one pixel high, divide one number by the other, and take the ATAN of the number. Therefore:

```
ATAN(2/1) == ATAN(2) == 63.4349 degrees
ATAN(1/2) == ATAN(0.5) == 26.5651 degrees
```

Therefore these are the real angles that can be used for a screen function. How does this relate to frequency? Well, if you remember the Pythagorean Theorem, the length of the third side of this angle, or length of all right triangles are defined by

```
length 2 = height 2 * width 2
```

Or

```
hypotenuse = square root (h2 * w2).
```

This formula gives the width of an theoretical square cell. Hang on, we're almost done. The hard part is over. Remember how you divide the DPI by the cell width to get the frequency? Well, it's the same for these odd angles. It's just that the cell width is a funny number. The exact formula for legal halftone angles is apparently a secret, because every book I checked, and every person I asked, didn't know it. Here it is :

```
Given
        height = an integer
        width = an integer
```

then

```
cellwidth = square root (height2+ width2)
angle = arctangent(height/width)
frequency = DPI/cellwidth
```

Having a table that lists every legal angle and frequency combination can be very useful. Nowhere else can you get this inside information! There are other reasons this table is useful. But I won't tell you yet. First, you have to memorize these numbers below. Just kidding! The combinations are there when you need them. I created the table so you don't need to know the formula above. (Am I a nice guy or what?) And there are other useful pieces of information they will help provide.

I didn't list every combination. I only listed those that used a cell 16 by 16 or smaller. Having a larger cell won't help, because a Level 1 PostScript printer only supports 256 different shades of gray. I also didn't list cells made up by doubling the size of a smaller cell. This would make the table twice the size, and your screen is too small as it is.

This table lists each angle, and the equivalent angles (by rotating 90 degrees, and taking the mirror image). So this table shows the basic angle, the three equivalent angles, the width and height of the triangle, the width of the cell and corresponding frequency:

Table A

Rational angle / frequency combinations for a 300 DPI printer							
Angle	90-angle	90+angle	180-angle	x	y	Cellwidth	Frequency
0.0000	90.0000	90.0000	180.0000	1	0	1.0000	300.0000
45.0000	45.0000	135.0000	135.0000	1	1	1.4142	212.1320
26.5651	63.4349	116.5651	153.4349	2	1	2.2361	134.1641
18.4350	71.5650	108.4350	161.5650	3	1	3.1623	94.8683
33.6901	56.3099	123.6901	146.3099	3	2	3.6056	83.2050
14.0363	75.9637	104.0363	165.9637	4	1	4.1231	72.7607
36.8699	53.1301	126.8699	143.1301	4	3	5.0000	60.0000
11.3099	78.6901	101.3099	168.6901	5	1	5.0990	58.8348
21.8014	68.1986	111.8014	158.1986	5	2	5.3852	55.7086
30.9638	59.0362	120.9638	149.0362	5	3	5.8310	51.4496
38.6598	51.3402	128.6598	141.3402	5	4	6.4031	46.8521
9.4623	80.5377	99.4623	170.5377	6	1	6.0828	49.3197
39.8056	50.1944	129.8056	140.1944	6	5	7.8102	38.4111
8.1301	81.8699	98.1301	171.8699	7	1	7.0711	42.4264
15.9454	74.0546	105.9454	164.0546	7	2	7.2801	41.2082
23.1986	66.8014	113.1986	156.8014	7	3	7.6158	39.3919
29.7449	60.2551	119.7449	150.2551	7	4	8.0623	37.2104
35.5377	54.4623	125.5377	144.4623	7	5	8.6023	34.8743
40.6013	49.3987	130.6013	139.3987	7	6	9.2195	32.5396
7.1250	82.8750	97.1250	172.8750	8	1	8.0623	37.2104
20.5561	69.4439	110.5561	159.4439	8	3	8.5440	35.1123
32.0054	57.9946	122.0054	147.9946	8	5	9.4340	31.7999
41.1860	48.8140	131.1860	138.8140	8	7	10.6301	28.2216
6.3402	83.6598	96.3402	173.6598	9	1	9.0554	33.1295
12.5288	77.4712	102.5288	167.4712	9	2	9.2195	32.5396
23.9625	66.0375	113.9625	156.0375	9	4	9.8489	30.4604
29.0546	60.9454	119.0546	150.9454	9	5	10.2956	29.1386
37.8750	52.1250	127.8750	142.1250	9	7	11.4018	26.3117
41.6336	48.3664	131.6336	138.3664	9	8	12.0416	24.9136
5.7106	84.2894	95.7106	174.2894	10	1	10.0499	29.8511
16.6993	73.3007	106.6993	163.3007	10	3	10.4403	28.7348
34.9920	55.0080	124.9920	145.0080	10	7	12.2066	24.5770
41.9872	48.0128	131.9872	138.0128	10	9	13.4536	22.2988
5.1944	84.8056	95.1944	174.8056	11	1	11.0454	27.1607
10.3049	79.6951	100.3049	169.6951	11	2	11.1803	26.8328
15.2551	74.7449	105.2551	164.7449	11	3	11.4018	26.3117
19.9831	70.0169	109.9831	160.0169	11	4	11.7047	25.6307
24.4440	65.5560	114.4440	155.5560	11	5	12.0830	24.8282
28.6105	61.3895	118.6105	151.3895	11	6	12.5300	23.9426
32.4712	57.5288	122.4712	147.5288	11	7	13.0384	23.0089

36.0274	53.9726	126.0274	143.9726	11	8	13.6015	22.0564
39.2894	50.7106	129.2894	140.7106	11	9	14.2127	21.1079
42.2737	47.7263	132.2737	137.7263	11	10	14.8661	20.1802
4.7636	85.2364	94.7636	175.2364	12	1	12.0416	24.9136
22.6199	67.3801	112.6199	157.3801	12	5	13.0000	23.0769
30.2565	59.7435	120.2565	149.7435	12	7	13.8924	21.5945
4.3987	85.6013	94.3987	175.6013	13	1	13.0384	23.0089
8.7462	81.2538	98.7462	171.2538	13	2	13.1529	22.8086
12.9946	77.0054	102.9946	167.0054	13	3	13.3417	22.4860
17.1027	72.8973	107.1027	162.8973	13	4	13.6015	22.0564
21.0375	68.9625	111.0375	158.9625	13	5	13.9284	21.5387
24.7752	65.2248	114.7752	155.2248	13	6	14.3178	20.9529
28.3008	61.6992	118.3008	151.6992	13	7	14.7648	20.3186
31.6075	58.3925	121.6075	148.3925	13	8	15.2643	19.6537
34.6952	55.3048	124.6952	145.3048	13	9	15.8114	18.9737
4.0856	85.9144	94.0856	175.9144	14	1	14.0357	21.3741
12.0948	77.9052	102.0948	167.9052	14	3	14.3178	20.9529
19.6538	70.3462	109.6538	160.3462	14	5	14.8661	20.1802
3.8141	86.1859	93.8141	176.1859	15	1	15.0333	19.9557
7.5946	82.4054	97.5946	172.4054	15	2	15.1327	19.8246
14.9314	75.0686	104.9314	165.0686	15	4	15.5242	19.3247

Phew! On a 300 DPI printer, there are 61 unique combinations. Now some of you may consider me to have a case of obsessive/compulsive behavior for listing all of them. Wrong-o. There are 51 additional combinations I didn't list, but could have, thank-you-very-much. (I do have some sensitivity to the readers, despite what the editors think).

As you recall, you can double the cell size, and half the frequency. Therefore, while I have listed one halftone screen at 45 degrees, there are eleven more frequencies which also work with 45 degrees. There are more than eleven, but again I limited this list to those cells smaller than 16 by 16. (See Table B)

Obviously, a frequency of 212.1320 isn't very useful, as there are only two colors. I also expect the smaller frequencies to be less useful, as it may be hard to distinguish between 128 shades of gray and 256 shades.

Table B

Halftone cell sizes for 300 DPI printer				
Angle	H	W	Cellwidth	Frequency
45.0000	1	1	1.4142	212.1320
45.0000	2	2	2.8284	106.0660
45.0000	3	3	4.2426	70.7107
45.0000	4	4	5.6569	53.0330
45.0000	5	5	7.0711	42.4264
45.0000	6	6	8.4853	35.3553
45.0000	7	7	9.8995	30.3046
45.0000	8	8	11.3137	26.5165
45.0000	9	9	12.7279	23.5702
45.0000	10	10	14.1421	21.2132
45.0000	11	11	15.5563	19.2847

Table B lists the valid frequencies for a 45 degree angle. 26.5 degrees has 6 different frequencies, 18.4 has 4, 33.69 has 3, 14.0, 38.9 and 11.3 have 2, while 21.8, 30.9, 38.6, 9.6, 39.8, 8.1 and 15.9 have one more frequency not listed.

Now that I've listed them, what can be done with these numbers? Especially if one doesn't have a printer with the same resolution. Here's the real secret - the resolution of the printer doesn't matter. The angles and cell width are the same. A 1 by 3 triangle always makes an angle of 18.4350, and a 3 by 1 triangle always makes the angle of 71.5650. The DPI will change the frequency, but you can calculate that by simply doubling or halving the frequency. Let me show you.

Here is a table for a 600 DPI printer, with the angle of 45 degrees.

Table C

Halftone cell sizes for 600 DPI printer				
Angle	H	W	Cellwidth	Frequency
45.0000	1	1	1.4142	424.2641
45.0000	2	2	2.8284	212.1320
45.0000	3	3	4.2426	141.4214
45.0000	4	4	5.6569	106.0660
45.0000	5	5	7.0711	84.8528
45.0000	6	6	8.4853	70.7107
45.0000	7	7	9.8995	60.6092
45.0000	8	8	11.3137	53.0330
45.0000	9	9	12.7279	47.1405
45.0000	10	10	14.1421	42.4264
45.0000	11	11	15.5563	38.5695

The frequencies change, but for each frequency at 300 DPI, the same frequency at 600 DPI doubles the cell width, which gives you 4 times the number of gray values. A 600 DPI printer has more combinations with a higher resolution, but the angles remain the same.

All together, there are 61 primary angle/frequencies, and multiples of the integers gives 51 additional combinations. The total remains the same at higher DPI, because there are a fixed number of combinations. Therefore there are 112 different angle/frequency combinations that generate cells smaller that 16 by 16 pixels, and table A lists all of the root angle/frequency combinations for a 300 DPI printer.

What else can be done with this table? How does the theoretical values compare to the real values? Let's look at the 5 combinations Apple uses. This time, I will fill in the exact angle and frequency, as well as the cell width (using Table A of course)

Angle	Frequency	Cell Width	Shades of Gray
45.0	53.0330	5.6569	33
0.0	75.0000	4	17
56.3099	83.2050	3.6056	14
45.0	106.0660	2.8284	9
0.0	150.0000	2	5

Now these numbers make a lot more sense. I counted the number of gray values by eye, and indeed, as the frequency increases, the shades of gray decrease. Angles 45 are preferred, and the angle of 56.3 is close to 45, which is why it was selected.

One piece of information that is very useful is determining how many different pixels are in each cell, which give you the shades of gray for each combination. Previously, I knew of three methods to calculate this. The first is to print it and count the number of grays. This is what my PostScript program below does. The second is to draw the square cell on the grid, and count the cells whose center is exactly inside the square. The third method is to work for a company that developed PostScript interpreters, and learn the secrets. This is highly confidential information. I could quit my job, join one of these companies, learn the secrets, tell you, and get sued. Well, maybe not.

All three methods were very unsatisfying. Now, I know you can calculate the number of pixels in a cell at angle 0 by using n squared. But what about these odd cells at a funny angle?

You know what? It works! The cell width, squared, plus one gives you the number of gray values. The trick, never before revealed, is to know the exact width and Table A provides this information. The cell width squared is also x squared + y squared, so you can take the two numbers from the table above, square each, add then together, and add one to get the numbers of available gray values. This surprised me at first, but if I stare at the numbers long enough, I'm sure it will make sense. Armed with this information, let's look at some other numbers Adobe uses for color separations.

When Adobe Separator creates four separate screens, it reads the Postscript Printer Description (or PPD) file to learn the characteristics of the printer which will print the separations. These PPD files are ASCII files, and on a Macintosh, there are stored in the printer description folder, inside the extension folder, inside the system folder. If all else fails, access the Adobe FTP site ftp.adobe.com in the /pub/adobe/printerdrivers directory, for hundreds of printers. I found the following set of lines inside the file for my old Apple LaserWriter Pro 630:

```
*% For 53 lpi / 300 dpi ================================

*ColorSepScreenAngle ProcessBlack.53lpi.300dpi/53 lpi / 300 dpi: "45.0"
*ColorSepScreenAngle CustomColor.53lpi.300dpi/53 lpi / 300 dpi: "45.0"
*ColorSepScreenAngle ProcessCyan.53lpi.300dpi/53 lpi / 300 dpi: "71.5651"
*ColorSepScreenAngle ProcessMagenta.53lpi.300dpi/53 lpi / 300 dpi: "18.4349"
*ColorSepScreenAngle ProcessYellow.53lpi.300dpi/53 lpi / 300 dpi: "0.0"

*ColorSepScreenFreq ProcessBlack.53lpi.300dpi/53 lpi / 300 dpi: "53.033"
*ColorSepScreenFreq CustomColor.53lpi.300dpi/53 lpi / 300 dpi: "53.033"
*ColorSepScreenFreq ProcessCyan.53lpi.300dpi/53 lpi / 300 dpi: "47.4342"
*ColorSepScreenFreq ProcessMagenta.53lpi.300dpi/53 lpi / 300 dpi: "47.4342"
*ColorSepScreenFreq ProcessYellow.53lpi.300dpi/53 lpi / 300 dpi: "50.0"
```

This specifies the precise combinations used for a frequency of 53 LPI. The value the person expects are first, while the exact values are in quotation marks. As I mentioned earlier, the process color and black are the same angle. These angles are not quite 15 and 75 degrees, but they are close. Using the information above, and Table A, the 4 screens are

Color	Angle	Frequency	Cell width	Number of colors
Black	45	53.033	5.6569	33
Process	45	53.033	5.6569	33
Cyan	71.5651	47.4342	6.3246	40
Magenta	18.4349	47.4342	6.3246	40
Yellow	0	50	6	37

Aha! There is some interesting information here. The frequency of yellow is 8% smaller than black. (This is a technique Adobe recommends, as I mentioned earlier). The frequency of cyan and magenta is 8% smaller than yellow. As you can see, the number of colors for magenta or cyan are 1/3rd of the number of black (or process color) values. Hmmm.

This demonstrates why it might be necessary to swap halftone screens around, to eliminate moiré and banding problems. If you used the standard patterns above, and your print contains a lot of flesh tones, you might notice some moiré patterns. This is because these tones are made from yellow and magenta, which only have a 18 degree difference. In this case, many professionals will switch the screens around, so that magenta is at 45 degrees, and black is at 75 degrees. This reduces the change of moiré patterns between yellow and magenta, because these angles are 45 degrees apart, the perfect angle! You can also see that this increases the number of colors of magenta, which is very important when it is desirous to have a high-quality flesh tones. This magenta for black swap is common for these reasons.

Another swap discussed is cyan for yellow. Therefore magenta and yellow are about 53 degrees apart, which does reduce the moirés in flesh tones. In addition, there is an increased change of moirés between cyan and yellow, as these screens are only 18 degrees apart. If you have a lot of green, this may increase the moiré patterns. This combination also doesn't give as many flesh values as the first variation, as there are only 11 * 11 combinations, instead of 33 * 37 combinations. I've read about these two different halftone swaps, but the books didn't explain the number of combination advantage swapping magenta/black swap has over swapping cyan/yellow. Table A, however, does show this advantage, once you understand how to use it. (I told you the table would be useful).

Looking at the above table, the halftone screen angle of 15.9454 (or 74.0546) and frequency 41.2082 might be useful. It is very close to 15/75 degrees, has 53 different values, and a slightly larger resolution of 41.2082 LPI. But that's only the theoretical guess. It might be worth experimentation, don't you think? It would be very hard to discover this combination by trial and error. Table A gives you freedom to experiment.

Remember, the information in Table A can be used for any resolution. Just multiply the cell width by 2, and divide the frequency by 2, for 600 DPI printers. 1200 DPI devices change the numbers by 4. 800 DPI devices changes the values by 800/300, etc.

There are two more points worth mentioning: you can always edit the PPD file if you use Adobe Separator to create special halftone screens. And if that doesn't work, save the output to a file, and edit it by hand.

The second point, and more important, is that halftone screens come from the analog days. Many print shops are unaware of the exact relationship between angles and frequencies. They may think the frequency is 53 LPI, but as the above example shows, they may get 47.4342 LPI. However, if you insist on being precise, and after reading this article you will be an expert, you might get a reputation as a smarty-pants

Spot Functions

Spot function - what are they? A digital approximation of an analog value. Typically it is a dot, whose size grows as the color approaches black. When setting halftone screens in PostScript, a function is specified. I won't go into too much detail here, except that I will mention several different spot functions, and you can try them out to see how they differ. The book "Real World Scanning and Halftones" lists several other spot functions you can try, including some fancy novelty shapes like butterflies and propellers. I don't want to go into too much detail, so I will mention that bad spot functions show up on blends, by creating noticeable changes at a particular place in the blend. If a simple dot function is used, there will be a noticeable change when the dots touch for the first time. Some use elliptical spots, which delay the time the spots touch, but they eventually do, and this eventually causes a noticeable change in the blend.

The most popular dot function, and the default in most printers, is the Euclidean spot. This changes shape several times as it varies from white to black. It starts out round, and when 50% is reached, it transforms into square. Later, it changes into a black square with a white dot. By transforming two times, the transformations are less noticeable, and therefore better.

If you want to see what the spot function does, print something using a very small LPI value. Ten lines per inch means each spot is 1/10th of an inch in size. This makes them easy to see, and easier to understand.

Experimentation Begins

Enough theory. Time to experiment with different values. First, I have to discuss the PostScript code. Setting the halftone screen in a Level 1 PostScript printer follows the following format:

frequency angle procedure setscreen

where a real example looks like: 53 45 {pop} setscreen The procedure sets the halftone screen to 45 degrees, and 53 DPI. The spot function is a primitive one that ignores one of the two coordinates, and therefore creates lines on the screen instead of spots. I am going to use a slightly different format, for reasons you will soon understand. I will define each spot function, and give it a name, i.e.:

```
/spot_line {
        pop
} def
```

I could define a spot function and load it by executing 53 45 /spot_line load setscreen But instead I will use the following PostScript code:

```
/ScreenSet {
        % set screen function
        % frequency angle /spot_function ScreenSet =
        dup str cvs /SpotFunctionName exch def
        load setscreen
} def

53 45 /spot_line ScreenSet
```

The difference between the two is this last one stores the name of the function in a variable called "SpotFunctionName". The reason I do this is because I will now describe a test program that prints out 100 values of gray, and also prints out the exact halftone screen used to generate the test pattern. I call this procedure "PrintPage". It prints out a matrix of gray squares, with 100 different values. The value of each square will be labeled, and the end squares will be duplicated on the next line, so you can see the effects of banding. The program also prints out the screen angle, frequency and name of the spot function. Trust me, when you have a stack of 100 different pieces of paper, it's easier to keep track of the differences when the values are automatically on the page. The program also prints out the time it takes to construct each screen. Some halftone screens are pre-built for efficiency. By printing out the rendering time, you can see how expensive each combination is.

The final part of the code exercises different halftone screens. There are hundreds of combinations, so I will only give you a sample. I urge you to experiment with different devices, different paper, and different screens.

Earlier I mentioned 5 different screen values for the Apple printers. To print out a sample page for each one, use the following:

```
53 45 /spot_euclid      ScreenSet PrintPage
75 0 /spot_euclid       ScreenSet PrintPage
83 56 /spot_euclid      ScreenSet PrintPage
106 45 /spot_euclid     ScreenSet PrintPage
150 0 /spot_euclid      ScreenSet PrintPage
```

To try the 10 different spot function I list below, use

```
53 45 /spot_round               ScreenSet PrintPage
```

```
53 45 /spot_iround              ScreenSet PrintPage
53 45 /spot_euclid              ScreenSet PrintPage
53 45 /spot_rhomboid    ScreenSet PrintPage
53 45 /spot_line                ScreenSet PrintPage
53 45 /spot_diamond     ScreenSet PrintPage
53 45 /spot_iellipt     ScreenSet PrintPage
53 45 /spot_rb          ScreenSet PrintPage
53 45 /spot_linea               ScreenSet PrintPage
53 45 /spot_lineb               ScreenSet PrintPage
```

The code above is at the end of the file, and prints out 15 pages. You can add as many additional combinations as you wish. (Let's see now, 112 different screens and 11 different spot functions. Well, you can print them all out if you wish.)

One more point. You can include the minimal code above before any PostScript code, and unless the file explicitly sets the halftone screen, you can specify the halftone screen to be precisely what you want. Therefore you can test it with scanned photographs or any other test you can think of, and see if certain halftone screens allow you to duplicate images better than others. Here's the halftone test program. It prints out 100 different shades of gray on a single sheet of paper. You can use this to see how many different grays you actually have (up to 100). You can also experoment with different postscript options to see what impact it has on your halftones. Enjoy!

Example 1

```
%!PS-Adobe-2.0
% Sample code to explore the different screen and spot functions
% Written by Bruce Barnett
% Inspired by Michael Thorne
% PostScript Language Journal Vol 1, Number 4
%
%

% define some abbreviations

/l /lineto load def
/m /moveto load def
/rl /rlineto load def
/rm /rmoveto load def
/sg /setgray load def
/sh /show load def
/slw /setlinewidth load def
/st /stroke load def
/tr /translate load def

% and some places to store some information
/str 300 string def % define a string
/ss 50 def     % defines square size
/fountstring 256 string def


% print a fountain
/PrintFountain {
       % create a string containing values from  0 to 255
       0 1 255 {fountstring exch dup put } for
       % scale a 1 by 1 image to the size that will spread across the page
       % first number is the width, second the height
       600 45 scale
       % construct/transform the image
       256 1 8 [256 0 0 1 0 0] {fountstring} image
} bind def


% Print a tinted box
/TintBox { %define a tinted box, size (ss by ss)
    tint 100 div sg
    newpath
    0 0 m
    ss 0 l
    ss ss l
    0 ss l
    closepath fill
} def

/LabelBox {                     % define a procedure to label a box
    10 ss 2 div  m              % move to (10, ss/2)

    lettercolor sg              % select color
    tint 3 string cvs sh
    (% grey) sh
} def

/NextLine {-1 ss mul 10 mul  -1 ss mul tr} def % goto next line

/NextLoc {ss 0 tr} def % goto next location (or place for a square)

/PrintMatrix {  % define a procedure to print a row of squares
                        % if at the end of the row, go to the next line
                        % else - go to the next location
 /tint exch def
 TintBox                                % draw the box
 LabelBox                               % add the label
 /count count 1 add def         % increase the count by one
 count 11 lt {                  % move to next spot
       NextLoc
  } {
       NextLine /count 0 def
  } ifelse
} def

% show details of layout (Halftone, etc.)
/ShowDetails {
       /str 300 string def
       100 750 m
       (Spot Procedure Name= ) show
       /SpotFunctionName load show             % => freq angle
       currentscreen                           % => freq angle proc
       pop                                             % ignore procedure
       (     Angle= ) show
       str cvs show                            % => freq
       (     Frequency= ) show
       str cvs show
} bind def

% A procedure to set the screen angle
% and remember the name of the function
% so we can print it
/ScreenSet {            % set  screen function
                                % ang freq /spot_function ScreenSet
       dup str cvs /SpotFunctionName exch def
       load setscreen
} bind def

/PrintPage {
       % also remember time to print page
       /time_start usertime def
       ShowDetails             % prints the halftone screen
       gsave                   % save the graphic state
       ss ss 10 mul tr
       0 0 m
```

```
        /count 0 def
        /lettercolor 1 def      % select white letters
        0 1 10 {PrintMatrix} for
        10 1 20 {PrintMatrix} for
        20 1 30 {PrintMatrix} for
        30 1 40 {PrintMatrix} for
        40 1 50 {PrintMatrix} for
        /lettercolor 0 def      % change to black letters
        50 1 60 {PrintMatrix} for
        60 1 70 {PrintMatrix} for
        70 1 80 {PrintMatrix} for
        80 1 90 {PrintMatrix} for
        90 1 100 {PrintMatrix} for
        grestore                %restore graphic state

        gsave                   % save it again, for the fountain
                PrintFountain
        grestore                % restore
        0 sg
        % now print the elapsed time
        100 775 m (elapsed time (milliseconds) = ) show
        usertime time_start sub str cvs show
        showpage                        % print the page
} bind def

% Here are the different spot functions
% These are suggested by Adobe

/spot_round { % simple round
                dup mul exch dup mul add 1 exch sub
} def


% Inverted Round
/spot_iround {
                dup mul exch dup mul add 1 sub
} def

% Euclidean Composite
/spot_euclid {  % default on many new PS printers
        abs exch abs 2 copy add 1 gt {
                1 sub dup mul exch 1 sub dup mul add 1 sub
        } {
                dup mul exch dup mul add 1 exch sub
        } ifelse
} def

% Rhomboid
/spot_rhomboid { % Rhomboid
        abs exch abs .8 mul add 2 div
} def

% Line
/spot_line {
        exch pop abs 1 exch sub
} def

% Diamond
/spot_diamond {
        abs exch abs 2 copy add .75 le
                {
                        dup mul exch dup mul add 1 exch sub
                } {
                        2 copy add 1.25 le {
                                .85 mul add 1 exch sub
                        } {
                                1 sub dup mul exch 1 sub dup mul add 1 sub
                } ifelse
        } ifelse
} def

% Inverted Elliptical
/spot_iellipt {
        dup mul .9 mul exch dup mul add 1 sub
} def


/spot_rb { % another from the "Red Book"
        180 mul cos exch 180 mul cos add 2 div
} def

/spot_line { % simple line
        pop
} def

/spot_line2 { % simple line going the other way
        exch pop
} def


% End of definitions, now to print

% Use 8 point Helvetica
8 /Helvetica-Bold findfont exch scalefont setfont

% each "PrintPage" prints one test page
% Print 5 test pages or different screens
% using the same spot function

53 45           /spot_euclid    ScreenSet PrintPage
75 0            /spot_euclid    ScreenSet PrintPage
83 56           /spot_euclid    ScreenSet PrintPage
106 45  /spot_euclid    ScreenSet PrintPage
150 0           /spot_euclid    ScreenSet PrintPage

% Now print 10 other spot functions, same screen

53 45   /spot_round     ScreenSet PrintPage
53 45   /spot_iround    ScreenSet PrintPage
53 45   /spot_euclid    ScreenSet PrintPage
53 45   /spot_rhomboid  ScreenSet PrintPage
53 45   /spot_line              ScreenSet PrintPage
53 45   /spot_diamond   ScreenSet PrintPage
53 45   /spot_iellipt   ScreenSet PrintPage
53 45   /spot_rb                ScreenSet PrintPage
53 45   /spot_linea     ScreenSet PrintPage
53 45   /spot_lineb     ScreenSet PrintPage

% I think you get the idea.....

% end of file
```

Example 2

Here is a more advanced program, thay lets you set the printer resolution It includes options to enable/disable the Apple LaserWriter pro options. Like the first, it prints 100 values of gray in a 10 by 10

grid, so you can compare the different gray values

```
%!PS-Adobe-2.0
%%BeginProcSet
% define some abbreviations
/f {
        findfont exch scalefont setfont
} bind def
% now define some abbreviations
/l /lineto load def
/m /moveto load def
/rl /rlineto load def
/rm /rmoveto load def
/sg /setgray load def
/sh /show load def
/slw /setlinewidth load def
/st /stroke load def
/tr /translate load def

% and some variables

/ss 50 def     % defines square size

% now for some procedures
%
/featurecleanup {

        % this procedure is used to test for a feature,
        % and clean up everything on the stack
        % when done.
        stopped
        cleartomark
        countdictstack exch sub dup 0 gt
        {
                {end}repeat
        }{
                pop
        }ifelse
} bind def


% this routine only works for printers with settable resolutions
/SetResolution {        % set resolution
                        %usage:         600 SetResolution =>
        /dpi exch def   % define first argument as dpi
        countdictstack[{
                1 dict dup
                % define HardWare resolution
                /HWResolution [dpi dpi] put setpagedevice
        }featurecleanup %]
} bind def

% this routine only works for some Apple LaserWriters (LaserWriter Pro)

/SetEnhancements {      % set FinePrint PhotoGrade
    %usage:     true true SetEnhancements =>
    % first bolean is FinePrint
    % second is PhotoGrade
    /PhotoGrade exch def        % define PhotoGrade
    /FinePrint exch def         % define FinePrint

% FinePrint
    countdictstack[{
        2 dict
        dup /PreRenderingEnhance FinePrint put
        dup /PreRenderingEnhanceDetails
        2 dict
        dup /Type 1 put
        dup /ActualPreRenderingEnhance FinePrint put
        put
        setpagedevice
    }featurecleanup %]
% photograde
        countdictstack[{
            2 dict
            dup /PostRenderingEnhance PhotoGrade put
            dup /PostRenderingEnhanceDetails
            2 dict
            dup /Type 1 put
            dup /ActualPostRenderingEnhance PhotoGrade put
            put
            setpagedevice
        }featurecleanup %]
} bind def

/TintBox { %define a tinted box, size (ss by ss)
    tint 100 div sg
    newpath
    0 0 m
    ss 0 l
    ss ss l
    0 ss l
    closepath fill
} def

/LabelBox { %define a procedure to label a box
    10 ss 2 div  m              % move to (10, ss/2)

    lettercolor sg              % select color
    tint 3 string cvs sh
    (% grey) sh
} def

/NextLine {-1 ss mul 10 mul  -1 ss mul tr} def % goto next line

/NextLoc {ss 0 tr} def % got next location

/PrintMatrix { %define a procedure to print a square
 /tint exch def
 TintBox
 LabelBox
 /count count 1 add def
 count 11 lt
 {NextLoc}
 {NextLine /count 0 def} ifelse
} def



% This procedure only works for some systems
% On other systems, it prints a "?"

/PrintProc {    % Print a procedure
        % usage:       proc PrintProc =>
        % Normally, you could use the pstack or == operators
        % to print a procedure. However, this sends the information to
        % the PostScript log. I don't want it there. I want it on the page
```

```
        % Therefore, I will use the little documented dictionary ==dict
        % and command typeprint.
        % See "Inside Postscript" page 6-3


        currentdict /==dict known {
                ==dict begin
                        typeprint
                end
        }{
                pop (?) show
        } ifelse
} bind def


% show details of layout (Halftone, etc.)
/ShowDetails {
        /str 300 string def

        100 750 m

% what is the product
        level2 {

                (Product= ) show
                systemdict /product get                         str cvs show

                (      PostScript level= ) show
                systemdict /languagelevel get   str cvs show

                (      Revision= ) show
                systemdict /revision get                str cvs show

                (      Serial Number= ) show
                systemdict /serialnumber get    str cvs show
        }{
                (PostScript Level 1) show
        } ifelse

% Second - print the current halftone setup
        100 725 m
        level2 { %ifelse
                % Postscript Level 2 has 5 types of halftone
                % must get dictionary to determine which type

                currenthalftone begin   % get halftone dict
                /HalftoneType load              % => HalftoneType
                dup 1 eq {                              % if type 1
                        ( Frequency=) show      Frequency str cvs show
                        (   Angle=) show        Angle str cvs show
                        (   SpotFunction=) show /SpotFunction load PrintProc
                } if
                % HalftoneType still on the stack
                dup 3 eq { % if dithered
                        ( Dithered Halftone ) show
                        (   Height=) show       Height str cvs show
                        (   Width=) show        Width str cvs show
                        (               Array = [) show

                        % show the array
                        1 1 Height  {           % go from 0 .. Height -1                                       1 add
                                /h exch def     % h = height
                                % remember currentpoint
                                currentpoint /curr_y exch def /curr_x exch def

                                1 1 Width  {    % go from 0 .. Width-1
                                        /w exch def             %w = width
                                        % calculate index into threshold array
                                        % i=(h-1)*(Height) + (w -1)
                                        h -1 add Height mul w -1 add add /i exch def
                                        /Thresholds load i get  % => threshold_value
                                        str cvs show ( ) show
                                } for
                                curr_x curr_y moveto
                                0 -10 rmoveto           % move 10 points down
                        } for
                        (    ] ) show
                } if
                % if type 2, 4 or 5, ignore
                pop
                end     % dictionary
        }{
                % PostScript level 1 only has one type of halftone
                currentscreen   % => freq angle proc
                (Spot Procedure= ) show
                PrintProc       % => freq angle

                (Angle= ) show
                str cvs show    % => freq
                (     Frequency= ) show
                str cvs show
        } ifelse


        level2 {
                100 700 m
                        currentpagedevice /HWResolution known { %if
                        currentpagedevice
                                (Resolution=) show      %
                                /HWResolution get       % => [dpi dpi]
                                dup 1 get str cvs show
                                ( by ) show
                                1 get str cvs show
                } if

                100 675 m

                currentpagedevice /PreRenderingEnhance  known { %if
                        currentpagedevice
                                ( FinePrint= ) show     %
                                /PreRenderingEnhance  get       str cvs show
                } if
                currentpagedevice /PostRenderingEnhance  known { %if
                        currentpagedevice
                                ( PhotoGrade= ) show    %
                                /PostRenderingEnhance  get      str cvs show
                } if
        } if
} bind def

/PrintPage {
        ShowDetails
        gsave
        ss ss 10 mul tr
        0 0 m
        /count 0 def
        /lettercolor 1 def  % select white letters
```

```
        0 1 10 {PrintMatrix} for
        10 1 20 {PrintMatrix} for
        20 1 30 {PrintMatrix} for
        30 1 40 {PrintMatrix} for
        40 1 50 {PrintMatrix} for
        /lettercolor 0 def    % change to black letters
        50 1 60 {PrintMatrix} for
        60 1 70 {PrintMatrix} for
        70 1 80 {PrintMatrix} for
        80 1 90 {PrintMatrix} for
        90 1 100 {PrintMatrix} for
        grestore
        0 sg

        showpage        % print the page
} bind def



% here are various spot functions.

/spot1 {
                dup mul exch dup mul add 1 exch sub } def

% simple round
/spot2 {
                dup mul exch dup mul add 1 exch sub } def
% Inverted Round
/spot3 {
                dup mul exch dup mul add 1 sub } def
% Euclidean Composite
/spot4 {
                abs exch abs 2 copy add 1 gt
        { 1 sub dup mul exch 1 sub dup
          mul add 1 sub }
        { dup mul exch dup mul add 1 exch sub }
        ifelse } def
% Rhomboid
/spot5 {
        abs exch abs .8 mul add 2 div } def
% Line
/spot6 {
        exch pop abs 1 exch sub } def
% Diamond
/spot7 {
        abs exch abs 2 copy add .75 le
                {
                        dup mul exch dup mul add 1 exch sub
                } {
                        2 copy add 1.25 le
                                { .85 mul add 1 exch sub }
                                { 1 sub dup mul exch 1 sub dup mul add 1 sub }
                ifelse }
        ifelse
        } def
% Inverted Elliptical
/spot8 {
        dup mul .9 mul exch dup mul add 1 sub
} def



% here are the Halftone dictionaries
/HalfDict_2 4 dict def
HalfDict_2 begin
        /HalftoneType 3 def
        /Width 2 def
        /Height 2 def
        /Thresholds (\000\100\200\300) def
end
/HalfDict_3 4 dict def
HalfDict_3 begin
        /HalftoneType 3 def
        /Width 3 def
        /Height 3 def
        % \252\343\161\034\000\125\216\070\307
        % \006\010\004\001\000\003\005\002\007
% this is in linear order
%       /Thresholds (\000\034\071\125\162\216\253\307\344) def
% this is in dithered order
        /Thresholds (\252\343\161\034\000\125\216\070\307) def
        end
/HalfDict_4 4 dict def
HalfDict_4 begin
        /HalftoneType 3 def
        /Width 4 def
        /Height 4 def
% dither array
%\000\010\002\012\014\004\016\006\003\013\001\011\017\007\015\005
% dither array scaled to values from 0-255 (\000-\377 octal)
%\000\200\040\240\300\100\340\140\060\260\020\220\360\160\320\120

%       /Thresholds (\000\020\040\060\100\120\140\160\200\220\240\260\300\320\340\360) def
        /Thresholds (\000\200\040\240\300\100\340\140\060\260\020\220\360\160\320\120) def
end
/HalfDict_5 4 dict def
HalfDict_5 begin
        /HalftoneType 3 def
        /Width 5 def
        /Height 5 def
        /Thresholds (\000\012\024\037\051\063\075\110\122\134\146\161\173\205\217\232\244\256\270\303\315\327\341\354\366) def
end
/HalfDict_6 4 dict def
HalfDict_6 begin
        /HalftoneType 3 def
        /Width 6 def
        /Height 6 def
        /Thresholds (\000\007\016\025\034\044\053\062\071\100\107\116\125\134\144\153\162\171\200\207\216\225\234\244\253\262\271\300\307\316\325\334\344\353\362\371) def
end
/HalfDict_7 4 dict def
HalfDict_7 begin
        /HalftoneType 3 def
        /Width 7 def
        /Height 7 def
        /Thresholds (\000\005\012\020\025\032\037\045\052\057\064\071\077\104\111\116\124\131\136\143\150\156\163\170\175\203\210\215\222\230\235\242\247\254\262\267\274\301\307\314\321\326\333\341\346\353\360\366\373) def
end
/HalfDict_8 4 dict def
HalfDict_8 begin
        /HalftoneType 3 def
        /Width 8 def
        /Height 8 def
        /Thresholds
(\000\200\040\240\010\210\050\250\300\100\340\140\310\110\350\150\060\260\020\220\070\270\030\230\360\160\320\120\370\170\330\130\014\214\054\254\004\204\044\244\314\114\354\154\304\104\344\144\074\274\034\234\064\264\024\224\374\174\334\134\364\164\324\124)
        def
%\000\040\010\050\002\042\012\052\060\020\070\030\062\022\072\032\014\054\004\044\016\056\006\046\074\034\064\024\076\036\066\026\003\043\013\053\001\041\011\051\063\023\073\033\061\021\071\031\017\057\007\047\015\055\005\045\077\037\067\027\075\035\065\025

%       /Thresholds
```

```
(\000\004\010\014\020\024\030\034\040\044\050\054\060\064\070\074\100\104\110\114\120\124\130\134\140\144\150\154\160\164\170\174\200\204\210\214\220\224\230\234\240\244\250\254\260\264\270\274\300\304\310\314\320\324\330\334\340\344\350\354\360\364\370\374)
 def
end

%%EndProcSet

userdict begin
        systemdict /languagelevel known {
                /level2 true def
        }{
                /level2 false def
        } ifelse
end

% select font
8 /Helvetica-Bold f

%%EndSetup
%%BeginProgram

% the following are for Apple's Pro printers only
%300 SetResolution
%true true SetEnhancements
% or
%false false SetEnhancements
%600 SetResolution


% the following pairs are suggested for 300 dpi
% frequency             53      75      83      106     150
% angle                 45      0       56      45      0

%53 45  /spot1 load setscreen PrintPage
%75 0   /spot1 load setscreen PrintPage
%83 56  /spot1 load setscreen PrintPage
%106 45 /spot1 load setscreen PrintPage
%150 0  /spot1 load setscreen PrintPage

% now for the Level 2 halftone dictionaries...

%HalfDict_3 sethalftone PrintPage
%HalfDict_4 sethalftone PrintPage
%HalfDict_5 sethalftone PrintPage
%HalfDict_6 sethalftone PrintPage
%HalfDict_7 sethalftone PrintPage
%HalfDict_8 sethalftone PrintPage
%150 0 /spot5 load setscreen  PrintPage

%true true SetEnhancements
%300 SetResolution
%HalfDict_8 sethalftone PrintPage
false false  SetEnhancements
600 SetResolution
HalfDict_8 sethalftone

PrintPage


%%EndProgram
%%Trailer
```