

.BMP Bitmap Circular Lettering and a Few Related Techniques

Don Lancaster

Synergetics, Box 809, Thatcher, AZ 85552

copyright c2008 pub 7/08 as [GuruGram #92](#)

<http://www.tinaja.com>

don@tinaja.com

(928) 428-4073

In a number of our recent [GuruGrams](#), we looked at techniques to dramatically improve small typography quality on limited resolution bitmaps.

Superb quality lettering can be especially useful for [eBay](#) product photography. Where the quality of your presented images can be dramatically upgraded. High quality low resolution typography can also be used for annotation and editing of technical illustrations.

In this [GuruGram](#), we'll review a few of our earlier tools. And then seek out our new solutions to [perspective lettering](#), [lettering on a cylinder](#), [rotated text](#), [circular lettering](#), and [dealing with logos](#).

The story so far

A tutorial review of all but our latest image post processing utilities appears as [POSTPROC.PDF](#) of [GuruGram #88](#). Tools and techniques covered include...

[Airbrushing](#)

[Architect's Perspective](#)

[Background Knockout](#)

[Background Slideunder](#)

[Bitmap Typewriter](#)

[Core Array-of-string Utilities](#)

[Dodging and Burning](#)

[Exploring the .BMP Data Format](#)

[Extreme Display Legibility Secrets](#)

[False Color & Rainbow Improvements](#)

[Gonzo PostScript Utilities](#)

[Imaginative Images](#)

[Inverse Graphics Transforms](#)

[JPG Artifact Reduction](#)

[Nonlinear Graphics Transforms](#)

[Pixel Interpolation Algorithms](#)

Punchthru Elimination
Photo Secrets for eBay
Step-by-step Image Prep
Using Distiller to run PostScript
Vignetting Utilities

A .BMP Bitmap Typewriter Review

Of these, our **Bitmap Typewriter** is your key secret to ultra legible limited resolution typography.

The Bitmap Typewriter provides the highest possible resolution full pixel typography and does so with incredibly legibility down to astonishingly small point sizes. Such tasks as relettering an entire test equipment panel are now quite feasible. As per **this example** that even includes rotated text.

In normal use, the needed lettering or relettering is created to a scratch bitmap and then cut-and-paste copied to the image being reworked. **The characters are absolutely pixel locked and fully antialiased** on a pixel-by-pixel basis **without** any damaging smoothing or low pass filtering in use.

Although any combination of **PostScript** fonts can be used, best low pixel results are often obtained using **Myriad Pro** or **Helvetica** families. Any number of letter colors can be fully blended to as many as four backgrounds. Your letters and background can be any reasonable color combination.

Font sizes are defined by the pixel count width and pixel count height of an upper case letter "A". Wider and narrower letters are proportionally forced into the appropriate nearest available width. The portion of the chosen character that maps into any particular pixel is sampled 36 times. From those samples, an appropriate antialiased blend of letter to background color is created. Repeating for each of the RGB planes.

The utility uses the following major variables...

- /targetfilename** — output short .BMP filename
- /targetfilenameprefix** — long prefix for output filename
- /globalkern** — sets the global kerning, often 1.
- /kern+char** — sets the positive kerning character, often "~".
- /kern-char** — sets the negative kerning character, often "'".
- /yinc** — sets the line to line vertical pixel spacing

That get interpreted by these routines...

- setfontfamily** — picks the current PS font family in use
- setbmsize** — sets upper case "A" pixel height and width
- curcolor** — defines the current RGB color
- setbackA** — maps the background color (also B,C,D).
- setgraystring** — images the current character string.

At present, imaging **x** and **y** positions input to **setgraystring** will start at that location. Use of **0, 0** instead will continue on the present line. Characters are generated on the fly as they are needed and then saved for possible reuse.

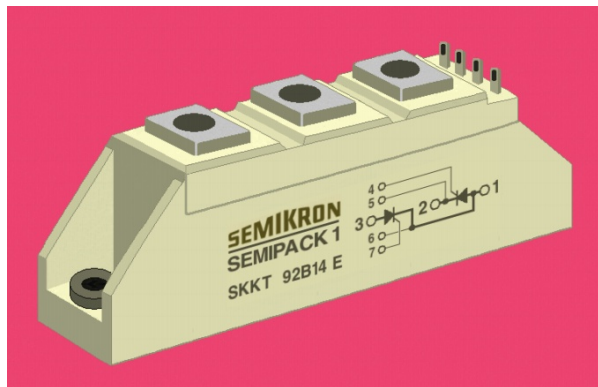
Near the end of any line, a character break will force a new combined linefeed and carriage return. With distance set by **yinc**. The background can be split into as many as four color zones. Kerning is normally done by inserting appropriate "~" or "" characters whenever a positive or negative one pixel kern is wanted.

The results of the bitmap typewriter are usually stunningly impressive. Especially when lettering drops down below "pseudolegible" sizes. Results can be further improved by going to **subpixel** techniques, but these can be rather complex and are strictly limited to specific LCD displays.

Some examples of the quality work the **Bitmap Typewriter** does can be found [here](#), [here](#), [here](#), and [here](#).

Perspective Lettering Pasteins

A recent addition that was newer than our **Post Proc Tutorial** lets you take a "flat" lettered bitmap and distort it so it can be pasted into a second in architect's perspective. Like so...



A tutorial on perspective lettering pasteins appears [here](#). With its underlying utilities [here](#).

The utility uses the following major variables...

/bmpinfilename — input "flat" bitmap input filename

/bmpinfilenameprefix — ... and prefix

/bmpoutfilename — input "pastein" bitmap output filename

/bmpinfilenameprefix — ... and prefix

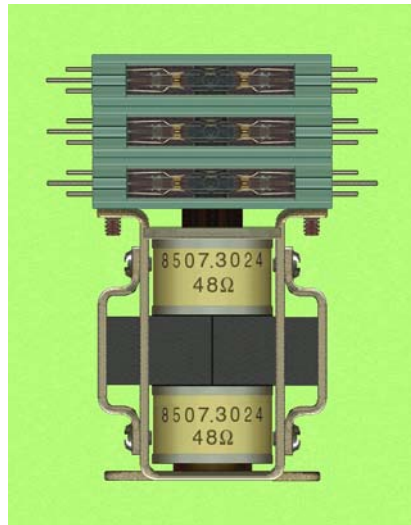
/xwidth — width of input "flat" bitmap
/yleftheight — left (and right) height of "flat" bitmap
/yrightheight — top right of "pastein" bitmap
/yrightclimb — bottom right of "pastein" bitmap

That can be activated with this command...

makperspaste — create new perspective "pastein" bitmap

Lettering on a Cylinder

Here's an example of **cylindrical lettering** that is fairly easy to fake using the **Bitmap Typewriter**...



Rather than do a full **Nonlinear Graphics Translation** on every letter, we'll use an approximation where **each full letter will have a fixed width and shape.**

To proceed, you can create the **full message** using the **equivalent width** of your **centermost character** in the message. Assume your center character is, say, **12** pixels wide by **16** high or **12x16**. Now, create a group of identical messages set to character widths of **11x16**, **10x16**, **9x16**, **8x16** ... on down to ... **1x16**.

Then cut and paste the appropriate width characters in their respective positions. If you are upgrading a photo or scam, any lower quality original lettering should give you clues to the correct character sizes and positions.

Rotated or Tilted Lettering

Sometimes it is necessary to tilt or rotate lettering. Such as this **ebay photo**...

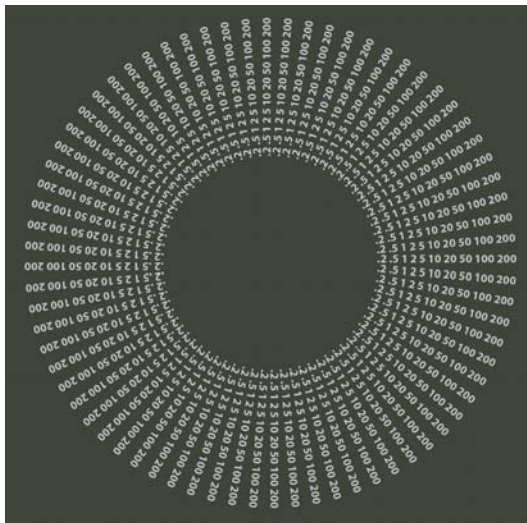


Once again, our **Bitmap Typewriter** can be used for the highest possible limited resolution .BMP typography. You create your lettering in its normal size to its normal "right side up" orientation. Then go into **Imageview32** or a similar program and use its **rotate** feature to orient your text properly.

Note that each custom rotation can give you **four** different alignments when you use rotations of **0, 90, 270, and 180** over and above your "by degree" rotation.

An Alternate Approach

Getting all the angles just right on a rotated dial may end up tedious. An interesting alternate method is to **create a catalog ahead of time for any and all possible rotations of all needed characters**. Such as this example...



A choice of 72 increments would give you a 5.0 degree rotation. Which translates to a 2.5 degree worst case error and a typical error of slightly over one degree. The color parameters in **Paint** or whatever are typically 256 times larger than those in PostScript and are easily and directly scaled.

Code generation in **Gonzo** is surprisingly short and fast...

```
/letcolor {202 255 div 201 255 div 206 255 div  
  setrgbcolor} store  
/backcolor {64 255 div 70 255 div 60 255 div  
  setrgbcolor} store  
/rotttext (.2 .5 1 2 5 10 20 50 100 200 ) store  
/leadspace 110 store  
/font1 /MyriadPro-Bold 12 gonzofont font1  
0 0 mt 1000 pu 1000 pr 1000 pd closepath backcolor fill  
gsave 310 300 translate letcolor 72 {leadspace 0 rotttext cl  
  5 rotate} repeat grestore
```

To use, you simply cut and paste the characters you need into your bitmap and ignore the rest. It is best to work oversize and reduce to pick up anti aliasing.

Circular Lettering

Here's an example of a circularly lettered bitmap...



While this can be done entirely within the **Bitmap Typewriter**, other "not quite as good" approaches may end up a lot faster and much less tedious. Often without degrading your resolution limited quality all that bad. And still providing more or less acceptable anti-aliasing between your lettering and background colors.

As usual around here, it is **PostScript** to the rescue.

If you really wanted to do the best possible small circular text, you would create your circular text message entirely in the **bitmap typewriter**. Then send it on to **Imageview32** and do repeated saves. Tilted once at the correct angle for each letter in the rotation. In general, dividing the subtended angle by half the number of characters will give a useful and much saner approximation. This is similar (but obviously more gruesome) to the rotated lettering of our previous section.

There are some circular **arc justify** text layout routines in our **Gonzo PostScript Utilities**.

An **Arc Justify** routine sets text along a **circular** path. This is useful for labels, for logos, badges and for large names on the back of jackets or T-shirts. The fancy arc justify routine even supports local and global kerning.

Here is how you activate a Gonzo arc justify...

```
-xpos- -ypos- radius (your message) karcjustify
```

A **positive** radius value causes the message to appear **clockwise across the top**. A **negative** radius value causes the message to appear **counterclockwise across the bottom**. In either case, the radius will be to the **baseline** of the text. Thus, **your upper and lower radius will normally differ by the average character height**.

Upper and lower messages are normally **centered along a vertical axis**. Any repositioning can be done by adding leading or trailing spaces. Since each and every character is separately positioned, the earlier and fancier **Gonzo** Justification features are **not** available for an arc justify.

A predefined **arckern** variable will add positive kerning to each and every letter and space. A custom **customkern** variable will create a (usually negative) space every time a **customkernchar** is called. Typically, arckern might be **+1**, your customkern **-1**, and customkernchar a (~) string. To reduce the space between two printing characters, you **place a "~" between them**.

Several detailed examples of arc justify projects appear **here**.

Here is one possible way to approximate a small sized circular bitmap text: Use the Gonzo **karcjustify commands** to create a **PostScript** program to produce the circular text **four to eight times larger than final size**. Distill the file to .PDF using Acrobat. Then use "save as" to save the file in a **.TIFF** format.

Sadly, **Imageview32** will not accept a .TIFF file, so load the .TIFF file into Paint and resave it as a .BMP 24 bit color file. Then, you bring your .BMP file back up in **Imageview32** and do a 4:1 to 8:1 reduction as needed. This should antialias your lettering and background colors properly and still retain fairly decent quality. Resave the reduced .BMP file and cut and paste as usual to your main image.

Once again, this will not be nearly as good as the Bitmap Typewriter for those extremely small font sizes at limited resolutions. But should end up acceptable for most uses. Quality should improve significantly for final fonts above ten pixels in height.

Dealing With Logos

Logos can often end up blurry and indistinct in a digital photo or a scan. Most especially at smaller sizes. What you'd really like may be more like this...



Logos get tricky in a hurry because they are **intellectual property**. Property whose rights often are aggressively enforced. There is a legal entity known as the **First Sale Doctrine** that says **you are allowed to use a logo specifically to resell a product that you already own**. For instance, you are allowed to use the Ford pony to sell your '65 Mustang. But are **not** allowed to use the image without getting permission for your historic autos calendar.

An interesting source for most modern logos is found [here](#). Which can sometimes give you clean and usable logo art. More often than not (as in our above image), an electronic logo may have changed a time or two over the years. Sometimes you can go to classic websites to find suitable originals.

Once you have found your logo art, you may want to change its foreground or background color. Here is a rather sneaky way to accomplish this: Bring the logo up in **Paint** and resave it as a **monochrome bitmap**. Then bring the "all gray" logo back up in **Imageview32**. **Resave as 24 bit color**. Then use the **color balance** tool to adjust your colors as needed.

You may also want to use the **Brightness/Contrast/Gamma** tool to adjust any foreground and background gray levels should these need further mods.

For More Help

Extensive samples of these techniques appear on our **eBay** sale pages.

Similar tutorials and additional support materials are found on our **PostScript** and our **GuruGram** library pages. As always, **Custom Consulting** is available on a cash and carry or contract basis. As are seminars and workshops. For details, you can email **don@tinaja.com**. Or call **(928) 428-4073**.