Don Lancaster's

ASK THE GURU

February, 1990

Secrets of Vinyl Lettering
5,000:1 PostScript Speedup
More on the Bezier Curves
A Third-Generation Printer
IBM/LaserWriter Interfacing

Meanwhile, consternation had broken out among all of those disgruntled mutineers encamped in the old Eskimo village. Er, whoops. Wrong column. Let's try again…

Plans are underfoot for a major new PostScript BBS board to open up on *GEnie*. The intent is to have at least a thousand heavy-duty free downloads, along with most of the routines found in my *Ask the Guru*, *LaserWriter Corner* and my *Hardware Hacker* columns. Its called PSRT.

Patches are at long last available for *AppleWriter* which let you have much longer resident textfiles on your IIgs. These patches are handy for downloading PostScript fonts. So far, the patches do seem to be compatible with *Don Thompson's WPL Toolkit* and many of my own patches. The package is shareware priced at $29 and is available from *Chester H. Page*.

Sadly, *Call A.P.P.L.E.* has ceased publication after all these years. My recommended substitute would be Tom Weishaar's great *A2 Central*. In yet another ominous development, *Sierra On-Line* announced they have discontinued all future IIgs product development, citing the intolerably frustrating development tools and the lack of sorely needed and long overdue IIgs hardware upgrades. Sierra, of course, was one of those Apple pioneers that long led the field in all of the high resolution graphic adventures. Sierra intends to continue to develop for all other major personal computers.

Over in the PostScript clone wars, *Freedom of the Press* has issued an upgrade that no longer breaks the tail off *Meowwrrr*, my PostScript Puss de Resistance. But *GoScript* still seems totally incapable of printing the PostScript fractal ferns that we looked at several columns ago. So much for PostScript compatibility.

That *Adobe Type Manager* is going

1. The printer shall speak genuine *Adobe* PostScript and use fully hinted downloaded Type I outline fonts.
2. The resolution is to be 415 DPI with an alternate 415/830 mode that allows 106 line halftone screens of 35 gray levels. Full histogram equalization shall be available.
3. The repeat print speed will be at least 12 pages per minute. Page makeready times shall be a minimum of 5 times faster than the *LaserWriter NTX.* A combination of new algorithms, software, and hardware is to be used for this speedup, including RISC-based or direct PostScript hardware engine chips.
4. An absolutely straight paper path will be provided as an option, allowing no-nonsense hand feeding of thicker materials. At least two 400 sheet minimum paper trays shall be provided, capable of automatically feeding heavy cover stock.
5. Envelope print quality is to be substantially better than on previous printers. The fusion rollers are to be separately accessible for use with *Kroy Kolor* and laminating materials.
6. As extra price options, an 11 x 17 capability will be offered, as shall the ability to print toner directly onto a 1/16th inch thick printed circuit board material.
7. Toner costs shall not exceed 0.5 cents per printed side, and shall be third-party reducible to under 0.2 cents per page. Toner cartridges must last a minimum of 10,000 copies and shall be conveniently end-user refillable a minimum of twelve times. The photosensitive toner drum shall have a *Mohs* scale surface hardness of not less than 9.0. Any and all parts of the toner and charging system must be low in cost, easy to get, and simple to replace.
8. The printer shall, in a single feeding, be able to duplex print on both sides of a sheet of paper. All rollers and all paper paths shall be capable of handling any previously printed material on a long-term basis. The reliability of the paper path and feeding mechanism shall be good enough to allow unattended overnight printing of book-on-demand published titles.
9. A minimum of 150 high quality, fully hinted, and outline fonts will be standard, with other fonts easily accessible as user-programmable EPROM cartridges and CD-ROM disks.
10. All descriptive font paths shall be available on all levels, including alterable plaintext representations of downloaded Type I fonts. The font paths must be easily interceptable and modifiable for such things as perspective and similar lettering or other nonlinear transformations.
11. There shall be no secrecy. *Everything* on *all* levels must be fully documented, fully unlocked, and fully available in plaintext form.
12. An optional hard disk shall have a fully documented, reliable, and user-friendly operating system that does not blow up during routine use. All disk files and formats shall be totally SCSI standard and fully documented. A mix of up to eight disks and CD ROM devices shall be allowed.
13. Provisions shall be made for direct hard disk access by the host, for the return of cached font characters to the host, and for use of the SCSI port as a direct and high speed data channel. The font cache is to be totally separable from other disk files and directories.
14. Minimum memory expansion will be 16 megabytes, and provisions are to be made for dual bitmaps that allow one page to compose while the other is printing.
15. Service manuals exceeding current *Hewlett-Packard* quality standards shall be readily available to the end user. These shall include full schematic diagrams, complete memory maps, and fully disassembled ROM listings. All source code shall be optionally available.
16. A minimum of 500,000 copies are required before minor repairs or parts replacements are needed. The machine is to last forever, simply by bolting on new parts.
17. A fully programmable video output of the final page bitmap shall be available. Any portion of any and all bitmaps shall be returnable to the host for recording, in both "open" and run-length encoded formats.
18. A compiling or pseudo-compiling provision is to be available that can dramatically speed up later reuse of any page image. A user system monitor is to be provided that gives complete and total access to all code levels.
19. Large carrying handles must be provided. Mode switches shall be rugged and easily changed. Toner density is to be adjustable from outside the machine. Unusual page formats as large as 8-1/2 x 24 shall be supported.
20. The end-user street price is to be under $3499.

Fig. 1 – An acceptable third generation PostScript printer.

gangbusters as the number one Mac utility and is now pushing the half million mark for unit sales. One discounting source is *MacWarehouse* at $53. The good news is that this gem can stunningly improve screen graphics and higher resolution dot matrix and most ink jet printouts. The bad news is that it only works on Type I downloaded fonts, and that it actually slightly degrades 10- and 12-point type on older *ImageWriter* printers. An IBM version of the ATM should be available soon.

Later versions of ATM fixed this *Imagewriter* bug.

As we have seen in our previous columns, *Hewlett-Packard* is one outstanding source of repair and service manuals for most of those Apple LaserWriter products. They have combined two of their older CX manuals into their new part #02606-90920. Their SX manual remains part #33440-90904.

A reminder about the *Midnight Engineering* magazine for all of you developing and marketing your own low end hardware or software. By way of a special arrangement, free sample copies are available to *Computer Shopper* readers.

As always, this is your column and you can get technical help and off-the-wall networking by calling (602) 428-4073. I do have a new and free laser printing *insider resources* brochure just for you and I am now shipping my brand new *LaserWriter Secrets* book and disk combo.

We seem to be into advanced topics this month. So, I thought we'd have us five really advanced contests to go with them. There'll be all the usual *Incredible Secret Money Machine* prizes for the dozen best entries, along with an all expense paid (FOB Thatcher, AZ) *tinaja quest* for two to the best of all.

For the first two contests, either (1) show me a way to use the SCSI port on the LaserWriter NTX as a ridiculously faster *AppleTalk* substitute, or else (2) show me a way to compatibly timeshare a single hard disk between an NTX and a host on a real time basis. Be sure to send your entries directly to me, and not to the *Computer Shopper* editorial. On to the goodies...

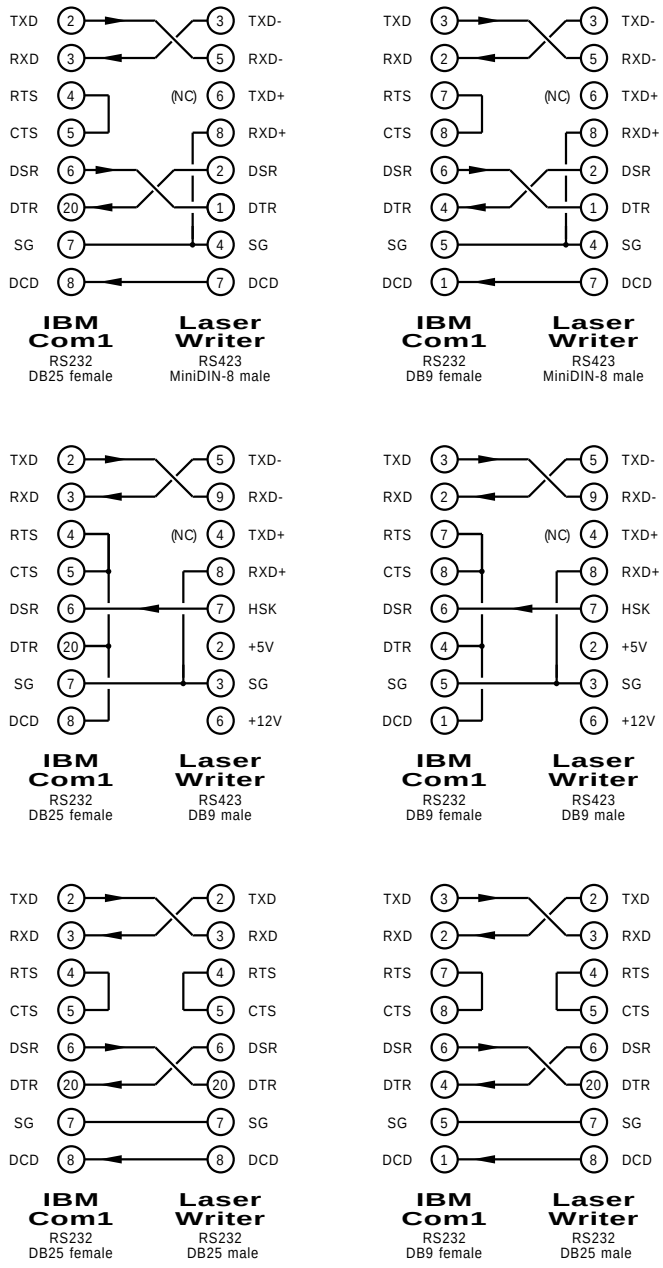


Fig. 2 – Six possible IBM to LaserWriter cables.

What do You Really Want In A Third-Generation Printer?

I never thought you would ask. Dozens of manufacturers seem to be on the verge of releasing a bunch of brand new PostScript laser printers. Sadly, most of them seem about to offer some monumentally stupid features for precisely the wrong reasons. Arrghhh.

To try and head things off at the pass, figure one gives you the bare minimum specs for an acceptable third-generation PostScript laser printer. This is based upon your

helpline calls and requests, on my own needs, and on lengthly talks with the other PostScript power users and developers. Every feature on this list is both technically and economically feasible here and now.

As I now see it, there are four main areas which severely cripple today's PostScript printers....

The toner atrocity – There's no sane reason why toner should cost more than ink, nor any reason whatsoever to exceed a toner cost of 0.2 cents per printed page. Instead, we now have these purposely undersized and overpriced hard-to-refill *Kamikaze* cartridges filled with excessively abrasive toner and an intentionally softened drum.

The font path paranoia – By depriving power users of a simple and direct access to the actual font paths used, many PostScript routines are slowed down by several orders of magnitude. Fancier effects such as perspective or star wars lettering are made unduly difficult.

The speed debacle – Many of the perceived speed problems in PostScript are caused by external hassles. Things such as the lack of display PostScript on your host. Or the sad absence of any direct SCSI replacement for unbearably slow *AppleTalk* or other serial comm channels, the lack of any general compiling routines, excessive host software overhead, and the failure to properly interact between the PostScript printer and the host.

The bitmap obstinticity – Although a device independence might be a laudable goal for PostScript, in no way should it get shoved down everyone's throat. By giving end users direct and simple access to any and all bitmaps, such things as step-and- repeat business cards can be ridiculously sped up. A video output of the bitmap as it is being generated would end up extremely handy for your debugging, and for finding any slower portions of your code. The ability to compact and save whole page bitmaps to CD-ROM would dramatically improve any Book-on-Demand publishing. And finally, some programmable line-by-line bitmap output would open up PostScript to previously unthunk of great opportunities, allowing a printer to interface to virtually any other output device, however specialized or limited its niche market.

Improving PostScript is kind of an ongoing dialog around here, so be sure to let me know what else you want to see in the way of any printer upgrades. We also offer professional services in this area.

Usually, all we need is an *accurate* sketch of what you want, sample input and output, and a written description of your problem. The turnaround time is typically a few weeks.

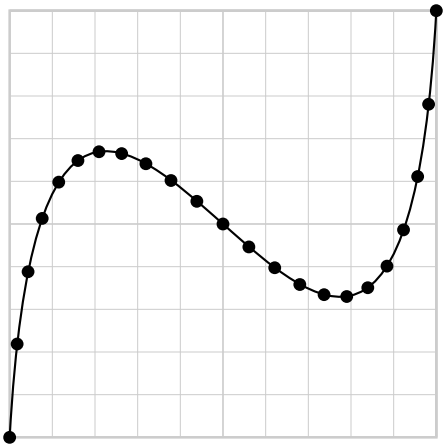
How About Another Round on IBM -to- PostScript Printer Interface Problems?

Here we go again. That helpline keeps ringing off the hook with IBM to PostScript printer interface problems. By far the overwhelming majority of these are caused by user inattention to detail...

"4073. Your PostScript helpline".

"Your #$@$#& Free Font will not

```
% Copyright c 1990 by Don Lancaster and Synergetics, 3860 West First Street, Box 809,
% Thatcher AZ 85552. (602) 428-4073. All commercial rights reserved. Personal use permitted so
% long as this header remains intact. Show and Tell disks for Apple, Mac, or IBM cost $39.50.
```



```
% Returns the approximate length of a selected Bezier curve to the host. Value /numpoints
% determines the accuracy. 100 is usually good enough. Result /blength holds the final length,
% usually to 0.1% accuracy For best results, use with the rubbergrid from my PostScript utilities.

/pcurveto {6 copy /y3 exch def /x3 exch def /y2 exch def /x2 exch def /y1 exch def /x1 exch def
currentpoint /y0 exch def /x0 exch def curveto} def

/xtt {x3 x2 3 mul sub x1 3 mul add x0 sub tt 3 exp mul x2 3 mul x1 6 mul neg add x0 3 mul add tt
dup mul mul add x1 3 mul x0 3 mul neg add tt mul add x0 add} def  % x coefficients as f(t)

/ytt {y3 y2 3 mul sub y1 3 mul add y0 sub tt 3 exp mul y2 3 mul y1 6 mul neg add y0 3 mul add tt
dup mul mul add y1 3 mul y0 3 mul neg add tt mul add y0 add} def % y coefficients as f(t)

/plotdots {0 1 numpoints 1 sub div 1.00001 {/tt exch def newpath xtt ytt moveto 0 0 0.2 0 360 arc fill}
for }def

/bezierlength {/oldx x0 def /oldy y0 def /blength 0 def 0 1 numpoints 1 sub div 1.0001 {/tt exch def
xtt ytt /newy exch def /newx exch def newx oldx sub dup mul newy oldy sub dup mul add sqrt
blength add /blength exch def /oldy newy def /oldx newx def} for }def % approximate curve with
line segments

% //// demo - remove before use. ////

200 200 moveto 10 dup scale 1 setlinewidth 0 0 moveto
1 18 9 -8 10 10 pcurveto stroke  % substitute desired curve here

/numpoints 23 def plotdots bezierlength blength == flush showpage quit
```

Fig. 3 – Approximate length of a PostScript Bezier curve.

work on my PC clone".

"What PostScript error message do you get back?"

"The lights blink for a while."

"What PostScript error message do you get back?"

"Huh?"

Back to square one. Yes, an IBM or a PC clone will easily support any PostScript laser printer, provided you do follow a few common sense rules. Firstoff, do **not**, under any circumstances **ever** use a parallel interface between your PC clone and your PostScript laser printer. To do so deprives you of crucial return error messages, severely limits what you can accomplish with your new PostScript language, and makes all of your printer drivers unbearably restrictive and klutzy.

Any software that only supports PostScript parallel interface is, by definition, defective and should be immediately flushed.

Second, do not ever, under any circumstances, try to print by copying out your COM-1 port. Instead, always use a full duplex comm channel, such as *PC Talk* shareware, that lets you view your return error messages and optionally record any PostScript output returned to your host. While the typical parameters are 9600 baud, 8 data bits, 1 stop bit, and no parity, you can later make your interface go as fast as 57,600 baud. Which is much faster than most parallel interfaces.

Third, be certain you do set up an XON / XOFF handshaking environment. If your handshaking is set wrong, errors will often be picked up after a page or two is printed. While you can, in theory, change your laser printer to do hardwire handshaking, there is absolutely no reason to ever do so. Always use XON / XOFF handshakes.

Fourth, after you have set up a reliable two-way serial communications with full error reporting, you install a printing and stack dumping error trapper. You can get one for $10 directly from *Adobe Systems*, pick one up off a BBS, find it in the green book, or else use the copy in my *PostScript Beginner Stuff*.

A shareware version of Adobe's EHANDLER.PS is also available as file #196 on *GEnie* PSRT.

Cables. It is normally considered proper form to have one between your computer and a printer. Figure two shows you the six most popular IBM to LaserWriter cable lashups.

The six combinations result since you can have a DB-25 or DB-9 connector on the host and a DB-25 using RS-232-C or (depending upon your LaserWriter model) a DB-9 or a Mini DIN-8 connector using the RS-423 interface serial standard.

These cables are available ready-to-go through the *Redmond Cable*, folks among other sources.

Some general guidelines here, in case you are trying out something special. When using RS-232 at both ends, you want to use what the IBM folks call a *null modem*, or what

Proc cacheing can give you a 12:1 up to a 3,000,000:1 speedup of your PostScript routines for any computation-intensive image that is to be reused at least once. To cache a proc, you define that proc as a character in a font, and then let the existing font machinery do the cacheing for you.

There are four possibilities…

(1) Font defined inside the job (speedup goes away with the job).
(2) Font persistently downloaded (fast so long as power applied).
(3) Font saved to NTX hard disk (stays fast until your disk blows up).
(4) NTX hard disk bitmap returned to host (can be permanently fast).

And here are some use rules…

1. Open up your font cache as wide as possible by using a *mark 1250 n setcacheparams*. The *n* value can be 48K (4 square inches) on a 3 meg NTX up to 188K (16 square inches) on a 12 meg NTX. For larger images, you can use several characters side by side.
2. A font definition is not allowed to contain an *image* operator; or a *setgray*, *setrgbcolor*, *sethsbcolor*, *settransfer*, or *setscreen* commands. Because a font dictionary ends up read only, a "floating" dictionary has to be used for any internal *save*, *restore*, or *def* commands.
3. One separate character needs to be created for each shade of gray in use. A white fill requires a separate mask as shown in figure five.
4. Use a font matrix of [1 0 0 1 0 0] and a point size of 1.0.
5. Font caches can be stored in run-length encoded format or as full bitmaps. The run-length encoding often uses much less memory with only a minor speed penalty. The *m* in *mark m n setcacheparams* decides whether full bitmaps or run length encoding will get used. To guarantee a bitmap, make *m* equal *n*. See the white book for details.
6. To verify that cacheing is taking place, run your routine twice, and measure the speed each time. Trip #2 should be ridiculously faster if the cache machinery really worked.
7. Should you get no image, this usually means you have inadvertently substituted *Courier* at 1/1000th normal size and that something is wrong with your font definitions.
8. Fonts used to define your proc cache are not in themselves cached.
9. The needed cache size can be estimated by multiplying the height in pixels by the width in pixels and adding around a thousand bytes. With enough side-by-side characters, even an entire page can be cached either in a bitmap or run length encoded form.

Fig. 4 – PostScript proc cacheing guidelines.

Apple will call a *printer cable* . Do note that your data channels and your handshaking are *crossed* in this type of cable.

IBM sometimes uses the RS-232-C auxiliary handshake on pins 4 and 5. But these are rarely used elsewhere and are best locally self-jumpered.

When using RS-232 at the host and RS-423 at the printer, note that the TXD- from the printer becomes RXD at your host, and that there is no connection to TXD+.

Note further that the transmitted data out will go to RXD- and that RXD+ must get *grounded* . Failure to ground the differential RXD+ input is far and away the most common problem on any RS-232 to RS-423 interface. Watch this detail.

Instead of any custom cables, it is often better to use a stock cable and add a terminating "pin rearranger," built up from a few short wires and a pair of *Radio Shack* male and female connectors. A pair of loops of #14 house wiring can be used to solder your two connector cases to each other.

Please note that you definitely must not ever use a "straight" or a "crossed" DB-9 to DB-9 cable, as your connections will be wrong.

Also note that some PC clone comm software may require a cold boot to change parameters. During your initial debug, always do a cold reboot and make sure your printer is a solid green before any retry.

Additional details on laser printer interface and debug appear in my *LaserWriter Secrets* , and the Apple White Book, otherwise known as the *LaserWriter Technical Reference Manual* . I do stock these.

Any More Info On Bezier Curves?

Lots of stuff. We do have a *tinaja quest* winner from our avuncular sleezoid contest. Michael Chaplin not only picked up the football on this one, but he ran out of the stadium and hasn't been seen since.

Mike's mind-blowing routines do include entire sleezoid alphabets and incredible metamorphically tweened sleezoid animation sequences. Write him for details.

It appears *Hewlett-Packard* has picked up on avuncular sleezoids in a very big way. They have upped their sleezoids to three dimensions and now call them *NURBS* instead. Leave it to HP to abandon the industry standard notation. A good introduction on this appeared in the October 1989 issue of the *Computer Graphics Review* .

I was sadly disappointed with the results from our find-the-length-of-a-Bezier-curve contest. A few math freaks claimed that a Bezier curve does not have a length, since a discontinuity is often possible. And several of those other entries pretty

```
%  Copyright c 1990 by Don Lancaster and Synergetics, 3860 West First Street, Box 809,
%  Thatcher AZ 85552. (602) 428-4073. All commercial rights are reserved. Personal use is
%  permitted so long as this header remains intact. Show and Tell disks cost $39.50.
```


```
%  This is my good old poison ivy can label, modified so it will be legal as a pair of font
%  characters.  Character (a) will be the white fill mask, while (b) will be the actual label…

/drawlabel {gsave stuffdict begin stuffdict exch /maskonly exch put stuffdict /overscan 1.0 put
/pixellineremap {0 1 overscan mul pxpwidth 300 mul 72 div cvi {stuffdict exch /sline# exch put
gsave gsave mappingproc newpath sline# 72 mul 300 div 0 moveto 0 pxpheight rlineto 0 0
rlineto 0 pxpheight neg rlineto closepath clip newpath pixelproc grestore grestore} for} stuffdict 3
1 roll put /mappingproc {sline# .24 mul dup neg exch pxpwidth 2 div sub canwide div 114.6 mul
dup sin canwide 2 div mul exch 3 1 roll add exch cos 1 exch sub tiltangle sin canwide 2 div mul
mul translate} stuffdict 3 1 roll put 94 18 translate stuffdict /tiltangle 14 put stuffdict /canhi 50
tiltangle sin div put stuffdict /canwide 200 put /NewCenturySchlbk-Bold findfont [30 0 0 33 0 0]
makefont setfont stuffdict /labelypos 8 put stuffdict /pxpwidth 238 put stuffdict /pxpheight 35 put
0 canwide 2 div tiltangle sin mul neg labelypos add translate /pixelproc { 0 0 moveto 0 pxpheight
rlineto pxpwidth 0 rlineto 0 pxpheight neg rlineto closepath maskonly {fill} {4 setlinewidth stroke
12 6 moveto 1 0 (FREE  FONT) ashow} ifelse} stuffdict 3 1 roll put pixellineremap end grestore}
def

/stuffdict 50 dict def  % a work place since internal def's are a no-no

% This is a more or less standard custom font builder, except for the matrix size…

/newfont 200 dict def newfont begin /FontType 3 def /FontMatrix [1 0 0 1 0 0] def /fontBBox
[0 0 188 54] def

/Encoding 256 array def 0 1 255 {Encoding exch /.notdef put} for Encoding dup (a) 0 get
/canwhite put dup (b) 0 get /canfont put

/Metrics 15 dict def Metrics begin /canwhite 0 def /canfont 0 def end

/BBox 8 dict def BBox begin /.notdef [0 0 0 0] def /canwhite [0 0 188 54] def /canfont
[0 0 188 54] def end

/CharacterDefs 10 dict def CharacterDefs begin /.notdef { } def /canwhite {true drawlabel} def
/canfont {false drawlabel} def end

/BuildChar {0 begin /char exch def /fontdict exch def /charname fontdict /Encoding get char get
def fontdict begin Metrics charname get 0 BBox charname get aload pop setcachedevice
CharacterDefs charname get exec end end} def

/BuildChar load 0 3 dict put /UniqueID 1111 def end /CanFont newfont definefont pop

% //// demo - remove before use. ////

/CanFont findfont [1 0 0 1 0 0] makefont setfont
/fullbitmap true def  % false for run length encoded

fullbitmap {mark 50000 23050 setcacheparams}{mark 50 23050 setcacheparams} ifelse

200 200 moveto 1 setgray (a) show 0 setgray (b) show showpage          % the slow one
200 500 moveto 1 setgray (a) show 0 setgray (b) show showpage quit     % absurdly faster
```

Fig. 5 – A 5000:1 PostScript speedup example.

much cancelled each other out.

I am still convinced that a simple closed expression for the length of a Bezier curve exists and that this is a trivially easy one to derive, given a creative enough transformation. Only I just can't seem to find it. So, for contest number three of this month, show me a closed formula for the length of a Bezier or other cubic spline curve.

Meanwhile, figure three shows you how to accurately approximate the length of a Bezier curve. You do this by using as many straight line segments as you think you need and adding them up. Sort of a *Simpson's Rule versus Pythagoras and the Teenage Mutant Ninja Night Nurses* type of thing. Since the points are closer together along all of the "more bent" portions of the curve, the results are surprisingly accurate. One hundred rapidly calculated points gets you well under an 0.1 percent error for most curves. This is more than good enough when you're plotting pixels.

Tell me all About Vinyl Lettering.

Toner sticks quite nicely to vinyl, which strongly suggests using your LaserWriter and an X-Acto knife to replace a $15,000 lettering machine.

Several tips here. Your vinyl must be carrier supported, and I would recommend trimming at least one-quarter of an inch or more off each edge before hand feeding. Your carrier itself should be precisely precut to 8-1/2 by 11 inches.

Dozens of sources for the vinyl lettering materials often advertise in *SignCraft* magazine; one typical supplier would be *Vinyl Express*. The usual pricing is around 50 cents to a dollar per square foot.

There are dozens of nice colors, including exotics which simulate etched or frosted glass, chromes, reflectives, translucents, and lots more. Some are self-adhesive, while others use application tapes.

Other options include the *Form-X Films* and the heavier *static cling* materials, such as those offered by *Joseph Struhl*. These are intended for long-term sticking to glass and to similar glossy surfaces simply by wetting the surface and squeegeeing them on.

Thicknesses vary from two to ten mils. I did seem to pick up some SX transfer assembly problems after I jammed a ten mil self-cling piece, so I would recommend either using the thinner materials, or else first creating a transfer pattern.

PostScript easily does backwards lettering by doing a *-1 1 scale*. This lets you put the letters on the inside of the glass, where they should be more vandal resistant. More details on this in *LaserWriter Secrets*.

I keep hearing persistent rumors of a magic new material which just might revolutionize both sign-making and PostScript laser pattern making in general. But every time I try to chase this down it seems to vanish. Poof even.

The story goes like this: You take a sheet of a carrier-supported and light-sensitive previnyl which is soft and only partially chemically cross-linked. You run this through your laser printer, placing toner only on the outlines needed.

Then you expose the pre-vinyl to the sun or strong light, which in turn hardens and completes the cross-linking into a vinyl polymer everywhere the toner was *absent*. Then you wipe on some glop which dissolves both your toner and the previnyl under the toner. Presto. Instant cut letters, ready for use.

A free *Incredible Secret Money Machine* if you can show me where to get this magic material.

So, What's the Cache?

I was going to show you how to double or triple your speed for this month's PostScript utility, but it appears our demo example ended up with a 5,000:1 speedup instead.

Adjectives such as "adequate" immediately come to mind.

I call my secret new technique *proc cacheing*, and it could work on any PostScript procedure that you want to reprint at least once at the same size at some future time. It is particularly good when used with any computation-intensive routines which involve irregular clipping intervals, custom logos, fancy font footwork, any extensive non-linear transformations, your curve-traced signatures, repeated randomizing, or extensive fractals.

Proc cacheing is more or less free, since all it requires is some re-thinking and a modest change or two in your programming style. While it works best on the NTX with a hard disk, you can apply it to any PostScript laser printer. You can even use your proc cacheing to capture and save your bitmap of an entire page.

PostScript has a built-in *font cache* which converts the outline descriptions of all fonts into bitmaps for later reuse. All you have to do to proc cache is *define your PostScript procedure as one or more characters in a custom font!* Any repeated use of your proc is then done as a bitmap, rather than by recalculating everything all over again. The typical run time speedups will range from 12:1 to 3,000,000:1.

Figure four summarizes the key proc cacheing rules, while a detailed example appears in figure five.

There are four major ways to use proc cacheing…

(A) You can define your font only inside your job, in which case the proc cacheing goes away with the job. This gets handy for all 12-up business cards where you do not want to permanently tie up any memory, or are using an older machine with limited memory.

(B) You can persistently download your font, so that the proc cacheing remains as long as power is applied. This does reserve some cache memory, and using lots of new fonts may in fact overwrite your proc cache.

(C) You can let the NTX hard disk automatically grab your proc cache. It will keep the bitmap for you until the next time your hard disk blows up. Your font should also be saved to disk for proper operation.

(D) You can read all your bitmaps saved to the NTX hard disk and

return them back to your host for recording. They appear on your directory as FC type textfiles. Which will give you a permanently fast version of your routine. This final route is especially attractive for Book-on-Demand publishing.

At any rate, your standard font cacheing is controlled by a *mark* 1250 12500 *setcacheparams*. Those 12500 bytes equal 12500 x 8 = 100,000 pixels, and sets a maximum possible size for any bitmap to be cached.

There are often two font cacheing mechanisms used. One generates a real bitmap, while the other uses a slower *run length encoding*, which produces a fairly fast image while using much less memory. The 1250 in the above example decides when to switch from real bitmaps into a run length encoding. The 1250 bytes will equal 1250 x 8 = 10,000 pixels.

Now, the total number of pixels per character bitmap will equal the height in pixels times the width in pixels. Assuming a square bounding box, the square root of the bitmap size gives us the height of the character. In this stock example, we could use real bitmaps up to 100 pixels high (or 24 points at 300 DPI). The run length encoding is used up to 316 pixels high (or 76 points at 300 DPI). Any of the characters larger than this will not go into the font cache, but will get recalculated each and every time.

Obviously, we need larger images than these to be really useful. The allowable maximum size for *n* in *mark m n setcacheparams* can change with the available printer memory. To find your limit, just keep on increasing *n* until you get a *limitcheck* error. On the 3 Megabyte NTX, you're allowed an *n* value around 48,000, which will translate into a bounding box of four square inches.

This quadruples to sixteen square inches on an NTX having a full 12 Megabytes installed.

These font cache size limits aren't nearly as bad as they sound. Note that you only have to proc cache your tightly trimmed slow stuff, which is often much smaller than your entire image. You are also free to use as many characters as you need to build up your full image.

Do note that as few as a mere *six* characters can proc cache an entire page bitmap when using a "full" NTX. Finally, note that you might use a full NTX to do all your proc cacheing, return the results to your host, and then rerun the final results on any old PostScript printer of any memory size.

There are several subtle gotchas, though. The *only* goal of your proc must be to make marks on the page. To qualify for font cacheing, your proc descriptions must not have any *setgray*, *sethsbcolor*, *setrgbcolor*, *settransfer*, *setscreen*, or any *image* calls present in them.

Since a font dictionary gets made read-only during its processing, you will have to provide an internal "floating dictionary" or two if you want to use a *save*, *restore*, or a *def* command. Otherwise, you will get one or more of the *invalidaccess* error messages.

So how do you handle grays or white fills? Simply by using a few additional characters. You use one character as a mask for each black, gray, or white level as needed. Our example of figure five uses the (a) character to erase the background to white and then overwrites the (b) character on top of it to create the actual label. Obviously, you do have to put the characters down in the proper sequence and in the correct colors to get the desired final result.

Two other differences between the proc cacheing and a "real" font: You use a font matrix of [1 0 0 1 0 0] and a point size of 1.0.

Because of pixel line remapping and all of the intensive non-linear transformations, the wraparound isometric label would normally print in 70 seconds on an NTX. As a run length encoded cache, it prints in 83 milliseconds. As a cached bitmap, it prints in 14 milliseconds. That's a speedup of 5,000:1. Which is not half bad for an amateur. Note that 6:1 speed penalty of your run length encoding over using the full bitmap. No big deal.

Your own speedups will depend on how calculation-intensive your proc is, and can range from a low of 12:1 for a simple 12-up business card, on up to several million to one or more for a fractal landscape.

Very handily, any other fonts used internally to create your proc font are *not* in themselves cached. Which can eliminate a major source of NTX hard disk blowups on such things as perspective letters. It can also slow down your first proc-as-a-font runtime, so watch this detail.

Some black magic does appear involved in tricking the NTX hard disk to actually cache your proc, even after it is obviously in the RAM cache in printer memory. Try (A) Persistently downloading your new proc, (B) Running your proc, along with a dozen standard font characters in an unused and oddball size; and then (C) Issuing a *control-c* to force a hard disk cache update, saving your proc.

I suspect this has something to do with Type 3 versus Type 1 Fonts. Let me know if you can come up with something better here, or at least an explanation of what is going on.

Once again, you can use several characters side by side for the larger images, and you can even cache an entire page as a nearly instant printing bitmap.

The newly announced PostScript level II does include several easy to use proc cacheing features. These are included in their forms and user path operators. More on this as the code becomes available.

Most of the PostScript routines you see here are now available on our new *GEnie* PSRT roundtable, in ready-to-run formats. You can call (800) 638-9636 for voice connect info. You can also get my *Ask the Guru* column preprints here as well.

As our final two contests for this month, either (4) show me a disgustingly sneaky new use for my proc cacheing, or else for you supergonzo PostScript folks out there, (5) find me a buyer for an 81 *VW* van with a mere 148,000 off-road miles on it. This last one really has got me stumped.

Let's hear from you. ❄