Fonts

Typesetting Modern Greek – An Update

Yannis Haralambous

I would like to announce that, as of March 1, version `1.1` of the reduced greek fonts and macros (cf. *TUGboat* 10, no. 3 (1989), 354–359) is available. New features include

- hyphenation patterns for modern greek following the rules mentioned in *op. cit.*,
- some refinements of the fonts,
- an italics font,
- a new version of `greekmacros.tex`, and finally
- a BONUS: an extended `logo10.mf` file for writing the METAFONT logo in greek!

This work has been done on a Mac Plus using OzTEX and MacMETAFONT. Many thanks to Peter Abbott and Anestis Antoniadis for their kind help.

The `RGR PACKAGE 1.1` is (or will soon be) available at the Aston Archive (UK) and from myself on Bitnet, at `yannis@frcitl71` (after June 1, at `haralamb@frcitl81`). To obtain a Macintosh version (fonts for Apple *Textures*), send a 3.5″ diskette (*"Coupons de Réponse Internationaux"* would be appreciated) to

Yannis Haralambous
101/11, rue Breughel
59650 Villeneuve d'Ascq
France

Graphics

Combining Graphics with TEX on IBM PC-Compatible Systems and LaserJet Printers

Lee S. Pickrell

Abstract

We describe a method for including graphics in TEX documents created on IBM PC computer systems with HP LaserJet printers (and compatibles). Although TEX has suffered from a perception that it does not handle graphics well, the intrinsic graphics ability of TEX is no different than that of any other word processing system. However, two particular aspects of TEX may exacerbate the **perception** of a graphics limitation: TEX is implemented over a broad range of computer platforms, and TEX files are explicitly processed in two distinct stages.

We maintain that TEX has an excellent intrinsic graphics capability, which has largely been unexploited. To demonstrate the graphics capability of TEX, we have chosen the IBM PC and the HP LaserJet as a natural configuration. Indeed, this article was produced using the PC/LaserJet combination, and includes graphics plots derived from several different sources. The caption of each plot explains how the graphics image was obtained. These figures were not "cut and pasted", rather they were included electronically on the device driver level.

After considering several possible methods for acquiring graphics, printer capture is selected because the LaserJet PCL language is well standardized [1]. Our premise is that it is wasteful to regenerate an image when many application programs generate document-quality graphics. This technique also provides a larger number of graphics sources than any other we considered. This article concludes with a detailed description of a graphics solution for the PC/LaserJet combination, that forms the basis for a new software product which we are introducing. The product name is **CAPTURE**, and it acquired all the graphics shown here. The intent of this article is to demonstrate, by example, that TEX is well suited for graphics.

1 The Perception of a Graphics Limitation in TEX

A common perception of TEX is that it is unable to incorporate graphics into typeset documents [2, 3, 4]. This belief is unfortunate, and is not true when taken in context. The technical problem of introducing graphics in TEX is no different than the problem facing any other word processing system.

Because a graphics image can take any form, its components cannot be standardized as, e.g., types of font are standardized. The lack of a universal graphics standard forces graphics inclusion to be done on the device driver level. Therefore, in order to be printed, a graphics image must be in a format compatible with the output device. This requirement is the same regardless of the document preparation system and applies equally to TEX and to other word processing systems.

Mixing graphics with text is done regularly on many PC-based word processing programs which are not perceived to have difficulty including graphics [5,

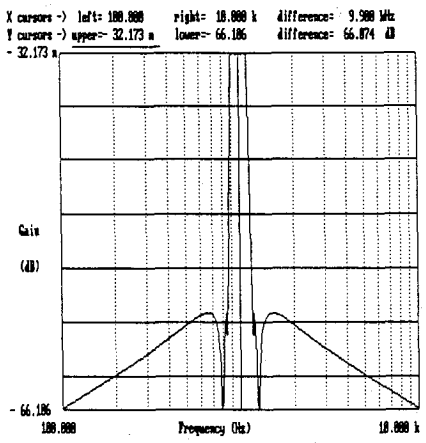

Figure 1: This is a plot from the Active Filter Design (AFD) Program by RLM Research, Boulder, Colorado.

6, 7, 8]. Conversely, the common belief is that TEX has a graphics limitation. Curiously, the technical problem is the same in both cases. This apparent dichotomy probably has more to do with psychology than computer technology, although our analysis is speculative. The issue is worth considering because this perception affects how TEX is used.

TEX is unique from other word processing systems because it is implemented across a broad spectrum of computing platforms. There are now implementations of TEX running on virtually all brands of mainframe computers and micro-computers. Conversely, most other word processing systems typically run on only one or two different computers. Take, for example, WordPerfect [8]. The original program only ran on IBM PCs, although it has recently been ported to the Macintosh. The specific graphics problem for the WordPerfect device driver only involves translating graphics images into the formats of the printers available on PCs. The available printers for the PC are quite limited in scope. Even with the addition of the Apple printer format, PostScript, the problem is still reasonable. Conversely, attempting to create a single program that would convert any graphics image to any of the printers used with TEX is intractable.

A solution to the perceived graphics limitation of TEX may not lie with any single piece of software or particular standard, but rather with an array of programs, each configured to the particular system on which TEX is run. In this sense, the graphics solution for TEX will be analogous to the development of device driver programs: a separate program for each computer/printer combination. This approach is intrinsically no different than that for other word processor systems, the only difference is the larger scope reflected by the broader application of TEX.

Another possible contribution to the psychology that TEX has difficulty with graphics is the explicit separation of the device driver component. A separate device driver program emphasizes that the inclusion of graphics is not intrinsic to the TEX program. Most other word processors use a single program which incorporates the device driver. However, functionally these programs must also have a distinct device driver component that is configured to the particular output device. Again consider the available PC-based word processors, for example, WordPerfect. During the installation the user must specify the output device [8]. This selection loads a particular code segment specific to that output device. The sections of code specific to other printers are retired. Functionally, a device driver program is chosen, although the operation is somewhat transparent to the user.

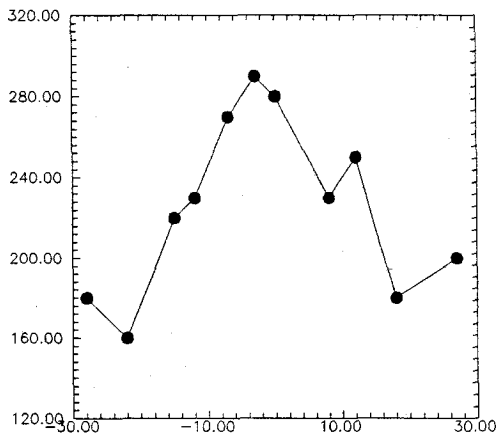

Figure 2: This is a plot from the popular scientific graphing program Grapher, by Golden Software, Inc.

2 Requirements for a General Graphics Solution for TEX

There are two requirements for creating a general graphics solution for TEX, and one has already been satisfied:

1. There must be a method, standardized in TEX, which allows the TEX program to communicate to the device driver. This communications path would enable TEX to instruct the device driver that a graphics file should be included. This condition is satisfied by the TEX

`\special{}` command. The argument of the `\special{}` command is passed verbatim to the device driver. When this argument is the name of a graphics file, the graphics image in the file is inserted into the output document. The procedure is functionally identical to that for any other word processor. For example, the other PC-based word processors also store the graphics images as separate files, and insert them at the device driver level. The only effective difference between TEX and these word processors is that historically it has been somewhat less convenient for TEX to communicate to the device driver.

2. There must be an array of software designed to create the graphics image files. This development would logically parallel the development of device drivers available for TEX. That is, a separate graphics program is needed for each computer/printer combination on which TEX is used.

The first condition is well satisfied by the `\special{}` command. Furthermore, more extensive TEX macros can be constructed which permit easy inclusion of the graphics files using the `\special{}` command as a root. Some of these are already available from TEX vendors. For example, Personal TEX, Inc., includes the file `setpcl.tex` with its product, which contains simple macro definitions for inserting graphics into TEX [3, 9].

We have also developed command definitions for including graphics in TEX, which are part of a new product called **CAPTURE**. The most primitive macro defines a command called `\insertplot{}{}{}` that creates an effective `\hbox{}` (or `\mbox{}` in LATEX). This box contains the graphics, and has the height and width of the image. The macro definition takes 3 parameters: the name of the graphics file, the vertical size of the graphics, and the horizontal size of the graphics. It can be treated as any other `\hbox{}` in TEX, (`\mbox{}` in LATEX). It inserts the graphics file and treats the image as a large letter of type. The definition of `\insertplot{}{}{}` is:

```
\def\insertplot#1#2#3{%
    \vbox to #2 true in{
      \vfill
      \hbox to #3 true in
        {\special{pcl:#1} \hfill}
      }% End of vbox
  }  %End of Definition
```

This particular macro definition works for the Personal TEX, Inc., drivers. Similar constructs have been made for the other PC-based TEX programs. For example, to use the μTEX or TEXPLUS drivers the "`pcl:`" must be replaced with "`hp:`" or "`hp: 300 insert`" respectively. Also, the `\vfill` and `\hbox to ...` may be reversed. However, the principal remains the same in all cases, an `\hbox{}` is created with the width and height of the graphics, and the graphics file is inserted.

We have also developed more extensive commands to be used in LATEX. These create plots that leave space for, position, caption, label, and insert graphical images. One particular macro is the `\pfig{}{}{}{}` command, which takes four parameters:

1. The name of the graphics file. This name is also used in a `\label{}` command so that the figure can be referenced by the file name using the `\ref{}` command.
2. The vertical size of the plot.
3. The horizontal size of the plot.
4. The text for the caption.

`\pfig{}{}{}{}` creates a floating block which has the dimensions of the graphics image. The plot is centered, captioned, and labeled with the file name. The plots used as examples in this article were all inserted using the `\pfig{}{}{}{}` command.

Using these simple command definitions, or others like them, mixing graphics with TEX is relatively simple, once the graphics image files are created. However, the problem has persisted of how to create the graphics image files to begin with.

3 The Problem of Choosing a Particular Processing Platform

The second condition which must be satisfied in order to mix graphics with TEX is the development of a broad base of graphics sources. Although a general solution to the TEX graphics limitation may require an array of programs, the proof-of-principle can be satisfied by demonstrating an effective graphics source on just one platform. Therefore, our first step was to choose the configuration we felt provided the largest opportunity for mixing graphics and TEX. This configuration is the IBM PC and the HP LaserJet printer. Our intent is not to establish the PC and LaserJet as the only configuration for including graphics in TEX, only that it is the best place to start.

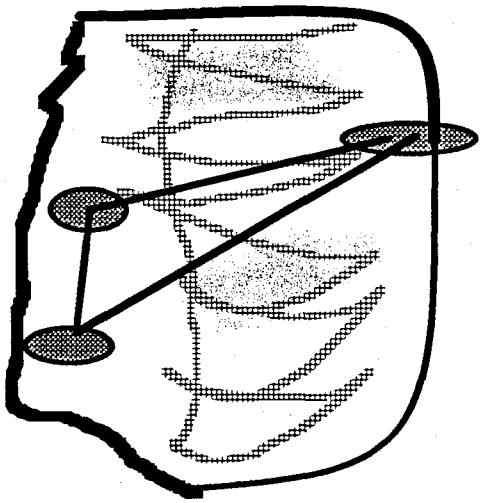

Figure 3: This is an utterly useless scribble done on PC Paintbrush. I did this just to prove I could capture the PC Paintbrush image.

3.1 Mainframe Compared to PC-Based TEX

The primary benefit of running TEX on a PC is the extensive source of graphics available on the PC, which far exceeds the possibilities from mainframes. Graphics is more common on PCs because the fast screen communications of microcomputers allow effective use of screen graphics. Graphics screens require very large communications bandwidths because each pixel (individual point of the image) is controlled by the processor. Conversely, the processor only needs to control characters to a text screen, because the local screen driver generates the fonts. Typically, the ratio of the graphics to text screen communications rate is about 500:1.

Most mainframe terminals are too slow to handle graphics well. The communications bandwidth between a terminal and a mainframe computer is typically 9600 characters per second. Conversely, the display on an IBM PC is part of the processor memory map. Communications with the screen are done at memory bus speeds, which typically run at 10 megahertz for AT class machines, or about a thousand times the speed of the mainframe terminal.

The fast screen communications allow microcomputer systems to be graphics based. Some notable examples are the Apple computers MacIntosh and Apple II, the WINDOWS operating system for DOS, and the new OS/2 operating system for IBM PCs. All of these operating systems run entirely in graphics mode. Therefore, we selected the IBM PC as the best choice for finding an abundant source of graphics.

3.2 Selecting from the Possible MicroComputer Printers

Because closely followed standards have evolved in the personal computer industry, there are only a few microcomputer printer standards which apply to this problem. In particular, there are three classes of printer which primarily support TEX on a PC: an Epson compatible dot matrix printer, a PostScript compatible laser printer, and a Hewlett-Packard LaserJet compatible laser printer.

PostScript systems (Apple or IBM) do not require a separate graphics program because PostScript-based TEX already supports a mixture of graphics and text. In one sense, PostScript is the ideal method for including graphics in TEX. PostScript is a complete page description language which defines both text and graphics, so the two can be mixed easily.

Unfortunately, PostScript is expensive, and its high cost has reduced the number of available applications. Adobe Systems charges large royalties on all applications which use PostScript as an output language, *and* on all of the laser printer manufacturers whose printers interpret the PostScript commands. PostScript laser printers typically cost a factor of two more than equivalent LaserJet compatible printers. Although laser printer prices have dropped in recent years, this ratio has remained constant. Also, PostScript based software applications typically cost more than those which do not support the language.

As a result of the high prices of PostScript, the number of PostScript applications are far fewer than those which support the LaserJet control language (PCL) [1]. PostScript has found a niche in the desktop publishing area, but this is a single application not necessarily well suited for scientists or engineers.

Another possible configuration is the IBM PC with an Epson compatible dot matrix printer. This combination is unsuitable because the dot matrix printer cannot switch modes between text and graphics as the laser printers can. Therefore, it is impossible to mix text and graphics, and the TEX dot matrix printer drivers do not support the `\special{}` command [3].

By elimination, we selected the HP LaserJet for the output device. The choice was fortuitous because the LaserJet (and equivalents) have become commonplace in the overall PC marketplace. This trend has accelerated because the prices of laser printers have dropped in recent years to levels equiv-

alent to the high-end dot matrix printers. Due to its wide use, almost all current applications now support the LaserJet PCL language in addition to the Epson. The LaserJet control language has become a de-facto standard in the PC industry [1].

4 Sources for the Graphics Image File

The next issue to be considered is the possible sources of graphics. We have identified three general methods for creating a graphics image file.

1. The program can be a file conversion utility. This method assumes that the source of the graphics will be one of the popular drawing or paint programs. The manufacturers of these programs publish the formats for the files which store the image. For example, the format for PC Paintbrush files is the PCX format, and the format for many image scanners is the TIFF format. Therefore, in order to use an image in one of these formats with TEX, all that is necessary is to convert from the format of the paint program to the format required by the LaserJet. Technically, this method provides a simple solution, but the range of graphics is severely limited. Although the file formats are published, only a few manufacturers use them and they are not standard. Even though typical file conversion programs attempt to cover many formats, the range of graphics is still limited.
2. The program can convert a graphics screen to the LaserJet control language. This approach is called screen capture, and is a popular method because it allows including graphics from any program which generates a graphics screen, whether or not the program supports the LaserJet. The disadvantage is that the image quality is poor. The best standard resolution for PC monitors is the 800 × 600 pixels VGA mode. This mode uses 800 × 600 pixels to generate the image across the entire 14 inch screen. By comparison, the LaserJet resolution is 300 pixels per *inch*, or about 7 times better. Screen capture offers the user two possibilities: a high resolution image that *at best* is about 2.5 inches wide on the paper, or a full size but very crude low resolution image.
3. The program can capture the printer output from the application program. This method has several benefits. First, the range of graphics is very large. Any program which supports the LaserJet can provide graphics for inclusion in a document. Because the LaserJet control language is a de-facto standard, almost all applications now support it [1]. Therefore, it is very likely that most application programs can provide graphics for TEX documents.

   Another benefit is that the full resolution of the LaserJet is used, and the graphics can be full sized. The quality of the graphics is much higher than for the screen capture method, and the range of graphics is much larger than for the file conversion method. As an example of the range and quality of graphics possible, we have included several examples in this document obtained with CAPTURE. The figure captions explain the source of each of the graphics plots. We have deliberately chosen application programs that are somewhat obscure to highlight the broad range of potential graphics sources.

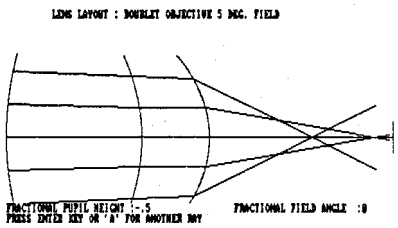

Figure 4: This is a plot of a badly aberrated, acromatic doublet lens at infinite conjugate. We used EZ-RAY, by Technical Software, Mountain View, CA., which is a ray tracing/lens design program. The screen image was converted to the LaserJet PCL language by GRAFLASR from Jewel Technologies, and captured by CAPTURE. It was then reduced to fit in the narrow columns of TUGboat.

5 Description of CAPTURE

The program which satisfies these conditions and can incorporate graphics into TEX documents is called CAPTURE. CAPTURE captures the output of any program which supports the LaserJet, writes the output to a file, and insures that the file can be inserted into TEX without disrupting the rest of the document.

The CAPTURE program consists of 4 files:

CPT.EXE This is the heart of the system. It captures the printer output from any application program which drives the HP LaserJet. Once the output has been captured to a disk file, it is automatically processed by the second program, FIXPIC.

FIXPIC.EXE This program processes the graphics file, so that it can be inserted into a TEX document. It removes all 28 control codes which

can disrupt the normal TEX output and allows re-positioning of the graphics according to the option switches.

`FIXPIC.OPT` This is an options file which contains the defaults the user wishes `CAPTURE` to use. The user can create `FIXPIC` with an ordinary text editor.

`PLOT.STY` This is a TEX style file which can be included in TEX and LATEX documents processed by any of the PC-based TEX systems. It has macros which simplify the graphics insertion.

To create the graphics file for inclusion into a TEX document, the command line for invoking the graphics software is prefixed with `CPT`. For example, say the user wishes to create a graphics file for TEX using Z-Soft's PC Paintbrush program. Normally, the user would type:

```
C:> paint
```

The computer would run the batch file `paint.bat`, which invokes PC Paintbrush. After the picture was created, the user would print it on the LaserJet using the output utility in PC Paintbrush.

In order to create a file to include in TEX, the user would do everything identically, except the command line would be prefixed with `CPT`. For example:

```
C:> cpt paint
```

After the image was created, the user would print it, just as he would otherwise. However, the output would not go to the printer, but would be written to a disk file. After the user exited PC Paintbrush, `FIXPIC` would automatically take over and sanitize the file. The sanitizing process would remove all the reset, positioning, and mode-changing commands, which would otherwise disrupt the rest of the TEX document. In addition, some commands may be added to the file such as horizontal positioning. The result is a file which can be directly included into a TEX document.

`CAPTURE` also calculates the width and vertical size of the graphics, which are printed at the end of the processing by `FIXPIC`. These numbers are used for positioning the graphics and leaving sufficient vertical space.

Using the printer capture, sanitizing, and macro definitions of the `CAPTURE` product, mixing graphics in TEX documents is quite simple.

6 Conclusion

The intent of this article is not to suggest that `CAPTURE` is the final solution for graphics in TEX. Rather, `CAPTURE` demonstrates a principle which can be applied more generally. Once graphics include files are generated, mixing graphics with TEX is relatively easy. Moreover, the generation of the graphics include files must be device specific. Just as separate device drivers are developed for each of the possible computer/printer combinations, so also must `CAPTURE`-like programs. We targeted `CAPTURE` for the IBM PC and the HP LaserJet because this configuration provides the largest possible spectrum of graphics sources at present and best illustrates the considerable graphics potential for TEX.

References

[1] *LaserJet series II User's Manual.* Hewlett Packard Corporation, Boise Division, P. O. Box 15, Boise, Idaho 83707, December 1986. Part No. 33440-90901.

[2] Stephan Lindner and Lutz Birkhahn. "Towards a Complete and Comfortable TEX System." *TUGboat*, **10**(3):368 – 372, November 1989.

[3] *Personal TEX News.* Personal TEX, Inc., 12 Madrona Ave., Mill Valley, CA 94941, Fall 1988. Volume 2, Number 2.

[4] T. L. (Frank) Pappas. "TEXnology on the IBM PC." *Computer*, 111 – 120, August 1989. Product Reviews.

[5] Kate Anderson. "Page-Layout Packages Boost Text Handling." *PC Week*, **6**(51):63, December 1989.

[6] Gus Venditto. "Pipeline." *PC Magazine*, **9**(1):63, January 1990.

[7] Edward Mendelson. "Two Aces and a King, The Big Three Word Processors Raise the Ante." *PC Magazine*, **8**(20):97 – 120, November 1989.

[8] *WordPerfect 5.0.* WordPerfect Corporation, 1555 N. Technology Way, Orem, Utah 84057, April 1989. ISBN 1-55692-200-0.

[9] Michael Spivak. *PCTEX Manual.* Personal TEX, Inc., 12 Madrona Ave., Mill Valley, CA 94941, November 1985.

◇ Lee S. Pickrell
Wynne-Manley Software, Inc.
1094 Big Rock Loop
Los Alamos, NM 87544
`pickrell%lsn.mfenet@ccc.nmfecc.gov`