Book Review

The LaTeX Graphics Companion and TeX Unbound—A Review of Two Books

Reviewed by Bill Casselman

The LaTeX Graphics Companion—Illustrating Documents with TeX and PostScript
Michel Goossens, Sebastian Rahtz, and Frank Mittelbach
Addison-Wesley, 1997
554 + xxv pages, $39.95

TeX Unbound—LaTeX & TeX Strategies for Fonts, Graphics, & More
Alan Hoenig
Oxford University Press, 1998
580 + ix pages, $35.00 (paper)
`http://www.oup-usa.org/docs/019509686X.html`

It is not easy to incorporate good mathematical figures in mathematical exposition—which is to say that the revolution in mathematical typesetting brought about by Donald Knuth's invention of TeX has not yet been matched by one in mathematical illustration. Curiously, at the same time Knuth gave us TeX he also gave us the graphics language METAFONT, but this has never enjoyed anywhere near the popularity of TeX itself.

There are many reasons why mathematical illustration is difficult. One's first impression is probably that the main difficulties are simply technical and that just around the corner will appear the perfect software tool. It is certainly true that in spite of the power of modern desktop computers, the technical tools available currently are either hard to use or of low quality, at least for mathematical purposes. But I would argue that the main difficulties are intrinsic to the problem—that mathematical illustration is a skill requiring practice and experimentation, if not natural talent. It may be that the awkwardness of the available tools has established an unnecessarily high threshold at which one is forced to begin, but I have trouble imagining that the task will ever be trivial. If one asks, for example, why the success of TeX has not been accompanied by success for METAFONT, then one possible answer is that typesetting (in spite of appearances!) is essentially a one-dimensional world where the number of choices is inherently limited.

There are roughly two separate phases to the technical difficulties of illustration: (1) producing the illustrations and (2) including them in mathematical papers written in TeX. The second step is largely distinguished from the first in that it does not involve the actual content of the illustrations. The lowest level of technical difficulty encountered in the second step is getting TeX to recognize the existence of an illustration, say, by constructing a box from it. Even this is occasionally frustrating, since the techniques used depend on the computer environment, and portability is not guaranteed. But the second step also frequently involves manipulating illustrations in various simple ways (scaling, rotating, perhaps coloring) which do not depend essentially on their content. Actually, the border between production and display of graphics in TeX is not so sharp as it might first appear, as anyone who has tried to construct complicated commutative diagrams knows from

Bill Casselman is professor of mathematics at the University of British Columbia, Canada. His e-mail address is `cass@math.ubc.ca`.

painful experience. The fuzziness of the boundary is also shown by current practices of font design. I like to think that one of the unsolvable philosophical problems of modern times is how to decide where text ends and graphics begin.

Both of the books under review are concerned with what might be called the middle ground of mathematical graphics. They describe in modest depth a large number of ways to produce illustrations and include in addition a briefer discussion of how to manipulate them once they are produced. Both limit themselves to techniques which can be used in almost all computer environments and without serious expense. Neither includes anything whatsoever on the intellectual process of making illustrations, neither discusses large commercial programs which one might wish to use to produce one's illustrations, and neither discusses seriously the details of page makeup that might lead one to abandon pure TEX and take up one of the commercial programs such as that used by the AMS, for example, to produce the final version of the *Notices*. Both books do, however, touch lightly on the question of how to produce mathematical graphics for display on the Internet, and both books also devote a fair amount of effort to explaining some aspects of font handling in TEX.

In this review I shall first discuss how the books handle what I call the second step of mathematical illustration—the incorporation of graphics already produced. I will then move on to the first step—that of producing mathematical illustrations—and talk about some options not covered in either book. Because the review weaves together discussion of both books, I have provided separate descriptions of the contents of each book in the last section, together with some closing remarks comparing the two books.

Manipulating Graphics

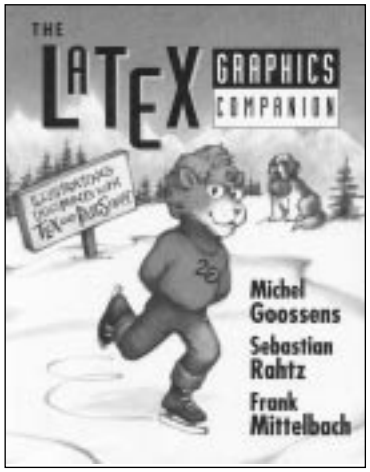
Once illustrations have been produced, it ought to be a mechanical process to incorporate them in a mathematical paper. This is essentially the case, but the difference between essence and reality can often be exasperating. Even here difficulties which appear at first merely technical are occasionally a matter of something deeper, such as questions of how figures are to be placed exactly where one wants them.

Hoenig's book begins with a somewhat discursive introduction to LATEX and other flavors of TEX. It does not attempt to give details of how to use TEX, but contents itself with an interesting survey which does a fairly good job of placing TEX in perspective. The book by Goossens et al. does nothing like this, but, after all, the same authors have covered this territory already in the authoritative manual *The LATEX Companion*. Well, not quite, because in this volume as well as in the earlier one, Goossens et al. do indeed restrict their attention to LATEX. This is probably a blessing for the large number who use only LATEX, but their book is therefore of limited use to the more technically sophisticated readers who would otherwise be attracted to it. The restriction to LATEX is especially frustrating since almost all of the advice they give can be paralleled in any flavor of TEX. However, figuring out the necessary adjustments in a non-LATEX environment might take a great deal of time. Those who do restrict themselves to LATEX will be able to use the LATEX `graphicx` package, which contains the convenient macro `\includegraphics`. This handles easily a very wide variety of input and handles well the problems of scaling and rotation one might encounter. Hoenig devotes a few pages to the LATEX graphics bundle, but Goossens et al. spend a whole chapter on it and do a more thorough job. Here, too, my impression is that this package is unnecessarily tied to LATEX and that it would not have been a difficult task for its developers to have made it available outside the LATEX environment.

In discussing the incorporation of graphics already produced, both books go on to lengthy discussions of font handling and to some comparison of the technical tools necessary for turning graphics into TEX boxes. Fonts make up, of course, one of the principal no-man's lands between graphics and text. Both books do a reasonably good job of explaining, for example, how to use PostScript fonts instead of the bit-mapped fonts that are often used currently by default in TEX. Hoenig spends more than 200 pages dealing with fonts, including a useful survey of the role of METAFONT in TEX's fonts, and his is one of the more interesting and valuable treatments currently available. Goossens et al. spend much less space on the topic, but perhaps what they say will be enough for most users of LATEX. Incidentally, font problems become more important when one takes up serious graphics work, because good mathematics illustration will not avoid labels and other textual inclusions, and it is not usually trivial to get text and figures to match well.

The problems of embedding a given graphic in a given TEX file are not always hard, but at times they can be formidable. This is largely because there is a wide variety in the kind of graphics file one wants to embed. Both books do well at explaining how to deal with the problem, given the assumptions of the authors. As I have already said, Goossens et al. explain primarily the `graphicx` package available with LATEX. Like many similar packages, it probably does not deal with all possibilities, but it does pretty well at hiding unnecessary complexities in those situations where it does work. In particular, it makes available a more or less homogeneous interface to the low-level programs such as `dvips` which it calls on to actually include graphics. The book is slightly frustrating here because they really do not tell one what

to do if one does not want to use the `graphicx` package. Hoenig has the virtue of dealing with all kinds of TEX, but does not really say much here except about the package `dvips`. This is a terrific program written and maintained by Tom Rokicki, once a student of Knuth's. In my experience it works best in a UNIX environment, where it can be incorporated easily into a `make` configuration, but even in other environments it often offers unique capabilities. At any rate, anyone incorporating complicated graphics in a paper should realize right from the start that publishers may have trouble dealing with them unless they are rendered into portable PostScript. There are pitfalls here—packages such as Mathematica are capable of producing stand-alone PostScript output, but it may take a little care to get it, since these packages can also produce semi-complete files which call on a special PostScript library that may be unavailable to a publisher. It is best to check portability and completeness by running pictures through a standard PostScript interpreter.



Neither of the books under review eliminates entirely the technical difficulties of incorporating graphics, but given the intrinsic complexity of the environment and given their announced assumptions, they do pretty well. Each of them also includes a few technical gems. I cannot resist mentioning in some detail the one that I find most useful, although it certainly might be considered unduly arcane by many. A common problem these days, dealt with briefly by both books, is that of rendering PostScript pictures into bitmap images (usually as `.gif` files) for embedding into Web pages. There are certainly several commercial packages that do this well, if expensively. In the low-cost domain I inhabit, the standard procedure is to use the workhorse program `ghostscript` (maintained by Peter Deutsch and Aladdin) to convert `.ps` to a simple but verbose bitmap format, from which another suite of programs of various kinds can produce the `.gif`. The main problem is that the initial conversion normally takes up an enormous amount of computer memory, because by default it works on a whole $8.5'' \times 11''$ page even if the image is quite small. I suppose I should have thought of it myself, but I was pleased to read on p. 458 of Goossens et al. how to insert a PostScript `setpagedevice` command into the `.ps` file like this

```
<< /PageSize [100 100] > setpagedevice
```

in order to shrink the size of the area converted (and hence stop my computer from spilling out petulant error messages about running out of memory).

Producing Graphics

In contrast to the technical problems mentioned above, producing the illustration itself is, I believe, an intrinsically difficult process, even if one discounts the higher intellectual activity required to get the picture to show what one wants it to. It does not perhaps have to be as difficult as it now often seems.

Here is a rough list of the options available to a mathematician who wants to produce mathematical illustrations:

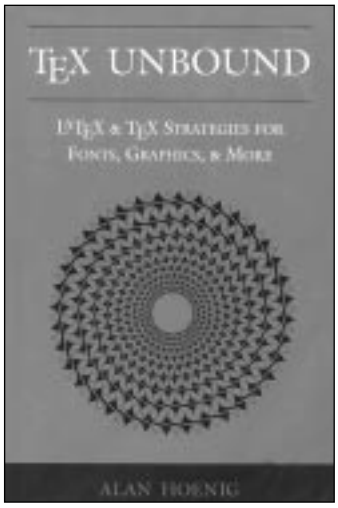
(1) Commercial drawing programs such as Adobe Illustrator or Corel Draw. Among these the most suitable will be those producing vector graphics, which are uniformly scalable, rather than bitmaps which show obvious defects when resized. In my experience these programs are not usually suitable for mathematics illustrations, since one often wants to exhibit a complicated structure they cannot easily deal with. There is one extremely important role which these programs can play in mathematical graphics, however. The most notorious problem one commonly confronts in this domain is that of embedding mathematical text in pictures. Of course, TEX is the only serious candidate for producing the text itself, but how does one then get the text into pictures? It is not difficult to use TEX and `dvips`, say, to produce what is called an *encapsulated* PostScript (EPS) file containing just a single label. Nearly all commercial graphics programs then allow one to import the EPS file into almost any figure using a graphical interface for correct placement. This is certainly in many ways the most convenient solution to the problem. It would be great if one of the free PostScript viewing programs, such as `ghostview`, allowed one to do this, but as far as I know none do yet. A recent release of Java includes a PostScript interpreter as a demonstration, and it ought not be too difficult a task to extend it to an EPS-importing tool.

(2) CAD (computer-aided design) programs developed primarily for engineering and architectural work. These often rely internally on a true programming language which can give pictures the required structure. However, they include a lot of capability which a mathematician will probably never use, and they are very expensive. It probably would not occur to most mathematicians to use one of these, but at least one person I know who does great graphics work relies almost entirely on

AutoCAD. Their 3-dimensional capability is pretty good.

(3) Mathematical software packages such as Mathematica, Maple, Matlab. They cost real money, but they can be used for a variety of purposes in addition to illustration. My major criticism here is that they are not quite flexible enough to produce highest-quality pictures in all circumstances, but after all this is an aesthetic judgment. They can get one a long way towards great pictures but if anyone has to resort to serious programming in one of these to draw pictures, he or she would probably be better off doing something else.

(4) Real graphics programming. For this, one might use some of the extensive graphics packages in C or Java and then write output in PostScript. One might even program directly in PostScript, although it is slow and severely limited in floating-point accuracy. The option of using a production programming language seems rarely to be seriously considered by mathematicians. Of course, programming is intrinsically difficult, but my own belief is that the difficulty of programming is not greater than the difficulties of designing good mathematical graphics in the first place and that the quality of output is almost always commensurate with the work put into it.



One other possibility is the graphics language METAFONT, which both books under review cover in some detail. I have already mentioned that METAFONT was designed by Donald Knuth to accompany TEX, and its use by Knuth in font design played a crucial role in TEX's success. For this reason alone, perhaps, it should occupy at least a small part of the heart and mind of every TEX user. In both these books some very elegant pictures produced by METAFONT are exhibited. However, I would not advise someone who dislikes programming to take it up, since it is really a rather complicated language; nor would I advise someone who likes programming to take it up, since I think it would be far more fruitful to take up C or Java or PostScript. Nonetheless, anyone who uses TEX extensively will probably find it useful to have at least a rough idea of what METAFONT is like, and each of these books offers a brief chapter on the topic. Both books also discuss PostScript, but more as an adjunct to printing rather than as a feasible way to produce pictures in the first place. They share also an apparent aversion to `ghostscript`, a freely available PostScript interpreter which I have found to be convenient and even invaluable.

(5) Several packages enabling one to do graphics more or less from within TEX. Both books cover a number of these. They generally have one great virtue pretty much missing from options (1)–(4), which is that they enable one to include TEX text inside the pictures they produce and often without a lot of fuss. In my opinion, all but one of the packages discussed in these books suffer from extremely low versatility and quality, however. The exception is the `PSTricks` package developed by Timothy van Zandt and Denis Girou, which comes with most free TEX distributions. This is essentially a TEX interface to PostScript. If explored in depth, it can do nearly anything that basic PostScript can, although I myself find the basic PostScript environment more pleasant. The great advantage of `PSTricks` is that it includes a large library of built-in routines that can produce spectacular effects. It also deals better than most with the problem of embedding mathematical text in figures.

One unfortunate but unavoidable fact is that no single tool does all tasks. It is not clear to me that one single tool ever will.

Summary

These two books have much in common, but they have their differences too. It might make a comparison easier if I summarize the contents of each.

The book by Goossens et al. opens with a chapter summarizing how to use graphics in LATEX. Chapter 2 describes the package of tools, such as `graphics` and `graphicsx`, that are bundled with LATEX. Chapter 3 describes METAFONT and a derivative program called `metapost`, a METAFONT-like interface to PostScript. Chapter 4 is concerned with `PSTricks`. Chapter 5 describes the package `Xy-pic`, which is a simple graphics language entirely embedded in TEX itself. Chapters 6–8 describe packages adapted to special areas such as chemistry, music, and games. Chapter 9 deals with the simple use of color in both drawings and text. Chapter 10 is concerned with how to use PostScript fonts, and Chapter 11 is a brief survey of other aspects of PostScript.

The book by Hoenig opens with a general description of TEX and LATEX. Chapter 2 tells how to obtain packages from the Internet. Chapter 3 is about METAFONT, and Chapter 4 describes the special features of LATEX, as opposed to other flavors of TEX. Chapter 5 covers the relations between TEX and other commonly used computer tools such as text editors and extensions of TEX that allow hyperlinks. Chapters 6–10 deal with fonts. Hoenig's treatment of graphics, with which the second half of the book is concerned, begins with a general discussion in Chapter 11. Chapter 12 discusses TEX-based graphics tools, Chapter 13 cov-

ers METAFONT and metapost, and Chapter 14 deals with PSTricks. (Thus Hoenig's Chapters 11–14 overlap closely with Chapters 1–4 of Goossens et al.) The final chapter is about a package mfpic, which is a TEX interface to METAFONT.

It will be apparent from this outline that the books overlap quite a bit, that Hoenig addresses a wider range of questions than Goossens et al., and that Goossens et al. are more specifically concerned with graphics questions. Given that they are addressing a somewhat narrow range of problems, both of the books under review do a fairly good job of explaining relatively simple solutions to the problems they do address. For those who use LATEX exclusively and are not interested in large-scale graphics production and font management, the book by Goossens et al. will be enough for most purposes. Hoenig's book is a more enjoyable read and suggests more distant journeys. The book by Hoenig, it seems to me, also provides more examples of figures useful to mathematicians.

Some other remarks: (1) In both books the figures of highest quality and interest were generally produced by PSTricks. The value of this package is perhaps not as clear as it would be if the books were to spend less time on less capable programs. (2) Presumably because it works only in a UNIX environment, neither book covers xfig (although Goossens et al. have a misleading reference to it, implying it is for some reason suitable only for computer scientists). (3) The book by Goossens et al. has not one but three separate indices. This eccentric and interesting organization is useful for some purposes, but none of the three qualifies as a traditional subject index, and this is occasionally annoying. (4) Both books lamentably seem to accept and encourage the current and widespread prejudice against doing serious programming in order to produce illustrations, but this is undoubtedly realistic in the current mathematical climate.

One final remark is that much of the most technical content of these books would be convenient to have in one public source on the Internet. This is especially true since this sort of information changes rather rapidly. For example, although Hoenig refers briefly to the CM fonts in PostScript form made available by Blue Sky Research, his reference is out of date, and Goossens et al. do not refer at all to them. This sort of thing is, of course, inevitable given the practices of traditional publishing.

References

The programs dvips and PSTricks are available at any of the CTAN archives. Some good sources of documentation are

http://tug.cs.umb.edu/dvipsk/

http://www.tug.org/applications/PSTricks/index.html

http://www.radicaleye.com/dvips.html

PostScript versions of mathematics fonts are indispensable for any serious integration of mathematical graphics and text. The Blue Sky fonts and a few others are available now from the AMS at

http://www.ams.org/index/tex/type1-cm-fonts.html

One source of useful technical information on TEX in general is the journal of the TEX User's Group, *Tugboat*. Information about it (and about TEX in general), including how to access some articles on line, can be found at

http://www.tug.org/

There are many sources for the programming graphics language PostScript on the Internet. A huge list can be found at

http://www.geocities.com/SiliconValley/5682/postscript.html#OTHER/

One reference of interest to mathematicians might be the text I have been using for several years to teach an integrated course on geometry and programming. This text is available at

http://sunsite.ubc.ca/DigitalMathArchive/Graphics/text/www/index.html

An extensive account of what Mathematica can do with graphics, which is of interest even if one does not use Mathematica, is contained in the book

Mathematica Graphics, Tom Wickham-Jones, Springer-Verlag, 1994.

For some of us, one of the most striking contributions of Donald Knuth is the observation that typography is of mathematical interest, in the sense that solving difficult technical problems in typography requires mathematical methods. The closest approximation to Knuth's style in the field of computer graphics is perhaps the column *Jim Blinn's Corner* published regularly in the IEEE journal *Computer Graphics and Applications*. Several of these columns have been collected together in *A Trip Down the Graphics Pipeline* (1996) and *Dirty Pixels* (1998), both written by Jim Blinn and published by Morgan Kauffmann. The author's home page is at

http://research.microsoft.com/~blinn/default.htm