A Century of Science Publishing
E.H. Fredriksson (Ed.)
IOS Press, 2001

Chapter 18

Developments in Technical Typesetting: TeX at the End of the 20th Century

Barbara Beeton
Editor, TUGboat, the Communications of the TeX Users Group

Introduction

The composition of mathematical and technical material has traditionally been considered "penalty copy", because of both the large number of special symbols required, and the complicated arrangement of these symbols on the page. While careful exposition has always been a valued skill for the mathematician or scientist, the tools were not available to allow authors to complete the presentation until relatively recently. The machinery has now come together: a desktop or portable computer; a text editor; an input language that is reasonably intuitive to a scientific author; a composition program that "understands" what mathematical notation is supposed to look like; a laser printer with a resolution high enough to print even very small symbols clearly. The composition engine is TeX, by Donald Knuth of Stanford University. This tool, originally developed specifically to produce the second edition of Knuth's *The Art of Computer Programming, Volume 2*, has been adopted by mathematicians, physicists, and other scientists for their own use. A number of scientific and technical publishers accept electronic manuscripts prepared with TeX and insert them directly into their production flow, while other publishers use TeX as a back end, continuing to accept and process copy in the traditional manner. TeX manuscripts are circulated in electronic form by their authors in e-mail and posted into preprint archives on the World Wide Web; they are portable and durable — a new edition of a work need not be typed from scratch, but can be modified from the file of the previous edition. In short, the impact on scientific authors is not just in the way their books and papers are published, but in the way they work.

From hot metal to cold type

By the end of the 1950's, the cost of composing technical material in the traditional manner was becoming insupportable for all but the most important or

high-volume publications. "Cold type" was replacing Monotype for some journals and books of transient importance, with a corresponding degradation of the perceived professional quality of their appearance. The editorial model remained the same, however: manuscripts prepared by (usually) the author's secretary were rigorously copy-edited, then turned over to the compositor (who might be a skilled typist working on a Varityper or IBM Selectric Composer) for conversion to camera-ready copy. The machines used were similar to typewriters, but instead of keys, they had interchangeable "fonts" and the spacing was defined in very small increments. In the case of a Varityper, two half-cylinders, each containing three rows of characters in one typeface, were mounted on a head that could be pivoted to change the style; the Selectric Composer used a separate "golf ball" for each typeface. The resulting copy was proofread and errors lightly marked in non-reproducing blue pencil (these were the days before photocopiers); it was then returned to the compositor for corrections. Typed copy was corrected by keying patches and "cutting them in" — superimposing the correction over the erroneous original on a light box, using a very sharp knife or razor blade to cut through the two layers, and taping the good parts together with opaque tape; touchup with white paint was often necessary so that the camera would not pick up shadows from the cuts.

Obviously, this was not something that an author would willingly undertake.

The rise of the computer

The American Mathematical Society (AMS) undertook some experiments in the early 1960's with highly stylized input prepared on paper tape according to a mnemonic system in which mathematical notation and text were separated into two "columns": commands and mathematical notation at the beginning of the line, and ordinary text following a tab character. The results, processed by a service bureau, were adequate, but the process was cumbersome, and didn't yield the hoped-for level of cost savings.

Over the next decade, other organizations developed computer-based systems, several of which became commercial products. A compactly-coded input scheme by Penta required dedicated hardware and highly trained keyboarders. A system from Bedford used early CRTs and pointing devices to manually position symbols; it too required dedicated hardware and skilled personnel, and according to anecdotal reports, the same mathematical expression in two chapters of the same book, prepared by two input technicians, might come out looking quite different in the two places.

Other general-purpose typesetting programs such as RCA's Page-1 and Page-2 were becoming fairly widely used, but while they were quite good for ordinary

prose and directories of various kinds, multi-level mathematics was quite beyond their capability.

In the academic world, the program TROFF, developed at Bell Laboratories as an adjunct to the UNIX operating system and the C programming language, was gaining a devoted following. The licensing of this system was extremely generous to academic users, requiring not much more than a token fee; however, the price to non-academic organizations (which included the AMS) was quite steep. Therefore, although TROFF and its companion equation processor, EQN, were very attractive, the financial implications were not.

The AMS had an opportunity to obtain the rights to a program by Science Typographers (STI) that would run on a general-purpose computer (an IBM 360 or clone). This system, like the Penta system, had rigidly coded input, but the ability to use the computer for other business processes made the investment more acceptable. An alternate input language was devised at AMS, based on the two-column (mnemonic) system developed earlier; it was thought that potentially this form of input might become usable by authors or their secretaries, and not require the highly-trained keyboarders that were hitherto indispensible. Input was initially on punched cards, with simulated distinctions between upper- and lowercase. Paper tape and mini-barcode Selectric type balls were also used to some extent. Bulk output was sent on magnetic tape to a service bureau for imaging on a VideoComp; an alternate output path involved conversion to Photon 713 machine code, with punched paper tape as the transfer medium. (Several acolytes of this system became quite adept at repairing torn or mispunched paper tapes in emergencies.) The STI program was used by AMS to prepare nearly all journals and many books from the mid-1970's through about 1984.

The birth of TeX

In 1978, the situation changed drastically. Donald Knuth, a computer scientist teaching at Stanford University, was invited to deliver the Gibbs Lecture at the annual meeting of the AMS. The lecturers for this series have been eminent scientists who are often known for their use of mathematics in disciplines other than mathematics; among the lecturers since the series began in 1924 were Albert Einstein, John von Neumann, Norbert Wiener and Paul Samuelson. The Gibbs Lecturer's topic is his or her own choice. Donald Knuth chose as his topic "Mathematical Typography" — this was the public unveiling of TeX.

Knuth's major published work is *The Art of Computer Programming* (TAOCP), a deep and broad survey of programming techniques and algorithms, originally intended to comprise seven volumes. However, by 1977 (when three volumes were

in print), the programming discipline had advanced with such great speed that Knuth had to revise volume two. When he first looked at the galley proof for this revision, he was appalled. Unlike the first edition of the first three volumes, which had been typeset by Monotype, the revised volume two had been phototypeset. Not only did it not look like the first edition, its quality was, to Knuth, unacceptable.

After recovering from the shock, Knuth thought about the problem, and determined that he, as a computer scientist, should be able to use his computer to do something about this. He realized that he would have to develop fonts as well as the computer program, but figured that if he planned well, the project might take him six months to a year.

Knuth had taken great interest in the typesetting of his books, and also in how mathematical notation appears on the printed page. When he began to think about assuming the responsibility for composing the new edition of TAOCP, he began to look more closely at published literature, not only technical, but the products of fine presses, to learn to recognize quality so that he could produce it.

Knuth began to think about his new program early in 1977; his journal for May 5 shows two entries: "Read about Bell Labs typesetting system", and "Major design of TEX started" [6, p. 482]. By the middle of May, he had devised the principal mechanism for placing type on pages, as well as the main features of the input language he would like to use; he recorded his decisions in a long memo saved as a computer file named "TEXDR.AFT" (because the computer he was using wouldn't allow it to be called "TEX.DRAFT"). As soon as he finished this memo, he went away with his wife on a library associates' tour of fine presses around Sacramento. Throughout the development of TeX he continued his education in typography, eventually working with several eminent type designers and typographers, and in the process creating a computer typography group at Stanford that nurtured several members of the next generation of type designers.

The first iteration of TeX was written in the SAIL language, developed at the Stanford Artificial Intelligence Laboratory. This language was specific to the mainframe "edusystems" of Digital Equipment Corporation (DEC). These machines were among the first timeshared systems, well suited to the needs of computer science research and experimentation. Access to computer time was almost no problem, but to get the best response, Knuth shifted to "night phase" for a lot of his work.

In his design of TeX, Knuth visualized an arrangement of boxes, containing single characters or composed groups, and flexible glue, which joins boxes within the framework of a page. At a low level, TeX manipulates these components according to rules appropriate for text or for mathematical expressions, with reference to

directives in the input file. The input vocabulary is mnemonic, with terminology usually selected to be familiar to a mathematician. Further, TeX is designed as a macro compiler, so it is possible to overlay the base commands with a different vocabulary, providing more complex operations through a single instruction. The most ambitious uses of this facility provide a user interface that consists entirely of logical markup and "hides" the underlying composition commands.

In mid-May 1977, Knuth left off work on TeX to develop the fonts that would be needed to put TeX's output onto paper.

Fonts for TeX

Stanford had an early laser printer, the Xerox XGP, with a resolution of about 180 pixels per inch; Knuth realized that with this kind of digital device, he could form fonts by treating each character as a matrix, filling in certain cells — pixels — while leaving others empty. A page of a book would then become a gigantic bit matrix.

The resolution of the XGP was not adequate, of course, to produce crisp images, but when Knuth looked through a magnifying glass at a sample produced on a high resolution machine (probably based on a cathode ray tube), he realized that the application of ink during printing will smooth out the edges when the raster is small enough. He observed that the critical density for ink smoothing was probably somewhere between 500 and 1000 pixels per inch [6, p. 35]; this value is very close to the density at which the human eye can no longer distinguish small irregularities.

It was important to Knuth to solve the problem of defining lettershapes in a purely mathematical way, so that as the precision of raster devices increased, the task of creating new fonts at higher resolutions did not have to begin again; the definition of the lettershapes would stay the same in a machine-independent form. The analogy he initially chose was to define the shape of a pen and describe in terms of cubic splines the path that it should follow. These shapes are defined in terms of parameters (such as pen dimensions and degree of slope) chosen by the designer; alteration of the parameters results in changes to the lettershapes. The first implementation of this program, Metafont, was usable early in 1979.

After considerable experimentation, and consulting with professional font designers, it became clear that the pen-path model was not how fonts were actually designed. Although type is based on writing, it is not the same, and a more satisfactory model consists in defining the outlines of the glyphs and then filling them in. The second version of Metafont adopts this model; released in 1985, it has remained unchanged since then except for a few bug fixes.

Using Metafont, Knuth developed a comprehensive family of fonts, known as Computer Modern, for use with TeX. In addition to alphabets in roman and italic shapes, in all the common weights, with and without serifs, there are a number of fonts devoted to symbols, so the collection of fonts is fully capable of handling nearly all mathematical typesetting needs. And if on occasion a symbol is required that doesn't yet exist, an author or publisher can use Metafont to create it.

One of the main problems in a raster environment is to determine which pixels are to be turned on; with Metafont, the rasterization is fixed at the time the program is run, and different renditions of a font are thus required for different output devices. The space required for storage of these files can be substantial. This is quite a different approach from the Type 1 (PostScript) and TrueType models, in which an outline is presented to a rasterizer at the time the composed image is rendered into output. These outlines are much more compact than Metafont raster fonts, and TeX users expressed a desire to both save space and get access to the wider variety of fonts available in outline formats. Since TeX itself requires only the font metrics, to make outline fonts usable by TeX, it suffices to create metric files in the required format and write suitable TeX-output-to-device programs to handle the imaging.

The Computer Modern fonts have been re-implemented in Type 1 form, and, at least in a publishing environment, TeX is now used almost exclusively with Type 1 fonts.

The spread of TeX

We left TeX in mid-1977, still under construction. Knuth's Gibbs Lecture in 1978 has already been mentioned. Reaction to this lecture, which introduced TeX and Metafont to a mathematical audience, was more enthusiastic than Knuth had expected. In particular, Richard Palais, then chairman of the AMS Board of Trustees, recognized the potential of these tools to typeset AMS publications. He interpreted their status as being ready for use, not still a research project. Arrangements were made for a small group of mathematicians and AMS employees to spend the month of July 1979 at Stanford to learn TeX, "bring it back and make it work". (These were my marching orders as I was presented with plane tickets and a copy of the first TeX manual.) Two members of this group were charged with developing TeX macro code for processing various AMS publications including Mathematical Reviews. The others were to create a package of macros suitable for use by authors, along with a user manual; this became AMS-TeX.

As it turned out, there were still some serious gaps in TeX, which, up to that time, had been designed for use by essentially one person, Knuth, to produce one

particular, well-delineated series of books. As more people heard about TeX and experimented with the original version, the gaps became known more clearly, and many requests were communicated to Knuth for extensions and "improvements". Where such changes were justified by reference to well-documented examples, the required functionality was usually provided. In a few instances, such as more flexible handling of diacritics, Knuth rejected the requests, giving his reasons for doing so; in the case of diacritics, he said that it would be more appropriate and robust to provide fonts with already accented letters than to apply piece accents.

The first version of TeX ran on only one type of hardware — the DEC "edusystems" for which it was written. It was, nonetheless, capable of typesetting the long-delayed volume two of TAOCP, so this was done, and the second edition appeared in January 1981.

In order to make TeX available to a wider audience, it had to be modified for other hardware and operating systems. A prototype was coded in Pascal, a programming language popular at that time for instructing computer science students; Knuth's graduate students developed this version under his direction. At this point, Knuth decided that he wanted TeX to be more than just a typesetting program; as a committed teacher, he felt that it should also be a model of a large program that would illustrate good programming techniques. To this end, Knuth devised yet another concept, "literate programming".

In using this technique, program code is interspersed with documentation, in a sequence that is logical from the point of view of explaining what is going on. The source file of a literate program can be processed in two ways: the documentation can be stripped out and the code reorganized into the sequence required by a compiler, or the entire file can be converted to a form that can be typeset by TeX. With this tool, Knuth started again from the beginning, and recoded both TeX and Metafont, taking care to use a "least common denominator" version of Pascal, to ensure that the programs could be compiled and installed on any platform that provided enough memory for the programs to run. This excluded a few smaller, older machines, but most popular machines were soon able to boast a standard implementation of TeX — not, however, before the compilation of TeX had flushed out a few bugs in nearly every Pascal compiler. This version is known as TeX 2 or TeX84.

After refining the documentation, Knuth typeset the two programs; these were published as two volumes of the series "Computers & Typesetting" (C&T) — *TeX: The Program* and *Metafont: The Program*. Of the former, one reviewer said that it was the first program he had ever been able to curl up with by the fire and enjoy reading it. These two volumes are fundamental illustrations of the "open source"

concept in programming.

Three other volumes complete the C&T series: two reference cum user manuals, *The TeXbook* and *The Metafontbook*; and *Computer Modern Typefaces*, a heavily illustrated book describing the fonts that Knuth created and providing a practical example of how Metafont might be used in practice. The five volumes were formally released in 1986 at a party held at the Computer Museum in Boston.

TeX became quite popular among mathematicians and scientists in the "hard" sciences, who require considerable mathematical content in their papers and books. It also acquired a substantial following among academics in Eastern Europe and other areas of the world where economic conditions do not permit purchases of expensive software. Two features are particularly attractive in this regard: first, TeX can be obtained with almost no monetary outlay; second, if suitable fonts don't exist, a user can create them (this has been accomplished for Greek, cyrillic, Amharic, and other scripts).

The heavy use of TeX in Europe made one limitation particularly onerous: TeX wasn't really designed to hyphenate words with accents and diacritics. In 1989, a delegation of TeX users from Europe approached Knuth at the annual TeX Users Group meeting and explained to him that some changes were needed to permit them to typeset their languages effectively. Although reluctant to make any more changes to TeX, Knuth was persuaded by their arguments, and he provided some major enhancements to support 8-bit input and effective hyphenation of languages other then English. This version is known as TeX3; after its release Knuth announced that, for his part, "TeX is frozen", and only the most egregious bugs would be corrected thereafter.

Knuth's custom has been to offer a reward ($2.56) to readers of his books for the first report of any error. In the case of TeX and Metafont, he also rewards the finders of programming bugs; this reward was doubled annually through the release of TeX3, and then stabilized at $327.68. Knuth now looks at bug reports only every two to three years; only one or two major bugs have been reported during each of the last few cycles.

TeX from an author's point of view

Arguably the best thing about TeX for an author is that it gives documents a professional typeset appearance.

Although becoming a skilled TeX user requires some effort, the ubiquity of computers and the decrease in availability of secretarial support in many academic departments provides incentive, especially for graduate students and younger faculty. For physics, perhaps even more than mathematics, the growth of preprint

databases, such as the arXiv at Los Alamos (http://xxx.lanl.gov), is a further incentive to master TEX (arXiv strongly prefers TEX submissions). For this use, TEX has several advantages: the source is plain ASCII, so it will not become unusable with the next release of the processing program; it is compact; it is freely available for all popular platforms; it generates high-quality output; and it retains contextual information, such as the relationship between equation numbers and cross references.

There are some other advantages for authors. TEX's vocabulary and syntax for mathematical notation is straightforward; it is nearly equivalent to the descriptive language that two mathematicians might use when communicating by telephone. This notation is also known to several important symbolic algebra programs, which are able to generate output in TEX form for direct use by an author. An author who is fluent in the TEX mathematics notation can combine all these tools to record an idea as it develops; it has been reported that this capability has radically changed the way that some mathematicians work.

Some authors prefer a "point and click" or WYSIWYG approach to input. Even if they agree that the results from using TEX are superior to those from word processors, they still struggle with the learning curve. For such people, tools have been developed to provide menus and on-screen formatting of mathematical expressions, with TEX code stored behind the scenes.

The situation is not so clear with respect to the document structure. "Plain" TEX is a low-level typesetting language, and with it, one can obtain almost any desired result. Although several highly functional front-end interfaces have been created, some authors still prefer to use raw typesetting commands. On the other hand, for a publisher, especially a publisher trying to compile journals out of submissions from multiple sources, article sources prepared using uniform structural markup is essential. Some authors can't be bothered with such "petty" concerns, but more and more publishers are using the reward of more rapid publication when a submission conforms to the rules, along with the threat of retyping a submission, slowing down its production and incurring the possibility of new typographical errors, for submissions that cannot be inserted smoothly into the normal production flow. It is still a challenge to get all authors to actually read instructions.

TEX from a publisher's point of view

For most publishers, the TEX flavor of choice is LATEX. This macro collection provides a user interface that defines logical structural markup for the top matter and sectioning elements of a document, and supports automatic numbering, cross-referencing, and interaction with standard tools for compiling bibliographies and

indexes.

While not as rigorous as SGML or XML, the LaTeX framework is reliable and permits authors to produce a well-structured document onto which a publisher can superimpose a house style. There are a few rough edges, however; the handling of author addresses in basic LaTeX, for example, does not conform to the style of some publishers, and in extending this area, different publishers have made some mutually incompatible decisions. Nonetheless, there is a common, dependable core, and public-spirited users have made extension packages for many desirable features, usually (if not always) doing their best to ensure inter-compatibility with other commonly used packages.

What (La)TeX doesn't provide is rigor. SGML and XML documents are required both to be well-formed and (at least for SGML) to conform to a DTD. There is no such requirement for (La)TeX documents, and authors are free to ignore the conventions of a document style if they so choose; TeX is quite forgiving in processing a mixture of logical markup and physical typesetting commands, as long as they properly reduce to TeX's somewhat arcane syntax rules. The disadvantage in this is that transformation other than by TeX for presentation in modes other than on paper (e.g., in HTML, or via audio) becomes quite difficult, and in some circumstances nearly impossible.

There are nonetheless some things that a publisher can do to increase the likelihood that a TeX manuscript submitted by an author can be processed with high reliability and minimal special handling. (1) Provide well-designed "author packages" and easy-to-follow instructions. (2) Make sure that the fonts the author is using are fully compatible with the ones used in-house. (3) Provide at least introductory TeX training to all production staff handling TeX submissions, and have at least one person with solid TeX expertise on staff to answer questions from authors and production staff; don't try to rely entirely on outside consultants. Of course, this changes the nature of the production environment, but that has changed already, and no traditional technical publisher would forgo thorough training in house style and copy-editing techniques; only the tools are different.

There is one more danger in using TeX. There are very few symbol fonts available, and few authors are likely to be willing to purchase new ones when a reasonable set is available for free. This limits the choice of text fonts unless the publisher is willing to do extra work to recheck line breaks and similar features which are particularly suscriptible to font changes. The document styles that come with LaTeX are very recognizable, as are the fonts. A publisher is advised to invest in some expert design effort to create styles with a distinctive look, yet are still compatible with what the author is using.

The future

Many technical journals and some books are now published in electronic form as well as, or even instead of, on paper. The style suitable for paper is often not best for reading on a screen. Even if pages are put online as PDF files, the capabilities of an online environment — an active contents list, links to internal cross-references, etc. — should be incorporated into a document. For certain kinds of documents, particularly instructional ones, a different design is desirable, one that presents material in small chunks, not all at once.

Math has always been a niche market, and future developments are likely to come, as TeX has, from a self-interested source. TeX users — both authors and publishers — want to participate in the electronic revolution, and that will take a while to become routine. The major problem is the absence of suitable fonts. A two-pronged effort undertaken by STIPub, a consortium of scientific societies and publishers, is now addressing this gap. One part of this project is directed at Unicode, to increase the population of symbols with standard Unicodes; the other is to create a font containing these symbols and make it freely available to any and all users of the Web.

As noted earlier, the variability of a (La)TeX file means it is not ideal for repurposing. For this, XML and SGML are superior. MathML, an XML application that defines a structure for expressing math formulas in a Web-compatible manner, is now an accepted recommendation of the World Wide Web Consortium. However, TeX is much less verbose and more easily handled by a human than any of these *MLs. For an author to switch from TeX, some very capable input tools will have to become available; authors, having caught the TeX habit, are not going to give up without a struggle. In the long run, an internal *ML format for archiving will probably become the norm, with translation by various tools to formats suitable for presentation in different media. The development of such tools is proceeding slowly. Like TeX, which took ten years rather than the originally envisioned six months, they will arrive, but not as soon as one hopes.

TeX is no longer novel, but it is still one of the most reliable and flexible typesetting programs available, and one of very few that is capable of producing publication-quality math. Stability is a virtue in this context. However, some missing features would make it even more valuable, especially to users working in languages other than English. Several projects are underway that will extend TeX to provide such features. pdfTeX has added a few features that are directed at hypertext and online presentation, and outputs PDF files directly. e-TeX has incorporated into the present TeX program some new elements such as right-to-left typesetting. NTS (a "new typesetting system") is re-implementing TeX in a "more modern" program-

ming language (Java, at last report) and then making additions. Omega uses Unicode for its input character set, and provides a number of filters to handle the special requirements of non-Western scripts. It remains to be seen which of these projects might result in a system that develops a critical mass of users to supersede TeX itself.

None of these TeX extensions directly addresses the special needs of flexible Web presentation. That will more likely be handled by some new internal format, like MathML.

In more mainstream developments, some of TeX's key ideas have been adopted for incorporation into new products. For example, TeX's line- and paragraph-breaking algorithm is now in Adobe's InDesign, by way of the hz program by Peter Karow and Hermann Zapf; however, this does not, and probably will never, handle math. Typesetting of math is too small an area, and much too demanding, for any of the major software houses to want to join in. There simply isn't any possibility of large profits, especially since a free alternative is available.

Where will math publishing be in twenty years? That is hard to predict. The only thing certain is that mathematicians and scientists will continue to have something to say to one another, and will demand that it be presented clearly and accurately.

References

[1] Knuth, D.E. (1984) *The TeXbook.* Volume A of Computers & Typesetting. Addison-Wesley, Reading, MA.

[2] Knuth, D.E. (1986) *TeX: The Program.* Volume B of Computers & Typesetting. Addison-Wesley, Reading, MA.

[3] Knuth, D.E. (1986) *The Metafontbook.* Volume C of Computers & Typesetting. Addison-Wesley, Reading, MA.

[4] Knuth, D.E. (1986) *Metafont: The Program.* Volume D of Computers & Typesetting. Addison-Wesley, Reading, MA.

[5] Knuth, D.E. (1986) Computer Modern Typefaces. Volume E of Computers & Typesetting. Addison-Wesley, Reading, MA.

[6] Knuth, D.E. (1999) *Digital Typography.* CSLI Lecture Notes Number 78, CSLI Publications, Stanford, CA.