

# TUGBOAT

Volume 20, Number 4 / December 1999

	339	Addresses
<b>General Delivery</b>	341	From the President / <i>Mimi Jett</i>
	342	Editorial comments / <i>Barbara Beeton</i> On being a fossil; Erratum: Mimi Jett's term of office; Gutenberg: the man of the millennium; Sebastian Rahtz leaves the <i>TUGboat</i> production team; International news: Greek, Russian and Vietnamese groups; Clarification of the CTAN "nonfree" classification; The origin of the @ sign; Communication by flags
<b>Typography</b>	344	Typographers' Inn / <i>Peter Flynn</i>
<b>Font Forum</b>	347	TrueType fonts in $\TeX$ / <i>Vladimír Koutný</i>
	348	The semaphore alphabet / <i>Vít Zýka</i>
<b>Software &amp; Tools</b>	350	The Paper Path: XML to paper using $\TeX$ XML / <i>Brian E. Travis</i>
	356	A WYSIWYG $\TeX$ implementation / <i>Igor I. Stokov</i>
<b>Book Reviews</b>	359	<i>The L<sup>A</sup>T<sub>E</sub>X Graphics Companion</i> and <i>TeX Unbound</i> —A review of two books / <i>Bill Casselman</i> <i>The L<sup>A</sup>T<sub>E</sub>X Graphics Companion</i> , by Michel Goossens, Sebastian Rahtz, and Frank Mittelbach; <i>TeX Unbound: L<sup>A</sup>T<sub>E</sub>X &amp; TeX Strategies for Fonts, Graphics, &amp; More</i> , by Alan Hoenig
	364	<i>Digital Typography</i> , by Donald Knuth / <i>Peter Flynn</i>
<b>Errata</b>	366	The good name of $\TeX$ ( <i>TUGboat</i> 20, no. 2, p. 93) / <i>Jonathan Fine</i>
	366	<i>TUG99</i> ( <i>TUGboat</i> 20, no. 3) / <i>Christina Thiele</i>
<b>Resources</b>	367	A CTAN search page / <i>Jim Hefferon</i>
<b>Hints &amp; Tricks</b>	367	Hey — it works! / <i>Jeremy Gibbons</i>
	370	The treasure chest / <i>Christina Thiele</i>
<b>L<sup>A</sup>T<sub>E</sub>X</b>	375	L <sup>A</sup> T <sub>E</sub> X News, Issue 12, December 1999 / <i>L<sup>A</sup>T<sub>E</sub>X project team</i>
	376	Scaled Pictures in L <sup>A</sup> T <sub>E</sub> X / <i>Bruce Shawyer</i>
<b>Tutorial</b>	378	Book design for $\TeX$ users: Part 2: Practice / <i>Philip Taylor</i>
<b>Report</b>	389	Preparation of documents for multiple modes of delivery—Notes from TUG'99 / <i>Ross Moore</i>
<b>Abstracts</b>	394	<i>Les Cahiers GUTenberg</i> , Contents of double issue 33/34 (November 1999)
	395	Euro $\TeX$ '99 Proceedings—Paperless $\TeX$
<b>News &amp; Announcements</b>	399	Calendar
	401	TUG2000—The 21 <sup>st</sup> Annual Conference
<b>Late-Breaking News</b>	400	Production notes / <i>Mimi Burbank</i>
	400	Future issues
<b>Cartoon</b>	340	Download free fonts! / <i>Roy Preston</i>
<b>TUG Business</b>	402	Institutional members
	403	Statement of ownership
<b>Advertisements</b>	403	$\TeX$ consulting and production services
	404	Y&Y Inc.
	c3	Blue Sky Research

## TeX Users Group

### Memberships and Subscriptions

*TUGboat* (ISSN 0896-3207) is published quarterly by the TeX Users Group, 1466 NW Naito Parkway, Suite 3141, Portland, OR 97209-2820, U.S.A.

2000 dues for individual members are as follows:

- Ordinary members: \$65; \$10 surcharge if payment received after May 1, 2000, to cover shipment of back issues.
- Students: \$35; \$10 surcharge if payment received after May 1, 2000.

Membership in the TeX Users Group is for the calendar year, and includes all issues of *TUGboat* for the year in which membership begins or is renewed. Individual membership is open only to named individuals, and carries with it such rights and responsibilities as voting in TUG elections. Contact the TUG office for information.

*TUGboat* subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. Subscription rates: \$75 a year, including air mail delivery; \$10 surcharge if payment received after May 1, 2000.

Periodical-class postage paid at Portland, OR, and additional mailing offices. Postmaster: Send address changes to *TUGboat*, TeX Users Group, 1466 NW Naito Parkway, Suite 3141, Portland, OR 97209-2820, U.S.A.

### Institutional Membership

Institutional Membership is a means of showing continuing interest in and support for both TeX and the TeX Users Group. For further information, contact the TUG office ([office@tug.org](mailto:office@tug.org)).

*TUGboat* © Copyright 1999, TeX Users Group

Permission is granted to make and distribute verbatim copies of this publication or of individual items from this publication provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this publication or of individual items from this publication under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this publication or of individual items from this publication into another language, under the above conditions for modified versions, except that this permission notice may be included in translations approved by the TeX Users Group instead of in the original English.

Copyright to individual articles is retained by the authors.

Printed in U.S.A.

## Board of Directors

Donald Knuth, *Grand Wizard of TeX-arcana*<sup>†</sup>  
Mimi Jett, *President*<sup>\*+</sup>  
Kristoffer Rose<sup>\*+</sup>, *Vice President*  
Don DeLand<sup>\*+</sup>, *Treasurer*  
Arthur Ogawa<sup>\*+</sup>, *Secretary*  
Barbara Beeton  
Karl Berry  
Kaja Christiansen  
Susan DeMeritt  
Stephanie Hogue  
Judy Johnson<sup>+</sup>  
Ross Moore  
Patricia Monohon  
Cheryl Ponchin  
Petr Sojka  
Philip Taylor  
Raymond Goucher, *Founding Executive Director*<sup>†</sup>  
Hermann Zapf, *Wizard of Fonts*<sup>†</sup>

<sup>\*</sup>member of executive committee

<sup>+</sup>member of business committee

<sup>†</sup>honorary

### Addresses

General correspondence,  
payments, etc.  
TeX Users Group  
P. O. Box 2311  
Portland, OR 97208-2311  
U.S.A.  
Delivery services,  
parcels, visitors  
TeX Users Group  
1466 NW Naito Parkway  
Suite 3141  
Portland, OR 97209-2820  
U.S.A.

### Telephone

+1 503 223-9994

### Fax

+1 503 223-3960

### Electronic Mail

(Internet)  
General correspondence,  
membership, subscriptions:  
[office@tug.org](mailto:office@tug.org)  
Submissions to *TUGboat*,  
letters to the Editor:  
[TUGboat@tug.org](mailto:TUGboat@tug.org)  
Technical support for  
TeX users:  
[support@tug.org](mailto:support@tug.org)

To contact the  
Board of Directors:  
[board@tug.org](mailto:board@tug.org)

### World Wide Web

<http://www.tug.org/>

<http://www.tug.org/TUGboat/>

### Problems not resolved?

The TUG Board wants to hear from you:  
Please email to [board@tug.org](mailto:board@tug.org)

TeX is a trademark of the American Mathematical Society.

Only the literate were in a position to concern themselves  
greatly with what would happen when the year  
dccccclxxxviiiij became a simple m, . . .

Robert Lacey and Danny Danziger  
*The Year 1000: What life was like  
at the turn of the first millennium*  
(1999)

# TUGBOAT

COMMUNICATIONS OF THE T<sub>E</sub>X USERS GROUP  
EDITOR BARBARA BEETON

VOLUME 20, NUMBER 4  
PORTLAND

•  
OREGON

DECEMBER 1999  
• U.S.A.

## **TUGboat**

During 1999, the communications of the T<sub>E</sub>X Users Group will be published in four issues. The September issue (Vol. 20, No. 3) contains the Proceedings of the 1999 TUG Annual Meeting.

*TUGboat* is distributed as a benefit of membership to all members.

Submissions to *TUGboat* are reviewed by volunteers and checked by the Editor before publication. However, the authors are still assumed to be the experts. Questions regarding content or accuracy should therefore be directed to the authors, with an information copy to the Editor.

## **Submitting Items for Publication**

The next regular issue will be Vol. 21, No. 1. Production and mailing have been delayed; the first issue for 2000 is expected by July, and the second should be sent to the printer before the annual meeting. Deadlines for future issues are listed in the Calendar, page 399.

Manuscripts should be submitted to a member of the *TUGboat* Editorial Board. Articles of general interest, those not covered by any of the editorial departments listed, and all items submitted on magnetic media or as camera-ready copy should be addressed to the Editor, Barbara Beeton, or to the Production Manager, Mimi Burbank (see addresses on p. 339).

Contributions in electronic form are encouraged, via electronic mail, on diskette, or made available for the Editor to retrieve by anonymous FTP; contributions in the form of camera copy are also accepted. The *TUGboat* “style files”, for use with either plain T<sub>E</sub>X or L<sup>A</sup>T<sub>E</sub>X, are available “on all good archives”. For authors who have no network FTP access, they will be sent on request; please specify which is preferred. Send e-mail to [TUGboat@tug.org](mailto:TUGboat@tug.org), or write or call the TUG office.

This is also the preferred address for submitting contributions via electronic mail.

## **Reviewers**

Additional reviewers are needed, to assist in checking new articles for completeness, accuracy, and presentation. Volunteers are invited to submit their names and interests for consideration; write to [TUGboat@tug.org](mailto:TUGboat@tug.org) or to the Editor, Barbara Beeton (see address on p. 339).

## **TUGboat Advertising and Mailing Lists**

For information about advertising rates, publication schedules or the purchase of TUG mailing lists, write or call the TUG office.

## **TUGboat Editorial Board**

Barbara Beeton, *Editor*  
Mimi Burbank, *Production Manager*  
Victor Eijkhout, *Associate Editor, Macros*  
Jeremy Gibbons, *Associate Editor*,  
*“Hey — it works!”*  
Alan Hoenig, *Associate Editor, Fonts*  
Christina Thiele, *Associate Editor*,  
*Topics in the Humanities*

## **Production Team:**

Barbara Beeton, Mimi Burbank (Manager), Robin Fairbairns, Michael Sofka, Christina Thiele

*See page 339 for addresses.*

## **Other TUG Publications**

TUG publishes the series *T<sub>E</sub>Xniques*, in which have appeared reference materials and user manuals for macro packages and T<sub>E</sub>X-related software, as well as the Proceedings of the 1987 and 1988 Annual Meetings. Other publications on T<sub>E</sub>Xnical subjects also appear from time to time.

TUG is interested in considering additional manuscripts for publication. These might include manuals, instructional materials, documentation, or works on any other topic that might be useful to the T<sub>E</sub>X community in general. Provision can be made for including macro packages or software in computer-readable form. If you have any such items or know of any that you would like considered for publication, send the information to the attention of the Publications Committee at [tug-pub@tug.org](mailto:tug-pub@tug.org) or in care of the TUG office.

## **Trademarks**

Many trademarked names appear in the pages of *TUGboat*. If there is any question about whether a name is or is not a trademark, prudence dictates that it should be treated as if it is. The following list of trademarks which appear in this issue may not be complete.

MS/DOS is a trademark of Microsoft Corporation  
METAFONT is a trademark of Addison-Wesley Inc.  
PC T<sub>E</sub>X is a registered trademark of Personal T<sub>E</sub>X, Inc.

PostScript is a trademark of Adobe Systems, Inc.  
techexplorer is a trademark of IBM Research.  
T<sub>E</sub>X and  $\mathcal{A}\mathcal{M}\mathcal{S}$ -T<sub>E</sub>X are trademarks of the American Mathematical Society.

*Textures* is a trademark of Blue Sky Research.

UNIX is a registered trademark of X/Open Co. Ltd.



## General Delivery

### From the President

Mimi L. Jett

Happy New Year!

Or shall we say New Decade? Some have argued that we are not *really* at the beginning of a new century or millennium, but simply in the last year of the former. Not so with TUG. We are clearly coming into the first year of our third decade, quite an accomplishment for a users group built around a typesetting language that was written at the onset of the digital revolution. The future looks bright for TUG as our growth and stability continue on an upward path. With a score of years comes some level of wisdom and maturity, enough at least to carry us comfortably forward.

Before I leave the subject of the new calendar, may I remind you that TUG has an official Y2K statement written by Barbara Beeton for *TUGboat* 20, no. 1, p. 45. To summarize, we do not expect anyone to experience life-altering trauma from using T<sub>E</sub>X before or after January 1, 2000. You may still choose to fill the basement with bottled water or place a time capsule in concrete, but T<sub>E</sub>X should be the least of your worries.

The second half of 1999 was eventful, with the 20<sup>th</sup> annual meeting in August in Vancouver BC and EuroT<sub>E</sub>X in Heidelberg just a month later. Both conferences were charged with the energy of new discovery and the joy of sharing. If you were not able to attend either of these gatherings, I encourage you to read the proceedings<sup>1</sup> to take in the breadth and depth of new development happening around T<sub>E</sub>X. One vital project is the NTS initiative. Originally started in 1991, this collaboration of great thinkers has truly developed a New Typesetting System that provides the things we wanted from T<sub>E</sub>X but couldn't have. The presentation in Heidelberg by Joachim Lammarsch, Jiří Zlatuška, Hans Hagen, Philip Taylor, and notably Karel Skoupý, laid out the history, emotions, logic and finally the resulting success: A complete T<sub>E</sub>X implementation running as object based code with Java. Tests showed exact matching files generated through NTS and "old fashioned" T<sub>E</sub>X. The importance of NTS is the power and extensibility it brings. The entire project has been funded through generous donations from DANTE, UKTUG, NTG, CSTUG, and others. I am happy to announce that the TUG Board of Directors has shown interest in contributing to this important

project and want to encourage you and your companies or schools to help as well. All donations are appreciated; please contact [zlatuska@muni.cz](mailto:zlatuska@muni.cz) to pledge your support. A small investment today will ensure better tools tomorrow.

Speaking of generosity, my mind is drawn to Germany, specifically to the community we know as DANTE. Here is a group of people always ready to help and support those in need, regardless of distance or other barriers. The latest gesture of goodwill from our friends was the donation of an UltraSparc server for use by the T<sub>E</sub>X Live development team. In addition to the computer, DANTE arranged for space and access through the University of Mainz, with system administration provided by Rainer Schoepf. Thank you very much to everyone who contributed to this wonderful outcome. The T<sub>E</sub>X Live CD is one of the premier benefits of TUG membership; we rely on the help of many people to make it possible. This gift from DANTE will be appreciated around the globe.

Looking forward into the year we have our 21<sup>st</sup> annual meeting TUG 2000 to be held at Oxford (UK) in Wadham College, a lovely place with rich history, from Saturday, August 12 through Friday, August 18. Bound to be one of the most memorable meetings yet, not only for the incredible venue, but also the cast of characters is known for high entertainment value. Remember the Poetry Reading in Vancouver?

In our tradition, when the annual meeting is on Europe or another continent, any meeting held in North America will be geared toward training and workshops in order to serve the needs of our members without distracting from the important annual conference. This coming year we are planning a 3-4 day workshop for beginners and intermediate users of T<sub>E</sub>X and associated tools. Once the venue is established we will send an email announcement to all North American members and post it on our website.

As we close this milestone year and prepare to open the door to the next, I wish to express my gratitude to everyone in our vast and virtual community for the work you are doing for the good of all. Every innovation shared, every patient explanation, and all the humility and kindness makes this a very nice place to be. Thank you, everyone. Happy New Year!

◇ Mimi L. Jett  
IBM  
T. J. Watson Research Center  
P.O. Box 218  
Yorktown Heights, NY 10598  
[jett@us.ibm.com](mailto:jett@us.ibm.com)

<sup>1</sup> *TUGboat* 20:3 and EuroT<sub>E</sub>X'99 books are available from the TUG office.

---

## Editorial Comments

Barbara Beeton

### On being a fossil

It was with considerable surprise that I read the kind words from several TUG presidents, past and present, in the last issue, recognizing my survival in TUG for 20 years.

The proceedings issue of *TUGboat* is put together by the Proceedings Editor (Christina Thiele, on this occasion), with the ever-present support of Mimi Burbank, the Production Manager; I get to relax a bit, although I do try to help out with copyediting and proofreading. The proceedings team does keep their Editor informed about what is being included, the page count, and other important matters. But this time they snuck this little piece—three whole pages—right past me.

It has been a truly satisfying experience to watch TUG develop through the years, and the most satisfying part has been to meet and work with so many fine people. To the authors of the tributes—Pierre MacKay, Nelson Beebe, Christina Thiele, Michel Goossens and Mimi Jett—I can only say a heartfelt “thank you”, and I’m pleased and proud to be able to count them among my friends.

To the perpetrators of this act, I will say “thank you” as well, and you guys need a good proofreader... (Cincinnati has only one “t”, you know).

### Erratum: Mimi Jett’s term of office

In the above-mentioned article, Mimi Jett’s term of office was given as 1997–2003 (pp. 313 and 315). Not quite; the term extends only through the annual meeting in 2001.

The term of the President, unlike the four-year Board terms, is only two years. Mimi’s first term was 1997–1999, and she chose to seek a second term, through 2001. There is no limit placed by the Bylaws on the number of terms any officer may hold, so if Mimi is not completely worn out by the summer of 2001, it *is* possible to extend that date, but we won’t force her into it.

### Gutenberg: the man of the millennium

As a year ends, the pundits consult their accumulated wisdom and name the “Man of the Year”. As the final digit of the year turns to zero, we are given the “Man of the Decade”, or of the century. At this year end, a greater span is upon us, and the designated “Man of the Millennium” has special meaning for us descendants of ink-stained

wretches—Johannes Gutenberg, purported inventor of movable type, has been ranked first among the men and women who shaped the past thousand years.

The use of movable type and the printing press to make books and other printed material accessible to ordinary people, not just the rich and high-born, wrought a basic change in the way knowledge was communicated and preserved. It can even be claimed that all later forms of communication have evolved from this development. Victor Hugo wrote, in 1831, “The invention of printing is the greatest event in history. It is the mother of all revolutions. . . . Thoughts, once printed, are indelible, liberated, impregnable and indestructible. They soar like a flock of birds, disperse to the four winds, and are everywhere at the same time.”

For more information on Gutenberg and his invention, see the Web page at <http://www.germanembassyottawa.org/news/whatsnew/bulletins/1999-07-09.0001.html>.

The University of Göttingen is preparing a CD-ROM, “Gutenberg digital”, containing a scanned reproduction of the Bible along with other related material. For a preview, visit <http://www.gutenbergdigital/vorversion/>.

This is quite a special treat.

### Sebastian Rahtz leaves the *TUGboat* production team

For five long years, Sebastian Rahtz has been an active and valued worker on the *TUGboat* production team. It is with real sadness that we see him go. He will not disappear from the TUG and  $\TeX$  scene—his current projects are compilation of  $\TeX$  Live 5 (he has also been the driving force behind the first four editions) and TUG 2000 in Oxford.

Sebastian, thank you, thank you, thank you! Your competence and good sense will be sorely missed.

### International news: Greek, Russian and Vietnamese groups

The third issue of “Eutupon”, the newsletter of the Greek  $\TeX$  Friends Group, is available to be downloaded from their Web site: <http://obelix.ee.duth.gr/eft/english/eutupon.html>. Several articles are in English, for those who don’t read Greek; but even if you don’t, the design of the newsletter is well worth a look.

CyrTUG, the Russian-speaking group, has inaugurated a  $\TeX$  discussion list to be carried on in Russian. To join the list, send e-mail to [cyr tug-subscribe@vsu.ru](mailto:cyr tug-subscribe@vsu.ru).

A new T<sub>E</sub>X group for Vietnamese users has been formed. Visit their home page at <http://iris.ltas.ulg.ac.be/vietttug>.

All the T<sub>E</sub>X groups we know about are listed on the TUG page [www.tug.org/lugs.html](http://www.tug.org/lugs.html). The president or a convenient contact is listed for each, along with a link to the group's home page, if one exists. If you learn of any new group that hasn't yet been listed, please do let us know; send a message to [webmaster@tug.org](mailto:webmaster@tug.org) and to [office@tug.org](mailto:office@tug.org) with an address of someone to contact for more information.

### Clarification of the CTAN “nonfree” classification

The June issue of *TUGboat* carried a note by Bernard Gaulle entitled “The french package on and off CTAN” (pp. 91–92); this was accompanied by editorial commentary regarding the purpose of the distinction between “free” and “nonfree”. There seems to have been some misunderstanding of precisely what is meant.

The categorization “nonfree” in CTAN terms means nothing more or less than that the license accompanying a package *limits redistribution or use* of the package.

It is the desire of the compilers of the T<sub>E</sub>X Live CD-ROM to include only files that can be redistributed without any restrictions, so that the mechanics of distribution do not create any undue administrative burden. (For T<sub>E</sub>X Live 4, this was evidenced in the TUG office by the necessity of telling some prospective purchasers that it was not permitted to distribute multiple copies, even for such worthwhile purposes as distributing them free to a computer science class.) The CTAN management have chosen to assist in this effort by making clear, by location in the archive, the status of all relevant material. Thus T<sub>E</sub>X Live 5, which will accompany the first 2000 issue of *TUGboat*, will be free of any restrictions, and can be supplied to anyone wanting a copy, unlike T<sub>E</sub>X Live 4, which could be distributed only to members of T<sub>E</sub>X user groups.

### The origin of the @ sign

A delightful, well-researched and -written article on the origin of the @ sign, by Michael J. McCarthy, appeared on the front page of the November 16 issue of the *Wall Street Journal*.

I thought the subject matter would be of interest to *TUGboat* readers, so I tried to obtain permission to republish it. This has turned out to be not a simple task, and ultimately it has proved impossible. The *Journal* requires payment for any republication; the fee requested was \$250. In the commercial world, that would be a mere token, but it is beyond *TUGboat*'s (nonexistent) budget for such items.

Robin Laakso at the TUG office tried her best to get the fee reduced, but to no avail. Thanks to her for a gallant attempt.

If you *can* find a copy of the November 16 *Journal*, do read the article. You will certainly enjoy it.

### Communication by flags

There are several flag codes still in use in nautical circles. One is described in this issue in an article on the semaphore flag code. Another system, the international flag code, contains a flag for each letter of the Latin alphabet and for each digit, with a few additional flags for good measure. These flags are brightly colored, with easily recognizable patterns; in addition to its meaning as a letter or digit, each flag is also assigned a meaning of a specific word or phrase.

One of these flags, the one representing the letter Z, has particular significance to our group; the extended meaning of this flag is “require a tug”. A drawing by Duane Bibby was commissioned by the 1999 TUG annual meeting committee to denote a “T<sub>E</sub>X Friendly Zone”.

The drawing, with the flag in gorgeous color, can be downloaded from the TUG Web site: [http://www.tug.org/publicity/tfz\\_master/](http://www.tug.org/publicity/tfz_master/).

Retrieve a copy, print it out, and hang it over your desk to proclaim that *you* inhabit a

T<sub>E</sub>X Friendly Zone.

◇ Barbara Beeton  
American Mathematical Society  
P. O. Box 6248  
Providence, RI 02940 USA  
[bnb@ams.org](mailto:bnb@ams.org)



## Typography

### Typographers' Inn

Peter Flynn  
University College Cork

### Reversed quotes

I am deeply indebted to the many kind people on the TYPO-L mailing list who responded to my query about 'reversed quotes' and where they came from (see the list archives). It's clear that they *do* exist as a special feature in some DTP packages, and that they *are* used out of a desire for a much misunderstood 'symmetry'.

I still think they are an unnecessary and pointless device, but like the misplaced apostrophe, probably impossible to stop. When I become dictator of the planet I'll be able to act on this:<sup>1</sup> until then I trust in your judgment to point out the evils of reversed quotes to anyone who uses them.

### Vulcan

This package, for which I presented a development report in Vancouver, has been having deep surgery. In the months between the start of work and the TUG meeting, half a dozen other packages were announced or upgraded which tackled—individually—several of the tasks *vulcan* was designed to deal with.

This is a twofold boost: it means that at least some other folk in the world thought seriously enough of the problems to want to solve them themselves; and it means we can now ditch our half-written or wholly-written patches and just `\RequirePackage` the relevant tools: an unintentional triumph of modular development! On the downside it means unravelling the developmental code and checking that none of it affects other parts.

All of which is by way of excusing the late non-appearance of the package on CTAN. The comments I got in Vancouver have led to a fairly deep-seated rethink of the way metadata was being handled in the titling—being one of the biggest parts of the package—and it's now being redone in a much more modular fashion.

**WinWord** Among the package options for *vulcan* which *didn't* get much of a mention in Vancouver there suddenly appeared a new one last April 1st:

---

<sup>1</sup> And on other important matters like the abolition of brussels sprouts, accountants (with a few honorable exceptions), and line dancing.

*winword*. This is something I had threatened to do quite separately a couple of years ago, and which I mentioned on `comp.text.tex` at some stage, but I hadn't expected a colleague to ask about it on that particular day.

After some discussion in Vancouver I decided to make this objectionable little item into a package. It invokes something akin to `pslatex` with the resizing of Helvetica and Courier to match the Times x-height; it implements the most basic built-in styles of `normal.dot`, including the wonky spacing on lists and the horrible superior ordinals; and it turns off hyphenation and justification. For good measure it reintroduces the notorious bug in the vertical spacing of the first lines of paragraphs, which you probably thought you'd seen the last of.

The net effect is to allow  $\LaTeX$  users to continue to manage their documents with  $\LaTeX$ , but to generate printout and PDF files that look exactly like all the other junk that comes out of wordprocessors. As I mentioned before, it was originally intended to help a former colleague who was trying to introduce  $\LaTeX$  by stealth in his organisation, and needed a temporary cover to avoid being the odd one out at meetings who submitted documents with 'that  $\TeX$  look' (incidentally another of the reasons for the origins of *vulcan*).

This way, management would remain unaware that he was subversively ferretting  $\LaTeX$  into the company, because his documents looked like everyone else's, but he could still reformat into a better style at some future date simply by replacing the `\usepackage` command.

So far I have used it for short reports and similar office documents which I circulate as PDF files (my refusal to send or accept *Word* attachments is notorious), and no-one has raised an eyebrow as they presumably believe it to have been done in *Word*.

### An academic 'How-To'

As part of a new drive to make life easier for postgraduates doing theses and academics writing articles and books in my institution, I have been concocting a document on how to use  $\LaTeX$  for academic writing. This is intended as an introductory text to complement *The Not So Short Introduction to  $\LaTeX$  2<sub>ε</sub>* [1], which is excellent for users in mathematics and the natural sciences but covers too much technical ground for users in the humanities, which is an area of expansion for  $\LaTeX$  right now. I have new thesis and letter style files to go with it (actually *vulcan* partly grew out of the need to do this, too).

However, there are two major holdups: *a*) the chapter on fonts needs detailed samples to go with instructions on how to get additional fonts working; and *b*) the chapter on cross-references needs a usable set of routines and macros for humanities-style bibliographic citations.

**‘Free’ fonts** The font sampler is close to finished, as recent readers of `comp.text.tex` may have noticed from my occasional grumps about fonts which are supposed to work but don’t. On one side of the page I have listed the common everyday typefaces that by now (I hope) come with every  $\text{\LaTeX}/\text{dvips}$  installation. On the other side I list most (perhaps not all) the other free fonts that are on CTAN, mostly in METAFONT format. I exclude any for which outlines or bitmaps are not freely available.

- Computer Modern with all its common variants (including Dunhill and Fibonacci but excluding Funny Font, for example).
- the ten typefaces representing the Adobe original ‘LaserWriter 35’,<sup>2</sup> shrinking Avant Garde and growing Chancery a little in order to compensate for the widely differing x-heights. It’s surprising how much this difference otherwise obtrudes when you stick all the examples close together like this. The alternative (leaving the 10pt samples as they are) is more ‘correct’ but in these circumstances would lead to more questions being asked before the reader is equipped to deal with them.
- the X Consortium members’ fonts (Utopia, Charter, Nimbus, Antiqua, and Grotesk). On my system (Red Hat Linux, using the default  $\text{\TeX}$  package to which I have of course added hugely over time) all these were broken as installed, even Charter. In all cases the `dvips` entry was for a PFA file and an encoding which didn’t exist, which in most cases didn’t match the `.fd` file created by `fontinst`, not did it match the `.tfm` and `.vf` files. I don’t know how this happened, but I’m glad I found it before the users did.
- the METAFONT fonts Pandora, Universal, and Concrete. I’d like to include more METAFONT text faces but space on one side of A4 is limited.
- In at the bottom come examples of Concrete and Euler math, done as EPS file because I haven’t figured out how to have three different

math encodings available in one file, nor am I sure I want to.

I have used the phrase ‘Typographia Ars Artium Omnium Conservatrix’<sup>3</sup> for the samples on this page, rather than strict alphabets, but for the other typefaces, a quotation relevant to their usage domain would be appropriate (so I’m open to suggestions for a phrase to illustrate Punk, for example).

On the other side comes the fun, therefore. I’m not fond of pigeon-holing typefaces for purposes of usage beyond simple classifications like serif and sans-serif. For typographic taxonomy, of course, there are many useful ways of highlighting the differences, but they would be inappropriate in a document aimed at beginners. I have settled for dividing what is available into ‘collections’, bearing in mind that scholars approaching the use of  $\text{\LaTeX}$  are likely to be most interested in what is useful for their field. The classification is not quite arbitrary but it will do until I find something better, and it’s incomplete because as I write I’m still working through CTAN...

**Germanic** Yannis Haralambous’ elegant Fraktur, Schwabacher, Gothic, and Decorative Initials all worked fine, as I would have expected, and I’ve pinched a bit of Goethe and a bit of Caxton for the sample text.

**Historical** Ditto Peter Wilson’s fine archaic fonts, for which he has kindly updated the Ugaritic (cuneiform). I’ve added the sean-chló (old Irish) EIAD font in this collection, along with Rustic, Bard, Uncial, and Ogham. These required some surgery to the `.fd` and `.mf` files, which was one of my chief problems, not being well versed in METAFONT. I even added Tengwar, which I managed to coax into action some years ago, with some homebrewed macros which let me typeset a readable transliteration of ‘Ash nazg durbatulúk...’.

**Symbols** The problems start when you try to use the symbol fonts like chess, backgammon, astronomical signs, cartography symbols, the BB dingbats, etc. Many of these fonts were designed for use with plain  $\text{\TeX}$  which loads fonts slightly differently from  $\text{\LaTeX}$ . Because I was being a good little boy and using or writing `.fd` files for each font, they were being loaded at an explicit 10pt regardless of whether or not the METAFONT code specified 10pt. As METAFONT’s default design size is 128pt I was getting microscopically small characters, and no amount of fiddling with the METAFONT code

<sup>2</sup> Without showing variants, just mentioning they exist — this is not for typographers: if readers haven’t grasped by this stage that italics slope to the right and that bold is blacker and heavier than roman then I need to hear from them.

<sup>3</sup> ‘Printing, the art which preserves all others’



**Bibliographics** Humanities bibliographic styles are significantly different from those used in the natural sciences. I mentioned what I wanted some while ago (in-footnote citations with author, title, year, and page range; with full citations unnumbered at the end). I couldn't find any package to do them, but I was pointed at `camel`. This is a prototype for doing a huge amount of stuff with citations, and looked very useful. It implements legal citations as well as appearing to handle most of what I wanted, but at the time it was alpha software and it conflicted with a number of other things I was trying to do.

After a recent brief but understandable diversion to investigate the interestingly named `humanbio` (which turned out to be nothing whatever to do with the humanities, but a package to format references for the journal *Human Biology*), I have ended up writing a spec for what is needed, and I guess I'll go hunting for someone who can program `.bst` files.

## References

- [1] Tobias Oetiker, Hubert Partl, Irena Hyna, and Elisabeth Schlegl. The Not So Short Introduction to  $\text{\LaTeX} 2_{\epsilon}$ . Technical report, CTAN, <http://www.tex.ac.uk/tex-archive/info/lshort/english>, Feb 2000. Web document.

◇ Peter Flynn  
University College Cork  
Computer Centre, University  
College, Cork, Ireland  
[pflynn@imbolc.ucc.ie](mailto:pflynn@imbolc.ucc.ie)  
<http://imbolc.ucc.ie/~pflynn>

different from those written in METAFONT usually bring some problems with compatibility or platform dependence. The most common problem is the need for manual font generation at a specific resolution, demanded by T<sub>E</sub>X, or you need a special device driver for this purpose.

An “easy” solution of all these problems is a conversion of a font into METAFONT format. Once the font is converted, you can use it the same way as regular METAFONT fonts. As long as I wanted to use some TTFs in T<sub>E</sub>X some time ago and I didn’t find any converter, I decided to write my own.

Let’s look at the TrueType fonts first. Each character is described by its outline, composed of Bezier curves. Some information used for scaling is also included. Currently, the converter reads only the glyph information for a character. The glyph consists of several closed paths. All paths are filled using invert-filter, i.e., the area filled twice will not be filled at all. These paths should not cross themselves, but, as long as the Windows OS doesn’t care about that, some fonts are not drawn properly. This causes problems in METAFONT, which treats this as an error in the input file. (Actually, this error can occur only when there is a crossing on a single path so that a “loop” comes up. The paths are processed independently by METAFONT, so crossing of two paths should not cause problems.) Because of this representation of TrueType fonts, the conversion program also generates a set of Bezier curves forming closed paths.

There is also another kind of glyphs, composed glyphs. These are specified as sets of several other glyphs, which are transformed and joined together. Many accented letters are stored in this way in TrueType fonts.

Different styles of the same font, like bold or italics, are usually stored as different TTF files. There are several possible encodings for use in TrueType fonts, but the one most widely used is UNICODE. Of course not all characters are included in a TTF file; those unused are mapped to a default “warning” glyph, usually an empty square. As long as T<sub>E</sub>X uses only 256 characters in a single font file, it might be desirable to create several MF files.

Now to the conversion program. The first version was based on “The FREE TrueType Font Engine” written by David Turner, but, because of some limits of this library (e.g., a 64KB memory limit), a new one was created, according to the specification for TrueType Fonts [1].

The conversion itself starts by loading all glyphs from the TTF file. All characters we want to export are specified in a single text-file. This file

## Font Forum

### TrueType Fonts in T<sub>E</sub>X

Vladimír Koutný

TrueType fonts are widely used these days; unfortunately, they are not supported by many non-Windows programs, like T<sub>E</sub>X. Generally, using fonts

An earlier version of this article appeared in *Zpravodaj* (the Bulletin of the Czech-Slovak T<sub>E</sub>X Users Group) **99/3**, pp.159–160, in Slovak, with the title “TrueType-fonty v T<sub>E</sub>Xu”; it has been translated by the author, and appears here with permission.

specifies the UNICODE number of each character, together with a character number that will be used in METAFONT. During the conversion, the program requires some free space in the working directory for temporary files, that are automatically removed at the end. Also, the output file will be written here, with the name of the original TTF file, but with an MF extension. Because of this, you should have write access to the working directory. The METAFONT output routines were designed by my schoolmate, Rišo Kráľovič.

The newly created font file can be used directly in T<sub>E</sub>X, maybe preceded by manually running METAFONT to create a TFM file. Unfortunately, there are some cases when METAFONT claims errors. The most common is the “Strange path (turning number is zero)” — this is the error mentioned above, caused by “loops” in the outline. Another reason for this error is the small size of the font, together with the resolution used for generation. This happens mostly with decorative fonts with many details. Possible solutions: use such a font in bigger size, or use higher resolution.

There are two more aspects we could look at. The first one is the kerning. In most TrueType fonts, there are stored kerning values. Normally, kerning is not supported by Windows, just some specific programs are capable of doing so. In T<sub>E</sub>X, the kerning is something natural, so this kerning information is quite useful. Therefore, the conversion generates also ligtables with kerning pairs.

Next, there are usually some ligature characters in a TTF font. In most fonts there are the fi (UNICODE 0xFB01) and fl (UNICODE 0xFB02) glyphs, as well as some others, like Œ, etc. There might also be still others, but the problem is that a regular TrueType font does not contain any information about ligatures. So if you know that this glyph is a ligature xy, then you can use it and write this ligature by hand into the MF file, but automatic generation of this kind of ligtables is not working yet.

There is also an extension to TrueType fonts, called TrueType Open [2]. This extension brings support for ligatures and also some other improvements, mostly useful for vertical, right-to-left and similar fonts. These fonts are usually treated as regular TrueType fonts, ignoring additional information. Unfortunately, only a few latin fonts are in this Open format; I’ve actually found only one (Tahoma; maybe it is because this font also contains Greek, Hebrew and Thai characters).

Another possible upgrade of this converter includes creating a somewhat more user-friendly interface, that will allow the user to select desired

characters interactively by shape, not only by UNICODE number, together with the possibility to enter ligature information.

The latest version of the converter can be found at <http://www.ksp.sk/textools>, or at <http://www.linxee.sk/ttf>. Feel free to report any bugs, comments or suggestions concerning the problems described here to my address [kluka@hotmail.com](mailto:kluka@hotmail.com).

## References

- [1] TrueType 1.0 Font Files, Technical Specification, Revision 1.66, November 1995, Microsoft
- [2] TrueType Open Font Specification, Version 1.0, July 1995, Microsoft

◇ Vladimír Koutný  
 Osuského 7  
 851 03 Bratislava, Slovakia  
[kluka@hotmail.com](mailto:kluka@hotmail.com)

---

## The Semaphore Alphabet

Vít Zýka

### History

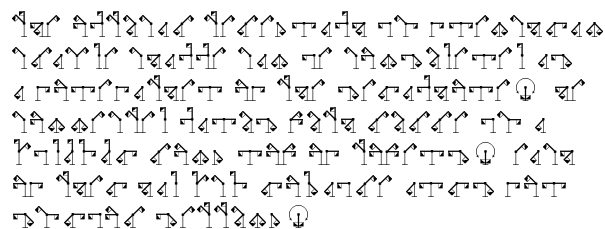
If a captain wanted to give a passing ship some navigation information, a message about an emergency, or a report about countries visited, he used to send a specialist in signalling to a good visible place. This person took two red-yellow flags, one in each hand, and sent a message using agreed flag configurations. One of seven possible positions for each arm meant a message character. Words and sentences were separated for better understanding by waving the flags once or twice in a circle, respectively. This enabled transmission of more general text than by the widely used signal flags (where a flag has a predetermined meaning of a word or a whole sentence), and is faster than using the Morse alphabet. Among the necessary skills of the receiver was surely

---

This article originally appeared in *Zpravodaj* (the Bulletin of the Czech-Slovak T<sub>E</sub>X Users Group) **99/3**, pp.157–158, in Czech, with the title “Semaforová abeceda”; it has been translated by the author, and is published here with permission.

(aside from excellent reading — which was not usual considering the level of sailors’ literacy) hawk eyes. To distinguish the flag position up to a distance of 7 kilometers on a swaying ship was very difficult.

Semaphore was limited to visible transmission only. With the development of electrical communication, the Morse alphabet superseded it for the majority of applications. Try to send a semaphore character by telegraph! It is not impossible, but the effective result is far inferior to the result achieved nowadays in the time of computers and Internet. While semaphore helped the captain to send and receive news, nowadays using this alphabet will probably make communication more difficult. Consider this:






*(The optical telegraph ... symbol setting.)*

The contemporary semaphore alphabet was suggested by an English Army colonel in 1822. But it was used for only a very short time in Britain. More use was made of it by the U.S. Navy during the Civil War.

**The Alphabet Code**

Seven positions of arms together give a combination of 28 signs. The basic alphabet set, which contains 26 characters, is split into six groups, called circles, in nearly alphabetical order. A circle is a group of signs for which the flag in the right hand has the same position.

There is no space in the code for digits, so they are signalled as the first alphabetic characters, but in front of the first digit is placed the sign ‘digit beginning’ , and following the last digit ‘digit ending’ . The second special sign for which there is a free space in the code is the sign of a ‘mistake’ . Its function is similar to the backspace key. The other signs, e.g. punctuation, are not contained in the semaphore and have to be communicated by words.

**Alphabet Usage**

The font ‘semaf’ was created by METAFONT. It consists of four shapes in three variants; see table. You can easily add a new font variant if you know METAFONT at a basic level. The only thing you need to do is set the font. You will not see some special characters in output — only the upper- and






































lowercase letters, digits, space, period, and two special semaphore signs are included. If you need e.g. ‘!’, you should define:

```
\def\!{} \catcode\!=13
\def!{ exclamation mark }
% or \def!{{\tenrm \!}}
% or in LaTeX: \def!{{\normalfont \!}}
```

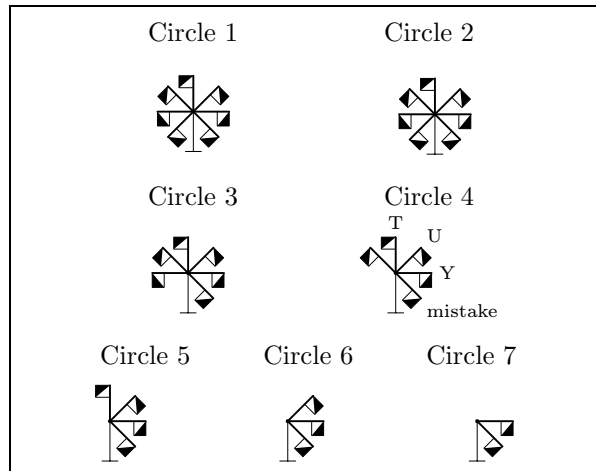
To include the semaphore font you can use the pre-prepared files `semaf.tex` or `semaf.fd` for plain  $\TeX$  or  $\LaTeX 2_{\epsilon}$  users, respectively. The font covers the IL2 coding table (`il2semaf.fd`) that doesn’t differ from Knuth’s OT1 coding in the seven low bits.

A concept of generalized ligatures enables an elegant solution of the digit typesetting. Beginning and ending digit signs are therefore included automatically, even without the need to write macros.

**Semaphore Alphabet Code**

A		O		W	
B		P		X	
C		Q		1	
D		R		2	
E		S		3	
F		T		4	
G		U		5	
H		Y		6	
I		J		7	
K		V		8	
L				9	
M		Z		0	
N				mistake	

**Semaphore Logical Arrangement**





---

**Semaphore Font Variants**

Variant		Roman r	Bold bf
Pillar	smf	⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮	⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮
Empty	smfe	⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮	⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮
Person	smfp	⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮	⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮
Variant		Monospace tt	Slanted sl
Pillar	smf	⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮	⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮
Empty	smfe	⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮	⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮
Person	smfp	⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮	⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮

The most important criterion of a good font is its legibility. It is true that this is too low in the case of the semaphore. In spite of this there is at least one area of usage: semaphore was incorporated into scout-life for boys and girls, together with its romantic background of sailors, to improve memory, perceptions, and coordination.

The font is available from CTAN and at:

<http://cmp.felk.cvut.cz/~zyka/zykatex.html>

- ◊ Vít Zýka  
Czech Technical University  
Faculty of Electrical Engineering  
Department of Cybernetics  
Center for Machine Perception  
Praha 2, 121 35, Czech Republic  
[zyka@cmp.felk.cvut.cz](mailto:zyka@cmp.felk.cvut.cz)  
<http://cmp.felk.cvut.cz/~zyka/zykatex.html>

and typesetting files into XML that adheres to your new schema. Now what?

This article describes a path that has many pieces that must fit together exactly. That's the down-side. The up-side, however, is a very powerful XML-to-paper path that will not cost you a penny, and runs on any platform that runs Java.

— — \* — —

So, you've gotten your valuable information assets described in terms of XML schemas (either DTD or some other form of schema), and you've taken the painful step of converting your information from word processors and typesetting files into XML that adheres to your new schema. Now what?

This article describes a path that has many pieces that must fit together exactly. That's the down-side. The up-side, however, is a very powerful XML-to-paper path that will not cost you a penny, and runs on any platform that runs Java.

As part of the information analysis phase of your project, you probably went through the task of looking for information hidden in your existing documents. This meant taking text that was in italic, for example, and tagging it as an emphasized phrase, foreign term, bibliographic reference, or legal citation. Your new XML-tagged content is rich in self-describing information objects, but contains no information about how to format those objects. For example, indicating to a formatting engine that a string of text is to be rendered in an italic font causes something to happen. That is, the formatting engine changes the font characteristics for the duration of the italicized phrase. However, tell a formatting engine to render something in "foreign phrase", and it will probably have a problem.

That is because there is an important piece missing. Sure, we know it is a phrase expressed in a language other than the default language being used in the current document, but we don't have information about how it is to be rendered on paper. We need some kind of mapping to translate from "foreign phrase" to "italic". What's more, we may even need to do more processing on the object. For example, we might want to collect all foreign phrases in the document, along with their translations in an appendix. We have even more opportunities for further processing when rendering the information in electronic form. For example, when creating an HTML rendering of the document, we may want our foreign phrase to be underlined and linked to a pop-up window with the translation. Or, we may want to build a system that causes a speech synthesizer to speak the word in its native tongue.

## Software & Tools

### The Paper Path: XML to paper using TeXML

Brian E. Travis

#### Abstract

So, you've gotten your valuable information assets described in terms of XML schemas (either DTD or some other form of schema), and you've taken the painful step of converting your information from word processors

This is only one example of processing a single element in many different ways based on the desires for information delivery.

So, how do we make the leap from “foreign phrase” to an italicized string of text with the translation collected in an appendix? Add to that the other several dozen elements that need to be translated into some kind of deliverable.

### Pagination Nation

The first thing to consider when putting information on paper is the page itself. This process is called “pagination”. Everything in your content must be rendered somehow on a two-dimensional frame bounded by the physics of the real world. A page contains a body area where the rendered text sits, plus a margin area where navigational information goes.

In the body of a page, there are blocks of text that have been rendered using centuries-old techniques. First, each line is set with text until it reaches an acceptable length, at which point the line is ended and the next word starts a new line. If the word doesn’t reach the acceptable range, and the next word causes the right margin to be overrun, the last word must be broken at a place that follows the rules of hyphenation.

Another task of the pagination program is making sure each line ends at the same place. This is called “justification”, and is preferred by some designers to make the page look symmetrical. This requires the typesetter to calculate the space left over at the end of a line, divide it by the number of spaces between words of that line, and add this new increment to each space.

These two processes are lumped together in the typesetting lingo as “hyphenation and justification”, or “H&J”. H&J is a basic requisite of any typesetting program, and all programs, from free to \$100K+ systems provide this service. Higher-end typesetters will also do a sophisticated analysis of the page after it is set in memory. One thing such typesetters look for is spaces between words that line up from one line to the next. Putting too many of these spaces in a row vertically causes an effect known as “rivers”, that might be distracting to the reader. Another check these high-end devices perform is hyphenation analysis. Some page designers don’t like to see more than two hyphenated lines in a row. In order to avoid this, the typesetter may need to reset the page many different times, using different word- and character-spacing values in order to eliminate multiple hyphens. Speaking of hyphens, some typesetters check to make sure there is no hyphenation between pages or columns. This is

something my third-grade grammar teacher, Miss Blankenship, would not tolerate.

Once the body area of a page is set, certain navigational features are placed on the page. The most common is the page number. The typesetter must keep track of the page number, and provide an incremental indicator on each page. This is more difficult than it looks, once you consider the many different ways pages can be numbered.

Running headers and running footers provide further navigational aids to the reader, and give the document designer a place to show off. Running headers usually provide some kind of indication of the title of the chapter, and maybe even the name of the document. Another type of running head is called a “dictionary header”. This is a header that changes depending on the contents of the page itself. The dictionary header is used in dictionaries, encyclopedias, and telephone books where the left header indicates the first entry on the page and the right header indicates the last entry on the page. This processing can be time-consuming, but leads to a better product to which consumers are accustomed.

All of these formatting conventions have been developed over a thousand years of page creation. We all grew up learning to navigate our way around a page, so these conventions should be followed to provide your users with a familiar interface.

### DSSSL and XSL

First, a little background. All of the techniques described above are oriented toward the delivery of information on paper. However, your XML documents are probably tagged according to each element’s meaning, not whether it should be italicized or placed in a dictionary header. We need to map the structure to a page layout. This requires a lot of decisions, which can be expressed in the language of the typesetting system.

Each typesetting system, however, has a different way of expressing such information. An ISO standard called DSSSL (Document Style Semantic Specification Language) was designed to normalize all of the rich formatting capabilities into a single syntax. The goal was to create a non-typesetter-specific formatting language that could be translated to any typesetter’s code. The benefits of this approach are twofold: first, a designer needn’t know the syntax specifics of a particular typesetting system to create pages using that typesetter. Second, creating stylesheets in a non-vendor-specific syntax allows a company to change their typesetters at will, without the costly process of converting from one syntax to another.

DSSSL took ten years to create, and was finally ratified as an international standard just about the time that XML was gaining momentum. DSSSL is based on SGML, and was never really implemented because software vendors were looking at XML as a replacement for SGML. The need for a vendor-neutral typesetting system was still there, however, so the DSSSL folks started work on a specification called XSL, the Extensible Stylesheet Language. XSL was intended to achieve the same goals as DSSSL, except it was expressed in XML syntax.

XSL uses a transformation process, which converts your XML document into another XML document expressed as a set of “formatting objects”. These formatting objects have element names like “block” and “character”, with attribute values like “bold”, and “green”. This formatting object XML document is exposed to the second XSL step, which translates the formatting object document into the codes of a particular typesetter.

This model assures that a designer need only be concerned with a single way of formatting a page (the formatting object model), and leaves the intricacies of the typesetter to each typesetter vendor. The biggest problem with XSL is that it is very difficult to express the full range of formatting options in a single, generic specification. The DSSSL people took ten years to do it.

XSL is still being developed, but has spawned another specification called “XSLT” (see Bob DuCharme’s “XML Linking and Styling: Standards Status Report”, <TAG> August, 1999). XSLT is only the first half of the XSL process. XSLT is designed to provide a generic tree-to-tree transformation of one document structure to another. Originally, as I mentioned, this resulting structure was the one defined by the formatting object schema. However, XSLT has been generalized to a point where it can create any arbitrary XML structure, and even non-XML structures. XSLT is truly a generic XML processing language.

### **T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X**

Now that we have a way of getting our XML documents into some other form, how do we produce pages? Simple, use a pagination program. There are many different pagination systems available for virtually any price you want to pay. In the 1980s, a computer science professor named Donald Knuth at Stanford University was working on a set of textbooks to describe The Art of Computer Programming. Knuth needed a more sophisticated way of paginating his document than the current state-of-the-art paginators were able to do. At that

time, scholarly works were being formatted using a rudimentary typesetting system called “troff”. Knuth was a student of the art of typesetting, and felt that a computer could be taught most of the mechanics of expressing that art. So, he embarked on an effort called T<sub>E</sub>X, which he describes in his book *The T<sub>E</sub>Xbook*.

Knuth used T<sub>E</sub>X to typeset his seminal multi-volume set of computer science textbooks, which has become the bible of computer science academia. Knuth also made the source of his new typesetting language available to the world to use and improve upon long before the concept of “open source” grabbed headlines. It didn’t take long before T<sub>E</sub>X became the syntax used to create scholarly and technical journals. The T<sub>E</sub>X mathematical syntax is very powerful, and is used to create technical papers with an accuracy unrivaled by any commercial typesetting system.

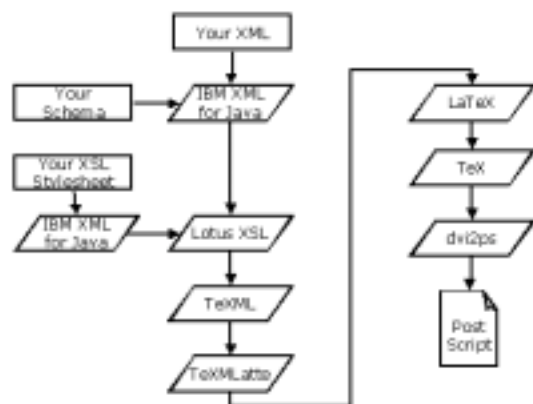
T<sub>E</sub>X is a very powerful typesetting language, with what I think is the best H&J logic available anywhere at any price. T<sub>E</sub>X produces beautiful pages and, because of the many add-ons that people have created over the years, has great flexibility.

Of course, you need to pay for this power and flexibility. The cost is learning the terse syntax and understanding all of the different settings and the way pages are created.

One of the most successful add-ons to T<sub>E</sub>X is a package called L<sup>A</sup>T<sub>E</sub>X, which provides an easier-to-use interface to the powerful T<sub>E</sub>X language. While T<sub>E</sub>X has commands for setting the font style to bold and left-justifying paragraphs, L<sup>A</sup>T<sub>E</sub>X has directives that allow you to indicate the title of a document, or the body of a section. For example, L<sup>A</sup>T<sub>E</sub>X uses the `\section` and `\subsection` commands to indicate where such breaks are made. What happens, however, if you call your structural objects “chapter”, or “lesson plan” or “appendix”? And what if your structure doesn’t map directly to those hard-coded into the L<sup>A</sup>T<sub>E</sub>X spec? If this is the case, you need an intermediary translation to indicate the complex structure-to-structure translation.

### **Alphabet Soup**

IBM, through an effort called AlphaWorks, is working on a number of projects to support XML and related standards. The AlphaWorks site makes available an XML parser, written in Java. On top of this, they provide, under their Lotus brand name, a product called LotusXSL. LotusXSL is an XSL processor that uses the transformation part of XSL (XSLT) to transform one type of XML to another.



**Figure 1:** Paper chase: XML to Paper using XML, XSL and  $\text{\TeX}$

Lotus engineers have created an XML schema that is designed to express the structure of a paper document in terms of the  $\text{\LaTeX}$  markup language. They have also created a processor to read this XML document and transform it to  $\text{\LaTeX}$  codes.

The path I am describing has many pieces that must fit together exactly. That’s the down-side. The up-side, however, is a very powerful XML-to-paper path that will not cost you a penny, and runs on any platform that runs Java.

The map is shown in Figure 1.

The XML document describes our information in terms of the information itself, not in terms of some eventual delivery platform. Therefore, we need to run a transformation step that translates this into some kind of format that can be interpreted as a two-dimensional, static form to be rendered to paper.

Translating directly to  $\text{\TeX}$  is difficult because of the complexity of the  $\text{\TeX}$  typesetting language. One of the problems is that  $\text{\TeX}$  uses the “<” and “&” characters to mean certain things. These characters are sacred to the XML parser, and shouldn’t be used because they might end up being interpreted as markup characters.

One solution, then, is to transform our XML document, which describes our information structure, into another XML document that describes the desired formatting characteristics of the information contained therein.

The  $\text{\TeX}$ XML system does this by defining an XML document that provides the full capabilities of the  $\text{\TeX}$  typesetting language. Actually, the  $\text{\TeX}$ XML system creates documents that can be expressed in  $\text{\LaTeX}$ , the  $\text{\TeX}$  add-on that is used to provide a high-level interface to  $\text{\TeX}$ .

## Structure to Pages

There are many steps involved. In this article, we will follow a fairly simple document through the steps required to go from XML to paper.

### 1. Create a directory to contain all programs and data.

Select some directory anywhere on your machine. I will describe all processes in relation to that root. You should be able to move the contents of the directory anywhere using the relative paths explained here.

### 2. Load the IBM4J parser.

Go to <http://www.alphaworks.ibm.com/tech/XML4J> and download IBM’s parser written in Java. The examples in this article use version 2.0.15 of IBM’s parser. Extract the files to the `xml4j_2_0_15` directory under the directory set above. If you are using a different version, you might need to change some environment variables and batch-file commands.

### 3. Load the Lotus XSL processor.

Go to <http://www.alphaworks.ibm.com/tech/LotusXSL> and download the Lotus XSL processor. The examples in this article use version 0.18.5 of the XSL processor. Extract the files to the `lotusxsl_0_18_5` directory under the directory set above. If you are using a different version, you might need to change some environment variables and batch-file commands.

### 4. Load the $\text{\TeX}$ XML Processor.

Go to <http://www.alphaworks.ibm.com/tech/texml> and download the IBM  $\text{\TeX}$ XML processor. The examples in this article use version 1.4 of the  $\text{\TeX}$ XML processor. Extract the files to the `TeXML_V1R4` directory under the directory set above. If you are using a different version, you might need to change some environment variables and batch-file commands.

### 5. Load a $\text{\TeX}$ Processor.

There are many excellent  $\text{\TeX}$  processors available for free or by commercial license. Check the  $\text{\TeX}$  Users Group at <http://www.tug.org> for a list of pointers to sites with  $\text{\TeX}$  implementations. I used the MiK $\text{\TeX}$  implementation for this article, which can be found at <http://www.miktex.de/>. Most  $\text{\TeX}$  implementations come with the  $\text{\TeX}$  processor, which will also read  $\text{\LaTeX}$  files. A  $\text{\TeX}$  processor creates a device-independent (DVI) output file. Most implementations also come with a program for converting the DVI to printable forms, like PostScript. The batch files in this article assume that `tex`, `latex`, and `dvi2ps` are in the system path, and that

there is a program registered to view DVI files. You might need to change some environment variables and batch-file commands.

#### 6. Select an XML document as a source.

Create or find an XML document that is suitable to transformation. XSL provides a powerful engine to transform from any XML structure to any other. For this example, we picked a straightforward example. Ours is shown in Figure 2.

```
<?xml version="1.0"?>
<bill-o-rights>
  <section>
    <title>Amendment I (1791)</title>
    <para>Congress <emph>shall make no law</emph>
    respecting an establishment of religion, or
    prohibiting the free exercise thereof; or
    abridging the freedom of speech, or of the
    press; or the right of the people peaceably
    to assemble, and to petition the government
    for a redress of grievances.</para>
  </section>
  <section>
    <title>Amendment II (1791)</title>
    <para>A well regulated militia, <emph>being
    ... {rest of bill of rights here} ...
    others</emph> retained by the people.
  </para>
</section>
<section>
  <title>Amendment X (1791)</title>
  <para>The powers not delegated to the United
  States by the Constitution, nor prohibited by
  it to the states, <emph>are reserved to the
  states respectively, or to the people</emph>.
</para>
</section>
</bill-o-rights>
```

Figure 2: XML document to be processed

#### 7. Write an XSL style sheet.

This document is processed using an XSL stylesheet that transforms it into an XML document adhering to the TeXML schema. Instead of using TeX directly, this system uses L<sup>A</sup>T<sub>E</sub>X, because it has a higher-level interface. L<sup>A</sup>T<sub>E</sub>X requires the creation of environments (env), commands (cmd), and parameters (parm). The XSL stylesheet identifies elements in the input XML document and outputs an XML document that consists of these env, cmd, and parm elements, plus some others to create the output

```
<?xml version="1.0"?>
<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/XSL/Transform/1.0">
  <xsl:output method="xml" indent="yes"
    encoding="UTF-8" xml-declaration="yes"/>

  <xsl:template match="/">
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="bill-o-rights">
    <TeXML>
      <cmd name="documentclass">
        <parm>article</parm>
      </cmd>
      <cmd name="title">
        <parm>U.S. Bill of Rights</parm>
      </cmd>
      <env name="document">
        <cmd name="date">
          <parm>1791</parm>
        </cmd>
        <cmd name="maketitle"/>
        <xsl:apply-templates/>
      </env>
    </TeXML>
  </xsl:template>

  <xsl:template match="section">
    <cmd name="section*">
      <parm>
        <xsl:value-of select="title"/>
      </parm>
    </cmd>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="section/title"/>

  <xsl:template match="para">
    <xsl:apply-templates/>
    <cmd name="par"/>
  </xsl:template>

  <xsl:template match="emph">
    <cmd name="emph">
      <parm>
        <xsl:apply-templates/>
      </parm>
    </cmd>
  </xsl:template>
</xsl:stylesheet>
```

Figure 3: XSL Stylesheet to create TeXML output

```

<DIR>    {\LotusXSL}_0_18_5
<DIR>    TeXML_V1R4
<DIR>    xml4j_2_0_15
  4,486  bill-o-rights.xml
  1,356  BOR2TeXML.xsl
  285    xml2tex.bat

```

**Figure 4:** Directory structure

```

java -cp xml4j_2_0_15\xml4j.jar;
lotusxsl_0_18_5\lotusxsl.jar
com.lotus.xsl.Process -in
%1.xml -xsl %2.xsl -out %1.texml
java -cp TeXML_v1r4\TeXML.jar;
xml4j_2_0_15\xml4j.jar
com.ibm.texml.TeXMLatte
%1.texml %1.tex
latex %1
dvips %1
start %1.dvi

```

**Figure 5:** Commands to run all processes

document. The  $\LaTeX$  language is described by the inventor, Leslie Lamport, in his book,  *$\LaTeX$ : A Documentation Preparation System User's Guide and Reference Manual*.

Our XSL stylesheet is shown in Figure 3.

The output from the XSL transform is an XML document that expresses the contents of the document as a series of environments, commands, and parameters, along with the text that is to be displayed according to the parameters. A program called  $\TeX$ MLatte processes this XML document and creates a  $\TeX$  input file. This  $\TeX$  file is processed by the  $\TeX$  processor, which creates a DVI file. The DVI file is transformed into a PostScript document with the DVI2PS program, and voila!, you've got paper!

#### 8. Process your files.

As we have seen above, several processes need to be executed to go from your XML to paper using the  $\TeX$  path. You should create a batch file or shell script that executes each one in turn to make the process automated. Using the directory structure shown in Figure 4, your commands look like those shown in Figure 5.

Notice that all of the paths are relative to the directory that contains directory structures for each component. Notice, also, that I have included the jarfiles directly in the java call using the `-cp` (classpath) command-line argument. If you have these in your CLASSPATH environ-

```

<figure>
<verbatim>
<?xml version="1.0" encoding="UTF-8"?>
<TeXML>
  <cmd name="documentclass">
    <parm>article</parm>
  </cmd>
  <cmd name="title">
    <parm>U.S. Bill of Rights</parm>
  </cmd>
  <env name="document">
    <cmd name="date">
      <parm>1791</parm>
    </cmd>
    <cmd name="maketitle"/>
    <cmd name="section*">
      <parm>Amendment I (1791)</parm>
    </cmd>
    Congress
    <cmd name="emph">
      <parm>shall make no law</parm>
    </cmd>
    respecting an establishment of
    religion, or prohibiting the free
    exercise thereof; or abridging the
    freedom of speech, or of the press;
    or the right of the people peaceably
    to assemble, and to petition the
    government for a redress of grievances.
    <cmd name="par"/>
    <cmd name="section*">
      <parm>Amendment II (1791)</parm>
    </cmd>
    A well regulated militia,
    <cmd name="emph">
      <parm>being necessary to the security
        of a free state</parm>
    </cmd>
    , the right of the people to keep and
    bear arms, shall not be infringed.
    <cmd name="par"/>
    ...
  </env>
</TeXML>

```

**Figure 6:**  $\TeX$ XML document

ment variable, you don't need to indicate them here.

#### 9. Check the output.

When the XSL stylesheet shown here is run against the XML document shown, it produces the  $\TeX$ XML file shown in Figure 6. When this document is processed with the  $\TeX$ MLatte

```

\def\TeXmath#1{\ifmode#1{\else$#1{}}\fi}
\def\TeXnomath#1{\ifmode\hbox{#1{}}\else#1{}}\fi}
\documentclass{article}
\title{U.S. Bill of Rights}
\begin{document}
\date{1791}
\maketitle
\section*{Amendment I (1791)}
Congress \emph{shall make no law} respecting an
establishment of religion, or prohibiting the
free exercise thereof; or abridging the freedom
of speech, or of the press; or the right of the
people peaceably to assemble, and to petition
the government for a redress of grievances.\par
\section*{Amendment II (1791)} A well regulated
militia, \emph{being necessary to the security}
of a free state}, the right of the people to
keep and bear arms, shall not be infringed.
\par
...
\end{document}

```

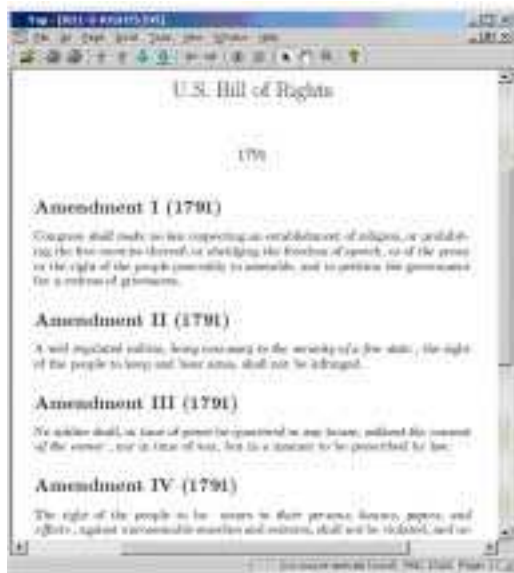
Figure 7: L<sup>A</sup>T<sub>E</sub>X document

Figure 8: The final document (shown in YAP DVI Viewer)

program, the result is a L<sup>A</sup>T<sub>E</sub>X file shown in Figure 7. After the L<sup>A</sup>T<sub>E</sub>X file is processed with T<sub>E</sub>X, it can be viewed using a DVI viewer, as shown in Figure 8. <end/>

◇ Brian E. Travis  
btravis@architag.com



---

## A WYSIWYG $\TeX$ implementation

Igor I. Stokov

### Abstract

A true WYSIWYG editor is implemented by means of minor modifications to canonical  $\TeX$ . The changes include the ability to start compilation from an arbitrary page and fast reformatting of paragraphs. The new features provide an immediate response for editing the typeset preview of a document.

### 1 Conditional compilation

$\TeX$  was designed and implemented as a document compiler (Knuth 1986); that is, one cannot preview a typeset document before the compilation of its source file, which means a relatively long response time between inputting the text and previewing the result. It does not matter much until one has to deal with the document's final appearance, making numerous source file corrections to achieve better-looking output. In other programming languages the problem of acceleration is often resolved by means of a 'conditional compilation', where a compiler tries to locate changes in the source text and perform only that part of the job relevant to the changes.

The same method evidently could work with  $\TeX$ : if, say, a user corrects page 10, then there is no need to recompile the first 9 pages, as they will remain the same.<sup>1</sup> If one could load the complete  $\TeX$  memory stage on the beginning of page 10 then the page of interest could be obtained much faster. Indeed  $\TeX$  already does something in this vein, loading a precompiled format on the job start. One need only generalize this technique for the intermediate stages of a  $\TeX$  run. And here is where the technical difficulties begin, involving an account of almost all global variables, arrays, open file pointers, etc. Besides, one cannot afford to store all these values, literally, if we're talking about ordinary hard disks.

So, let us see how it is done in the WYSIWYG  $\TeX$  prototype program with the tentative name ' $\TeX$ lite'. The memory dump is made after every page completion when the page is thrown out and the memory is relatively empty. In addition, an extra dump refers to the beginning of the first page (after loading all styles or `\input` files, at the moment of first switching into horizontal mode). So,

---

<sup>1</sup> Sometimes they will not, for example, if a table of contents goes at the beginning and the document requires two runs. This case is discussed below.

we have memory stage  $j + 1$  after each  $j$ -th page, plus the initial stage 1 to be dumped. Indeed, every  $j$ -th dump ( $j \geq 1$ ) records only the differences to a basic memory stage  $k$ , where  $k = (j/8) * 8 + 1$ . The basic memory stages (whose numbers form the sequence 1, 9, 17, 25, ...) in turn are stored as the differences with respect to memory stage 0, which occurs just after loading the precompiled format. Here a precompiled format is handled as a special case of a memory dump—the only one made, regardless any other basic memory stage.

The two-level hierarchy of basic memory stages makes it possible to keep the total number of dumps space almost linear, with respect to document size. In fact, close memory stages generally differ less than distant ones as differences tend to be collected. By confining the distance between compared memory stages (to 8 in our case) one can set up a certain differences limit. Two levels of comparison definitely slow down memory dumping and reading although there is the positive effect resulting from smaller space requirements and fewer disk addressings.

Rough measurements were done on a 535-page book, *T<sub>E</sub>X: The Program* (Knuth 1986). T<sub>E</sub>Xlite was tested in both canonical T<sub>E</sub>X and WYSIWYG modes, where the first case required 24 seconds and the other 41 seconds. All 536 dumps took 29,280,000 bytes of virtual memory. These values, of course, indicate plenty of scope for T<sub>E</sub>Xlite optimization.

As the memory stages are dumped, it is known which line  $l_j$  in a source file was being read by T<sub>E</sub>X on the completion of  $j$ -th page. If a user has edited line  $l$ ,  $l_j < l \leq l_{j+1}$ , then T<sub>E</sub>X does not need to recompile the first  $j$  pages; it can already start from  $j + 1$ . There is one exception however: T<sub>E</sub>X might produce or change some `\output` files and wish to read their contents again at the next run (as happens, for example, with the L<sup>A</sup>T<sub>E</sub>X command `\tableofcontents`). Let the user enter or correct some T<sub>E</sub>X clause and press a certain key to watch the result. T<sub>E</sub>Xlite notices the least line number  $l_j$  subjected to changes and retrieves the corresponding page number  $j + 1$ . Then it loads the memory stage  $j + 1$  and starts T<sub>E</sub>X which behaves as if it has just processed page  $j$  and is going to continue the compilation. If T<sub>E</sub>X is not interrupted by another user demand then it will run until the end of a document and check whether any `\output` files have been updated since the previous run. If they have, then T<sub>E</sub>X is run again, this time from the zero stage.

So, in response of the user ‘recompile’ command, T<sub>E</sub>X is run once or twice. Each time it produces, among others, a page which can be pre-

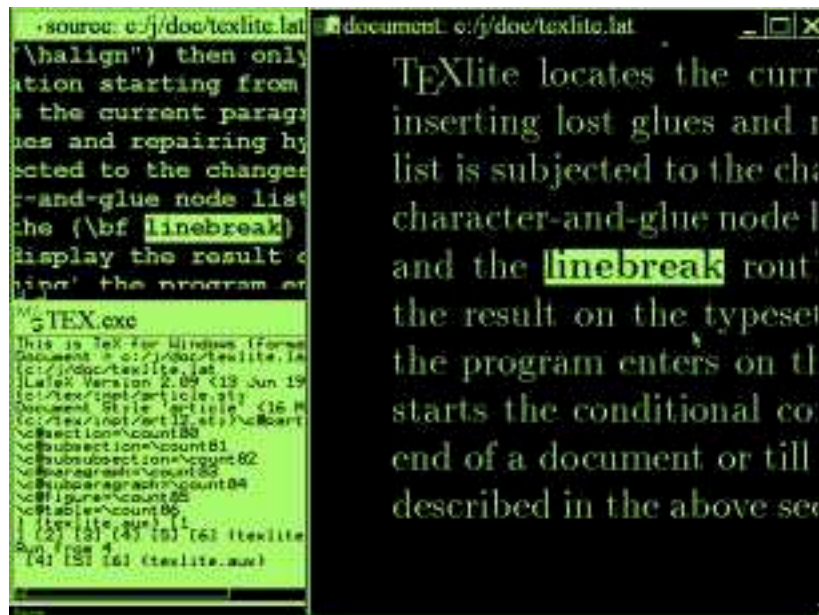
viewed by the user. Before displaying the page T<sub>E</sub>Xlite compares its past and present virtual views and composes a map of changes. This map (which may be, and usually is, void) helps to both reduce redraw time and avoid flickers. In practice it means that a user may enter or edit some consistent T<sub>E</sub>X clause and get a very fast (in a split-second) and precise response regardless of what page number he is working with. Though there still remains a chance for a page view to be altered later, the possibility is small and the change is gentle.

## 2 WYSIWYG T<sub>E</sub>X

Although the conditional compilation already provides significant advantages, it still leaves two major problems rooted in a human psychology untouched. First, it is wrong to share one’s attention between two views (source text and typeset document). Moreover, most people (all but us T<sub>E</sub>X users) do not like programming languages and avoid learning them despite all the accrued benefits. Thus there is a certain need to provide a way to work directly and solely with the typeset view of a document, leaving intervention in the source file for extreme cases.

The simplest (and probably only) decision lies in keeping the back link (the authors of the Mac implementation *Textures* call it ‘synchronicity’<sup>2</sup>) from the typeset document to the source text. The problem, of course, is rather technical and is resolved in T<sub>E</sub>Xlite by keeping track of source file characters to their corresponding memory nodes in a special array. All operations with memory nodes (including node lists copying, rebuilding, etc.) address this array as well. Finally, the information on character locations in the source text (line and column numbers) is stored in the typeset pages output (an analog of the DVI file). Using this information one can synchronize positions in a document view and its source text. Users may work with the document view and mark a current position in it with a flashing caret. Upon performing some editing operation one could apply a corresponding action to the source text and initiate the conditional compilation as described above. On fast machines (starting from a Pentium-200) this process (compilation of one page) is often fast enough to achieve no perceptible delay between pressing a key and obtaining a visible result. However, one should not rely upon fast machines only. Besides, there may be various slowing down factors, such as complex page formatting, slow macros involving vast calculations, etc.

<sup>2</sup> See <http://www.bluesky.com/sync.html> for details.



**Figure 1:** A screen shot of a TeXlite run. A selection in the typeset document window is shown to be mapped into the source text.

Although any procedure providing a fast and fairly accurate result will help here, it is better to use native TeX algorithms for this purpose. Difficulties in this choice follow from the fact that canonical TeX does not keep parameters it has used to build boxes and paragraphs. That is, one could not correctly rebuild a box or a paragraph from its contents alone. TeXlite resolves this problem by storing necessary data in special ‘whatsit’ nodes. It does not take too much extra space, as many parameters (penalties, glues, parshape, etc.) remain the same throughout a document and can thus be omitted. In addition, TeXlite is more verbatim in its output of typeset pages as it preserves all the nested lists structure (however, the common DVI file is also optionally output).

Let us see what happens when a user edits a typeset document. First of all, TeXlite decides (with the aid of ‘whatsit’ nodes) which paragraph, if any, the current position belongs to. If no paragraph is recognized (that is, it may happen within `\halign`) then only the enclosing box is rebuilt and the conditional compilation starting from the current page is initiated. Otherwise, TeXlite locates the current paragraph and unwraps it back into the hlist by inserting lost glues and repairing hyphenation aftermaths. The unwrapped list is subjected to the changes followed from the user input (one may insert

a character-and-glue node list or delete several nodes from the current position) and the *linebreak* routine is called to rebuild the paragraph and display the result on the typeset document view. After this ‘emergency repair’ the program enters the source text, performs parallel changes there and starts the conditional compilation which runs from the current page to the end of the document or until the user presses a key once more. Here the scenario described in the above section is repeated in detail. If TeX manages to build the current page before a next key hit (usually it does) and the new page happens to be different (usually it does not) from the repaired one then the view is accurately updated.

### 3 Implementation

At present TeXlite is implemented under Win32 although without any specific Win32 virtues, which hamper porting to other platforms, are used. The program spawns four threads, where the most important one is TeX, slightly modified in five aspects:

1. It can be interrupted from outside and fall asleep until an external wake-up command.
2. It dumps its own memory stage after every page completion.
3. For every paragraph it stores all the data required to unwrap the paragraph and break it into lines again.

4. It outputs typeset pages in a form of nested lists along with a common DVI file.
5. It traces the ancestry of nodes in the memory and in typeset pages from the source text.

Another thread answers for the user interface (which is more than just bare-bones now) and owns the source text and typeset document windows (see Figure 1). Two other threads, running on a higher priority, do asynchronous mapping and scaling of typeset pages to the previewer, which allows no bottlenecks in the path from a user action to a visible result.

Thus in  $\text{\TeXlite}$  one can edit a typeset document in true WYSIWYG mode without addressing the source text, at least while dealing with a narrative text. Still, there are many apparent improvements worth adding: language constructions handled by menu commands or by application of ‘wizards’, linkage of  $\text{\TeX}$  messages to the source text to allow faster and more intuitive error corrections, and so on. Further application of the WYSIWYG mode for  $\text{\TeX}$  also promises some more substantial benefits whose exploration, however, requires more extensive efforts.

#### 4 Availability

An alpha release of  $\text{\TeXlite}$  is available by emailing the author under the condition to report all bugs and problems to him. A self-contained distribution of  $\text{\TeXlite}$  takes about 600K bytes.

#### References

- [1] Knuth, D.E. *Computers & Typesetting, Vol. B,  $\text{\TeX}$ : The Program*. Reading, Mass.: Addison-Wesley, 1986.

◇ Igor I. Stokov  
Novosibirsk Institute of Organic  
Chemistry  
Siberian Branch of Russian  
Academy of Science  
Lavrentiev avenue 9  
Novosibirsk 90, Russia  
`stokov@nioch.nsc.ru`

## Book Reviews

### *The L<sup>A</sup>T<sub>E</sub>X Graphics Companion* and *T<sub>E</sub>X Unbound* — A Review of Two Books

Bill Casselman

Michel Goossens, Sebastian Rahtz, and Frank Mittelbach, *The L<sup>A</sup>T<sub>E</sub>X Graphics Companion: Illustrating documents with T<sub>E</sub>X and PostScript*. Addison-Wesley, Reading, Massachusetts, 1997, ISBN 0-201-85469-4, 554 + xxv pages, \$39.95.

Alan Hoenig, *T<sub>E</sub>X Unbound: L<sup>A</sup>T<sub>E</sub>X & T<sub>E</sub>X Strategies for Fonts, Graphics, & More*. Oxford University Press, New York, 1998, ISBN 0-19-509686-X 580 + ix pages, <http://www.oup-usa.org/docs/019509686X.html>, \$35.00 (paper).

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 T<sub>E</sub>X has not yet been matched by one in mathematical illustration. Curiously, at the same time Knuth gave us T<sub>E</sub>X, he also gave us the graphics language METAFONT, but this has never enjoyed anywhere near the popularity of T<sub>E</sub>X 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 T<sub>E</sub>X 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

---

This review originally appeared in the *Notices of the American Mathematical Society*, **46**:11 (December 1999), pp. 1402–1406, and appears here by permission.

the illustrations, and (2) including them in mathematical papers written in  $\text{T}\text{E}\text{X}$ . 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  $\text{T}\text{E}\text{X}$  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  $\text{T}\text{E}\text{X}$  is not so sharp as might first appear, as anyone who has tried to construct complicated commutative diagrams knows from 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 make-up that might lead one to abandon pure  $\text{T}\text{E}\text{X}$  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  $\text{T}\text{E}\text{X}$ .

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

of the review, 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  $\text{L}\text{A}\text{T}\text{E}\text{X}$  and other flavors of  $\text{T}\text{E}\text{X}$ . It does not attempt to give details of how to use  $\text{T}\text{E}\text{X}$ , but contents itself with an interesting survey which does a fairly good job of placing  $\text{T}\text{E}\text{X}$  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  $\text{L}\text{A}\text{T}\text{E}\text{X}$  Companion*. Well, not quite, because in this volume as well as the earlier one, Goossens et al. do indeed restrict their attention to  $\text{L}\text{A}\text{T}\text{E}\text{X}$ . This is probably a blessing for the large number who use only  $\text{L}\text{A}\text{T}\text{E}\text{X}$ , but their book is therefore of limited use to the more technically sophisticated readers who would otherwise be attracted to it. The restriction to  $\text{L}\text{A}\text{T}\text{E}\text{X}$  is especially frustrating since almost all of the advice they give can be paralleled in any flavor of  $\text{T}\text{E}\text{X}$ . However, figuring out the necessary adjustments in a non- $\text{L}\text{A}\text{T}\text{E}\text{X}$  environment might take a great deal of time. Those who do restrict themselves to  $\text{L}\text{A}\text{T}\text{E}\text{X}$  will be able to use the  $\text{L}\text{A}\text{T}\text{E}\text{X}$  `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  $\text{L}\text{A}\text{T}\text{E}\text{X}$  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  $\text{L}\text{A}\text{T}\text{E}\text{X}$ <sup>1</sup> and that it would not have been a difficult task for its developers to have made it available outside the  $\text{L}\text{A}\text{T}\text{E}\text{X}$  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  $\text{T}\text{E}\text{X}$  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,

---

<sup>1</sup> Editor's note: This misunderstanding is cleared up in the Afterword.

for example, how to use PostScript fonts instead of the bit-mapped fonts that are often used currently by default in  $\text{\TeX}$ . Hoenig spends more than 200 pages dealing with fonts, including a useful survey of the role of METAFONT in  $\text{\TeX}$ '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  $\text{\LaTeX}$ . 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  $\text{\TeX}$  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  $\text{\LaTeX}$ . 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  $\text{\TeX}$ , 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 `.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 program such as Adobe Illustrator or Corel Draw. Among these, the most suitable will be those producing vector graphics, which are uniformly scalable, rather than bit maps 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  $\text{\TeX}$  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  $\text{\TeX}$  and  $\text{\dvips}$ , 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  $\text{\ghostview}$ , 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  $\text{\TeX}$ , and its use by Knuth in font design played a crucial role in  $\text{\TeX}$ 's success. For this reason alone, perhaps, it should occupy at least a small part of the heart and mind of every  $\text{\TeX}$  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  $\text{\TeX}$  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 a feasible way to produce pictures in the first place. They share also an apparent aversion to  $\text{\ghostscript}$ , 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  $\text{\TeX}$ . Both books cover a number of these. They generally have one great virtue pretty much missing from all of the options (1)–(4), which is that they enable one to include  $\text{\TeX}$  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  $\text{\PSTricks}$  package developed by Timothy van Zandt and Denis Girou, which comes with most free  $\text{\TeX}$  distributions. This is essentially a  $\text{\TeX}$  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  $\text{\PSTricks}$  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  $\text{\LaTeX}$ . Chapter 2 describes the package of tools, such as



`graphics` and `graphicx`, 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 covers 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://www.tug.org/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  $\TeX$  Users 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 Kaufmann. Blinn's home page is at

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

### Afterword

Note to the reissue in *TUGboat*:

Since the original publication of this review in the *American Mathematical Society Notices*, it has been called to my attention that although both books here under review indicate strongly that the `graphicx` package is tied to  $\LaTeX$ , this is not in fact the case. A version for use with plain  $\TeX$  should be available at any of the CTAN archives.

It has also been called to my attention that I might have mentioned the package `psfrag`, which helps embed  $\TeX$  labels in PostScript figures. It is not in my view a perfect program, but it is nonetheless impressive. The original version was written, apparently, in `perl`, but the current version has been written in  $\TeX$  itself. As Hoenig says of another program in his book, in this program you can see  $\TeX$  do things it may never have been intended to do (although who of us can presume to read the mind of Don Knuth?). Reading the source for `psfrag` might bring a shudder to any programmer who values readability and flexibility highly, but of course it trades these virtues for another — namely, portability.

One of the referees of the original article claimed that my advice to use 'raw' PostScript for drawing mathematical pictures was in some way a betrayal of the highest standards of mathematical elegance. I would like to say with even more emphasis here that in my view good mathematical

graphics requires an input of mathematical concepts. Programming directly in PostScript, if a reasonable library is at hand, suits admirably, and can without doubt produce the best possible output. I have used it for this purpose for several years, and have managed even to teach my techniques over the years to hundreds of mathematics undergraduates. Almost everyone who has tried it has come away quite pleased.

Finally, the last line of the original review has been misinterpreted as saying that my home page is at `microsoft.com`. Nothing is further from the truth, and that line has been changed in this version.

◇ Bill Casselman  
Mathematics Department  
University of British Columbia  
Vancouver, Canada V6T 1Y4  
[cass@math.ubc.ca](mailto:cass@math.ubc.ca)

---

### Book review: *Digital Typography*

Peter Flynn

Donald Knuth, *Digital Typography*. CSLI Publications, Stanford, CA, 1998, 1-575586-010-4.



Over two decades of METAFONT,  $\TeX$ , and  $\LaTeX$  have left the world with a wealth of material about to the digital nature of type, typography, typesetting, and type design related to these systems. However, although the standard manuals [1, 2, 3]

---

This illustration appears on page 1 of *Digital Typography*, and is used here by permission. The file contains this note: "Fake woodcut I picked up somewhere in early 80s. If anybody can identify the source, I'll gladly give credit..."

-- Don Knuth

mention some of the topics where they are immediately relevant, and over half of them have been printed in *TUGboat* at some stage, a lot of the material is not part of the actual programs or their documentation. Instead, it forms part of the background or history of  $\text{\TeX}$  and friends: some of it is transient, being posted to newsgroups or mailing lists; some is anecdotal; and some has probably even reached the status of myth.

I am therefore particularly pleased to be able to review this book by Knuth himself, which not only describes the development of METAFONT and  $\text{\TeX}$  but sets out many of the fundamental principles of digital type design and typography, and explains those aspects of the theory and practice which underpin the programs and affect how they get used.

The book is arranged in 34 chapters, each being an article or note on a specific topic. The subjects covered range wide over the field, from the internal details of the algorithms for breaking paragraphs into lines to a simple way to do diagonal fractions for weights and measures in cooking. Along the way we are treated to dissertations on the design of the letter S; the origins of the Euler, Concrete, and Punk fonts; typesetting tricks like flowing text around an image; the use of fonts in Indian script and in right-to-left languages; digital half-tones; the real origins of the first drafts of  $\text{\TeX}$ ; and Knuth's views on the past and future of his creations, taken from Q&A sessions with users (see the Table of Contents on page 365).

The old Army phrase, 'on parade, on parade; off parade, off parade', can just as well be applied to mathematicians and computer scientists: this book is Knuth off-duty. The formal papers and articles are mixed with teaching notes and recipes, diary entries and interviews. His lucid, fluent, and exact prose makes it a pleasure to read, and the occasional diversions give us an insight into some of the byways Knuth explored on his journeys from drafts and early ideas to finished programs and fonts. Along with the aesthetics there is some programming (I had to dust off what Pascal I once knew!), and of course some mathematics, so designers and programmers alike will find plenty to read—as will every user. Much of it is not tied to  $\text{\TeX}$ , and the developers of less competent systems would do well to read the book to see where they are going wrong.

The first two chapters deal with a little typographic history and the demands that mathematics places on typography (the original reason for  $\text{\TeX}$ 's existence). Chapters three and four explain in great detail how paragraph formatting works: not line by line but by treating the paragraph as a whole. In chapters five to nine we have a series of useful macros (in plain  $\text{\TeX}$ ; but  $\text{\LaTeX}$  equivalents exist in almost

every case) covering a variety of small formatting needs. Chapters 10 and 11 handle some internal details of  $\text{\TeX}$  and WEB.

From chapter 12 to chapter 20 we are deep in fontland, on letter design, the meta-ness of METAFONT and the lessons it teaches, and the design and use of math fonts. Chapters 21–23 may come as something of a surprise: half-tones and the digitization of angles are unusual topics in books on typography, and yet they are very welcome because they explain clearly the problems of rasterization: fitting the dots when the dots won't sit in a line.

The historical chapters 24–26 contain probably the least-known material: Knuth opens his diary on the days when he was designing and nurturing  $\text{\TeX}$ . As Tom Lehrer says, some of you may have had occasion to run into mathematicians, and to wonder therefore how they got that way [4]... the labor of love expressed in these chapters may go some way towards explaining it!

Chapter 27 contains another small diversion: the bitmaps for the canonical icons for  $\text{\TeX}$  and friends. Chapter 28 explains how Knuth got from computing into typesetting. From chapter 29 to 33 we find the future: in a series of question-and-answer sessions at conferences, Knuth explains his views on how things happened and where they go from here. Finally, he begins the process of winding down on  $\text{\TeX}$ , as the number of new errors discovered asymptotes to zero.

I have found this book a wonderful source of both information and knowledge. Whether you're new to type or you've been using it for 20 years or more, there's something here you didn't know. Go and buy it now.

Here is the table of contents:

1. Digital Typography
2. Mathematical Typography
3. Breaking Paragraphs Into Lines
4. Mixing Right-to-Left Texts with Left-to-Right Texts
5. Recipes and Fractions
6. The  $\text{\TeX}$  Logo in Various Fonts
7. Printing Out Selected Pages
8. Macros for Jill
9. Problem for a Saturday Morning
10. Exercises for  $\text{\TeX}$ : The Program
11. Mini-Indexes for Literate Programs
12. Virtual Fonts: More Fun for Grand Wizards
13. The Letter S
14. My First Experience with Indian Scripts
15. The Concept of a Meta-Font
16. Lessons Learned from METAFONT

17. AMS Euler—A New Typeface for Mathematics
18. Typesetting Concrete Mathematics
19. A Course on METAFONT Programming
20. A Punk Meta-Font
21. Fonts for Digital Halftones
22. Digital Halftones by Dot Diffusion
23. A Note on Digital Angles
24. `TEXDR.AFT`
25. `TEX.ONE`
26.  $\TeX$  Incunabula
27. Icons for  $\TeX$  and METAFONT
28. Computers and Typesetting
29. The New Versions of  $\TeX$  and METAFONT
30. The Future of  $\TeX$  and METAFONT
31. Questions and Answers, I
32. Questions and Answers, II
33. Questions and Answers, III
34. The Final Errors of  $\TeX$

## References

- [1] Donald E. Knuth. *The METAFONTbook*. Addison-Wesley, Reading, MA, 1986.
- [2] Donald E. Knuth. *The  $\TeX$ book*. Addison-Wesley, Reading, MA, 2nd edition, 1986.
- [3] Leslie Lamport. *L<sup>A</sup>T<sub>E</sub>X, a document preparation system*. Addison-Wesley, Reading, MA, 2nd edition, 1994.
- [4] Tom Lehrer. Lobachevsky. In *Tom Lehrer Revisited*. Reprise Warner, Burbank, CA, 1959.

◇ Peter Flynn  
Computer Centre, University  
College, Cork, Ireland  
[pflynn@imbolc.ucc.ie](mailto:pflynn@imbolc.ucc.ie)  
<http://imbolc.ucc.ie/~pflynn>

<quote>

The author of T<sub>E</sub>X, Donald Knuth, has made it perfectly clear that he does not object to anyone revising T<sub>E</sub>X (or METAFONT) just as long as the resulting program is called something else. However, he also says “nobody is allowed to call a system T<sub>E</sub>X or METAFONT unless that system conforms 100% to the manuals to the TRIP and TRAP tests.”

</quote>

should correctly read:

The author of T<sub>E</sub>X, Donald Knuth, has made it perfectly clear that he does not object to anyone revising T<sub>E</sub>X (or METAFONT) just as long as the resulting program is called something else. However, he also says “nobody is allowed to call a system T<sub>E</sub>X or METAFONT unless that system conforms 100% to my own programs, as I have specified in the manuals to the TRIP and TRAP tests”.

---

## TUG 99

Christina Thiele

The TUG99 Conference was organized by several committees working together and independently, a point which was perhaps not clearly indicated the proceedings issue, *TUGboat* 20, no. 3, (pp. 156–57).

In addition to the Program Committee (co-chaired by Anita Hoover and Stephanie Hogue), which included the Proceedings Committee, there was a Conference Committee, co-chaired by Patrica Monohon and Susan DeMeritt. This latter committee took care of all the pre-conference site research and visits, made the arrangements with the university (rooms, meals, banquet, etc.), put together the registration package material (mugs, T-shirts, the printing and binding of the preprints etc.), and in particular took care of the budget to make the conference a financial success, so that for future conferences there would be a pocket of funds available to facilitate planning, and pay for the advance purchases of materials and services without individuals having to put stresses on their own personal reserves while waiting for reimbursement.

Our thanks to Wendy McKay for bringing these details to our attention.

## Errata

### The good name of T<sub>E</sub>X

Jonathan Fine

Editor’s note: The letter written by Mr. Fine to the editor, published in *TUGboat* 20, no. 2, p. 93, inadvertently had a missing \ before an % sign, thereby rendering the next to the last sentence unintelligible. The second paragraph:

## Resources

### A CTAN Search Page

Jim Hefferon

A Comprehensive T<sub>E</sub>X Archive Network (CTAN) site is a big place. Keeping track of what files are where, even just in the L<sup>A</sup>T<sub>E</sub>X subtree, and of what those files do, is a big job. Recently I (and others; see below) have added a resource—a web page—that you can use to find what you want.

#### 1 An Offer of Help

One thing that this page will do for you is what you'd think it would do: locate files by name. The other day I needed `cwebmac.tex`. I went to <http://tug.ctan.org/CTANfind.html>, submitted the filename, and was rewarded with a results page such that clicking on the filename downloaded the file. Besides the file I asked for, also shown on that page is some more information, the surrounding directory and the file's date, so that I can decide among (possible) multiple choices. There are also links to download the entire directory packaged as a zip file or as a tarred and gzipped file.

By the way, my download wasn't made from [tug.ctan.org](http://tug.ctan.org) but rather from my favorite CTAN mirror. The first time I used `CTANfind`, before seeing the results page I got a page that asked me to select the mirror. That's how the system found out my preference. How that preference is remembered is that when I selected a mirror, my browser got a cookie with the information in it. Now, every time that I go to this same `CTANfind` page, the browser presents this cookie and the information is used in the generated results.

Another thing that this page will do for you is to help answer questions like, "Is there an already written solution to ...?" That's because it has the ability to search Graham Williams's wonderful Catalogue<sup>1</sup> of information on T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X. For instance, one of the things I struggled with first in trying L<sup>A</sup>T<sub>E</sub>X was to have a letter place the closing on the left. If only I had `CTANfind`—searching the Catalogue for 'letter' gives perhaps two dozen responses, at least one of which, `block`, does the job.

There is one more thing that this tool can do for you. A design target for this page was that from it you could quickly answer half of the questions from a day's `comp.text.tex`. So included at the

end are a few links, to the UK TUG FAQ, to the *Short Introduction to L<sup>A</sup>T<sub>E</sub>X*, etc., that seem to me to be the most often referenced.

I hope that you find this tool useful.

#### 2 A Request For Help

Here are the technical details: Behind the web page are two CGI scripts that in turn rely on data files generated as cron jobs (the programs are in Perl 5). These were written to be easily set up at other sites. If you are a CTAN mirror and you are willing to try hosting your own version then [joshua.smcvt.edu/ctan/install\\_search.shtml](http://joshua.smcvt.edu/ctan/install_search.shtml) should make installation on any UNIX system straightforward; no editing of Perl source.

Having a version of `CTANfind` running on each continent would be great (although Antarctica might be tough!). Right now, you can go to [tug.ctan.org/CTANfind.html](http://tug.ctan.org/CTANfind.html) and [joshua.smcvt.edu/ctan/CTANfind.html](http://joshua.smcvt.edu/ctan/CTANfind.html) in the USA, and [www.tex.ac.uk/CTANfind.html](http://www.tex.ac.uk/CTANfind.html) in England.

#### 3 With a Little Help From My Friends

The net can be such a fine place. A number of people let me steal, err . . . , borrow their ideas (some of which they were patient enough to explain to me at length first). I'd particularly like to thank Karl Berry, Robin Fairbairns, and Graham Williams.

◇ Jim Hefferon  
 Department of Mathematics  
 Saint Michael's College  
 Colchester, VT 05439, USA  
[tex@joshua.smcvt.edu](mailto:tex@joshua.smcvt.edu)  
<http://joshua.smcvt.edu/hefferon.html>

<sup>1</sup> in `\help\Catalogue\catalogue.html` on CTAN

## Hints & Tricks

**“Hey — it works!”**

Jeremy Gibbons

Welcome again to “*Hey — it works!*”, a column devoted to  $\LaTeX$  and  $\text{META}$  tips and tricks. This issue is devoted to a single topic, a sequel to an earlier article on creating ornamental rules: we show how to construct ornamental boxes out of individual symbols.

Please note that I have moved. My new contact details are given at the end of this article. Unfortunately, mail is not being forwarded from my old address; I apologize profusely for any inconvenience

that this may have caused. This column is being archived at <http://users.comlab.ox.ac.uk/jeremy.gibbons/hiw/>.

### Ornamental boxes

In this column in *TUGboat* 19:4, Christina Thiele showed how to produce ornamental rules constructed from ordinary characters:

```
*****
```

It is also fun to generate ornamental *boxes* out of ordinary characters:

```
*****
* shake the yoke of *
* inauspicious stars *
*****
```

This article shows how. We start off with a simple macro, and elaborate on it in stages.

#### First attempt: a single symbol

Our first attempt constructs an ornamental box out of copies of a single symbol, as in the example above. The macro `\boxitA` takes two arguments: the contents to be boxed, and the symbol (in fact, any horizontal material) that will be used to surround it. The first step is shift these two boxes vertically, if necessary, so that they have zero depth.

```
\def\boxitA#1#2{%
  \setbox0=\hbox{#1}% the box contents
  \setbox0=\hbox{\raise\dp0\box0}%
  \setbox1=\hbox{#2}% the ornament
  \setbox1=\hbox{\raise\dp1\box1}%
```

Unfortunately, Christina's elegant use of leaders doesn't work as well for boxes as it does for rules; we have to achieve the same effects manually. We compute precisely how many instances of the symbol are required, horizontally and vertically, to exceed the dimensions of the contents; call these two numbers *m* and *n*. Each number is the size of the contents divided by the size of the symbol, rounded up to the nearest integer; we round upwards by first adding the size of the ornament less one.

```
\count0=\wd0 \advance\count0 by \wd1
\advance\count0 by -1 \divide\count0 by \wd1
\count1=\ht0 \advance\count1 by \ht1
\advance\count1 by -1 \divide\count1 by \ht1
```

The dimensions of the contents may not be exact multiples of the size of the symbol, so we wrap the contents in the smallest enclosing box with such dimensions:

```
\setbox0=\hbox to \count0\wd1{%
  \hfil\vbox to \count1\ht1{%
    \vfil\box0\vfil}\hfil}%
```

Finally, we construct the ornamental box, with *m*+2 instances of the symbol at the top and bottom, and *n*+2 instances at the left and right:

```
\hbox{\vbox{\offinterlineskip
  \hbox{\copy1%
    \duplicate{\count0}{\copy1}%
    \copy1}
  \hbox{\vbox{\duplicate{\count1}{\copy1}}%
    \copy0%
    \vbox{\duplicate{\count1}{\copy1}}}}
  \hbox{\copy1%
    \duplicate{\count0}{\copy1}%
    \copy1}
}}%
}}
```

Here, the macro `\duplicate` generates a given number (its first argument) of copies of a given text (its second argument):

```
\def\duplicate#1#2{% #1 copies of #2
  \count2=#1%
  \loop
  #2%
  \advance\count2 by -1
  \ifnum \count2>0 \repeat}}
```

For example, the box at the start of this article was generated by the code

```
\boxitA{\begin{tabular}{c}
  shake the yoke of \\
  inauspicious stars
\end{tabular}}
{${ast$}}
```

#### Second attempt: multiple symbols

The first attempt gave the general idea; however, it would be nice to be able to use different symbols for the four edges and the four corners. In this second attempt, we provide such a facility. However, for simplicity we assume that all eight symbols are the same size. For example:

```
\ \downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\swarrow
\rightarrow bring me my \leftarrow
\rightarrow arrows of desire \leftarrow
\nearrow \uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\searrow
```

As before, we start by making the contents and all eight of the symbols sit on the baseline:

```
\def\boxitB#1#2#3#4#5#6#7#8#9{%
  % contents TL T TR L R BL B BR
  \setbox0=\hbox{#1}% the box contents
  \setbox0=\hbox{\raise\dp0\box0}%
  \setbox1=\hbox{#2}% #2 to #9 the ornaments
  \setbox1=\hbox{\raise\dp1\box1}%
  \setbox2=\hbox{#3}%
  \setbox2=\hbox{\raise\dp2\box2}%
  \setbox3=\hbox{#4}%
  \setbox3=\hbox{\raise\dp3\box3}%
  \setbox4=\hbox{#5}%
  \setbox4=\hbox{\raise\dp4\box4}%
  \setbox5=\hbox{#6}%
  \setbox5=\hbox{\raise\dp5\box5}%
  \setbox6=\hbox{#7}%
```



```

\setbox6=\hbox{\raise\dp6\box6}%
\setbox7=\hbox{\#8}%
\setbox7=\hbox{\raise\dp7\box7}%
\setbox8=\hbox{\#9}%
\setbox8=\hbox{\raise\dp8\box8}%

```

(It would be nice to do this with a loop, but unfortunately you cannot use a counter to access a macro parameter.) Then we compute the number of symbols required, horizontally and vertically, and pad the contents accordingly:

```

\count0 = \wd0 \advance\count0 by \wd1
\advance\count0 by -1 \divide\count0 by \wd1
\count1 = \ht0 \advance\count1 by \ht1
\advance\count1 by -1 \divide\count1 by \ht1
\setbox0=\hbox to \count0\wd1{%
  \hfil\vbox to \count1\ht1{%
    \vfil\box0\vfil}\hfil}%

```

Finally, we construct the ornamental box, taking care to use the correct symbol for each position:

```

\hbox{\vbox{\offinterlineskip
  \hbox{\copy1%
    \duplicate{\count0}{\copy2}%
    \copy3}
  \hbox{\vbox{\duplicate{\count1}{\copy4}}%
    \copy0%
    \vbox{\duplicate{\count1}{\copy5}}%
  \hbox{\copy6%
    \duplicate{\count0}{\copy7}%
    \copy8}
}}%
}}

```

In order to use this macro, we need a means of making all eight symbols the same size. The macro `\resizeW` solves this problem: it yields its first argument, but centred in the width of its second argument. (Fortunately, all eight arrows are the same height, so no vertical adjustment is necessary.)

```

\def\resizeW#1#2{% #1, but to width of #2
  \setbox0=\hbox{\#2}%
  \rlap{\hbox to \wd0{\hfil#1\hfil}}%
  \phantom{\box0}%
}

```

Then the box constructed out of eight arrows can be generated by

```

\boxitB{\itshape
  \begin{tabular}{c}
    bring me my \
    arrows of desire
  \end{tabular}}%
{\resizeW{\$searrow$} {\$searrow$}}
{\resizeW{\$downarrow$} {\$searrow$}}
{\resizeW{\$swarrow$} {\$searrow$}}
{\resizeW{\$rightarrow$} {\$searrow$}}
{\resizeW{\$leftarrow$} {\$searrow$}}
{\resizeW{\$nearrow$} {\$searrow$}}

```

```

{\resizeW{\$uparrow$} {\$searrow$}}
{\resizeW{\$narrow$} {\$searrow$}}

```

### Third attempt: different shapes

A little reflection suggests that there is no need for all eight ornaments to be the same size; all that is required is for those symbols that will be aligned together to have matching sizes in the appropriate dimension. Thus, if we call the four edge symbols T, B, L and R, and the four corner symbols TL, TR, BL and BR, then:

- TL, L, BL should have the same width;
- T, B should have the same width;
- TR, R, BR should have the same width;
- TL, T, TR should have the same height;
- L, R should have the same height;
- BL, B, BR should have the same height.

The only change required to the macro is to make sure the appropriate symbols are used when it comes to computing the number of symbols required:

```

\def\boxitC#1#2#3#4#5#6#7#8#9{%
  % contents TL T TR L R BL B BR
  \setbox0=\hbox{\#1}% the box contents
  ...
  \count0 = \wd0 \advance\count0 by \wd2
  \advance\count0 by -1 \divide\count0 by \wd2
  \count1 = \ht0 \advance\count1 by \ht4
  \advance\count1 by -1 \divide\count1 by \ht4
  \setbox0=\hbox to \count0\wd2{%
    \hfil\vbox to \count1\ht4{%
      \vfil\box0\vfil}\hfil}%
  ...
}

```

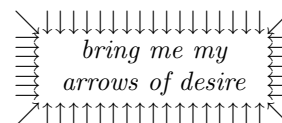
Now it is possible to dispense with the resizing; we can write simply

```

\boxitC{\itshape
  \begin{tabular}{c}
    bring me my \
    arrows of desire
  \end{tabular}}%
{\$searrow$}{\$downarrow$}{\$swarrow$}
{\$rightarrow$}{\$leftarrow$}{\$nearrow$}
{\$uparrow$}{\$narrow$}

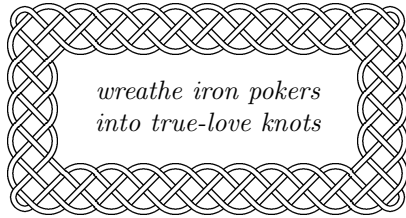
```

to generate

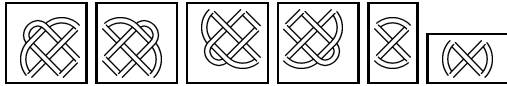


### Knotwork

Sadly, there is a shortage of good symbols for creating such ornaments; not many typographic elements come in eight different orientations! However, there is nothing to stop you designing your own symbols:



This ornamental box uses a font of six different knotwork components:



(the top and bottom edges use the same symbol, as do the left and right edges). The designs are based on those in the book *Celtic Knotwork Designs* by Sheila Sturrock (Guild of Master Craftsman Publications, 1997).

- ◇ Jeremy Gibbons  
 Oxford University Computing  
 Laboratory  
 Wolfson Building, Parks Road  
 Oxford OX1 3QD, UK  
[jeremy.gibbons@comlab.ox.ac.uk](mailto:jeremy.gibbons@comlab.ox.ac.uk)  
[http://www.comlab.ox.ac.uk/  
 oucl/people/jeremy.gibbons.  
 html](http://www.comlab.ox.ac.uk/oucl/people/jeremy.gibbons.html)

out about new or upgraded packages, as well as continuing (in future editions) to feature packages that deal with specific topics.

It also seemed only logical to include an overview of CTAN's structure, as well as a few tips on how to try and identify packages that may well prove invaluable to your work: the trouble with a rich resource is that it's very hard to digest, and CTAN certainly can prove frustrating when you want to find a solution *now*!

Another new feature (end of the column) will be a list of upgrades to packages already described in past editions of this column so that you can confirm that you have the latest versions at your disposal.

So, let's start with some CTAN pointers, some notes on its structure, and then plough on into the inventory.

### CTAN pointers

If you haven't visited CTAN or aren't sure there's anything there for you, here are some points to keep in mind.

If you're getting *TUGboat*, you've also been getting periodic updates of the CTAN holdings on a CD-ROM (produced by DANTE). Track it down if it's not on your shelf and use it as a quick way to access and examine packages. Of course, it's only a 'snapshot' but it already contains a massive quantity of material; if you find something you like, then all you need do is check CTAN periodically for any upgrades.

2. The easiest way to access CTAN itself is to use your web browser and head to [www.tug.org](http://www.tug.org), and then follow the links to CTAN (middle of the opening page, middle column, under 'Software'). Remember to add the useful pages to your bookmarks!

The `ftp` links to the three backbone servers are there, as well as links to mirror sites around the world, and also to some handy search engines.

3. Some packages are contained within a single file while others may require several files to be fetched and then processed correctly. It's best to read any `readme` file to make sure you pick up all that's needed.

Process the files on your machine, move the resultant files to where the instructions tell you, and then print up the documentation and put it in a binder. No point in having it somewhere in e-form if you prefer making notes by hand.

4. There are lots of ways of doing (almost) the same thing, so if one package doesn't quite work for you (provided you've done the right things to install it) it's quite likely that another package is lurking somewhere. The Treasure Chest



## The Treasure Chest

CTAN, the Comprehensive T<sub>E</sub>X Archive Network, is a huge beast! And the T<sub>E</sub>X community is constantly feeding the creature with new packages and fonts and implementations and tools and utilities ... to the point where it seems impossible to know what's available and even then, *where* it's to be found.

So, in an effort to try and show you the richness of the feast available at the Archive, this edition of the Treasure Chest column will focus on a new component — a quarterly inventory of new packages. And while it's true that there are search mechanisms on the web for CTAN, not everyone has time to browse with their browser. We'd like this column to become another means of letting T<sub>E</sub>X users find

column is one small attempt at bringing packages to the fore, mainly with reference to a general topic: headings, or floats, or special layouts (critical editions, linguistics, music, etc.), and so on. If you can't find what you're looking for, a post to one of the newsgroups or lists can often dislodge a suggestion or two.

5. Another source of information on packages is the UK TUG's FAQ, also available via the TUG webpages.
6. You should also make a note that a directory may have a further `misc/` directory; this is often where very small packages are stowed.
7. And remember: it's often likely that a layout problem you're encountering has stumped someone else too, to the point where they sat down, solved their problem, and then decided to make their work available to the whole community.

If you're able to find your solution via CTAN, take a moment and send a thank-you note to the developer — they'll surely appreciate the acknowledgment.

### CTAN and its subtrees

The top of the CTAN tree is `tex-archive`. The table on the next page shows the 16 subtrees (at present) and the purpose of each.

It helps to become familiar with these general categories so that you don't waste time looking for something in an unlikely subtree. For example, material related to `BIBTEX` is found in the main subtree `biblio/bibtex/contrib/`, and not down in `macros/latex/contrib/supported/`.

To help you navigate around the archive and locate files there are some excellent webpages — I'd recommend bookmarking these two:

- `index.html`, found in `tex-archive`, lists the intent of each sub-tree
- In the above-mentioned HTML document, go to 'Help', and click on the 'catalogue', which is Graham Williams' 'The `TEX` Catalogue Online', which provides incredible options for locating and accessing packages old and new. The Brief Index (300K) has a short description of each entry and a link to the full entry.

If you have a favorite tool, let me know and we'll mention it in a future edition.

### Packages posted to CTAN

Notes to keep in mind:

1. Packages are listed in the month of the latest upgrade; that is, if a version is posted in Oct. and a new version in Dec., only the newer one will be listed.

2. Monthly entries are in alpha-order, for easier reference.
3. Unless otherwise stated, packages are located in:

`macros/latex/contrib/supported/`

However, there are two main branches under `contrib/`: `supported/` and `other/`. The default used here is `supported`, and where `other` is the location, its path will simply be noted as `.../other/`, to cut down on path length.

### October 1999

**bibunits:** A package to provide multiple references in one document (upgrade to v. 2.1 provides compatibility with `overcite` and `natbib`, and makes `cite` more robust).

**cmactex:** in `nonfree/systems/mac/`

Upgrades: Two files, `doc.sit.bin` and `texmflib.sit.bin`, are replacements and contain minor corrections to existing archives. `Pdftex14c.sit.bin` is the Macintosh port of v. 0.14c of `pdfTEX`.

**comment:** in `.../other/`

A package to selectively in/exclude pieces of text: the user can define new comment versions, and each is controlled separately (upgrade to v. 3.6).

**ctib4tex:** in `language/tibetan/`

`ctib4tex` contains "Tibetan for `TEX` and `LATEX 2ε`", a package using a modified version of Sirlin's Tibetan font. The great advantage of this Tibetan implementation is that all consonant clusters are formed by `TEX` and `METAFONT`. No external pre-processor is needed.

**dotlessi:** in `.../supported/bezos/`

`Dotlessi` provides dotless i's and j's for use in any math font (`\mathrm`, `\mathsf`, etc.).

**easy:** A family of packages for equation environments, block matrices, tables, vectors, and customising bibliographies.

**epmtfe:** in `systems/os2/`

EPM `TEX` Front End is an integrated `TEX` environment for OS/2, based on the EPM editor. Some support was added for using the `TEX` Front End with `VTEX/2` and for viewing PDF files (upgrade to v. 2.5).

**fonteinf.pdf:** in `info/german/`

This is a translation into German of `tipos.pdf`, kindly made by Thomas Ruedas.

**geometry:** This package provides an easy and flexible user interface to customize page layout. It implements auto-centering and auto-balancing mechanisms so that users have only to give the least description for the page layout (upgrade to v. 2.2).

**gloss:** `gloss` is a package which allows the creation of glossaries using `BIBTEX` (upgrade to v. 1.4).

**hyperref:** This package is used to emend cross-referencing commands in `LATEX` to produce some sort of `\special` commands. Upgrade v. 6.67 is a stabilizing release, committing all the small test enhancements made over the last few months. This includes

### CTAN sub-trees

**biblio/** systems for maintaining and presenting bibliographies within documents typeset using  $\TeX$

**digests/** collections of  $\TeX$  mailing list digests,  $\TeX$ -related ‘electronic magazines’, and indexes, etc., of printed publications

**dviware/** printer drivers and previewers, etc., for  $.dvi$  files

**fonts/** fonts written in METAFONT, and support for using fonts from other sources (e.g., those in Adobe Type 1 format)

**graphics/** systems and  $\TeX$  macros for producing graphics

**help/** FAQs and similar direct assistance; the catalogue

**indexing/** systems for maintaining and presenting indexes of documents typeset using  $\TeX$ .

**info/** manuals and extended how-to information; errata for  $\TeX$ -related publications, collections of project (e.g.,  $\LaTeX$  and NTS) documents, etc.

**language/** support for various languages

**macros/**  $\TeX$  macros; several directories have significant sub-trees:  
     **macros/context/** the Context distribution  
     **macros/generic/** macros that work in several environments  
     **macros/latex/** the  $\LaTeX$  distribution and contributed matter  
     **macros/plain/** Donald Knuth’s example macro set

**support/**  $\TeX$  support environments and the like

**systems/**  $\TeX$  systems; organised by operating environment, but also including:  
     **systems/knuth/** Donald Knuth’s current distribution  
     **systems/generic/** Complete systems that can potentially operate in more than one operating environment

**tds/** the  $\TeX$  Directory Structure standard (the output of the TUG TDS working group)

**tools/** tools of use for the archive maintainers (including mirrors of the source of the compression tools the archives use)

**usergrps/** information supplied by  $\TeX$  user groups

**web/** ‘Literate Programming’ tools and systems

For your information, the following names are found along with the 16 subtrees under `tex-archive` but are symbolically linked to one of the actual subtrees:

```
archive-tools --> tools/
bibliography --> biblio/
dante         --> usergrps/dante
documentation --> info/
languages     --> language/
```

restructuring the source tree, putting test files and documentation in their own directories.

**mathsPIC:** in `graphics/pictex/`

MathsPIC is a DOS filter program for use with  $\text{P}\text{I}\text{C}\text{T}\text{E}\text{X}$  (v. 1.8f). There is a 37-page manual in `.tex`, `.dvi`, and `.ps` formats.

**pdfscreen:** This package helps to redesign the PDF output of your normal documents fit to be read in a

computer monitor while retaining the freedom to format it for conventional printing.

**pdfslide:** This is a package for use with  $\text{pdf}\text{T}\text{E}\text{X}$ , to make nice presentation slides.

**pitthesis:** This is a  $\text{L}\text{A}\text{T}\text{E}\text{X} 2\epsilon$  document class package for writing theses at the University of Pittsburgh, PA.

**pstoedit:** in `support/`  
**pstoedit** converts PostScript and PDF files to other vector graphic formats so that they can be edited graphically.

**rtf2latex2e:** in `support/`  
**rtf2latex2e** converts Rich Text Format files to  $\LaTeX$  2 $\epsilon$ . It runs on Mac, UNIX, Linux, and Win (upgrade to v. 0.263).

**tipos.pdf:** in `info/spanish/`  
 Describes (in Spanish) the large variety of types of files for fonts (*tipo* means ‘font’ in Spanish).

**topfloat:** A package to move any type of float material to the top of the page.

### November 1999

**bizcard:** A package for typesetting business/visiting/calling cards (upgrade to v. 1.1).

**ccfonts:** A package and some of the necessary `.fd` files to use the Computer Concrete fonts with  $\LaTeX$ .

**expressg:** in `graphics/metapost/contrib/macros/`  
 This METAPOST package provides facilities to assist in drawing diagrams that consist of boxes, lines, and annotations. Particular support is provided for creating EXPRESS-G diagrams.

**latex2rtf:** in `support/`  
**latex2rtf** is a translator program that translates  $\LaTeX$  text into the RTF format used by several text processors, including Microsoft Word and Word for Windows (patches to v. 1.8a). The distribution is made for use within the MS-DOS window of Win95 and Win3.11, but all sources can be compiled on UNIX computers having GCC compilers.

**makefonts:** in `fonts/utilities/`  
 The package contains shell scripts which cause `.pk` files to be generated (upgrade to v. 2.0).

**merlin:** in `.../supported/custom-bib/`  
 Part of the `custom-bib` package for generating customized  $\BIB\TeX$  styles from a generic file by means of the `docstrip` program that is part of the  $\LaTeX$  installation (upgrade to v. 4.00).

**Metafp.ps:** in `info/`  
 A PostScript article entitled “Some Experiences in Running METAFONT and METAPOST.”

**multirow:** Bug fixes to `multirow.sty`, and new package, `bigdelim`, an application of `multirow` for delimiters inside arrays and tabulars.

**parskip:** in `.../other/misc/`  
 Package to be used with any document class at any size, which produces the following paragraph layout: zero `\parindent` and non-zero `\parskip`. The package is no more than a quick fix; the proper way to achieve effects as far-reaching as this is to create a new class.

**permute:** The `permute` package inputs, outputs and composes permutations (upgrade to v. 0.12).

**qbibman:** in `biblio/bibtex/utills/`  
**qbibman** is a graphical front-end to `BibTool` based on the Qt library.

**rmligs:** in `support/`  
 This is a program for removing incorrectly used ligatures from  $\LaTeX$  documents. This version is intended for German-language texts only.

**scrnger:** in `.../supported/koma-script/contrib/`  
 The Koma-Script packages seek to implement European rules of typography and paper formats, as documented in Tschichold. `scrnge` adds support for the language `ngerman` to the current versions of `scrlettr.cls` and `scrdate.sty`.

**snapshot:** This package helps users obtain a list (a ‘snapshot’) of the external dependencies of a  $\LaTeX$  document, in a form that can be embedded at the top of the document. Such a dependency list makes it possible to arrange that the document be processed is always with the same versions of everything, in order to ensure the same output.

**Songbook:** The package provides a core set of functions for the production of songbooks (upgrade to v. 3.1). See also: [cyberus.ca/~crath/Misc/Songbook/](http://cyberus.ca/~crath/Misc/Songbook/).

**TeEncontreX:** in `documentation/spanish/`  
 Documentation that attempts to collect and centralize all the available data about  $\TeX$  so that anyone can find information in one place. People can add new articles (very easily) to the database. All its contents may be found at [ctv.es/USERS/irmina/TeEncontreX.html](http://ctv.es/USERS/irmina/TeEncontreX.html).

### December 1999

**aeguill:** A package intended to add French guillemets to the `ae` package. The guillemets are built with the `wncyr` fonts (by default), or with either the `lasy` fonts or the EC fonts (upgrade to v. 0.97).

**bakoma:** in `nonfree/systems/win32/`  
 Upgrade of `BaKoMa  $\TeX$`  (v. 2.21). For more information about changes: [.../win32/bakoma/dst/changes.html](http://.../win32/bakoma/dst/changes.html).

**BibTool:** in `biblio/bibtex/utills/`  
**BibTool** allows the manipulation of  $\BIB\TeX$  files in a way that goes beyond the possibilities—and intentions—of  $\BIB\TeX$  (upgrade to v. 2.44).

**braket:** in `.../other/misc/`  
 Macros for Dirac bra-ket `<|>` notation and sets `{|}` (update). Fixed and expanding sizes provided. This minor revision will use  $\epsilon$ - $\TeX$ 's `\middle` primitive if it is available.

**calxxx:** in `.../other/`  
**calxxx.tex** prints a card-size calendar for any year, AD or BC, with Gregorian or Julian leap rules (useful for years before the adoption of Gregorian rules).

**contour:** The package generates a colored contour around a given text in order to enable printing text over a background without the need for a color box around the text (upgrade to v. 1.03).

**dichokey:** The package can be used to construct dichotomous identification keys (used especially in

- biology for species identification), taking care of numbering and indentation of successive key steps automatically. Run the example file!
- dvi:** in `dviware/`  
**dvi** (.dvi file information utility) is a utility written in C that extracts information from a  $\TeX$  .dvi file (upgrade to v. 0.27).
- extsizes:** in `.../other/`  
 This package provides two classes: **extarticle** and **extreport**, which allow for documents with a base font of size 8–20pt.
- fotex:** in `macros/`  
 Package used to process XSL formatting objects when serialized as XML by an XSL processor (package updates).
- grverb:** in `languages/greek/package-babel/`  
 This package addresses the issue of writing both Latin and Greek verbatim text, particularly useful for computer listings and, more generally, in computer science-related texts. A public domain font is used by both the command and the environment. The font conforms to the ISO-8859-7 encoding for the Greek language.
- jadetex:** in `macros/`  
 Package used to process the output of the Jade DSSSL processor in its  $\TeX$  mode (package updates).
- lgrind:** in `support/`  
**LGrind** is a source code pretty-printer; it converts program text from many languages into  $\LaTeX$ . This new version features a Python mode (upgrade to v. 3.64).
- ltx2x:** in `support/`  
 The **ltx2x** program (written in C) replaces  $\LaTeX$  commands in a document by user-defined strings. In essence,  $\LaTeX$  tags can be replaced by other kinds of document tags (e.g., HTML/SGML or RTF) or can be removed altogether (**de $\TeX$** ). The program also has an unsophisticated pretty-print capability (upgrade to v. 0.92).
- matlabweb:** in `web/`  
 A literate programming system for the Matlab language. Similar to **CWEB**, created with a slightly modified version of the Spider system. Can be used with plain  $\TeX$  or  $\LaTeX$ , the latter with help from the **webfiles** package.
- miktex:** in `systems/win32/miktex/1.20`  
**Mi $\TeX$**  is a free  $\TeX$  distribution for Windows 9x and Windows NT. The latest release is 1.20, to be found in the 1.20 sub-folder.
- nestquot:** in `.../other/`  
 Quotes that change between double and single according to their nesting level.
- passivetex:** in `macros/`  
 Shared macros for **Jade $\TeX$**  and **FOT $\TeX$** , include the XML parser and UTF8 handler (written by David Carlisle), and all the general support for characters in Unicode (package updates).
- psrip:** in `support/`  
 A Perl script that extracts images from PostScript files.
- rsc.patch:** in `.../rsc/contrib/`  
 A patch for the **rsc** package, which utilizes the inclusion of RCS-supplied data in  $\LaTeX$  documents.
- refcheck:** **refcheck.sty** is intended to check references. It looks for numbered but unlabelled equations, for labels not used in the text, for unused bibliography references (upgrade to v. 1.6).
- texshell32:** in `systems/win32/`  
 This program is a free shell for the typesetting system  $\TeX$ . It runs under Windows 95, 98 and NT (upgrade to v. 0.61).
- ut-thesis:** A class file for formatting documents according to the School of Graduate Studies' (SGS) guidelines (07/97) for theses at the University of Toronto (upgrade to v. 1.8).
- webfiles:** in `web/`  
 A  $\LaTeX$  package for inclusion of several **CWEB** and/or Spidery **WEB** documents in a single  $\LaTeX$  document.

### Tour package upgrades

- acronym:** New version: 1.4, dated 2000/02/09.  
 The package is for: acronyms, mottos, topical quotations; full and short versions; auto-generation of acronym listing (*TUGboat* 20,1).
- epigraph:** New version: 1.5, dated 2000/02/20.  
 The package is designed for typesetting epigraphs — pithy quotations often found at the start (or end) of a chapter (*TUGboat* 20,1).
- hanging:** New version 1.1, dated 1999/05/01.  
 The package provides facilities for defining hanging paragraphs and hanging punctuation (*TUGboat* 20,1).
- paralist:** New version: 1.9, dated 2000/03/05.  
 The package provides itemized and enumerated lists that can be typeset within paragraphs, as paragraphs and in a compact version (*TUGboat* 20,1).
- soul.sty:** New version: 1.3, dated 1999/05/15.  
 The package provides hyphenate-able letterspacing, underlining, and some variations on each (*TUGboat* 19,4)

◇ Christina Thiele  
 15 Wiltshire Circle  
 Nepean, Ontario  
 K2J 4K9 Canada  
 cthiele@ccs.carleton.ca

# L<sup>A</sup>T<sub>E</sub>X News

Issue 12, December 1999

## *LPPL update*

Since the release of the L<sup>A</sup>T<sub>E</sub>X Project Public Licence version 1.1, we have received a small number of queries which resulted in some minor changes to improve the wording or explain the intentions better. As a consequence this release now contains LPPL 1.2 in the file `lppl.txt` and the previous versions as `lppl-1-0.txt` and `lppl-1-1.txt`.

## *fixltx2e package*

This package provides fixes to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> which are desirable but cannot be integrated into the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> kernel directly as they would produce a version incompatible to earlier releases (either in formatting or functionality).

By having these fixes in the form of a package, users can benefit from them without the danger that their documents will fail, or produce unexpected results, at other sites; this works because a document will contain a clear indication (the `\usepackage` line, preferably with a required date) that at least some of these fixes are required to format it.

## *Outcome of TUG '99 (Vancouver)*

The slides from the TUG'99 presentation we gave on *a new interface for L<sup>A</sup>T<sub>E</sub>X class designers* are available from the L<sup>A</sup>T<sub>E</sub>X Project website; look for the file `tug99.pdf` at:

<http://www.latex-project.org/talks/>

Please note that this document was intended only to be informal “speaker’s notes” for our own use. We decided to make them available (the speaker’s notes as well as the slides that were presented) because several people requested copies after the talk. However, they are *not* in a polished copy-edited form and are not intended for publication.

Prototype implementations of parts of this interface are now available from:

<http://www.latex-project.org/code/experimental/>

We are continuing to add new material at this location so as to stimulate further discussion of the underlying concepts. As of December 1, 1999 the following parts can be downloaded.

**xparse** Prototype implementation of the interface for declaring document command syntax. See the `.dtx` files for documentation.

**template** Prototype implementation of the template interface (needs parts of **xparse**).

The file `template.dtx` in that directory has a large section of documentation at the front describing the commands in the interface and giving a ‘worked example’ building up some templates for caption formatting.

**xcontents** Interface description for table of contents data (no code yet). Coding examples have been thoroughly discussed on the `latex-1` list.

**xfootnote** Working examples for generating footnotes, etc. Needs **xparse** and **template**.

All examples are organised in subdirectories and additionally available as **gzip tar** files.

Please remember that this material is intended only for experimentation and comments; thus any aspect of it, e.g., the user interface or the functionality, may change and, in fact, is very likely to change. For this reason it is explicitly forbidden to place this material on CD-ROM distributions or public servers.

These concepts, as well as their implementation, are under discussion on the list L<sup>A</sup>T<sub>E</sub>X-L. You can join this list, which is intended solely for discussing ideas and concepts for future versions of L<sup>A</sup>T<sub>E</sub>X, by sending mail to `listserv@URZ.UNI-HEIDELBERG.DE` containing the line

SUBSCRIBE L<sup>A</sup>T<sub>E</sub>X-L *Your Name*

This list is archived and, after subscription, you can retrieve older posts to it by sending mail to the above address, containing a command such as:

GET L<sup>A</sup>T<sub>E</sub>X-L LOGyy`mm`

where `yy`=Year and `mm`=Month, e.g.

GET L<sup>A</sup>T<sub>E</sub>X-L LOG9910

for all messages sent in October 1999.



# L<sup>A</sup>T<sub>E</sub>X

## Scaled Pictures in L<sup>A</sup>T<sub>E</sub>X

Bruce Shawyer

Three years ago, I wrote macros to extend the picture environment, to allow for the automatic re-sizing of the picture. It behaved almost exactly like the picture environment, and was used thus:

```
\begin{scalepicture}{n}(xlen,ylen)
[(xllcorner,yllcorner)]
% picture commands
\end{scalepicture}
```

Here

1. “*n*” is the percentage of a “full picture”. A “full picture” is one that fills up the line, leaving a small margin on each side, and has  $n = 100$ ;

Exactly as in the picture environment:

2. “*xlen*” is the coordinate length needed across the breadth of the picture (must be a positive integer);
3. “*ylen*” is the coordinate length needed across the height of the picture (need not be an integer);
4. “(xllcorner,yllcorner)” are the coordinates of the bottom left hand corner of the picture. If these are (0,0), then this can be omitted.

The reason for this macro is to eliminate the need to calculate the `\unitlength`, and to enable pictures to be re-sized easily.

Thanks to comments from A.S. Berdnikov, O.A. Grineva and S.B. Turta, in *TUGboat*, Vol. 17, #2, pp. 229-232, 1996, I changed the macro to use `\hsize` and not `\textwidth` for the calculation, and so it works properly inside minipages and in tabular paragraph mode, etc.

However, there were two problems with changing the percentage number that required further attention:

1. The size of the font remained the same, no matter the percentage of a “full picture”;
2. The positions of the labels usually needed adjustments.

I have tried to address both these problems and now have a working version, available from CTAN under the name **fullpict.sty**.

1. I have created a new environment:

```
\begin{scaledpicture}{n}(xlen,ylen)
[(xllcorner,yllcorner)]
% picture commands
\end{scaledpicture}
```

This addresses the font size problem in the following way: for a normal line of text, such as North American 8.5×11 paper, or European A4 paper, with full page or standard article style,

Percentage number	Font size
< 39	<code>\tiny</code>
≥ 39 and < 49	<code>\scriptsize</code>
≥ 49 and < 59	<code>\footnotesize</code>
≥ 59 and < 74	<code>\small</code>
≥ 74	<code>\normalsize</code>

2. Within minipages and in tabular paragraph mode, the choice of the font size is further adjusted to reflect the smaller size of the picture.
3. Also, the placing of labels can be helped with the commands:

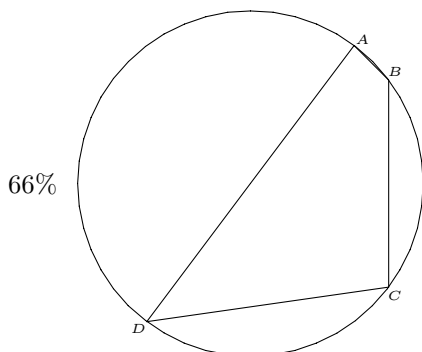
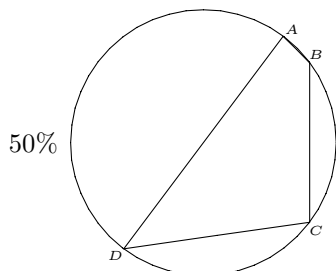
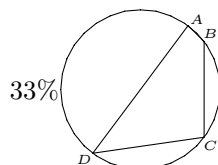
- (a) `\cput(a,b){label}` — put centered at the point  $(a, b)$ .
- (b) `\eput(a,b){label}` — put to the east (right and centered) of the point  $(a, b)$ .
- (c) `\nput(a,b){label}` — put to the north (above and centered) of the point  $(a, b)$ .
- (d) `\wput(a,b){label}` — put to the west (left and centered) of the point  $(a, b)$ .
- (e) `\sput(a,b){label}` — put to the south (below and centered) of the point  $(a, b)$ .
- (f) `\neput(a,b){label}` — put to the north-east of the point  $(a, b)$ .
- (g) `\nwput(a,b){label}` — put to the north-west of the point  $(a, b)$ .
- (h) `\swput(a,b){label}` — put to the south-west of the point  $(a, b)$ .
- (i) `\seput(a,b){label}` — put to the south-east of the point  $(a, b)$ .

4. If you want a more accurate placement of the label, or to use a different radius from the default, make use of the command `\angleput{degrees}[radius](a,b){label}`.

Here, **degrees** represents the angle from the positive direction of the  $x$ -axis (may be positive or negative), and the optional **[radius]** allows multiplication of the default radius (1) by **radius** (may be positive or negative).

It may be of interest to note that, instead of using a circle for the placement of points, a polygon with 24 sides was used. This made the calculations considerably easier, using linear functions instead of trigonometric functions. This becomes noticeable, only for larger values of **radius**. It seems unlikely that such values would be used in practice.

These are used in the diagrams illustrated below:



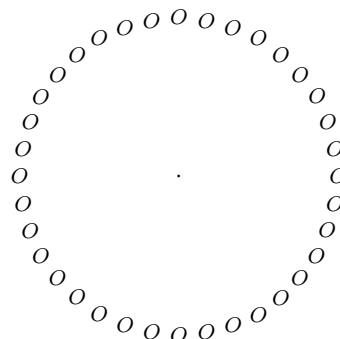
Geographic positions



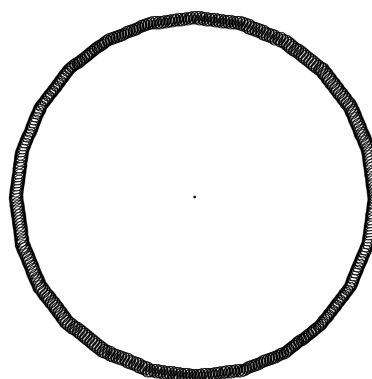
24 points at distance zero — 15 degree intervals



24 points at distance 2.2 — 15 degree intervals



36 points at distance 10 — 10 degree intervals



360 points at distance 13.5 — 1 degree intervals

The environment `scaledpicture` automatically centers the picture. If you do not want it centered, use the environment `Scaledpicture`. These macros are incorporated in `fullpict.sty`, which is available from CTAN `tex-archive/macros/latex/contrib/supported/fullpict/`, or by anonymous ftp from

**isthmus.math.mun.ca**

directory `/pub/cruX`.

Please report any bugs or criticisms to me.

- ◇ Bruce Shawyer  
 Department of Mathematics and  
 Statistics  
 Memorial University of  
 Newfoundland  
 St. John's, NF  
 Canada A1C 5S7  
**bshawyer@math.mun.ca**

# Tutorial

## Book Design for $\text{\TeX}$ Users

### Part 2: Practice

Philip Taylor

Every book will have one figure that cannot be seen from its point of reference.

#### Abstract

In the predecessor to this paper,<sup>1</sup> three fundamental concepts of *uniformity*, *information* and *structure* were introduced, and general guidance given on each of them. In this paper, more practical advice is given, specifically in two areas: guidance on actual dimensions, proportions and layout; and guidance on implementing some of the ideas through the medium of the  $\text{\TeX}$  language. Finally, some difficult (and even insoluble) problems in layout are discussed.

-- \* --

#### 1 How big is a book?

Just as we are all familiar with the general concept of a book, we are also familiar with practical upper- and lower-bounds on its size; a book that measures 3 centimetres by 2 centimetres is of as little use to most of us as a book measuring 3 metres by 2 metres. Looking at my bookshelves as I write, and ignoring only those volumes whose dimensions lie beyond the  $3\sigma$  points of the distribution, I can safely suggest that the majority of ‘normal’ books lie in the range 18 cm  $\times$  10 cm to 35 cm  $\times$  25 cm. In terms of more traditional printers’ units (picas), we can re-express this range as 42 pc  $\times$  24 pc to 80 pc  $\times$  64 pc (in all cases I have approximated rather than taking any exact measurements). What is more interesting, however, is the aspect ratio of each these books: almost without exception they are in portrait orientation rather than landscape. Why should this be?

There are, I suggest, two answers to this; one intensely practical, the other slightly theoretical. The practical answer is easily demonstrated: take any book that is *not* in portrait orientation (i.e. one that is in landscape orientation), hold it in one hand and attempt to open it: if the book is small, or tends to

square rather than being overtly landscape, it will be reasonably stable in the hand, but if it is large, or markedly landscape in aspect ratio, it will tend to fold back on itself as the centres of gravity of the two halves fall outside the span of the opened hand. For certain classes of book (i.e. those intended to be read from a desk or lectern, or perhaps opened on the reader’s lap), this is of little consequence; but for those books which are most likely to be read whilst being held in the hand (which includes the vast majority of books published), such instability would render them almost unreadable, and therefore such combinations of size and aspect ratio are generally avoided.

The theoretical reason hearkens back to material covered in the predecessor to this paper, and is concerned with the optimal length of line. In that paper it was suggested that between 40 and 70 characters per line is the target, with the ideal somewhere near the upper end of that range. Given that most normally sighted people can read without difficulty 9 point to 12 point typefaces at the normal distance associated with reading books, but find anything much smaller somewhat difficult to read (and tend to regard anything much larger as ‘insulting’, in the sense that it appears to have been intended for children), this suggests that most books will tend to have a *measure* somewhere in the approximate range 12 picas to 30 picas, but will tend to cluster nearer the upper end of that range. When we compare this with the range of book sizes cited above, these figures seem reasonable; the smallest book encountered was 24 pc in width, measured across the cover, whilst the largest was 64 pc, similarly measured. Allowing for trimmed pages fitting comfortably inside the cover, and ‘sensible’ margins (as yet to be defined), we find that the smallest book has a measure of 17 pc whilst the largest has a measure of 48 pc (and is set in an abnormally large font; it would be more usual to find a book of this size set in double-column format). Clearly there is a reasonable correspondence between theory and practice.

In practice, some sizes are more ‘desirable’ than others; traditionally, books were printed in a restricted range of sizes, and some of the terms used are still extant today; examples include ‘quarto’, ‘folio’, etc. Others, for example ‘elephant’ and ‘royal’ have fallen into disuse, and there is today far greater freedom in choosing the final size of a book. However, practical realities intrude here, as everywhere else, and ultimately the printer will have to produce the pages of the book by sub-dividing a much larger sheet of paper; as such large sheets of paper are produced in a fixed range of sizes, it is obvious

---

This is the second part of a talk delivered to a SOFSEM meeting in Hrdonov (Czech Rep.). It is reprinted with permission from the SOFSEM Organizing Committee, Masaryk University, Brno, Czech Republic.

*Keywords:* Design, typography, layout

<sup>1</sup> Book Design for  $\text{\TeX}$  Users; Part 1: Theory; see *TUGboat* 19:1, March 1999, pp. 65–74.

that some final page sizes will result in much less wastage than others, and such sizes are therefore to be preferred; your printer will give you advice on ‘ideal’ page sizes if asked, and will almost certainly tell you if your preferred size leads to gross wastage.

In determining the dimensions of a book, there are essentially three variables: the overall area of the text, including headers and footers; the margins; and the trimmed dimensions of the final page. Clearly at most two of these can be arbitrarily determined, and the third must follow by the simple rules of arithmetic and geometry. In practice one tends (if given total freedom) to determine the final page size and the text area first, and then to calculate the margins based on the difference; but in so doing it is important to remember that the margins are just as important as every other element of the made-up page, and cannot simply have arbitrary size. ‘Sufficient, but not too much’ is an excellent axiom to bear in mind when determining the size of margins; for example, a small book whose trimmed width is 23pc might have an outer margin of 3pc and a measure of 17pc; the actual inner margin will therefore also be 3pc, but the *perceived* inner margin will be somewhat less, as some portion of it is taken up by the binding. In general, the thicker the book the greater the apparent loss of inner margin, but binding technique is even more significant, and a well bound thick book may lose less space on the inner margin than a poorly bound thin book.

As the overall dimensions of the book increase, so may the margins; but they do not increase in direct proportion to the increase in page size: rather, if anything, they increase quite slowly, perhaps in proportion to the square root of the increase in page size, or to its logarithm. Once again, ‘sufficient but not too much’ is the key.

So far we have concentrated on the inner and outer margins, and it is worth pointing out before considering the top and bottom margins that, if symmetric perceived margins are required, this inherently requires asymmetric actual margins; but the asymmetry alternates between verso and recto pages. That is, in order to allow for the binding loss, the right margin on the verso page and the left margin on the recto page must each be increased by the binding loss. This is achieved automatically in the ‘book’ style of L<sup>A</sup>T<sub>E</sub>X, but plain T<sub>E</sub>X users will need a modified output routine. In order not to need any knowledge of the existing output routine, the following code hooks into the `\shipout` primitive, and can therefore be used in conjunction with *any* output routine, no matter how complex, unless

it, too, adjusts `\hoffset` on the fly (in which case more sophisticated code would be required).

```
\newdimen \rectohoffset
\newdimen \versooffset

\def \bindingloss {2 pc}
  %% adjust to suit actual book
\let \Shipout = \shipout
  %% need an alias so as to steal primitive
\let \then = \relax
  %% just syntactic sugar (sorry, Kees!)

\rectohoffset = \hoffset
\advance \rectohoffset by \bindingloss
\versooffset = \hoffset
\advance \versooffset by -\bindingloss

\def \shipout
  {\ifodd \count 0
   %% can't use \pageno in LaTeX
   \then
     \hoffset = \rectohoffset
   \else
     \hoffset = \versooffset
   \fi
   \Shipout
  }
```

Before considering actual dimensions for the vertical margins, it is worth considering the simpler question of proportion, and here, as in many elements of book design, two schools of thought obtain: the first would advocate that the top margin should be less than the bottom, the second just the converse! The argument in each case is based on visual balance: those who would place the text block asymmetrically towards the top of the page claim that, visually speaking, it ‘sinks down under its own weight’, whilst the alternative school claim that unless it is set asymmetrically towards the bottom of the page, it makes the page look top-heavy and therefore unstable. My own belief is that once the effects of head- and footlines are considered, the two schools can to a certain extent be reconciled; if, however, there are no head- and footlines, then my sympathies incline more towards the ‘lower-is-better’ school than towards its opponents.

The reason for considering the head- and footlines whilst discussing the margins is that whereas the left and right margins are what I will term ‘simple’ (that is, they each occupy a single band of white space), the top and bottom margins are effectively composite: there is white space above the headline,

white space below the headline, and similarly white space above and below the footline (if present; if not, then the bottom margin is simple). But in terms of visual density, the footline is usually very light — frequently no more than an unornamented page number — whilst the headline is frequently quite dense (see the predecessor to this paper for a fuller discussion on the possible contents of a headline). The effect of this is that the two lower margins are perceived by the eye/mind as being a single band of white space, whilst the two upper margins are perceived as separate entities. The eye/mind therefore takes the sum of the two bottom margins as representing the white space at the bottom of the page, whilst more or less ignoring the lower of the two upper margins and seeing only the upper component as representing white space.

We must now attempt to summarise the preceding discussion and to come up with some firm recommendations. In general the space above the headline is significantly greater than the space below, and is of the same order of magnitude as the mean of the left and right margins (assuming for the moment that these are not exaggerated; discussion on exaggerated margins occurs later in this section). The space below the headline is fairly small: perhaps 1 pica or thereabouts. At the bottom of the page, the situation is reversed: there is relatively little space above the footline, but rather more space below. But here caution must prevail: if we were to leave the same space above the footline as below the headline (e.g., 1 pica), we would overconstrain the page makeup process, for although any page could still run one line light, it could not run one line over without interfering with the footline (or, worse, displacing the footline vertically downwards); it is therefore necessary to leave additional white space above the footline on a normally made-up page, so that an overrun of a single line can be permitted *in extremis*. Thus a gap above the footline of perhaps 2 picas is appropriate, with an additional margin of 3 or 4 picas below. Bear in mind that these figures represent only a first-order approximation, but that only relatively small adjustments would be needed for a fairly wide variation in page size.

All the discussion on margins up to this point has reflected a fairly traditional, orthodox and conservative perspective. But the size and symmetry of margins is one of those areas in which *avant garde* designers feel obliged to express their individuality. Until the advent of the so-called ‘DTP revolution’, most books had conservative margins of the order of magnitude suggested above; but at about the time when DTP was becoming widespread, a new gener-

ation of designers suddenly found the need to adopt quite enormous margins, sometimes out of all proportion to the other material on the page.<sup>2</sup> The reasons for this sudden interest in wide margins are probably quite interesting, but I suspect not well understood. I can think of several possible reasons: (1) Each generation of designers feels obliged to express its creativity in some overt manner; simply to follow the guidance of its predecessors is felt at best to be pastiche, and at worst plagiarism. (2) The liberating effect of what I will term ‘Design through DTP’<sup>3</sup> allowed designers to experiment with designs that might previously have been consigned to the dustbin, either because the wasteful nature of their extremes became only too apparent as real paper models were made of the design, or because the time which elapsed between the creation of a design and its first physical realisation allowed the designer time for retrospection; many, I am sure, toned down their own excesses during this cooling-off period. (3) Many of the realisations of these designs were accomplished using early DTP systems, which were themselves fairly limited in their page makeup ability; having large margins into which oversize elements could flow allowed the designers additional flexibility to work within the constraints of the DTP system.

But there is a fourth consideration, quite independent of the DTP revolution, which may also dictate the use of large margins, and this final discussion on margins concentrates solely on the page makeup problems associated therewith. Text, tables, graphics, equations and formulæ all have different, and sometimes conflicting, requirements — text, as we have seen, will normally fit best into a measure somewhere in the range 12 pc to 30 pc; tables possessing multiple columns may well not fit into such a restricted measure, a problem that also can affect complex graphics (which although generally scalable can become illegible if over-reduced); equations and formulæ may also require a measure well in excess of 30 pc if they are not to be split over more than one line. With the exception of equations and formulæ, the problems are not insoluble,

<sup>2</sup> It is a sad reflection of our times that this also occurred during a period when awareness of the ecological effects of the loss of the world’s forests was becoming increasingly widespread; thus on the one hand we had the environmentalists urging us to save trees, whilst on the other we had a generation of designers apparently hell-bent on destroying the world’s forests purely to provide large asymmetric white borders for their books. . .

<sup>3</sup> by which I mean the use of an Apple Macintosh or similar system to produce an on-screen mock-up of a proposed design without any need for a physical realisation to become available.

or even difficult: where it is known in advance that a measure well in excess of 30pc will be required, the text can be set in two columns whilst overwidth tables and graphics can be allowed to span both columns; as tables and graphics are generally regarded as ‘floating’ entities (that is, they can migrate in the text without causing the reader difficulty, as reference to them is almost invariable by name or by number rather than by implicit physical association), they can appear on a page in their own right, or at the top or bottom of the page on which they are referenced, without interrupting the flow of the text. But equations and formulæ (and similar entities, such as program fragments and algorithms) frequently *cannot* be allowed to float: the author will almost invariably write the text on the assumption that the equation/formula will always occur exactly where it does in the manuscript, and will simply allow his or her text to ‘fall through’ to the equation or formula; if such an equation/formula is overlong and cannot be wrapped, then both columns of the two-column text will need to be interrupted, to the great inconvenience of the reader, for it will not necessarily be at all apparent whether the text is to be read up to the equation/formula and then continued below in the same column, or the text is to be read up to the point of the equation/formula and then continued from the top of the next column. Worse, if the equation/formula occurs not in the first column of text but the second, as the reader progresses down the first column he/she is suddenly stopped dead in his/her tracks by a completely irrelevant equation/formula; not only does the reader now not know from where to continue, he/she also does not know why the interruption occurred in the first place. Only on reading down the second column does the reason for the interruption become clear.

Therefore, in such works, an alternative approach is required, and one such approach is the use of oversized margins: the text is set to a fairly wide single-column measure, but the trim dimensions of the page are such as to allow the longest equation or formula to extend out into the (usually right) margin as necessary. The designer is then faced with another problem: how to justify to the reader the presence of these margins on pages where no such equations or formulæ occur. It is by no means unusual to find section heads pushed out into the margins in these circumstances, nor to find marginal notes which might otherwise occur as foot- or even endnotes. Anything which can justify the presence of the anomalous margin is regarded as fair game!

Finally gutters: the internal ‘margins’ that separate columns from each other in multi-column for-

mats. Generally speaking, a gutter should be no wider than the mean of the left and right margins; if anything, it can be somewhat narrower. Some designers prefer to divide their gutters vertically by a narrow rule; I would tend to avoid this unless rules were used elsewhere in the design. Here, as in many places, the desire for uniformity provides excellent guidance.

## 2 The elements of a book

Having established guidelines for the overall dimensions of our book, it is now appropriate to consider the various elements which make up that book. At the most superficial level (and ignoring the covers, spine and dustjacket), a book consists of the *front matter* (also referred to as ‘prelims’), the *text*, and the *back matter* or *end matter* (the last is clearly ambiguous, as a book has two ends, but traditionally ‘end matter’ is used in preference to the less ambiguous ‘back matter’).

The front matter is composed of such elements as the half and full title pages; the copyright and cataloguing-in-publication data page; a table of contents (and sometimes other analogous tables); and perhaps a preface. Also frequently included in the front matter (particularly with the advent of the DTP revolution, since which we have all become far more aware of typefaces and typography in general) is a ‘colophon’, which strictly speaking should occur as the very last element of the book, but now more usually occupies space on the copyright and cataloguing-in-publication page; the colophon contains details of the typefaces and leading used, and may also give details of designer, printer, etc.

Amongst the end matter are found appendices; one or more indexes; a bibliography (if such is not associated with each chapter, or if an overall bibliography is desired as well as one per chapter); and perhaps a glossary or similar.

Finally, the text is composed of the body of the book; usually divided into chapters, it may also be divided at a higher level into parts.

It is fair to say that the boundaries between these three zones are not entirely rigid: an author may choose to regard a preface as a part of the text, rather than as a part of the front matter, and this will need to be reflected in the page numbering, as we shall see. Similarly some writers may regard their appendices as forming a part of the text; this may affect their page numbering but is less likely so to do. Indeed, an author may choose to write a preface, a prologue, an introduction, a conclusion, an epilogue, and one or more appendices; the designer and author

will need to liaise carefully to ensure that each is appropriately classified.

The primary reason for this division concerns page numbering: front matter is traditionally numbered in *roman* style, using lower-case roman numerals (i, v, x, l, c, d, m) which are often set as dropped folios, whilst the text proper is usually numbered using *arabic* numerals (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Appendices and other end matter usually continue in the same sequence and style as the main text, but it is permissible to re-start the numbering for the appendices and prefix the page number with an letter 'A'; if this latter course is taken, the index (assuming that the index forms the very last element of the end matter) will need to have unnumbered pages, as it would clearly be inappropriate to continue using 'A'-style numbering whilst it would be equally inappropriate to resume the main numbering scheme. Fortunately indexes are not required to be self-referential (although I confess to once padding out an index that would otherwise not balance with an entirely spurious reference to 'loop, infinite', whose sole page number was that of the entry for 'loop, infinite' in the index. . .).

There are also conventions as to which elements are required to occur recto, which verso, and which require to be preceded or followed by a blank page. A typical book might be numbered as followed (remember that odd numbers indicate recto, whilst even numbers indicate verso):

1. half title;
2. blank;
3. full title;
4. cataloguing-in-publication, copyright, colophon;
5. preface to the edition;
6. general preface;
7. ditto, continued;
8. blank;
9. table of contents;
10. ditto, continued;
11. glossary;
12. blank;
- 1 first chapter.

Of these, the half and full titles are required to occur recto, (whence the blank page between, which also affords a nice contrast to the complexity of the full title page); the copyright and c-i-p page frequently occurs on the reverse of the full title page; the preface is not required to start recto, but it may be the designer's wish that it should so appear; the table of contents is normally recto, as here; the first chapter invariably opens recto, and except in the most

casual of styles all subsequent chapters must open recto as well. The page number of the first chapter page could equally well have been '13'; it is a design decision as to whether to continue the numbering sequence from the prelims or whether to start afresh with the main text.

There are fewer conventions concerning the end matter, but it would be normal for the *first* appendix to start recto; subsequent appendices may start recto or verso as necessary; and the index would also normally start recto.

### 3 Laying out the pages

Although by far the majority of pages in a book are 'normal' pages, it makes a certain amount of sense to start by considering the opening chapter pages, since these contribute a great deal to the book's visual identity and allow a fair degree of artistic licence in their creation. (It is also fair to say that one can waste an enormous amount of time trying to design them!)

When designing one's first book, it is by no means uncommon for people to align the main chapter header (be it 'Chapter 1' or 'Introduction') with the top of a normal page. For some books, particularly those with with very short (less than two pages) chapters, this makes enormous sense, for otherwise one can run to far more pages than are strictly necessary (there are also aesthetic reasons why such a design is to be preferred in these circumstances). However, the vast majority of books have chapters whose page count often runs into double figures, and for such books it is customary (although not essential) to start the opening chapter heading some way down the page. Typically a quarter to a third of the page depth may be reserved for the above-heading space.

There next comes the question of what to put in the heading. If chapters are numbered, one has to decide between 'Chapter 1', 'One', '1' or some similar variant; and if named, whether to also number or just to use the name (and if one uses both names and numbers, then which numbering style to use). It is thought that 'Chapter 1' is a little old-fashioned, but I do not hold to this view. If both numbers and names are used, and if just the arabic number is chosen, then there is also the option of placing the two on the same line, perhaps separated by a colon and the space of the line; if they are put on separate lines, then it is customary for the number line to precede the name line.

Next the question of font: in which font(s) are these headings to appear? In almost all cases, a large bold font will be used, but 'large' is very much in the eye of the beholder; it is probably safe to say

that L<sup>A</sup>T<sub>E</sub>X uses rather larger fonts for this purpose than more conservative designers might choose. The use of a *sans serif* font for such headings is most certainly justifiable, but not essential.

Placement: should the headings be centered or ranged left (or even ranged right)? Generally speaking, centered headings are either slightly old-fashioned or are more suitable for works in the arts; modern scientific publications frequently adopt a ranged-left theme which runs throughout the book, including headings such as these. Ranged-right probably shrieks *avant garde*, but cannot be discounted on that score; if used, there should probably be other elements in the design which echo the ranged-right theme, or there should be a contrasting ranged-left theme to balance. If an epigram is used, it is probably better to have the headings ranged left and the epigram ranged right, as the converse would over-emphasise the epigram to the detriment of the chapter title.

There is another element to placement which also requires discussion: is the white space above the heading to be regarded as belonging to the heading or to the page? By this I mean the following: if the chapter title normally occupies  $n$  lines (typically one or two), but a pathologically long title for a particular chapter requires one or more additional lines, from where should the space for these lines be taken? Should the title be allowed to extend *up* the page, encroaching on the reserved white space, or *down* the page, displacing the starting point of the main text downwards? Neither is ideal, but if authors insist on writing pathologically long titles, one or other solution must be taken. Although the following is not cast in stone, it is perhaps worthy of consideration: if the opening chapter page starts with a line containing only the *number* of the chapter (or with the word ‘Chapter’ followed by the number), then that should *always* occur in the same vertical position (and thus the main text will get displaced downwards); but if the page starts with the *title* of the chapter, then that title may be allowed to extend upwards, thereby ensuring that the main text always starts at exactly the vertical position on the page.

And rules: should the headings be set off from the text by a horizontal rule? Here we probably need to return to the theme of uniformity: if rules form a recurring theme throughout the book, then a rule between heading and text is probably fine; if not, then it may seem intrusive.

Finally, before leaving the subject of opening chapter pages completely, it might be worth recapitulating on the advice given in the predecessor to

this paper concerning running heads and folios: generally speaking, a running head has no place on an opening chapter page; the white space above the title should merge imperceptibly into the top margin. This means that the folio, if normally on the outer edge of the running head, must (on an opening chapter page) either be omitted completely, or must be relegated to the footline. Omitting the folio is highly undesirable, as it renders the table of contents virtually useless (and also reduces the usefulness of the index, if any entry in the index refers to an opening chapter page); the solution is therefore to set the page number as a dropped folio, centered in the footline. Sometimes such folios are given a little additional ornamentation, for example en-dashes on each side set off by a thin space; although this convention is taken directly from typewriter practice it does, in the opinion of the present author, render the folio a little more obvious, and therefore has something to commend it.

Having completed opening chapter pages, the next most significant element in the design of the book is the normal text page; such pages usually make up over 90% of the book, and it is therefore worth expending considerable effort ensuring that they look ‘right’. We have already dealt with margins, gutters, head and footlines, so we may concentrate on the text proper, and in particular on the fonts and leading to be used.

#### 4 Fonts and leading

As suggested above, the text will normally be set in a 10pt *serif* font, often on a 12pt leading (here, at least, plain T<sub>E</sub>X gives sensible defaults, except in the excessive measure used). There appears to be a widespread belief that Times Roman is the font of choice, yet this font, designed as it was for use in the exceptionally narrow measure of newspaper columns, has little to commend it apart from widespread availability. The font is too narrow for the generous measure of most books, and if it *must* be used can benefit enormously from being anamorphically scaled by a factor of 24/25 in the vertical direction. Such scaling, whilst anathema to purists, converts the somewhat narrow letterforms of Times Roman into rounder, softer, shapes, and enables a near optimal combination of font size and leading to be used on measures up to 27pc and beyond. 11/12.5 Times Roman, when anamorphically scaled by a factor 24/25, yields 10.56/12 which in the opinion of the present author results in a highly readable text.

But far better than anamorphically scaling Times Roman is to select a font which already



has the appropriate properties (rounded letterforms, suitability for use with wide measures, etc.); examples are legion, but amongst the most obvious candidates are Baskerville, Bembo, Caslon, Garamond, and Palatino. To be avoided are fonts which are highly idiosyncratic: it is to be remembered that the *sole* purpose of the font is to convey information; if the reader is distracted by the idiosyncratic nature of the font, information transfer will be less than optimal and the book's value reduced as a result.

It may be worth digressing at this stage to discuss briefly one particular book which I first encountered on being asked to review it, Knuth's *3:16 Bible Texts Illuminated*. My first reaction on opening this book was to ask myself rhetorically "why on earth did he set it in Computer Modern?". I was familiar with Computer Modern from the *Computers and Typesetting* quintology, and had, of course, set much of my own material in Computer Modern whilst learning about T<sub>E</sub>X; but I had reached the point where I felt that other fonts had much more to offer, and had not, for some time, typeset anything in Computer Modern at all; it therefore came as a nasty shock to find a book on Bible Study typeset entirely in Computer Modern, particularly by someone whose opinions I value so greatly.

And yet, the strange thing is that having read no more than half a dozen pages of *3:16* I suddenly discovered that I was no longer seeing the font at all; it had, to all intents and purposes, ceased to exist as a typeface, and become purely a medium for the communication of facts. Now Computer Modern, based as it is on Monotype 8a, is not everyone's ideal font; and particularly when rendered on low resolution devices such as laser printers can be quite unpleasant indeed, with the thin strokes breaking up or disappearing completely and the thick strokes somehow seeming out of proportion. Yet when rendered on a high resolution typesetter, the contrast between thick and thin contributes much to the aesthetics of the font, and the overall effect is to yield an unintrusive design, pleasantly devoid of idiosyncrasies, which suppresses its own personality and allows the information to shine through. Perhaps there is no such thing as a bad font; what we perceive as bad may simply be a good font used inappropriately, or rendered using inappropriate technology.

But to return to the question of design, and in particular to the design of the normal text pages of a book. Having selected our primary font and leading, we will need to select appropriate variants of that font for particular purposes (we may also need to select one or more other fonts for special purposes, but as a general rule the fewer fonts used in a document,

the better the document will be). For emphasis, and for foreign words and phrases within the text, it is customary to use an italic variant of the font; the use of bold for emphasis is to be strongly deprecated, with such fonts being reserved for headers and similar. Italics may also be used for book titles, for the names of ships, and for other analogous entities. It goes without saying that underlining, too, has no place in the running text of a book, and very little place anywhere else either; just as the use of bold for emphasis is an artifact of early word-processing systems (which were incapable of italics and therefore had to create an alternative convention for achieving stress), underlining is an artifact of handwritten and typewritten text, and has no place in a typeset document.<sup>4</sup>

If it is necessary to stress a word or phrase within a longer structure that is already being typeset in italics, it is customary to revert to a roman font for the stressed section; but the present author can find no reason why in these circumstances the stressed section should not be set in bold italics, if such a font variant is available (and with the advent of PostScript fonts, such variants are usually to be found); if the bold stressed section is being compared or contrasted with another section of text in the book which is physically nearby, then it may be necessary to set that section too in bold italics, even if it occurs in a context in which italics are *not* being used; in that way, the reader will be given appropriate typographic cues as to which two sections are being compared or contrasted.

Italics (which are a highly stylised variant of a font) should not be confused with *slanted* or *oblique* variants, both of which involve no original design but result from a simple geometric transformation of the roman form of the font. Whereas italics and oblique forms both have an honourable ancestry (oblique normally being reserved for *sans serif* fonts whilst italics are normally a variant of a *serif* form), slanted fonts appear to be another artifact of the DTP revolution. In the opinion of the present author they have little to offer in the way of aesthetics, and even though they are sometimes used where it is

---

<sup>4</sup> Of course, like almost every rule, these rules too admit of exceptions, and it would be a brave author indeed who wrote that *every* instance of underlining, or of the use of bold within running text for emphasis, was categorically wrong; the most that can be said is that generally speaking such (ab)uses are regarded as infelicitous or inappropriate, and that should the designer none the less decide to adopt such a convention, he or she should be aware of the 'rules' that are being flouted, and take a conscious decision to flout them rather than simply being unaware of their existence.

deemed desirable to differentiate typographically between two entities which would otherwise both have to be rendered in italics, as a general rule I would caution against their use. Designers have managed for centuries to convey considerable amounts of information without having recourse to slanted fonts; it is to be hoped that future generations of designers will conclude that they represent no more than what Fowler might have termed ‘elegant variation’, and are therefore a luxury without which we can all happily do.

It is sometimes necessary, particularly in books on linguistics or other subjects in which language is both used and discussed, to differentiate typographically between the two uses. Sometimes simple quotations marks will suffice; sometimes italics; but there are also times when both of those forms are already reserved for other typographic differentiation, and some third form is needed to clarify which text is being discussed and which text is performing the discussion. In these circumstances (and in very few others), it is justifiable to introduce a new font which may be used as a part of the running text. If the main text is set in a *serif* font (as it almost invariably will be), then a second *serif* font would *not* be suitable; even though two *serif* faces may be as different as chalk and cheese, the risk of confusion is still too great (and the æsthetic clash too severe) to permit two distinct *serif* faces to appear in juxtaposition. The second font must therefore be a *sans serif* face, chosen to blend in with, whilst being clearly differentiable from, the main text face. The second font will need to be matched for weight (visual density), ex-height and caps-height; and because of the variation in the semantics of *design size*, will probably need to be loaded at a fractional size.

## 5 Headings

The motto for the predecessor to this paper was “There can never be too little space below headings, only too much!”, and in those few words can be summarised the bulk of the received wisdom concerning headings. As previously pointed out, a heading *must* be tied to the text with which it is associated, and that text is invariably the text which immediately follows. Headings are frequently hierarchical in nature, and lower-level headings are more closely bound to the following material than higher-level; thus the white space which separates low-level headings from the text is usually less (and never more) than the white space which separates higher-level headings and text. In the limiting case, the heading is *run in*, that is to say literally forms a part of the text and does not occupy a line in its own

right. For run-in headings, it is essential that the author be consistent in usage, since such headings can either participate in the grammar of the text or remain a distinct grammatical entity; in the former case it is customary to indicate the extent of the heading by a change of font (italics, or bold, or even caps and small caps), but by no extra horizontal white space or punctuation; for headings which are grammatically distinct from the text which they introduce, a change of font is also indicated, but punctuation (e.g. a colon) or additional white space (e.g. one quad) is also frequently used. Such a heading might be set off by as little as 1ex additional white space from the preceding text, and certainly by not more than one blank line.

At the next level in the hierarchy, the heading usually occurs on the immediately preceding line, and occupies a line in its own right. It is not set off by any additional vertical white space, but simply separated from the text by the normal leading for the paragraph. Again a change of font is indicated, and the font options applicable to run-in heads are equally applicable here, although the use of caps and small caps would be unusual. The extra vertical white space above the heading is of the same order of magnitude as for run-in headings.

A level higher and perhaps a larger font is indicated. Assuming a base setting of 10/12, a 12pt font might be suitable for such a heading. If a bold font has been used for any lower level, then this font too must be bold, otherwise ambiguity will result (the same is true at *all* levels in the hierarchy: once a bold font has been used at a lower level, bold fonts must be used at all higher levels. In the same way, no font used in a higher level heading may be smaller than a font used in a lower level heading; it may be the same size, but only if it is bold and the lower level is not, or if there is other clear typographic indication of the hierarchy). Above such a heading a little extra white space might be allowed, perhaps between one and one-and-a-half blank lines.

Beyond this point, simple extrapolation is sufficient: as we move up the hierarchy, headings get bigger, bolder, more distinctive. The white space below them may increase, but only very slightly; the white space above increases, but not to ridiculous limits. Anything in excess of three blank lines is almost certainly excessive, and two blank lines are normally more than sufficient.

At this point it is appropriate to consider the implications of the above set of rules on  $\text{T}_{\text{E}}\text{X}$  implementations. In order to allow successful page makeup in  $\text{T}_{\text{E}}\text{X}$ , it is customary to allow the vertical white space associated with headings to be flexible

(i.e., ‘rubber lengths’, in L<sup>A</sup>T<sub>E</sub>X’s quaint terminology); but T<sub>E</sub>X has two quite distinct rules when dealing with flexibility: if a dimension is given a negative flexibility (i.e., is allowed to shrink), then T<sub>E</sub>X will take advantage of the stated shrinkability if necessary to achieve optimal page makeup, but will never attempt to shrink it by more than the permitted amount; however, if a dimension is given *positive* flexibility (i.e., is allowed to stretch), then T<sub>E</sub>X will first of all take advantage of that flexibility to achieve optimal page makeup, and if that flexibility is insufficient, *will continue to stretch it until optimal page makeup has been achieved*, even if this involves stretching it by many times its stated stretchability. Of course in these circumstances T<sub>E</sub>X issues a warning, but by then it is too late: the evil deed has been done.

The implications of this behaviour for successful implementations of design are quite severe: T<sub>E</sub>X must *never* be given positive stretchability to use if it is required to exercise any automatic control over the upper bound by which white space will be stretched; shrinkability can be used, but T<sub>E</sub>X is noticeably asymmetric in this respect, and whereas `\vfill` and its friends can be used to pad out underfull pages whilst preventing embedded ... plus \$n\$ pt constructs from contributing white space, there is no equivalent which can be used to negatively pad pages whilst preventing ... minus \$n\$ pt constructs from shrinking (the reason is that T<sub>E</sub>X will not allow what it terms ‘infinite glue shrinkage’ to occur in unrestricted horizontal or vertical modes). Thus there are severe problems in inhibiting T<sub>E</sub>X from taking excessive advantage of permitted flexibility, and in the end only careful observation of the log file, and manual intervention where T<sub>E</sub>X has exceeded its brief, will be sufficient to keep matters under control.

But recalling for a moment the discussion on grid-based layouts which took place in the predecessor to this paper, it will be appreciated that simply preceding and following header lines by `\vskip` commands will not necessarily have the desired effect. A far more satisfactory method of placing headers, whilst ensuring that they occupy an integral number of blank lines (i.e. an integral multiple of `\baselineskip`) relies on a technique which I refer to as a ‘pseudobox’: this is a T<sub>E</sub>X construct which is in reality a box whilst behaving like glue; the following code fragment illustrates the technique in use.

```
\newbox \headerbox
\newdimen \headerheight
\newdimen \headerdepth
\def \header #{\afterassignment \afterheader
```

```
\setbox \headerbox = \vtop}
\def \afterheader {\noindent
\aftergroup \reallyafterheader}
\def \reallyafterheader
{\headerheight = \ht \headerbox
\headerdepth = \dp \headerbox
\advance \headerheight by \headerdepth
\headerdepth = \headerheight
\ht \headerbox = 0 pt
\dp \headerbox = 0 pt
\advance \headerheight by 0.5\baselineskip
\divide \headerheight by \baselineskip
\multiply \headerheight by \baselineskip
\ifdim \headerheight < \headerdepth
\advance \headerheight by \baselineskip
\fi
\vskip 0 pt
\box \headerbox
\vskip \headerheight
\noindent
\ignorespaces
}
```

If this code is used to typeset a large bold header within the text of this paragraph, as in `\header {\Huge Header}`, the effect *should* be to leave the remainder of the paragraph set on its natural grid;

## Header

whether or not it has achieved this effect is left to the reader to see! Perhaps a brief explanation of the code is in order, as so far as the author is aware the technique has not previously been published. The `\header` macro takes no parameter, but the terminal hash of its parameter list causes it to require an open brace to immediately follow its use; on the assumption that the open brace is the open brace of a brace-delimited parameter (which it should be, if the macro has been properly used), the macro sets `\headerbox` to a `\vtop` containing the parameter. However, an additional token is introduced into the `\vtop` just prior to the parameter by means of the `\afterassignment`, that token being `\afterheader`. This token itself expands into three further tokens, `\noindent` (to prevent the parameter from being indented within the box), `\aftergroup` (to allow the following token to be expanded not within the box but outside it, once it has been set), and `\reallyafterheader`, which is the macro that does all the real work. Thus the combined effect of the `\afterassignment` and the `\aftergroup` is to inhibit any indentation of the parameter, and to cause

`\reallyafterheader` to be expanded once the box has been set. `\reallyafterheader` commences its work by saving the height and depth of the box in which the header has been set, and then computes their sum; the height and depth are set to 0pt. Using Knuth's algorithm from A15.8, the combined height + depth is rounded to the nearest integral multiple of `\baselineskip`, and if the result of this rounding is less than the original sum, a further increment of `\baselineskip` is added. The result of this computation is the smallest integral multiple of `\baselineskip` within which the entire contents of the box can be set. A vertical skip of 0pt is carried out (to force `TeX` into vertical mode), and then the box is typeset (remembering that it has zero apparent height and depth, and therefore occupies no space), after which a further `\vskip` of the calculated integral multiple of `\baselineskip` is carried out to leave room for the contents of the box whilst not disturbing the regularity of the baseline grid. Finally `\noindent` and `\ignorespaces` ensure that the first paragraph following the header is typeset correctly.

A real-life instance of this code would require parameterisation to indicate the level of header, from which it could ascertain (by means of a look-up table) how to distribute any required additional space around the header; in addition, it would enable ragged-right setting within the header box, and would need to deal correctly with a header immediately followed by another header (the spacing should not be additive). Many other refinements are possible.

## 6 Paragraphs

In trying to make practical recommendations for real-life book design, it is necessary to alternate between those entities which occur fairly rarely (opening chapter pages, headers, etc.) and those which form the bulk of the book (regular pages, paragraphs, etc.). Here we consider material which makes up the vast bulk of the book, to wit the paragraph.

Fortunately the 'rules' for paragraphs are fairly straightforward, but as so many examples may be seen which either blatantly ignore the rules or are simply unaware of them, some discussion is nonetheless necessary. It should be noted, however, that these rules are inherently culturally based, and I am advised by one eminent French authority<sup>5</sup> that the rule stated below concerning the first paragraph of

any new section would be incorrect were it to be applied to material published in French.

- The first paragraph of a new section is not indented. This rule is so often more honoured in the breach than in the observance that I sometimes wonder whether its existence is widely known at all. For reasons entirely unclear to me, `LaTeX` whilst doing its best to honour this rule indents abstracts, which seems to me at best inconsistent and at worst inexcusable. I am very pleased to see that these proceedings avoid that error.
- A paragraph is either indented, or is set off by vertical white space from preceding material. It is normally considered infelicitous to do both; it is a gross error to do neither. The reason why the latter is so severe a crime is that if paragraphs are neither indented nor set off by vertical white space, then any text in which a paragraph just happens to end flush with the right margin will be followed by a paragraph whose existence can barely be guessed at. There will be *no* typographic clues to indicate that a new paragraph has started.
- The leading and font within a paragraph are uniform. This may seem to go without saying, but if a document is set with the minimum leading necessary for unadorned text, then an accented capital letter may well be enough to force down the entire line on which it occurs. In such circumstances either the leading must be increased for the entire document, or special steps taken to conceal the height of the accented letter (whilst ensuring that it does not unfortunately co-incide with a descender from the line above). By 'uniform', when applied to the font, I do not suggest that every glyph in the paragraph must be set in the same font; clearly there may be a need for italics, or even for a *sans serif* font at points, as indicated above. But all the glyphs within the paragraph should *appear* uniform, and must therefore come from closely related or well chosen fonts. For example, the first phrase of each paragraph in a book may be set with an initial full cap and then small caps; provided that these blend in with the main text font, there can be no objection to this. Similarly the first letter of the paragraph may be a dropped cap; provided that it too blends in with the main text font, that is a perfectly valid design decision (and sometimes very stylish, if I may express a personal opinion).

<sup>5</sup> Bernard Gaulle, past and future President of GUTenberg, the French-speaking `TeX` Users' Group.

- A paragraph should not end with only a part-word on the last line. Assuming that hyphenation is permitted at all (which it will need to be if fully justified text is specified), then the last line of a paragraph should end with at least one full word and preferably more. Plain T<sub>E</sub>X's (and L<sup>A</sup>T<sub>E</sub>X's) setting for `\parfillskip` do not encourage this; a more felicitous setting might be `\parfillskip = 0 pt plus 0.7\hsize`, which encourages longer last lines at the expense of setting some such lines slightly loose.

## 7 Graphics, figures, and other 'floating' entities

Although there is much more that can (and should) be said about book design in general, I feel that there is one area which must be treated before I close, and that is the whole area of insertions, or L<sup>A</sup>T<sub>E</sub>X terms them, 'floats'. These are, in some general sense, graphic entities, although they may turn out to be purely textual in content. What really typifies them, however, is that they are invariably indirectly referenced; that is, they are referenced by the author in terms of *see Fig. 1* or *See Table 2.4*, rather than being implicitly referenced by position in the text as in, for example, *as shewn below*. By virtue of the indirect nature of their reference, they can be physically remote from the point of reference, but one of the major skills of page makeup is the careful placement of such entities. The cardinal rule for these insertions is that *they must be capable of being seen from the point of reference*. One of the little appreciated strengths of T<sub>E</sub>X is how well it carries out this task for footnotes, which are a very simple instance of insertions; if you look carefully at T<sub>E</sub>X-set material which has many footnotes, you will probably be surprised at the number of times that a footnote reference occurs on the very last line of the page (before the footnotes themselves appear, that is). If you have not thought about this problem before, you may casually remark to yourself "that's lucky; another line and the footnote marker and its text would have appeared on different pages". But now try to find an instance where that has happened; try as you might, I suggest that you won't. And that surely suggests that it is more than luck that causes that particular juxtaposition of footnote marker and start of footnotes to occur so regularly, so reliably, and so consistently. And of course it is more than luck; all the while that T<sub>E</sub>X is accumulating material in galleys, it is carefully tracking how much space is occupied by footnotes and how much by the main text; and as soon as the combination of the two exceeds the available space on the page,

T<sub>E</sub>X knows that it must cut the galley at or near that point and start a new page.

Now footnotes are, as I said, a particularly simple instance of such insertions; no-one minds if the text of a footnote is started on its page of reference but continued on the next (no-one but a pedant, that is). But figures, tables, graphics, etc., are a very different kettle of fish; they are essentially indivisible entities, and can therefore either appear on a given page or not appear on that page; there are no half measures which would allow a part of the figure/table/graphic to appear, and the remainder to appear on the next page.

So now put yourself in the position of T<sub>E</sub>X, this time not accumulating text and footnotes, but accumulating text and (say) figures. T<sub>E</sub>X continues to accrete material in its galley as before, and encounters a reference to a figure; say that the page is only a third full. If the figure is less than two-thirds the depth of the page, there is no problem: T<sub>E</sub>X simply adds the figure to the list of things that appear on that page and carries on. But now let there be a second figure reference, maybe two-thirds down the page: T<sub>E</sub>X looks to see how big the figure is, and discovers it needs a half a page to itself. What does T<sub>E</sub>X do? The first choice is trivially ruled out; you can't have the reference to the figure followed by the figure, because (a part of) the figure would fall off the bottom of the page. OK, what's the next choice? Remember that the figure can *float*. So, let's try floating the figure to the top of the page on which it was referenced: no problem there, the figure appears at the top of the page, pushing the textual material material down. Some of the textual material will fall off the bottom of the page, of course, because we already know that we have 2/3 of a page of text, and 1/2 of a page of graphics, so 1/6 of a page of text falls off the bottom. But that's no problem, because textual material can normally be split at almost any point: so T<sub>E</sub>X chooses the nearest valid breakpoint and carries the remaining material over to the next page.

Then what happens? Well, think about what is *on* the material that has been carried over: the reference to the figure that caused the trouble in the first place! So now we have the figure on page  $n$ , and the reference to the figure on page  $n+1$ . If  $n+1 \equiv 1 \pmod{2}$  (sorry, if  $n+1$  is odd!), then there is no real problem, for the reference to the figure occurs on the recto half of the spread, and the figure itself occurs on the verso half of the spread, so all is well. But if  $n+1$  is *even*, then all h@ll breaks loose: because the figure is on the recto half of a spread, and the reference to the figure is on the verso half of the

next spread; and when the reader finally encounters the reference to the figure, the figure itself can no longer be seen. And no matter what  $\text{\TeX}$  were to do in those circumstances, it would not be able to solve the problem without assistance.

So there are some problems in page makeup that simply *cannot* be solved by naively applying rules; rules are all very well, but eventually the time will come when the author's text and the rules of design are simply incompatible, and in those circumstances you will have little option but to liaise with the author and attempt to persuade him or her to re-write the offending portion of the text. If the author is dead, and the text is cast in tablets of stone, then you will have to do a lot of work by hand, maybe setting a whole series of paragraphs one line looser than ideal, just to force a reference onto a more appropriate page. But when you've done it, and the finished book is printed, and you look at it and *know* that there are no further improvements that you could have made, then a great warm glow will fill your body and you'll know that it's all been worthwhile. Good luck!

◇ Philip Taylor  
The Computer Centre,  
Royal Holloway and Bedford New College,  
University of London,  
Egham Hill, Egham, Surrey TW20 0EX,  
United Kingdom.  
`P.Taylor@Vax.Rhbnc.Ac.Uk`

time required to do the necessary work, the idea was largely ignored. Also there was the problem of obtaining appropriate versions of the manuscripts, some of which were still undergoing editing revisions, so were not yet finalised.

For some time I've been routinely preparing mathematical course materials for paper (via  $\LaTeX$ ) and in HTML, using  $\LaTeX2HTML$ . Recently I started using *pdfTeX* as well, and organise the manuscripts to process seamlessly with all three tools, while exploiting the best features of each. Having some time available, working on a proceedings collection in PDF seemed like an appropriate thing to do, and could provide valuable experience for similar work in the future.

There are two main tasks here:

- Prepare a PDF version of each paper.
- Somehow combine the papers, using active hyperlinks, to present as if part of a unifying electronic document.

As each task separately requires some amount of editing within each author's manuscript, I wanted to develop a method which would minimize the number of times each file need be manually edited. It was felt that any decisions concerning styles and layout should be able to be applied to all the preprints, without any need to make edits in the individual files. To a large extent this was achieved. The results of this work can be viewed at <http://www.tug.org/TUG99-web/program.pdf> which has links to .pdf files for the papers, in the directory <http://www.tug.org/TUG99-web/pdf/>.

Below I describe the techniques developed, and lessons learned. Some of these lessons and techniques are doubtless known already to experienced  $\LaTeX$  and  $\TeX$  users; others are new and can surely be refined to become even more useful. I'm writing this article with hindsight<sup>2</sup>, after the TUG'99 meeting has concluded; indeed some of the work on individual preprints was done on returning home after the meeting. Advice is given, to help authors simplify the tasks of editors, which in turn leads to reducing the time required for a publication to be prepared.

### Preparation Notes

The main issues for creating PDF, as distinct from DVI, versions of papers submitted for a proceedings (or any other) volume relate to

- A. bookmarks—navigation to sections, subsections, figures, etc.;

## Report

### Preparation of Documents for Multiple Modes of Delivery—Notes from TUG'99

Ross Moore

#### Background

As the theme of the TUG'99 meeting concerned preparing documents for Web-based delivery, and the  $\TeX$ -related tools recently developed for this purpose, it had been suggested<sup>1</sup> that it would be nice to apply some of these tools to the preprints for the meeting, in order to show off the effectiveness of these tools. As no single person had the expertise in all of *pdfTeX*,  $\LaTeX2HTML$  and  $\TeX4ht$ , nor the

<sup>1</sup> by Mimi Burbank

<sup>2</sup> ... and at the request of Christina Thiele

- B. active internal hyperlinks for citations and cross-references;
  - C. active hyperlinks to external URLs mentioned in the paper, and/or other papers in the same proceedings set;
  - D. incorporation of included graphics.
- Of course, also of relevance is:
- E. how the papers, as individual .pdf files, will be linked back to a common document which serves as a wrapper, including a Table-of-Contents with active hyperlinks to each paper.

Since most of the submissions for TUG'99 were prepared using L<sup>A</sup>T<sub>E</sub>X, and the `hyperref` package already provides an automatic solution to issues A and B (provided the author has used `\label`, `\ref` and `\cite` appropriately) it was decided to use *pdf*T<sub>E</sub>X, via the `pdflatex` command, and `hyperref` for *all* the papers. This includes the papers originally submitted as T<sub>E</sub>X source, rather than L<sup>A</sup>T<sub>E</sub>X, for which there would necessarily be some extra editing required. Thanks to Sebastian Rahtz and other authors, the packages and drivers to tackle issues C and D were already available, so it was not necessary to write any complicated macros to implement these effectively.

To obtain a consistent style across all the papers, and to ensure that the same packages are available for handling citations, URLs, graphics etc. it was decided to use a common “driver” file, implemented as follows.

- Each submitted paper was stored in a separate subdirectory, along with any styles, graphics and bibliography files. (This structure was already in place, due to earlier phases of the editing process.)
- A common file, called `TUG99pdf.pre` was located in the common parent of these subdirectories. This file would be `\input` at the beginning of each job. This file contains the `\documentclass` command, and commands to load suitable packages. Parts of its contents will be described in due course.
- For each paper a “mini-driver” file was made, to load `TUG99pdf.pre` and subsequently `\input` the author’s original source (or rather, the current version available in the editing process). This file was named e.g. `rowley.ltx`, where the current source revision is `rowley5.ltx`. It is this mini-driver file that is actually typeset, to produce `rowley.pdf` and auxiliary files.

For example, the mini-driver for most of the L<sup>A</sup>T<sub>E</sub>X submissions was as follows:

```
\input ../TUG99pdf.pre
\input{\jobname\revisionLevel.ltx}
```

Notice that the name of the paper to be processed does not occur explicitly within this file. It is constructed from `\jobname` and `\revisionLevel` (set to 5 within `TUG99pdf.pre`). Thus it is sufficient to have a single file `tug99art.ltx` within the parent directory. Then `rowley.ltx` is just a symbolic link to `../tug99art.ltx`.

For those authors that chose to use the Harvard style of citation, there is a similar mini-driver, called `tug99harv.ltx`, with contents:

```
\PassOptionsToClass{harvardcite}{ltugproc}
\input ../TUG99pdf.pre
\input{\jobname\revisionLevel.ltx}
```

Notice the use of `\PassOptionsToClass`, to ensure that appropriate code is used when `\documentclass` is subsequently encountered.

To prevent `\documentclass` being run twice in the same job, the file `TUG99pdf.pre` concludes with:

```
\renewcommand{\documentclass}[2] [] {}
\let\usepackage\RequirePackage
\let\newcommand\providecommand
```

This way packages loaded from within the author’s source do not cause conflicts (e.g. with options or drivers) when already loaded from `TUG99pdf.pre` or from `ltugproc.cls`. Similarly by forcing the use of `\providecommand`, instead of `\newcommand`, within the author’s manuscript, name-clashes are avoided when the author tries to define a command-name that is already available. Indeed the author’s attempt is ignored completely, so that a consistent style is maintained across all the submitted papers. For example, `\DVI` is defined by `ltugproc.cls` to expand to `\acro{dvi}`, however an author may try to define `\newcommand{\DVI}{\texttt{dvi}}`. Using `\providecommand`, the author’s attempt is ignored, so that any adjustments to the expansion of `\acro` will be applied in this paper also.

**Advice to Authors:** *Get into the habit of using `\providecommand` for stylistic markup, whenever it is conceivable that your document may become part of a journal issue or edited volume. Reserve use of `\newcommand` for text-replacements or macros that are guaranteed to be specific to your own manuscript.*

*Similarly, use `\RequirePackage` whenever possible, rather than `\usepackage`, as this allows easier integration of your source with packages and styles for the journal or edited volume.*



This use of a driver-file has effectively implemented Kaveh Bazargan's idea<sup>3</sup> of using two different class files. An author uses `ltugproc.cls` while preparing his/her manuscript, while the editors use whatever class is requested from `TUG99pdf.pre`. For the record, `TUG99pdf.pre` starts as follows:

```
\PassOptionsToPackage{pdfTeX,colorlinks,
  linkcolor=blue,citecolor=magenta}{hyperref}
\documentclass{ltugproc}
\RequirePackage{latin1}{inputenc}
\RequirePackage{url}
\RequirePackage{html}
\RequirePackage{graphicx}
\RequirePackage{enumerate}
\RequirePackage{alltt}
...
```

Notice that the `hyperref` package is not explicitly requested, since it will be loaded automatically from the `html` package, available with the most recent revisions of `LATEX2HTML`, when the processing is being done by `pdfTeX`.

**Bookmarks.** Automatic bookmarks are created for section and subsections, and also (optionally) for figures and tables, which provides a useful alternative to a Table of Contents, and List of Figures, etc. However, only plain text is allowed for the text of the active hyperlink in the bookmark window. This means that section headings cannot contain styled text, or mathematics, unless an alternative simplified optional argument is supplied. Similarly an optional argument should be provided for complicated, or long, figure captions.

**Advice to Authors:** *Get into the habit of providing optional arguments to section titles and figure/table captions, if only as a comment to be used if required. For example:*

```
\section
%[pdfTeX and LaTeX] % uncomment if needed
{\pdfTeX{} and \LaTeX}
...
```

**Internal Hyperlinks.** `LATEX`'s `\label` and `\ref` mechanism translates directly into active hyperlinks in the PDF document when the `hyperref` package has been loaded. Similarly `\cite` commands produce active links to the bibliography listing, at least with some of the available packages for formatting citations and bibliographies. Patrick Daly's `natbib` package is generally the best to use, and is fully supported by `hyperref` for `pdfTeX` (and `LATEX2HTML`). The Harvard style of citation is also supported by `natbib` by loading it with an optional argument:

```
\usepackage[nharvard]{natbib}
```

so there is no excuse for the die-hards not to use it.

**Advice to Authors:** *Learn to use L<sup>A</sup>T<sub>E</sub>X's symbolic \label-\ref mechanism, if you don't already do so. With electronic documents processed by either pdfTeX or L<sup>A</sup>T<sub>E</sub>X2HTML, the cross-references will become active hyperlinks, which are far more useful than a number or other passive marker. Similarly learn to use natbib for the bibliography and citations.*

**External Hyperlinks.** The best package for formatting URLs is undoubtedly Donald Arseneau's `url.sty`, which can be used with either `LATEX` or `TEX`. It is supported by both `hyperref` and `LATEX2HTML`, to create active hyperlinks in PDF and HTML documents respectively.

A common practice among authors is to typeset URLs using `\texttt` or `{\tt ...}`. This is *visual* markup, not *logical* markup, and should be avoided within the body of the document. It is better to use a `LATEX`-like notation: `\myurl{...}` even if the definition is just `\def\myurl#1{\tt #1}`. This allows an editor to load `url.sty` and insert a single line: `\let\myurl\url` into the preamble of the document to make the hyperlinks active.

There are two quite common errors with URLs. Firstly, don't forget the `http://` at the start, or `ftp://`, or whatever else is appropriate. Acrobat Reader, or a Web-browser, interprets `www.tug.org` as a *relative* URL, resulting in an error.

If a relative reference is indeed intended, e.g. to a directory relative to the author's home-page, then make sure that a valid URL to the home-page is provided within the document preamble. The syntax used by `hyperref` for this is

```
\hyperbaseurl{http://www.tug.org}
```

Even if your document doesn't use `hyperref`, it is useful to include such a line, commented-out, where it can be easily found by the journal editor.

The second common pitfall is in using a notation such as: `CTAN/macros/latex/supported`. While any *TUGboat* reader will understand exactly what is meant, the resulting hyperlink will fail in a browser, since it will be assumed to be a relative URL. If you really wish that string to be displayed, mark it up as:

```
\texttt{CTAN/}\url{macros/latex/supported}
```

and provide a valid `\hyperbaseURL`, such as:

```
ftp://ctan.tug.org/ctan
```

**Advice to Authors:** *Read and understand the issues discussed in the preceding paragraphs.*

<sup>3</sup> in this volume

Another type of active hyperlink can be very effective in an electronic document. For example, every mention of “Adobe Acrobat” or perhaps just the first, can be a hyperlink to the download page to obtain the latest version of the software. Such links are especially useful in bibliography listings, where they can provide a direct link to an electronic version of a cited paper, or to a preprint archive, or a publisher’s Web site. Commands for this are `\href` from `hyperref` and `\htmladdnormallink` from `html.sty`.

**Included Graphics.** Using `pdfTeX` to create the PDF files, it is not possible to include PostScript graphics directly. Instead they must first be converted to PDF, then these can be included as part of the job. The conversion can be done using either Ghostscript, or with Acrobat Distiller. A *Perl* script `epstopdf`, by Sebastian Rahtz and Thomas Esser, creates the correct command for Ghostscript, after having first read the `%%BoundingBox` comment to establish the correct size for the translated image. Alternatively the script `ps2pdf` uses Ghostscript to convert full pages to full PDF pages; if this is more than what is required, it should still be possible to crop the image when it is included in the PDF document. For PostScript files which are not EPS, or for which there is no `%%BoundingBox` comment, then Ghostscript can create a valid EPS version, prior to using `epstopdf`.

As for including the image within the document, the best  $\LaTeX$  command to use is the version of `\includegraphics` from the `graphicx` package. Its optional argument is flexible enough to be able to do anything that is possible with other commands, such as `\psfig` or `\epsfbox`. Furthermore, with `\includegraphics` it is not necessary to include the `.eps` suffix with the filename, since this is the default when a graphics file of this type exists. Similarly when `pdfTeX` is used, the default is `.pdf`, or `.jpg` when there is no appropriate `.pdf` file in the search paths. Hence the codeline

```
\includegraphics[scale=.5]{myimage}
```

suffices to include the correct version of the graphic, either `myimage.eps` with DVI, or `myimage.pdf` or `myimage.jpg` with the PDF version.

**Advice to Authors:** *Check all Encapsulated PostScript graphics for correct `%%BoundingBox` information. Load the `graphicx` package and become acquainted with the possibilities available with the optional argument to `\includegraphics`. Also look at the `\DeclareGraphicsRule` command, if `.jpg` or other graphic formats are to be used.*

## Proceedings Issues

For the individual papers to appear as are of a collection, such as a Journal or Proceedings volume, each paper must contain some things that can only be provided by the editor(s); for example, page numbers and running-heads or footers. For a collection of `.pdf` files, there also needs to be navigation back to a document which provides an overall Table-of-Contents, or other unifying material.

The driver and mini-driver setup makes it very easy to do this, with minimal editing within the individual manuscripts. Firstly the driver assigns a code-number to each job. This is done within `TUG99pdf.pre` by  $\TeX$  code that loops through all the values for `\jobname` until it finds a match with the current document, as follows:

```
\newcount\jobCode
\let\thisJobNum\relax
\edef\thisJobName{\jobname}
\edef\thisJobName{\meaning\thisJobName}
\loop\advance\jobCode by 1\relax
\getAuthorName{\the\jobCode}%
{\ifx\authorName\emptyJob
\gdef\thisJobNum{0}\fi
\edef\testjob{\authorName\revisionLevel}%
\edef\testjob{\meaning\testjob}%
{\ifx\thisJobName\testjob
\xdef\thisJobNum{\the\jobCode}%
\else \ifx ...
\else ...
\fi\fi\fi}}\relax
\ifnum\jobCode >50 \let\thisJobNum\emptyJob\fi
\ifx\thisJobNum\relax\repeat
```

where the `...` denotes extra code that copes with authors having written two or more papers. Notice the technical trick of using `\meaning`, to overcome differences in the category codes of letter-tokens in the expansions of for `\testjob` and `\jobname`. The macro-name `\thisJobNum` holds the required code-number after exiting from the loop, else is `\relax` if there has been some mistake (termination being guaranteed by the arbitrary maximum value of 50 for `\jobCode`).

The value for `\authorName` is supplied via:

```
\def\authorName{}
\def\getAuthorName#1{\edef\authorName{%
\ifcase #1\relax\or
fulling\or
ion\or
...
panelC\else\fi}}
```

in which the authors are listed within the `\ifcase` in the order that the talks were given, or will appear within the proceedings, or whatever other order is most convenient.

Now page-numbers or other things can be obtained from similar `\ifcase` listings; e.g.

```
\def\getTalkPage{\edef\authorPage{%
\ifcase\thisJobNum ???\or % something is wrong
  1001\or %fulling
  1006\or %ion
  1015\or %lovell
  ...
  1158\else
\fi}}
```

This is particularly convenient, as it is no longer necessary to set the page-number explicitly within each author's file, as was being done previously.

Similarly, the date and time scheduled for each talk were recorded in `TUG99pdf.pre`:

```
\def\getTalkDate{\edef\authorDate{%
\ifnum\thisJobNum=0 ??? % something is wrong
\else\ifnum\thisJobNum<10\relax Monday, 16%
\else\ifnum\thisJobNum<15\relax Tuesday, 17%
\else\ifnum\thisJobNum<25\relax Wednesday, 18%
\else\ifnum\thisJobNum<33\relax Thursday, 19%
\fi\fi\fi\fi\fi}}
```

```
\def\AM{\noexpand\,am}
\def\PM{\noexpand\,pm}
\def\getTalkTime{\edef\authorTime{%
\ifcase\thisJobNum ???\or % something is wrong
% Monday
  8.30\AM\or
  9.00\AM\or
  ...
  ...
  3.45\PM\else
\fi}}
```

This information was inserted automatically into the footer of each paper. Furthermore, the footer was made as an active hyperlink to the daily schedule, within `program.pdf`. Thus `program.pdf` serves as the wrapper, apparently combining all the papers into a single volume. A little bit of arithmetic was programmed to correlate the value in `\thisJobNum` with symbolic `\label` names used for anchors in `program.pdf`.

A significant advantage of using the driver file in this way is immediately apparent. Suppose the order of the papers is changed, a paper is withdrawn, or the page-lengths are modified. It is only necessary to make suitable adjustments within the driver file; the author's manuscripts need not be changed at all.

### To $\TeX$ or not to $\TeX$ <sup>4</sup>

Several papers for TUG'99 were submitted using  $\TeX$ , rather than  $\LaTeX$ . These were among the most troublesome to prepare for PDF. It is not difficult to adjust definitions of `\title` and `\author`

to cope with a different syntax. For example, a mini-driver `tug99tex.ltx` copes with the rudimentary book-keeping:

```
\input ../TUG99pdf.pre
\let\latextitle = \title
\let\latexauthor=\author
\let\latexaddress=\address
\let\latexnetaddress=\netaddress
\def\title ##1*{\latextitle{##1}}
\def\author ##1*{\latexauthor{##1}}
\def\address ##1*{\latexaddress{##1}}
\def\netaddress ##1*{\latexnetaddress{##1}}
\def\article{\begin{document}\maketitle}
\def\endarticle{\end{document}\endinput}
\def\head #1\endhead{\section{##1}}
\def\subhead #1\endhead{\subsection{##1}}
\def\subsubhead #1\endhead{%
\noindent\textbf{##1}\ignorespaces}
\let\entry=\bibitem
```

```
\input{\jobname\revisionLevel.tex}
\end{document}
```

What is more difficult is to adapt or edit markup commands used within the body of the manuscript (in particular `\item` and `\itemitem`), or commands used for visual, rather than logical, effect.

### Advice to Authors: *Please use $\LaTeX$ ...*

It is not an issue of pride as to whether an author can typeset beautiful pages himself/herself, or that the default  $\LaTeX$  styles are ugly. Rather, it is imperative to recognise that the author is *not* in control of the ultimate page-layout and style in which his/her words will be typeset.  $\LaTeX$ 's main strength lies in the use of logical markup constructions within the body of the manuscript. This way the author's desires or intentions can be expressed, even when the implementation may be deficient or lacking altogether. Use XML, we can hear Sebastian saying.<sup>5</sup>

**Advice to Authors:** *... at least use  $\LaTeX$ -like markup syntax in the body of the document.*

The need for logical markup is even more imperative with the possibility of different types of output: author's manuscript on paper, printed preprint version, printed proceedings volume, electronic version in PDF and/or HTML. For example, the electronic interpretation of `\url` is very different, and much richer, than the interpretation for paper. Figures and tables should always be floated, no matter how much you detest using this for your own publications; layout is the editor's problem, not the author's.

<sup>5</sup> He is not wrong; we just don't yet have enough robust tools or the experience with it to make this a convenient path to follow.

<sup>4</sup> with apologies to Fred Bartlett [sic].

L<sup>A</sup>T<sub>E</sub>X, through its use of packages, already has well-defined markup conventions for just about everything that might appear in a manuscript. To not take advantage of this means that editors, in trying to give the richest possible interpretation for the particular medium, may not fully understand an author's intentions. This can result in outright errors, or delays in publication while an attempt is made to gain clarification. Instructions like “no macros” (which is clearly ludicrous for a journal about T<sub>E</sub>X-related things) really mean “don't worry about the formatting, but logical markup is quite OK, provided we can change the definition to impose our own styles”. Since the latter is too hard to enunciate, and yet harder still to quantify, it usually comes out as “no macros” which is then largely ignored.

◇ Ross Moore  
Macquarie University  
NSW 2109, Australia  
`ross@maths.mq.edu.au`

## Abstracts

*Les Cahiers GUTenberg*  
**Contents of Double Issue 33/34**  
**(November 1999)**

MICHEL GOOSSENS, *Éditorial : XML ou la démocratisation du web* [Editorial: XML or, the democratisation of the web]; pp. 1–2

The editor sets the scene for the arrival of XML: the realization towards the end of 1996 that there was a very real need to bring consistency and transparency to web page markup across all web browsers. Within two years (10 Feb. 1998) XML emerged, addressing three critical issues raised by Jon Bosak (Sun Microsystems), who also chaired the XML working group: extensibility of the markup, sufficient depth of structures being marked, and validation of the markup. Bosak's article (in English) can be found at [xml.com/pub/w3j/s3.bosak.html](http://xml.com/pub/w3j/s3.bosak.html).

The remainder of the editorial relates these points to the articles: the specification itself is reproduced in a very useful format (French on left-hand pages, the definitive English text on the right). To be read along with the specification is a very thorough introduction, written by Michel Goossens. Together, these provide a very useful tool to the French-speaking user, as so much terminology is being either invented from scratch or

new precise meanings assigned old familiar words. Such pairings are indispensable to those who must work in both languages — or those who must translate from one into the other!

The specification covers 90 pages, the introduction another 124. And this double issue has still more: a comparison between SGML and XML, an introduction to Document Object Models (interfaces for XML documents), generating MathML in Omega, a program to generate MathML-encoded mathematics, and finally, the issue closes with a translation of the XML FAQ (v.1.5, June 1999), maintained by Peter Flynn.

In all, over 300 pages devoted to XML.

MICHEL GOOSSENS, *XML et XSL : un nouveau départ pour le web* [XML and XSL: A new venture for the Web]; pp. 3–126

Late in 1996, the W3C and several major software vendors decided to define a markup language specifically optimized for the Web: XML (eXtensible Markup Language) was born. It is a simple dialect of SGML, which does not use many of SGML's seldom-used and complex functions, and does away with most limitations of HTML. After an introduction to the XML standard, we describe XSL (eXtensible Stylesheet Language) for presenting and transforming XML information. Finally we say a few words about other recent developments in the XML arena.

[Author's abstract]

As mentioned in the editorial, this article is intended to be read in conjunction with the actual specification, provided later in the same issue.

SARRA BEN LAGHA, WALID SADFI and MOHAMED BEN AHMED, *Comparaison SGML-XML* [An SGML-XML Comparison]; pp. 127–154

The media hype surrounding the eXtensible Markup Language (XML) Leads us to hope that future Web documents will be better structured and easier to re-use. The XML specification, which addresses the wish of the Web community to have a language more flexible than HTML without necessarily adopting the rigidity and complexity of SGML, is considered a step forward since it incorporates technical advances of both the HTML and SGML worlds. In the present article we explain the differences (improvements) between XML and SGML. Since each XML document is by construction a valid SGML document, we review the basic principles of both standards and present a detailed comparison of XML and SGML.

[Authors' abstract]

FRANÇOIS ROLE and PHILIPPE VERDRET, Le Document Object Model (DOM) [The Document Object Model (DOM)]; pp.155–171

The present article gives an overview of the Document Object Model (DOM), a hierarchy of standard interfaces proposed by the W3 Consortium. It allows application programs to access the structure of XML documents and manipulate their content. We start with a brief theoretical description of the DOM. Then we have a look at a few use cases expressed in three languages (Java, Perl and JScript). The parallel treatment in these three languages should allow you to get an idea of the functionality offered by the DOM, as well as emphasize its programming language-neutral character. At the end of the article we discuss the present limitations of the DOM and its foreseeable future evolution. [Authors' abstract]

YANNIS HARALAMBOUS and JOHN PLAICE, Produire du MathML et autres \*ML à partir d' $\Omega$  : Omega se généralise [Generating MathML and other \*MLs from  $\Omega$ : The Generalization of Omega]; pp.173–182

Nowadays, the Omega typesetting system not only lets you generate typographically excellent documents in many scripts, but you can also use it to transform the input of your Omega files into SGML. In particular, mathematics expressions will be translated automatically into MathML while through a redefinition of the  $\LaTeX$  macros any kind of SGML tags can be obtained, thus turning the editor into a powerful system. [Authors' abstract]

BENJAMIN JENNES AND RAPHAËL MARÉE, Un compilateur d'expressions mathématiques générant du MathML [A compiler to generate MathML from mathematical expressions]; pp.183–190

In this article we look at the problem of publishing mathematical expressions on the Web. We present a solution based upon the use of compiling techniques and the MathML language. After a general description of the approach, we describe the different stages of the compilation with the help of an example. We conclude with a discussion of the advantages and limitations of `maje`, the program we have implemented. [Author's abstract]

THE W3 CONSORTIUM, La spécification XML [The XML specification]; pp.191–280

This document contains the original English (odd-numbered pages) and French translation (even-numbered pages) of the eXtensible Markup Language (XML) “recommendation” v.1.0 ([www.w3.org/TR/1998/REC-xml-19980210](http://www.w3.org/TR/1998/REC-xml-19980210)) by the W3 Consortium, dated 10 February 1998.

The French text may contain errors not present in the original, due to the translation effort. Thus, the English text is the definitive version.

The following people worked on the translation: Patrick Ion, Samira Cuny, Alain La Bonté, Nicolas Lesbats, and François Yergeau.

An HTML file of the translation (along with other material) can be found at [http://babel.alis.com/web\\_ml/xml/](http://babel.alis.com/web_ml/xml/).

[Translation of opening paragraphs]

PETER FLYNN, Foire aux questions XML [An XML FAQ (v.1.5, June 1999)]; pp.281–311

This document contains the most Frequently Asked Questions about XML — along with the answers. The FAQ attempts to provide users, developers and others with an entry level of information but in no way is it part of the XML standard itself.

[Translation of French résumé]

This translation was provided by Morgane le Bihan and Dreves Ewen. The original English-language version can be obtained in various formats from [www.ucc.ie/xml/](http://www.ucc.ie/xml/), as well as being “available in oil-based toner on flattened dead trees” — Peter’s humour still in fine form!

— \* —

Articles from *Cahiers* issues can be found in PostScript format at the following site:

<http://www.gutenberg.eu.org/pub/gut/publications/publis.html>

[Compiled by Christina Thiele]

---

## Euro $\TeX$ '99 Proceedings Paperless $\TeX$

The Euro $\TeX$  '99 Conference took place in Heidelberg, Germany, 20–24 September 1999. It was organized by the University of Heidelberg, represented by the Institute of Psychology and the University Computing Center. GÜNTER PARTOSCH and GERHARD WILHELMS, Words from the Editors; pp.1–2

### Paper-less $\TeX$

JACQUES ANDRÉ and HÉLÈNE RICHY, Paper-less editing and proofreading of electronic documents; pp.3–16

This paper describes a system for editing and proofreading electronic documents with a pen-interface: with the help of a mouse, or, better, an

electronic stylus on a display tablet, an author may indicate how he wants to edit his text (e.g. cross-line words to delete them, underline to compose in italics, etc.). There is no learning step: however, there is a dictionary of simple and natural drawings which are easy to remember. [In] this way, some natural gestures allow a comfortable interface for correcting digital documents.

This prototype is based on studies on ergonomy of editing signs, localization into a structured document, sign recognition in an editing context, and modeling of correcting commands. The pen-interface is integrated within *Amaya*, the W3C interactive Web editor, for correcting Web pages. A version for correcting structured documents is experimented upon within a structured editor.

HANS HAGEN, The NTG MAPS bibliography — from SGML to  $\text{\TeX}$  to PDF; pp. 17–39

A few years ago the NTG decided to put their MAPS volumes on the internet in the PDF file format. At about the same time, it was decided to build the associated bibliography, in such a way that it could be used to produce both an HTML and PDF document.

Recently the MAPS bibliography has been converted to a proper XML document source. In the process the descriptions were made as consistent as possible. The XML source was used as input for a PDF document with extensive browse and search options. This PDF file, along with the MAPS articles, is provided to NTG members as an additional service.

In this article, the electronic NTG MAPS will be presented and the specific characteristics of the production process will be explained. Also, some of the complicating aspects will be discussed. I assume that the reader is familiar with SGML and  $\text{\TeX}$ . The focus will be on the interfacing between SGML,  $\text{\TeX}$  and PDF.

HANS HAGEN, Which way are we heading? — In search for the holy grail; pp. 40–49

Is  $\text{\TeX}$  really out-of-date? Are we making a fool out of ourselves when we stick to using  $\text{\TeX}$ ? Does  $\text{\TeX}$  gain the attention it deserves? In this paper I will elaborate on these questions. The oral counterpart of this paper will be illustrated with some examples.

HEIKO OBERDIEK, PDF information and navigation elements with `hyperref`, `pdf $\text{\TeX}$`  and `thumbpdf`; pp. 50–68

The PDF format offers additional possibilities for information and navigation through paperless on-line documents. This paper shows how the

navigation features bookmarks and thumbnails can be created automatically or manually by powerful packages like `hyperref` and `thumbpdf`. The problems and solutions are described that arise from converting  $\text{\TeX}$  strings to the PDF ones used in the general document information or in the outlines.

KRISTOFFER ROSE, Towards an XML DTD for  $\text{\LaTeX}$  — Technical Workgroup initiative for reformulating  $\text{\LaTeX}$  as an XML application; pp. 69–70

We propose the creation of a working group with the goal of making it possible to process  $\text{\LaTeX}$  documents in XML.

SERGEY A. STRELKOV, Testbed for Preparation of a Russian Patent Document in XML Format; pp. 71–85

An experimental technology for Russian patent document preparation in XML is described. For this purpose the special draft XML PatDoc DTD is used. This technological DTD is the analog of the subset SGML DTD of the World Intellectual Property Organization (WIPO) Standard ST.32. This compound XML document may contain mathematical expressions and chemical formulae which are presented by reference to the appropriate files. The paper patent document image is stored in TIFF format. Chemical formulae are simultaneously stored in one of these formats: `mol`, `cml`, `wmf`, `gif`, `eps`. Images of mathematical expressions are stored in the formats `gif` or `eps`, and are placed at the end of the document in the codings: Plain  $\text{\TeX}$ , `MathType`, `MathML`. The components of the patent documents are typed in MS-Word 97 (Windows NT 4.0 or Windows 95). `MathType` is used to type the mathematical expressions; `ISIS/DRAW` — for the chemical formulae. The compound `rtf` file is converted to an `xml` file with PatDoc DTD using these programs: `rtf2xml`, `OmniMark LE 4.01`,  `$\text{\TeX}$ 4ht`, `MiK $\text{\TeX}$` , `NSGMLS`, `SGMLpm`, `IE 4.01`, `msxsl.cab`. The programs are glued using VBA macros for Word 97 and a set of Perl scripts.

ANSELM LINGNAU, `TkDVI`: DVI Previewing with `Tcl` and `Tk`; pp. 86–100

Application-level scripting is a powerful method for structuring software. This paper introduces `TkDVI`, a  $\text{\TeX}$  DVI previewer based on the `Tcl`/`Tk` scripting language and graphics toolkit. After a brief introduction to `Tcl`/`Tk`, we present the design and major components of the previewer, pointing out the specific advantages gained by using `Tcl`/`Tk`. A number of extensions and future projects are also discussed.

MATTHEW BAKER, Visualization of electrophoresis gels using  $\TeX$ ; pp. 101–108

This paper describes a  $\TeX$  system for creating interactive PDF files to visualize electrophoresis data. A Perl program processes greyscale electrophoresis images, segmentations derived from these images and computed numerical data to create a hyperlinked document in  $\TeX$ . This paper describes the steps involved.

### Fonts

ALEXANDER BERDNIKOV, Fonts for paperless  $\TeX$ : How to make them?; pp. 109–116

‘Paperless’  $\TeX$  requires  $\TeX$  fonts to be in Type 1 format rather than in METAFONT format. While METAFONT is still the most flexible tool for designing fonts, direct conversion of .mf files into Type 1 or TTF binary files is still not a routine procedure. We will discuss what can be done in this area to make it the standard procedure for any user.

These problems are discussed:

1. What ‘standard’ ways of MF to PFB conversion exist now and what are the limitations of these tools?
2. What are restrictions for the METAFONT source to be converted into PFB using the ‘standard’ converting routines?
3. Is there some technology which enables to create METAFONT and Type 1 fonts in a parallel manner?
4. Is there some chance that someday a freeware universal tool like METAPOST or pdf $\TeX$  will appear to make such conversion easy and flexible, or to organize the parallel design of MF + PFB in a comfortable style?

BOGUSŁAW JACKOWSKI, JANUSZ M. NOWACKI and PIOTR STRZELCZYK, Antykwa Półtawskiego: a parameterized outline font; pp. 117–141

There have been several attempts to generate outline fonts from a METAFONT or METAPOST source. It looks as if such an approach has necessitated manual tuning. The aim of this paper is to share our experiences with preparing a replica of a traditional Polish type, Antykwa Półtawskiego (Półtawski’s Antique) as a METAPOST “metasource”, i.e., in such a way that a variety of outline instances (Type 1) of the font can be generated on the fly.

KAREL PÍŠKA, Fonts for Neo-Assyrian Cuneiform; pp. 142–154

This paper presents  $\TeX$  and PostScript fonts for typesetting cuneiform (Akkadian, Hittite, Old

Persian, and Ugaritic) in a form similar to script of the Neo-Assyrian period. The fonts have been developed in the Type 1 font format; fonts for *Syllabary A* were also created in METAFONT.

TACO HOEKWATER, An extended maths font set for processing MathML; pp. 155–164

In the autumn of [1998], work started on a new set of mathematical fonts that are intended to cover the full range of characters included in MathML as well as those included in the proposals for maths extensions in the next version of Unicode.

This paper presents the first result of that work: A new Times-compatible maths fonts set consisting of about 1500 symbols and a few alphabets, along with a collection of  $\TeX$  macros to use them.

These fonts are donated to the public domain by Kluwer Academic Publishers and are available in both METAFONT source and Adobe Type 1 formats.

ALEXANDER BERDNIKOV and OLGA LAPKO, Old Slavonic and Church Slavonic in  $\TeX$  and Unicode; pp. 165–196

The characteristic features of Cyrillic (Old Slavonic and Church Slavonic) writing systems are analyzed and compared. The old numbering rules and the difference between the canonical orthodox Church Slavonic and ‘old believer’ Church Slavonic are considered as well. It is shown that Old Slavonic and Church Slavonic differ strongly, and should at the very least be considered as two well distinguished dialects of the same writing system. An analysis of the current state of the Unicode 04xx encoding page shows that it is not sufficient to represent the Old Slavonic and Orthodox Church Slavonic writings adequately. The project of T2D encoding which enables the representation in  $\TeX$  of out-of-date Bulgarian texts (from the middle of the 19th century till 1945), Russian texts (1703–1918 and emigrant literature) and Church Slavonic/Old Slavonic texts, is described.

### Maths

RICHARD W. D. NICKALLS, mathsPIC — A filter program for use with P $\TeX$ ; pp. 197–210

This article presents an overview of the mathsPIC utility for the P $\TeX$  drawing engine. MathsPIC facilitates the drawing of mathematical diagrams by allowing the manipulation of points by name rather than by coordinates. Some familiarity with the P $\TeX$  package is assumed.



VALENTIN ZAITSEV, ANDREW JANISCHEWSKY and ALEXANDER BERDNIKOV, Russian typographical traditions in mathematical literature; pp. 211–227

Although the general Russian typographical traditions are already reviewed in several publications, the specifics of mathematical publication and mathematical formula presentation in Russian books and journals is still not described in full detail. This paper describes the traditions of mathematical publications and, especially, the characteristic features of the style and the graphical forms of the mathematical symbols used in mathematical notation. The ways in which these specifics can be implemented in  $\text{\TeX}$  are discussed briefly as well.

### New Typesetting System ( $\mathcal{N}\mathcal{T}\mathcal{S}$ )

JOACHIM LAMMARSCH, The history of  $\mathcal{N}\mathcal{T}\mathcal{S}$ ; pp. 228–232

Beginning in 1991, a project was initiated by DANTE e.V. to design and construct a successor to  $\text{\TeX}$ . This overview discusses the historical and political framework in which the project has been taking place.

HANS HAGEN, Some  $\mathcal{N}\mathcal{T}\mathcal{S}$  thoughts; pp. 233–240

There are already several extensions to  $\text{\TeX}$ , including  $\varepsilon\text{\TeX}$  and  $\text{pdf}\text{\TeX}$ , and the re-implementation of the Pascal source of  $\text{\TeX}$  in Java is nearly complete. Some thoughts are presented concerning how the next steps should be coordinated to ensure consistency and continuity, and what features needed for contemporary document production are not now well supported and should be provided. Since the present  $\text{\TeX}$  implementations do their job well and reliably, there is time for the  $\mathcal{N}\mathcal{T}\mathcal{S}$  team to approach their task carefully, since we are talking about life-long tools.

JIRÍ ZLATUŠKA,  $\mathcal{N}\mathcal{T}\mathcal{S}$ : Programming languages and paradigms; pp. 241–245

Developments in computer software and hardware since  $\text{\TeX}$ 's creation have changed the ground rules and rendered many assumptions obsolete. This essay presents the considerations and discussion that informed the decision to use Java as the language in which the  $\mathcal{N}\mathcal{T}\mathcal{S}$  is being implemented.

KAREL SKOUPÝ and PHILIP TAYLOR, The implementation of  $\mathcal{N}\mathcal{T}\mathcal{S}$ ; pp. 246–260

This paper addresses the actual implementation of  $\mathcal{N}\mathcal{T}\mathcal{S}$ ; it is intended to provide the reader as much detail as can reasonably be accommodated in a paper intended to appear in the Conference Pro-

ceedings. A considerably more detailed version will eventually be available as an accompaniment to (or possibly integrated in) the JavaDoc documentation which will accompany the released version of  $\mathcal{N}\mathcal{T}\mathcal{S}$ .

### Miscellaneous

EDRMUTHE MEYER ZU BEXTEN and JENS HILTNER,  $\text{\LaTeX}$ : Das Satzsystem für sehgeschädigte Studierende [ $\text{\LaTeX}$ : The text processing system for visually handicapped students]; pp. 261–280

There are many blind and visually handicapped people in the Federal Republic of Germany who would like to go to university. Especially in the natural and technical sciences, mathematics is very important. The question is, how can mathematics be made more understandable for these students? Computer assisted mathematical writing systems for the blind have been conceived for many years, but they have also shown a variety of problems. In the new center for blind and visually handicapped students of the Fachhochschule Gießen-Friedberg, a different direction is being taken, by using the globally recognized and accepted program  $\text{\LaTeX}$ .

MICHAEL PIOTROWSKI, JENS KLÖCKNER and JÖRG KNAPPEN, Is  $\text{\LaTeX}2_{\varepsilon}$  markup sufficient for scientific articles?; pp. 281–289

The markup of the standard article class is compared with the requirements of several standard SGML DTDs (majour, docbook, iso12083), concentrating on header information (author/address markup) and bibliographic information.

WŁODEK BZYL, Detection and correction of spelling errors in marked-up documents; pp. 290–307

This paper discusses the problem of detecting and correcting spelling errors in marked up documents. We divide the problem into three separate tasks and propose solutions to all of them. Based on this division, a simple system that provides the ability to deal satisfactorily with any  $\text{\TeX}$  markup is presented. It is the first system of which the author is aware that is able to deal with multilingual documents.

### Appendix

Author index; pp. 308–312

[Compiled by Barbara Beeton]

## Calendar

### 1999

- Dec 11 NTS talk by Hans Hagen, Masaryk University, Brno, Czech Republic.
- Dec 13 Tutorial, “All the nice things we can do with pdf(T<sub>E</sub>X)”, Hans Hagen, Masaryk University, Brno, Czech Republic. To attend, register with [secretary@cstug.cz](mailto:secretary@cstug.cz).

---

### 2000

- Feb 7 *TUGboat* **21** (1), deadline for technical submissions.
- Feb 7–11 Seybold Seminars Boston/Publishing 2000, Boston, Massachusetts. For information, visit <http://www.seyboldseminars.com/Events>.
- Feb 21 *TUGboat* **21** (1), deadline for reports and news items.
- Feb 23 Johannes Gutenberg’s 600<sup>th</sup> Birthday! Born 23 February 1400
- Feb 27 – Mar 2 XTECH 2000, the XML Developers Conference: “Looking back, going forward”, San Jose, California. For information, visit [http://www.gca.org/attend/2000\\_conferences/xttech\\_2000/](http://www.gca.org/attend/2000_conferences/xttech_2000/).
- Mar 8–11 DANTE 2000 and 22<sup>nd</sup> meeting, Technische Universität Clausthal-Zellerfeld, Germany. For information, visit <http://dante2000.itm.tu-clausthal.de/>.
- Apr 1 – Jun 11 Exhibition, “Sumner Stone, Calligraphy and Type Design in a Digital Age”, Ditchling Museum, Ditchling, Sussex, UK. For information, visit <http://www.letteringtoday.co.uk/>.
- Apr 11 *TUGboat* **21** (2), deadline for technical submissions.

- Apr 30 – May 2 BachoT<sub>E</sub>X 2000, 8<sup>th</sup> annual meeting of the Polish T<sub>E</sub>X Users’ Group (GUST), “T<sub>E</sub>X on the turn of the 20th century”, Bachotek, Brodnica Lake District, Poland. For information, visit <http://www.gust.org.pl/>.
- May 9 *TUGboat* **21** (2), deadline for reports and news items.
- May 10–12 GUTenberg 2000, “L<sup>A</sup>T<sub>E</sub>X et XML : coopération pour l’internet”, Toulouse, France. For information, visit <http://www.gutenberg.eu.org/manif/gut2000/gut2000.html>.
- May 19–21 Typography Forum, “Navigation durch Text Bild Raum”, Museum der Arbeit, Hamburg, Germany. For information, visit <http://www.forumtypographie2000.de>.
- Jun 1–3 Society for Scholarly Publishing, 22<sup>nd</sup> annual meeting, Baltimore, Maryland. For information, visit <http://www.sspnet.org>.
- Jun 12–16 XML Europe 2000, Palais des Congrès de Paris, France. For information, visit [http://www.gca.org/attend/2000\\_conferences/europe\\_2000/](http://www.gca.org/attend/2000_conferences/europe_2000/).
- Jun 15 NTG 25<sup>th</sup> Meeting, Rijksuniversiteit Groningen, The Netherlands. For information, contact [ntg@ntg.nl](mailto:ntg@ntg.nl).
- Jun 16–18 TypeCon 2000, Westborough, Massachusetts. For information, visit <http://www.typesociety.org>.
- Jun 21–23 Typo[media]2000, “Links to Minds, Mainz, Germany. Linotype’s design conference; for information, visit <http://www.typomedia.com>.

*Status as of 15 March 2000*

For additional information on TUG-sponsored events listed above, contact the TUG office (+1 503 223-9994, fax: +1 503 223-3960, e-mail: [office@tug.org](mailto:office@tug.org)). For events sponsored by other organizations, please use the contact address provided.

Additional type-related events and news items are listed in the Sans Serif Web pages, at <http://www.quixote.com/serif/sans>.

- Jul 21–25 ALLC-ACH 2000: Joint International Conference of the Association for Literary and Linguistic Computing, and Association for Computers and the Humanities, Glasgow, Scotland, UK. For information, visit <http://www.ach.org/>.
- Jul 23–28 SIGGRAPH 2000, New Orleans, Louisiana. For information, visit <http://www.siggraph.org/calendar/>.
- Aug 12–18 **TUG 2000** — The 21<sup>st</sup> annual meeting of the T<sub>E</sub>X Users Group, “T<sub>E</sub>X enters a new millennium”, Wadham College, Oxford, UK. For information, visit <http://tug2000.tug.org/>.
- Aug 28–  
Sep 1 Seybold San Francisco, San Francisco, California. For information, visit <http://www.seyboldseminars.com/Events>.
- Sep DK-TUG, 2<sup>nd</sup> Annual General Meeting. For information, visit <http://sunsite.auc.dk/dk-tug/>.
- Sep 11–12 PODDP '00: 5<sup>th</sup> International Workshop on Principles of Digital Document Processing, Munich, Germany. For information, visit <http://www.cs.uwm.edu/~poddp00>.
- Sep 12 *TUGboat* 21 (3), deadline for reports and news items.
- Sep 13–15 DDEP00: 8<sup>th</sup> International Conference on Digital Documents and Electronic Publishing, Munich, Germany. For information, visit <http://www11.in.tum.de/DDEP00>.
- Sep 19 *TUGboat* 21 (4), deadline for technical submissions.
- Oct 17 *TUGboat* 21 (4), deadline for reports and news items.
- Oct 20–21 MathML and Technologies for Math on the Web, Urbana-Champaign, Illinois. For information, visit <http://www.mathmlconference.org>.
- Nov 17–19 Conference: Eric Gill & St. Dominic’s Press, University of Notre Dame, Notre Dame, Indiana; three concurrent exhibitions of Gill’s and related work will be held in the University museums and library. For information, visit <http://www.nd.edu/~jsherman/gill/>.
- Dec 3–7 XML 2000/Markup Technologies 2000, Washington, DC. For information, visit [http://www.gca.org/attend/2000\\_conferences/XML\\_2000/](http://www.gca.org/attend/2000_conferences/XML_2000/).

## Late-Breaking News

### Production Notes

Mimi Burbank

Once again, the issue is late due largely to circumstances beyond our control. In January of 2000, SCRI was “transmogrified” into the School of Computational Science and Information Technology (CSIT) at Florida State University, and some adjustments have been needed.

For the first time, all of the articles in this issue were submitted as L<sup>A</sup>T<sub>E</sub>X files. Ten different packages were required, and when attempting to run *all* files as one source document, I had a 300i stack overflow. I removed from the main file the most likely culprit — the article using the `html` and `hyperref` packages — with the desired effect. Most of the articles used the `graphicx` package, and several files using `epsf.sty` were recoded to use `graphicx`. As a result, I wound up with four different PostScript files: two from L<sup>A</sup>T<sub>E</sub>X source and two from plain T<sub>E</sub>X source.

The problem which prompted me to attempt to run all of the L<sup>A</sup>T<sub>E</sub>X files as one job had to do with the vertical spacing of documents. I was able to “knit” files together much better when processing them as a single job. Curiously enough, when an article was removed from the unified file, the vertical spacing was completely different. We will definitely be experimenting with this more in the future.

**Output.** The final camera copy was prepared at CSIT on a Sun Enterprise server running Solaris 7.0, using the *T<sub>E</sub>X Live* 4 setup, which is based on the *Web2c* T<sub>E</sub>X implementation version 7.3 by Karl Berry and Olaf Weber. PostScript output, using outline fonts, was produced using Radical Eye Software’s `dvips(k)` 5.85, and printed on an HP LaserJet 4000 TN printer at 1200dpi.

**Coming In Future Issues.** The March 2000 issue will contain the T<sub>E</sub>XLive 5 CD-ROM, and we hope to have the latest update on X<sub>M</sub>T<sub>E</sub>X, the macro package for drawing chemical structural formulae by Shinsaku Fujita. The Treasure Chest will include not only more package tours, but also the inventory of packages posted to CTAN during the first quarter of the year 2000.

◇ Mimi Burbank  
CSIT, Florida State University,  
Tallahassee, FL 32306–4120  
[mimi@csit.fsu.edu](mailto:mimi@csit.fsu.edu)

# TUG 2000

Wadham College, Oxford, UK  
August 13th–16th, 2000

The 21st Annual Conference of the T<sub>E</sub>X Users Group will take place at Wadham College, Oxford, between Sunday 13th August and Wednesday 16th August 2000. Tutorials will be given on the 17th and 18th August.

---

## *The Location*

Oxford is a small, pleasant city with an internationally famous university. The city is full of ancient buildings, beautiful gardens, libraries and bookshops. The conference will be held in Wadham College, a traditional college (founded 1613) in the centre of the city. Oxford is easily reached from London, and is a good starting point for visiting much of southern England.



## *The Conference*

The conference will feature talks on all aspects of T<sub>E</sub>X and its relationship to both traditional and electronic document preparation and processing. The Annual General Meeting of the T<sub>E</sub>X Users' Group will be held during the period of the conference.

We expect the cost to a typical delegate to be about £300, including accommodation and meals; cheaper accommodation and bursaries will also be available.

The conference chairman is Sebastian Rahtz (Oxford University Computing Services) and local organisation is led by Kim Roberts (Oxford University Press).



---

## *Dates and Contacts*

<b>15th January 2000</b>	Proposals for papers
<b>31st January 2000</b>	Acceptance of papers
<b>15th February 2000</b>	Publication of booking form and prices
<b>31st March 2000</b>	Delivery of papers for refereeing
<b>31st May 2000</b>	Delivery of final papers
<b>General enquiries:</b>	<a href="mailto:tug2000-enquiries@tug.org">tug2000-enquiries@tug.org</a>
<b>Paper submissions:</b>	<a href="mailto:tug2000-papers@tug.org">tug2000-papers@tug.org</a>

*Sebastian Rahtz*  
*OUCS*

*13 Banbury Road*  
*Oxford OX2 6NN, UK*

*Tel: +44 1865 283431*  
*<http://tug2000.tug.org/>*

## Institutional Members

American Mathematical Society,  
*Providence, Rhode Island*

CNRS - IDRIS,  
*Orsay, France*

College of William & Mary,  
Department of Computer Science,  
*Williamsburg, Virginia*

CSTUG, *Praha, Czech Republic*

Florida State University,  
Supercomputer Computations  
Research, *Tallahassee, Florida*

Hong Kong University of  
Science and Technology,  
Department of Computer Science,  
*Hong Kong, China*

IBM Corporation,  
T J Watson Research Center,  
*Yorktown, New York*

ICC Corporation,  
*Portland, Oregon*

Institute for Advanced Study,  
*Princeton, New Jersey*

Institute for Defense Analyses,  
Center for Communications  
Research, *Princeton, New Jersey*

Iowa State University,  
Computation Center,  
*Ames, Iowa*

Kluwer Academic Publishers,  
*Dordrecht, The Netherlands*

KTH Royal Institute of  
Technology, *Stockholm, Sweden*

Los Alamos National Laboratory,  
University of California,  
*Los Alamos, New Mexico*

Marquette University,  
Department of Mathematics,  
Statistics and Computer Science,  
*Milwaukee, Wisconsin*

Masaryk University,  
Faculty of Informatics,  
*Brno, Czechoslovakia*

Max Planck Institut  
für Mathematik,  
*Bonn, Germany*

New York University,  
Academic Computing Facility,  
*New York, New York*

Princeton University,  
Department of Mathematics,  
*Princeton, New Jersey*

Space Telescope Science Institute,  
*Baltimore, Maryland*

Springer-Verlag Heidelberg,,  
*Heidelberg, Germany*

Stanford University,  
Computer Science Department,  
*Stanford, California*

Stockholm University,  
Department of Mathematics,  
*Stockholm, Sweden*

University of Canterbury,  
Computer Services Centre,  
*Christchurch, New Zealand*

University College, Cork,  
Computer Centre,  
*Cork, Ireland*

University of Delaware,  
Computing and Network Services,  
*Newark, Delaware*

Universität Koblenz-Landau,  
Fachbereich Informatik,  
*Koblenz, Germany*

University of Oslo,  
Institute of Informatics,  
*Blindern, Oslo, Norway*

University of Texas at Austin,  
*Austin, Texas*

Università degli Studi di Trieste,  
*Trieste, Italy*

Uppsala University,  
Computing Science Department,  
*Uppsala, Sweden*

Vanderbilt University,  
*Nashville, Tennessee*

Vrije Universiteit,  
*Amsterdam, The Netherlands*

## T<sub>E</sub>X Consulting & Production Services

Information about these services can be obtained from:

**T<sub>E</sub>X Users Group**  
 1466 NW Naito Parkway, Suite 3141  
 Portland, OR 97209-2820, U.S.A.  
 Phone: +1 503 223-9994  
 Fax: +1 503 223-3960  
 Email: [office@tug.org](mailto:office@tug.org)  
 URL: <http://www.tug.org/consultants.html>

### North America

**Hargreaves, Kathryn**

135 Center Hill Road,  
 Plymouth, MA 02360-1364;  
 (508) 224-2367; [letters@cs.umb.edu](mailto:letters@cs.umb.edu)

I write in T<sub>E</sub>X, L<sup>A</sup>T<sub>E</sub>X, METAFONT, MetaPost, PostScript, HTML, Perl, Awk, C, C++, Visual C++, Java, JavaScript, and do CGI scripting. I take special care with mathematics. I also copyedit, proofread, write documentation, do spiral binding, scan images, program, hack fonts, and design letterforms, ads, newsletters, journals, proceedings and books. I'm a journeyman typographer and began typesetting and designing in 1979. I coauthored *T<sub>E</sub>X for the Impatient* (Addison-Wesley, 1990) and some psychophysics research papers. I have an MFA in Painting/Sculpture/Graphic Arts and an MSc in Computer Science. Among numerous other things, I'm currently doing some digital type and human vision research, and am a webmaster at the Department of Engineering and Applied Sciences, Harvard University. For more information, see: <http://www.cs.umb.edu/kathryn>.

**Loew, Elizabeth**

President, T<sub>E</sub>Xniques, Inc.,  
 675 Massachusetts Avenue, 6th Floor,  
 Cambridge, MA 02139;  
 (617) 876-2333; Fax: (781) 344-8158  
 Email: [loew@texniques.com](mailto:loew@texniques.com)

Complete book and journal production in the areas of mathematics, physics, engineering, and biology. Services include copyediting, layout, art sizing, preparation of electronic figures; we keyboard from raw manuscript or tweak T<sub>E</sub>X files.

**Ogawa, Arthur**

40453 Cherokee Oaks Drive,  
 Three Rivers, CA 93271-9743;  
 (209) 561-4585  
 Email: [Ogawa@teleport.com](mailto:Ogawa@teleport.com)

Bookbuilding services, including design, copyedit, art, and composition; color is my speciality. Custom T<sub>E</sub>X macros and L<sup>A</sup>T<sub>E</sub>X<sub>2 $\epsilon$</sub>  document classes and packages. Instruction, support, and consultation for workgroups and authors. Application development in L<sup>A</sup>T<sub>E</sub>X, T<sub>E</sub>X, SGML, PostScript, Java, and BC++. Database and corporate publishing. Extensive references.

### Outside North America

**DocuT<sub>E</sub>Xing: T<sub>E</sub>X Typesetting Facility**

43 Ibn Kotaiba Street,  
 Nasr City, Cairo 11471, Egypt  
 +20 2 4034178; Fax: +20 2 4034178  
 Email: [main-office@DocuTeXing.com](mailto:main-office@DocuTeXing.com)

DocuT<sub>E</sub>Xing provides high-quality T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X typesetting services to authors, editors, and publishers. Our services extend from simple typesetting and technical illustrations to full production of electronic journals. For more information, samples, and references, please visit our web site: <http://www.DocuTeXing.com> or contact us by e-mail.