

TUGBOAT

Volume 23, Number 3/4 / 2002

	243	Addresses
General Delivery	245	From the Board of Directors / <i>Karl Berry</i>
	245	Editorial comments / <i>Barbara Beeton</i> TeX 3.141592, METAFONT 2.71828; Glitch in <i>TUGboat</i> 22:4; Farewell, Michael Downes; Last cartoon by Roy Preston; A web site for Arabic typography; Recognition for the Plantin Museum; An alphabet game for children of all ages; <i>Making TeX Work</i> on CTAN; <i>TeX for the Impatient</i>
	247	Hyphenation exception log
	249	Donald Knuth: All questions answered (University of Oslo, 30 August 2002)
Dreamboat	261	Floating point numbers and METAFONT, METAPOST, TeX, and PostScript Type 1 fonts / <i>Claudio Beccari</i>
Typography	269	The Octavo package / <i>Stefan A. Revets</i>
Philology	276	The teubner LATEX package: Typesetting classical Greek philology / <i>Claudio Beccari</i>
	282	Typesetting in Bengali script using TeX / <i>Palash B. Pal</i>
Electronic Documents	288	interactiveworkbook: LATEX-based interactive PDF on the Web / <i>Jonathan Kuhn</i>
Font Forum	291	Multiple Master math extension fonts / <i>D. Men'shikov, A. Kostin, and M. Vulis</i>
	294	TrueType fonts in PostScript / <i>Thomas H. Barton</i>
	296	The Kerkis font family / <i>Antonis Tsolomitis</i>
	301	Euler-VM: Generic math fonts for use with LATEX / <i>Walter Schmidt</i>
Software & Tools	304	Rambutan: Literate programming in Java / <i>Prasenjit Saha</i>
Graphics Applications	309	Eukleides: A geometry drawing language / <i>Christian Obrecht</i>
	313	METATEX / <i>Ramón Casares</i>
Hints & Tricks	319	The treasure chest / <i>Mark LaPlante and William F. Adams</i>
Tutorials	329	Introduction to pdfTeX / <i>Thomas Feuerstack</i>
LATEX	335	Constructing circuit diagrams with <i>pst-circ</i> / <i>Herbert Voß</i>
	341	Absolute positioning with <i>textpos</i> / <i>Norman Gray</i>
	344	Multilingual bibliographies: Using and extending the <i>babelbib</i> package / <i>Harald Harders</i>
Abstracts	354	<i>Les Cahiers GUTenberg</i> : Contents of Thematic Issue 41 (November 2001)
News &	355	Calendar
Announcements	358	Practical TeX 2004: training and techniques
	c3	TUG 2004 announcement
Late-Breaking News	360	Production notes / <i>Mimi Burbank</i>
	360	Future issues
Cartoon	244	Type Design as Art / <i>Roy Preston</i>
TUG Business	360	Report of TUG election / <i>Arthur Ogawa</i>
	365	Financial statements for 2001 and 2002 / <i>Robin Laakso</i>
	357	Institutional members
	375	TUG membership application
Supplement	367	Bugs in <i>Computers & Typesetting</i> , 6 July 2003
Advertisements	376	TeX consulting and production services
	359	TeX Live, 2003 Edition

T_EX Users Group

Memberships and Subscriptions

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

2003 dues for individual members are as follows:

- Ordinary members: \$75.
- Students/Seniors: \$45.

Membership in the T_EX 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. For membership information, visit the TUG web site: <http://www.tug.org>.

TUGboat subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. Subscription rates: \$85 a year, including air mail delivery.

Periodical-class postage paid at Portland, OR, and additional mailing offices. Postmaster: Send address changes to *TUGboat*, T_EX 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 T_EX and the T_EX Users Group. For further information, contact the TUG office (office@tug.org).

T_EX is a trademark of the American Mathematical Society.

TUGboat © Copyright 2002, T_EX Users Group

Copyright to individual articles within this publication remains with their authors, and may not be reproduced, distributed or translated without their permission.

For the editorial and other material not ascribed to a particular author, permission is granted to make and distribute verbatim copies without royalty, in any medium, provided the copyright notice and this permission notice are preserved.

Permission is also granted to make, copy and distribute translations of such editorial material into another language, except that the T_EX Users Group must approve translations of this permission notice itself. Lacking such approval, the original English permission notice must be included.

Printed in U.S.A.

Board of Directors

Donald Knuth, *Grand Wizard of T_EX-arcana*[†]

Mimi Jett, *President*^{*+}

Arthur Ogawa^{*+}, *Vice President*

Don DeLand^{*+}, *Treasurer*

Susan DeMeritt^{*+}, *Secretary*

Barbara Beeton

Karl Berry

Kaja Christiansen

Stephanie Hogue

Judy Johnson⁺

Ross Moore

Cheryl Ponchin

Kristoffer Rose

Michael Sofka

Philip Taylor

Raymond Goucher, *Founding Executive Director*[†]

Hermann Zapf, *Wizard of Fonts*[†]

^{*}member of executive committee

⁺member of business committee

[†]honorary

Addresses

General correspondence,
payments, etc.

T_EX Users Group
P. O. Box 2311
Portland, OR 97208-2311
U.S.A.

Delivery services,
parcels, visitors

T_EX 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

Submissions to *TUGboat*,
letters to the Editor:
TUGboat@tug.org

Technical support for
T_EX users:
support@tug.org

To contact the
Board of Directors:
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 board@tug.org.

[printing date: January 2004]

He who first shortened the labor of Copyists by device of
Movable Types was disbanding hired armies, and cashiering
most Kings and Senates, and creating a whole new
Democratic world: he had invented the Art of printing.

Thomas Carlyle
Sartor Resartus (1833-1834)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP
EDITOR BARBARA BEETON

VOLUME 23, NUMBER 3/4 • 2002
PORTLAND • OREGON • U.S.A.

TUGboat

This issue (Vol. 23, No. 3/4) is the last of the 2002 volume year. For 2003, three issues will be published. Two will contain proceedings—of EuroTEX 2003 and of TUG 2003; there will be one regular issue.

We are unfortunately not able to set a definitive schedule for the appearance of the next few issues.

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

Owing to the lateness of the present issue, and the scarcity of material submitted for future issues, suggestions will be accepted and processed as received.

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-in-Chief, Barbara Beeton, to the Managing Editor, Robin Laakso, or to the Production Manager, Mimi Burbank.

The *TUGboat* “style files”, for use with either plain TeX or L^AT_EX, are available from CTAN. For authors who have no network access (browser or FTP), they will be sent on request; please specify which is preferred. Send e-mail to *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*.

TUGboat Editorial Board

Barbara Beeton, *Editor-in-Chief*

Robin Laakso, *Managing Editor*

Mimi Burbank, *Production Manager*

Victor Eijkhout, *Associate Editor, Macros*

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 243 for addresses.

Other TUG Publications

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 TeX 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* or in care of the TUG office.

TUGboat Advertising

For information about advertising rates or publication schedules, write or call 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.

METAFONT is a trademark of Addison-Wesley Inc.

PostScript is a trademark of Adobe Systems, Inc.

TeX and *AMS-Tex* are trademarks of the American Mathematical Society.

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

General Delivery

From the Board of Directors

Karl Berry (Director)

Greetings,

Welcome to another issue of *TUGboat*! Due to *TUGboat*'s unfortunately-delayed publishing schedule, this general delivery will mostly cover events in 2002, despite being written in December 2003. (We do have hopes for *TUGboat* to catch up to present time, as will be discussed in the Editorial Comments.)

2002 board meeting

The TUG board of directors met in Portland, Oregon on July 20, 2002. (Several directors lived in that area.) Two notable events from that meeting:

- The bylaws were amended to provide for more timely decisions by email, based on our experience over the past years. In particular, instead of requiring a full two weeks of discussion followed by a full two weeks of voting, there is now a one-week period of discussion, followed by at most one week of voting. The result is announced as soon as it is determined, i.e., a majority of directors have voted.

The full bylaws are available on the TUG web site: <http://tug.org/bylaws>.

- We retroactively supported the 2002 EuroTEX conference sponsored by GUST with a donation of \$2,000. TUG has traditionally supported EuroTEX conferences with this amount. We regret overlooking making the donation in time for the conference itself.

A number of other items from the meeting were discussed in the general delivery for the last regular *TUGboat* issue (volume 22, number 4).

Board resignations

In October 2002, Patricia Monohon and Wendy McKay resigned from the board for personal reasons.

We appreciate their service, and were especially grateful that they remained on the TUG 2003 conference committee, and in fact performed the bulk of the work for that conference.

Project Gutenberg

Finally, I would like to mention a worthy text (and TEX) project: Distributed Proofreaders (<http://www.pgdp.net>) is the main source of public domain

electronic books. It is part of Project Gutenberg (<http://www.gutenberg.net>). The project always needs savvy TeX and L^AT_EX folk to correct OCRed texts, especially of math books.

Thus, if you have Internet access, a browser, and a spare ten minutes now and then, you can make a significant contribution to mathematics. The finished e-books are free, downloadable, and computer-searchable. Sign up at the web site, <http://www.pgdp.net>.

The TUG board welcomes input or questions at any time; please email us at board@tug.org.

◊ Karl Berry (Director)
685 Larry Ave. N
Keizer, OR 97303
USA
karl@tug.org

Details of the changes are recorded in the files `errata.tex`, `tex82.bug`, `mf84.bug`, `cm85.bug`, `errorlog.tex` and `errata.ten`.

Most changes will not affect most users, so upgrading is not urgent. The new versions will be included in the 2003 edition of **TeX Live**.

Glitch in *TUGboat* 22:4

Again a glitch...

The article on “Eukleides: A geometry drawing language”, by Christian Obrecht (pages 334–337), was published without the proper graphics, only the graphics calls. (It was processed without using Eukleides, which was required.)

A corrected version appears in this issue.

Farewell, Michael Downes

Michael John Downes, the principal author of the AMS-L^AT_EX packages, and an employee of the American Mathematical Society since 1985, died on March 8, 2003. He had been diagnosed with an aggressive form of brain cancer in December 2002.

Michael was born August 2, 1958, grew up in Lansing, Michigan, and attended Michigan State University, graduating from the University’s honors program with a degree in Russian and a minor in mathematics. After graduation, Michael moved to Rhode Island, where he worked for 17 years as both an editor and a publications technical specialist.

Members of the AMS may have seen Michael at several annual meetings, where he presented sessions for AMS authors in the use of AMS-L^AT_EX; he also wrote an article on this topic for the *AMS Notices* (**TeX** and L^AT_EX 2_E, volume 49, number 11, December 2002, pages 1384–1391). He was a member of the L^AT_EX team, helping to plan its future direction; this involvement led to the acceptance of AMS-L^AT_EX as a “required” component of L^AT_EX.

Michael was active in his church and community, and looked for opportunities to help others. He leaves three daughters, his former wife, his mother, four brothers, three sisters, and many nieces, nephews, aunts, uncles and cousins. He will be greatly missed.

Last cartoon by Roy Preston

This issue contains the last of the type-related cartoons drawn by Roy Preston that we’ve been able to enjoy for the past few years. Roy has “retired” from cartoon-drawing for personal reasons. We will miss his wit and wry observations.

Thanks for brightening our days, Roy.

A web site for Arabic typography

The web site <http://www.arabictypography.com> is a non-commercial site that was developed as an academic project in the Netherlands. Designed and maintained by a young Lebanese designer, it provides information on typography and calligraphy, recommendations for resources, links to related sites, and animated examples. A Flash plugin is required to view the site.

Recognition for the Plantin Museum

The Plantin Museum, in Antwerp, Belgium, has been entered in UNESCO’s Memory of the World Register. This lists documentary heritage which has been deemed of world significance. The citation for the Plantin Museum reads as follows:

The Officina Plantiniana can be regarded as the most important printing and publishing house that Belgium has ever had. It was founded in 1555 by Christoffel Plantin who, in one of the most turbulent periods of Western history, succeeded in making himself the greatest typographer of his day, and was continued until 1876 by his descendants, the Moretuses.

The rise and the heyday of the Officina in the sixteenth and seventeenth centuries coincide with an era in which scholars from the Low Countries—present-day Belgium and Holland—were able to play an extremely important part in the development of Western thought.

The history of the Officina Plantiniana is therefore more than an account of the fortunes of a large capitalist enterprise: it also reflects and is part of the great cultural currents of the West. Since the business archives of the house have, providentially, been preserved almost intact it is possible to illuminate three hundred years of book history in all its aspects and problems with an incredible wealth of detailed and accurate data.

The Register can be accessed at <http://www.unesco.org/webworld/mdm/>; in the panel to the left, click on the entry Memory of the World: Register.

The museum is well worth a visit, both for the ambience of an early printing house and type foundry and for the exhibits from the extensive library, including not only works printed there, but also many other important early books.

The museum’s web site is found at http://museum.antwerpen.be/plantin_moretus/.

An alphabet game for children of all ages

Another web offering is this charming little puzzle made in Flash: <http://www.orgdot.com/abc/>.

It invites one to construct letters of the alphabet from line segments that rotate on a grid. In addition to the usual 26 letters of English, there's a bonus (it was created in Norway): Ø, Æ, and Å.

Making T_EX Work on CTAN

A free html version of Norm Walsh's book, originally published by O'Reilly, is given in the file `info/makingtexwork/mtw-1.0.1-html.tar.gz`.

The root of the html tree is `makingtexwork/html/index.html` in the expanded archive.

Many thanks to Norm for submitting this package to CTAN.

T_EX for the Impatient

This book was originally published in 1990 by Addison-Wesley, written by Paul Abrahams, Kathryn Hargreaves, and Karl Berry. It contains tutorial and reference information on primitive and plain T_EX, as well as introducing Eplain (<http://tug.org/eplain/>). It does not discuss L_AT_EX.

It has been officially declared out of print and Addison-Wesley has reverted all rights to the authors, who decided to make the book available in source form, under the GNU Free Documentation License, as their way of supporting the community which supported the book in the first place.

The book will be included in future editions of T_EX Live. It is also part of the GNU Project.

Primary distribution point:

`ftp://tug.org/tex/impatient`

CTAN location:

`http://ctan.org/tex-archive/info/impatient`

Source repository: <http://savannah.gnu.org/projects/teximpatient>

Email contact: `impatient@tug.org`.

Information about the free edition is included in the preface, the copyright page, and the `README`.

The authors would be interested in hearing about any projects undertaken with this material. They do not plan to make any further changes or additions to the book, except for correction of any outright errors reported to them, and perhaps inclusion of the illustrations.

- ◊ Barbara Beeton
American Mathematical Society
201 Charles Street
Providence, RI 02904 USA
`bnb@ams.org`

needed in your document, should be inserted with discretionary hyphens.

The reference used to check these hyphenations is *Webster's Third New International Dictionary*, Unabridged.

English hyphenation

It has been pointed out to me that the hyphenation rules of British English are based on the etymology of the words being hyphenated as opposed to the “syllabic” principles used in the U.S. Furthermore, in the U.K., it is considered bad style to hyphenate a word after only two letters. In order to make *T_EX* defer hyphenation until after three initial letters, set `\lefthyphenmin=3`.

Of course, British hyphenation patterns should be used as well. A set of patterns for UK English has been created by Dominik Wujastyk and Graham Toal, using Frank Liang's PATGEN and based on a file of 114925 British-hyphenated words generously made available to Dominik by Oxford University Press. (This list of words and the hyphenation break points in the words are copyright to the OUP and may not be redistributed.) The file of hyphenation patterns may be freely distributed; it is posted on CTAN in the file `tex-archive/language/hyphenation/ukhyph.tex` and can be retrieved by anonymous FTP or using a Web browser.

Hyphenation for languages other than English

Patterns now exist for many languages other than English, including languages using accented alphabets. CTAN holds an extensive collection of patterns in `tex-archive/language/hyphenation` and its subdirectories.

Converting this list into a list of hyphenation exceptions

Werner Lemberg has created a script that will convert this article into a real `\hyphenation` block that can be incorporated into a document either directly or by inputting a file. His work has necessitated some changes to the macros used to format the list, but the appearance of the list will not change. Many inflected forms will be included

automatically, some evident in the printed version, but many included silently.

The script, `hyphenex.sh`, runs under Unix and is posted on CTAN, in `tex-archive/info/digests/tugboat/hyphenex.sh`

The List — English words

an-ti-holo-mor-phic	anti-holo-mor-phic
au-toround-ing	auto-round-ing
biomed-i-cal	bio-med-i-cal
book-seller	book-sell-er
caller	call-er
coas-so-cia-tive	co-asso-cia-tive
dam-selfly(ies)	dam-sel-fly(ies)
falling	fall-ing
geostrophic	ge-o-strophic
halfway	half-way
hotbed	hot-bed
id-iosyn-cratic(ally)	idio-syn-cratic(-i-cally)
in-stallers	in-stall-er
lovestruck	love-struck
metabo-lite	me-tab-o-lite
nephews	neph-ews
nephrite(ic)	neph-rite(ic)
pheny-lala-nine	phenyl-ala-nine
pho-tooff-set	pho-to-off-set
pro-peller	pro-pel-ler
pro-pelling	pro-pel-ling
quasiequiv-a-lent	qua-si-equiv-a-lent
rect-an-gu-lar	rec-tan-gu-lar
seller	sell-er
speller	spell-er
spelling	spell-ing
teller	tell-er
wingspan	wing-span

Names and non-English words used in English text

Florid-ian	Flor-i-d-ian
Ghostsript	Ghost-script
GhostView	Ghost-View
Grass-man-nian	Grass-mann-ian
Lo-jban	Loj-ban
Maxwell	Max-well
Mi-crosoft	Mi-cro-soft
Netscape	Net-scape
Palatino	Pala-tino
Pfaf-fian	Pfaff-ian
philosophis-che	phil-o-soph-i-sche
Rad-hakr-ish-nan	Ra-dha-krish-nan

◊ Barbara Beeton
American Mathematical Society
`bpb@ams.org`

Donald Knuth: All questions answered
University of Oslo, 30 August 2002

Tom Lyche: It is a great pleasure for me to introduce Professor Donald Knuth. Professor Knuth is a Professor Emeritus in The Art of Computer Programming at Stanford University. He is one of the leading researchers in computer science, and has made many fundamental contributions in many areas, including combinatorial algorithms and discrete mathematics. His monumental books, *The Art of Computer Programming*, have been seminal in computer science, and his typographical system *TEX* is used heavily, especially in journals requiring mathematical typography.

He has many honors. He is a member of three national academies in the United States, and he is a foreign associate of the French, Norwegian and Bavarian science academies. He has received the Turing Award, the National Medal of Science, the John von Neumann Medal, the Steele Prize, the Adelsköld Medal, and the Kyoto Prize from Japan, moreover the Harvey Prize, . . . , I hope I get them all. He holds honorary doctorates from Oxford University, the University of Paris, the Royal Institute of Technology in Stockholm, the University of St. Petersburg, the University of Marne-la-Valée, Masaryk University, St. Andrews University, Athens University of Economics and Business, the University of Tübingen, and from Monday, also from the University of Oslo. [applause]

Professor Knuth has a long-lasting and close relationship to Norway. In '67 he came to an IFIP conference in Oslo where, among other things, SIMULA67 was presented. He spent the academic year '72–73 at the University of Oslo, and this visit was influential for further development of computer science in Norway.

So, a little bit now about this session. The format of this session is informal. Anybody who has questions will ask them directly to Professor Knuth; also, those who had sent me questions before can pose them directly to Professor Knuth, and he will repeat the question for the recording. And I'll start by asking a question myself. [laughter]

This Q&A session was held in conjunction with the celebration of the bicentennial of the birth of Niels Henrik Abel, as well as the 25th anniversary of Oslo University's computer science department.

Thanks to Dag Langmyhr for providing a copy of the recording and especially for correcting the Norwegian references.

The video can be accessed at <http://www.ifiuio.no/aktuelt/arkiv/2002/09/allquestions.html>.

When you came here in 1972, you gave a proof that Norway should not become a member of the European Union. [laughter]

DEK: yeah, yeah, yeah . . .

Lyche: Is this proof still valid?

DEK: [laughs] Okay, very good question, Tom. Yes, I came, and I gave a tongue-in-cheek lecture—it was my first lecture at the University—and I didn't realize that it was something that you shouldn't joke about, because as I rode the trikk¹ back to my apartment, I noticed that I was moving from "Stem nei" to "Stem ja"² country.

The idea of the lecture was really mathematical. It's interesting to economists, the study of a three-way duel, where there are three players. And in my presentation, if I remember correctly from 1972, there were three players; they were named Petersen, Quisling, and Rasmussen, and they had probability p , q , and r that when they fire a gun, they would hit the person they shoot at. So I worked out the theory of what's the probability of survival as a function of p , q , and r . And the answer was that the one who had the smallest probability had the best chance of living, that the big powers were shooting each other, and then the other one would be left. Anyway, that was my "proof", and it had nothing really to do with the Common Market. And I'm not sure I'm glad you brought it up or not. [laughter]

It does, in fact, seem to happen only with three players; the phenomenon doesn't occur with two or with four, and economists are still unsure about this, but maybe that's the reason . . . In English we have a word "truce", and it seems to start with "tr", which is the prefix for "three". There's this uneasy truce, where actually all players do best by firing in the air, every time it's their turn to fire. But the result doesn't seem to be true for other numbers.³

We're trying here to get the connection to the Internet going, so that I can show you my home page. Before I start with more questions, I just want to say how much "jeg elsker dette landet."⁴ [laughter] My ancestors came from Schleswig-Holstein, which is a little bit close to Scandinavia, but my academic ancestors are almost entirely Norwegian. The first time I came to Norway, in 1967, I fell in love with the country and decided I wanted to be here a lot. My academic ancestors . . . my thesis

¹ electric tram

² "Vote no", "Vote yes"; this refers to the 1972 referendum on whether to join the European Union.

³ *Journal of Recreational Mathematics* 6 (1973), 1–7.

⁴ "I love this country"—a play on the Norwegian national anthem, which begins, "Yes, we love this country."

adviser was an American, but his adviser was Øystein Ore, at Yale University, and Ore was a student of [Thoralf] Skolem, Skolem was a student of [Axel] Thue, Thue was a student of Elling Holst, who was a student of Sophus Lie, and Sophus Lie was a student of [Carl] Bjerknes, and next week I'm going to find out who Bjerknes was a student of,⁵ and so on. [laughter] But anyway, in that sense I'm a son of Norway.

And pretty soon we're going to be able to see my home page ...

The other thing I should say before we start out is that I want to pay respect to Ole-Johan Dahl. We had, of course, a special memorial session for him this morning, and both Dahl and [Kristen] Nygaard are extremely important in computer science. I had a very close relationship with Ole-Johan; he's one of the five people who had the strongest influence on my whole life. We were very close personally, and I invited him to Stanford for a year after he had invited me to Norway for a year. I looked at my diary before coming here, and found out that on my previous trip to Norway, which was 1994, I spent five nights at his house playing piano with him.

[laughter; DEK looks at screen at the front of the room, which shows a regular television channel.]

So we have a real hacker here getting us to the outside world. Well this is great — maybe I can get a copy of this video.

When this building was dedicated, Ole-Johan and I played the piano in the lounge, and we ... [looks at screen again] X-watch — all I want is Netscape!

Eskil Brun (AV technician): I'll log in as me, in case you don't remember your password. You only got your account yesterday.

DEK: This is true. You don't have to log in. Just get me Netscape, and I'll destroy your files.

Brun: It's only 4.75.

DEK: The reason I wanted to know is if I can use Ghostview if necessary.

Let's get to my home page. Let's just see what you've got here under bookmarks. You don't have any bookmarks. [laughter] [up comes Google on the screen, then DEK's home page⁶]

Today my subject is called "All questions answered", and it's based on an idea that was started at Caltech when I was on the faculty there. It was started by Professor Feynman. At the end of all his

physics classes, the last day of class was optional; students didn't have to come if they didn't want to, but if they came, he would answer any questions they had, on any subject except religion or politics. I liked that idea too, and I kept it up when I went to Stanford. If any of you ever were there in one of my classes, you'll know that the last day of my class was always called "All questions answered" on anything except religion or politics. Or the final exam — that was the other excluded thing.

Now today I already talked about politics, and it was a disaster. [laughter] I'll talk about religion only if there's a strong feeling for it, but basically, I want to answer any question that anyone in this room wants to ask — except the "Frequently asked questions", because you can always look up the answers on my home page. So ask me an "Unfrequently asked question" preferably. [looks at monitor] What else have we got here ... Recent news ... I was a million years old at the beginning of the year [Web page for 1 000 000₂ year Knuthfest⁷ pops onto screen], if you use binary notation. And there are other things that are on this page — you can explore these things to your heart's content.

This page⁸ shows all the books that I've got out there, with errors in them, and anybody who finds an error the first time, I really want to know about it. Then I write you a check if you're the first one, and if I believe that it's an error. And unfortunately, it usually is. [laughter] But the way I write books, I try to maximize my chance for error. I mean, a book is more useful when there are more ways that it could have been wrong. And so, instead of saying that something is better, I say, "oh, it's 12.8% better." And maybe it really is 12.7% better, so that's an error. So there's many chances all the way through to be wrong, and I try to get it right the first time, but the fact is, when I was a college student, I did not get 100% on all of my exams. Sometimes I would get 99, you know. Now, with several hundred chances to make an error on every page, you can see how many errors there probably are in my books. I'm trying to get them fixed. And also the errors in software — there's a rumor that somebody found the first error in TEX since 1994, and if so, I have to pay \$327.68, which is an amount that kept doubling until it reached 2¹⁵, and then I froze it at that point. [laughter] So anyway, now I'm ready for real questions. If you ask in Norwegian, somebody here will translate, so don't be afraid.

⁵ The list is given on DEK's web page. Bjerknes was a student of Bernt Holmboe, who was a student of Søren Ras-mussen, who was self-taught.

⁶ <http://www-cs-faculty.stanford.edu/~knuth/>

⁷ <http://forum.stanford.edu/events/knuthfest02.html>

⁸ <http://www-cs-faculty.stanford.edu/~knuth/books.html>

[Q]: [in Norwegian; translated]

DEK: Do I think Sweden is a better country than Norway?

No, but it's interesting.

They have some good ... What was the name of the ... I was a fan of one of the skiers. Goodness, I'm forgetting ... [some prompting] In the winter Olympics that were held in Lillehammer, there was ... I sort of fell in love with this one Swedish skier.⁹ [more prompting and laughter] I can't even remember any more, so I ... but I never got to meet her. I did meet the king once; that was nice, and he said that his daughter likes computers. To me, when I was here, living in Norway, we went to all parts of the country, and it's certainly the greatest year in my life.

[Q]: [paraphrased by DEK] The question is about generic libraries, what is the role of little toy algorithms, instead of staying on the high level?

DEK: I think, in all fields, not only computer science, people can make the mistake of saying you should always stay on the high level. Everyone who is best at their field seems to forget the way they learned it, and they'll find out that some parts of the things they learned are more useful to them in later life. So then they'll say, I won't bother my students; they won't have to learn all the stuff I had to do when I started. And as a result, I think their students are missing a lot. And I think that even more so in computer science, it's very important to keep track of many levels at once. What was the word that your professor had for this? Goodness, now I can't even remember the name of the man who spoke about it,¹⁰ but he had coined a new word, which sort of means "look at things at all levels". The professor on information design. [prompting from audience] "Polyscopic". It's more than a telescope—it's a polyscope. So you see, the people who are the best computer scientists have a certain kind of talent that is not very strong in the general population. I think one of the properties of this talent is the ability to shift levels, to see something in the small at the same time you're seeing it in the large—to know that in order to solve a big problem you want to add 1 to a little counter. And in order to add 1 to a counter, it's actually better to have it in a certain cache or something like this—to understand what's going on at many levels at once, and effortlessly to convert, to chain. So the high levels are great, but in principle, the more that part of our brain is also able, if necessary, to open the

box and look under the covers and see what's there, the better we are. And the people who have that skill discover that it also correlates well with being able to make computers do tricks, and they can really resonate with computers. I mean, like Eskil here had to go back into a config file and change all kinds of settings on monitor and things like this, so he had to dive down into rather low levels of the system in order to get these pictures on the screen. That's not an isolated event. Every day, I'm still going to things that are at low level, even though I'm using high-level stuff all the time too. So I believe when you build a building, you start from the foundation and you build up. You don't start at the roof and build down.

I spent a lot of time five years ago designing a computer to replace the MIX computer in my books. My book, *The Art of Computer Programming*, when I started writing it—that was 1962, 40 years ago—I knew that I had to have a low-level machine in there, and I took all the machines that I could find in the world in 1962 and I found something that would be nicer than all of them but similar to all of them. I put that into the book, and now it's extremely obvious that's it's quite different from the machines of today. And machines, in fact, ... computers went through a period where they got to be horrible to program at the low level, because people stopped doing much assembly programming. When almost all the programs were being written by compilers, then people changed the design of the machines so that only compilers could enjoy writing programs. But when RISC computers came out, and I read the book of Hennessy and Patterson about ten years ago, I was so happy, because all of a sudden you could have a real ... today's computers were actually beautiful. You could look at the programs and you could enjoy seeing what the bits were actually doing in the low level. And so there was a window in time when machine language was nice again, although it will probably get bad before long. I mean, the Itanium seems to be a disaster from this point of view. But I was glad that I could find a computer for the present time and explain its low-level operation, and give an entire machine design that I could use instead of MIX, and that's this MMIX machine.

If you look here [displays MMIX web page¹¹ on screen] this book is just the documentation of all the software that goes with it and makes it appear as if it was a real computer. It's just a textbook computer, but I had a lot of people helping me on the

⁹ Pernilla Wiberg

¹⁰ Dino Karabeg

¹¹ <http://www-cs-faculty.stanford.edu/~knuth/mmixware.html>

design of it—people who designed the MIPS chip and the Alpha chip both worked with me on MMIX. I believe that the future will prove that students who learn something about what's going on at this low level are going to turn out to be the ones who are going to be important in their careers, and the student who never learns about those levels and only learns how to apply somebody else's generic methods is not going to be of fundamental importance. The number of people who have the skill that I mentioned, of going through all these levels, is a small part of the population. The percentage seems to have been constant, as far as I know, at about 2%—one person out of 50 who's born seems to have this combination of abilities that makes them really resonate with computing. That's not enough people to solve all the problems that computers are able to solve. So those who have it should be sure to buy my books [laughter] and to use it. But the rest of the world, they should make friends with geeks like us, and then the other people can use these things. But just staying on the top level is as bad as a mathematician who only knows the statement of theorems and not the proof of theorems.

I gave too long an answer to that question, but it's something I feel rather strongly about, that you don't want to cut out either the low level or the high level.

[Q]: Is it true that you believe that it's not desirable to have bugs in programs? And what do you think about a new law in the United States that companies should be able to say that they don't take responsibility for any [problems]?

DEK: I didn't know about this law, but there have been a lot of worse laws passed, probably. As I said, politics—I have no talent for it whatsoever. I like to try to think, though, what is the best for the world eventually, and I came to the conclusion that it's almost impossible to get a program that doesn't have bugs in it. The reason is, I've never seen a program that exceeds a certain size that I could really say was bug-free. So we have to learn to live with programs that aren't 100% debugged, no matter how much I love to have programs that are bug-free. I would love to have TeX be one of the first examples of a program of more than a hundred lines—it's 15,000 lines of code or something is all—but I would love to say that this is one program that is absolutely solid. It hasn't been *proved* correct, but that doesn't mean anything. Well, it means something, but it doesn't mean that you're at the end, because there might be a mistake in your proof.

When I looked at the first published papers on proving correctness of programs, Tony Hoare's paper¹² on proving that the *find* program was correct, there were two bugs in it. He had proved it correct, but there were two bugs in his original proof. So I'm a strong believer that formal methods are helpful, but I don't ever want to say that now I've done this and I've got it right. You can prove the program meets its specifications, but how do you know the specifications are correct? It's almost as hard to write specifications as to write a program. So when you get to saying that something has no bugs in it, it seems to be impossible to get to that level. We could just say we want to get as close to that as humanly possible, or something. A friend of mine works for the government—he's the head of all the software that goes into air traffic control; there's a team, but he's the leader of it. He's based in California—he's being mandated by Congress that these programs have to be completely bug-free. And the Senators don't want to believe that this is impossible, so they'll only listen to the people who tell them it's possible, and the people who tell them it's possible just are saying this because they know that it'll pay their salary.

I wrote a paper called "The Errors of TeX"¹³, which was the entire history of the debugging of this small-scale program that I wrote for typesetting. The beginning of the paper said, "I make mistakes. I always have made mistakes and I probably always will." I have to learn how to live with a life where I'll never be sure that I've got it right, but still get better and better.

Another friend of mine works for what we call BART around San Francisco; it's the subway system. They're building a new extension that goes to the San Francisco airport. And his boss is requiring him to have no bugs in this software. I don't know ... you've seen the movies, you hear the message that says "This airplane is being flown completely automatically, but don't worry, it can't go wrong, can't go wrong, can't go wrong..." The thing is, Norway has a history of innovative ways to improve the ... The man (wasn't he in Bergen?) 20 years ago who introduced "bebugging",¹⁴ where he would say I have a program, and you want to see how robust it is, so you introduce random errors in it and see actually how long it takes before anybody notices. As a way to get some feeling as to how well you've checked out a program—I can't remember the name of the product—but anyway, that was quite influential.

¹² Communications of the ACM **14** (1971), 39–45.

¹³ Software—Practice & Experience **19** (1989), 607–685.

¹⁴ Tom Gilb, Software Metrics (1967).

So anyway, I think if lawyers have to get into it, they should have some way so that an insurance company can't say it doesn't have to take a risk for being sued that somebody's saying, "Well, my child died because of a bug in this program, and you're at fault." There's a certain level of care that is reasonable to expect, but another level that I would say is unreasonable. So it's not necessarily impossible to have some reasonable law around the problem that you describe. But I would say the chances that we've found the right balance at the beginning are very low. I don't know the details of the law.

Okay, good questions.

[Q]: Relating to a problem in the Norwegian government where there are a lot of legacy systems, where they want to have them cooperate and work together, is it better to start over from scratch instead of trying to run the programs that were patched together over the years?

DEK: In my personal experience, every time I started over from scratch I was happier afterward that I did it. In fact, TeX I scrapped entirely. After five years I took everything I learned and said, okay, let me start over again, and I'm *not* going to try to be compatible with the other. If this system is going to be important, it won't be more than a year or two before there will be more users of the new system than ever care about the old system, never heard about it. So no matter how much time had been invested in the other, it would be a small percentage compared to how much better the new one would be. I know many, many anecdotes over the years where this is true, and very few of the opposite, where preserving the past turned out to be successful. And part of is because of this problem of bugs. Every time you look at the old programs, you see that they don't really do what you thought they were doing.

Ken Thompson¹⁵ told me, at Bell Labs he went through one of Bell Labs' most important applications that had been built up over a period of years—I don't remember if it was something about how they charge for telephone lines, or what it was—but he took this program and took a look at it and within one month he had identified several serious bugs and he had also been able to make it much smaller, more reliable, give more flexibility to it, and so on.

When I was a college student, one of my first jobs was to ... There was a company in Cleveland, Ohio, that made what they called bearings; it's part of the motor of a car. This company had what they called a "load study" program that would

study ... the engineers would put in the design of one of the bearings and they would simulate on the machine whether it would be strong enough to take the pressure of the thing. This was one of the leading manufacturing companies in the United States, and they had been using this program for some years. I was hired by Burroughs Corporation, who wanted to sell a Burroughs machine, because the engineers had an IBM computer before. So I took the program that was running on their IBM computer, and all I had to do was convert it to the Burroughs language. I thought, okay, it paid \$200 or \$300, I can make some money, I'm a student; but I didn't realize that the Burroughs computer didn't have floating point subroutines, so I had to take a month to write programs to calculate arctangent and everything else that Burroughs didn't have in its library. I finally got to the point that I could take the program from the IBM and put it into the Burroughs machine; and I *didn't* get the same answer as they did.

I figured out how their program worked, and I made it run ... supposedly it was running faster; instead of being able to have four parameters, I was able to give it ten parameters; and so on. But I didn't get the same answer. I was going nuts about it, so I finally found a place that had an IBM machine, and it traced the program line by line, through thousands of instructions, and printed it out on an old line printer—nowadays you can't imagine what people had to cope with back then—and I compared the intermediate results with my program on the Burroughs machine, until I finally got to a point in the middle of the thing where the engineers' IBM program was overlaying some of their floating point data with a machine language instruction. I hadn't realized this conflict of the code. So what happened was that they were ... if I had to get the same answer as their program, I would have to clobber a Burroughs floating point number with some IBM machine language instruction. [laughter] So I had to come to the engineers and say, "Well, you know, do you realize that your answers have been more than 10% off all the time you've been using this program? And wouldn't you prefer to have the correct answer?" [laughter] I should have charged another \$50 probably for this [laughter] but this was just my first experience of many where, every time I took an existing program, I found that it was less work and a better result—you know, total work—to redesign it and to not ... Now, you still have to deal with the problem of old data files, and being able to deal with different formats, but that's as far as I would go, and I would get out of the old data format as soon as possible.

¹⁵ In the talk, DEK mistakenly said Ken Knowlton.

[Q]: How does “second system syndrome” impact on the answer to this question?

DEK: You must define for me “second system syndrome”, which sounds like a great thing I should have known. [laughter]

[Q]: It’s something that Frederick Brooks wrote in the *Mythical Man Month*.¹⁶ The “second system syndrome” is the tendency of the person who has written a system once and is trying to rewrite an equivalent system a second time, to try to do all the things that he didn’t do in the first one, so Brooks’s theory is that the second system of one particular type that you write is almost always a disaster. The first one works, but is limited, and the second one is a disaster, and the third one, you learn from both the first and the second.

DEK: Okay. I knew that *Mythical Man Month* was sort of a “three-M”, but I didn’t realize that he also had his “three s’s in there. Of course, I read that book so I should have remembered. I have a great admiration for Fred Brooks, and he has lots of experience as a manager, which is orthogonal to mine, because my experience has been with small groups of people. I mean, to me, having a group as large as two—Dahl and Nygaard—is impossible; it’s the only example I know in the whole world where you have two people with a strong personality complementing each other and working together to make a great product.

But it is certainly true that if you . . . When I went, for example, to make the second version of TeX, I did not want to go much further than the first. It’s very easy to be tempted, after you’ve understood one problem, to say okay now, that’ll scale up, and in my new project I will be able to, now that I understand the small thing, now I know everything and so I can try a much bigger project, and my superior knowledge is going to make this bigger project really fly. So that’s an important danger to watch, to extrapolate on your own knowledge. Like my MMIX computer, which I have here now, is my second system, actually. I started out with the design of MMIX ten years ago and it was a 32-bit computer; I realized it should be a 64-bit computer. But then I tried to hold back on things that I was putting into it just because they seemed to be cute. It’s a strong temptation to do that, so I had to have a large focus group sort of keeping the control on.

Before I stop on this screen, by the way, I want to mention MMIX to people who are doing curriculum

here. I would hope that somebody here would take a look at this, because it’s become quite a . . . There are several universities in Germany that are using it thoroughly in their teaching now, and a new book is coming out in German next month—introduction to computer programming, where they learn MMIX before they learn Java. They learn how to do simple programming, they get a feeling for how caches behave, and so on. Because in this computer you can design computers that we don’t know yet how to build. You put in not only your program, but you put in a specification, a configuration of the machine. You say how many types of cache you have with different caching strategies, how many functional units you have, how much parallelism you’re going to have, how many instructions are you going to execute simultaneously, and find out if that speeds up your programs very much. It’s a meta-computer; it’s a computer that has many parameters in it, and, well, John Hennessy said it was the cleanest design he ever saw of a machine language. I tried to make it so that it wasn’t hard to learn and to keep in your mind. And the programs are kind of fun to write. So it’s also a machine that I still think is five or ten years ahead of the state of the art, as far as actually building the chip for it. But the main idea was to make it of maximum use in the educational environment and for experimental purposes, so that people could play around with it. Now we’ve got a C compiler—it’s in the gcc standard distribution now—to get code for it, as I understand, and a lot of good tools have been made for simulating it, and we had some experimental tools for visualizing the pipeline. It’s something I recommend people here taking a look at, because I think it has use especially in pedagogy and learning things that are going to stay with people who make a career of computing.

[Q]: You commented that you haven’t worked in many very large groups. But a lot of modern software development, especially in the open source world, is done by *very*, very large groups if a lot of people are contributing source, and basically [...] each other. How do you view the success of these projects, especially from the scale of the projects themselves.

DEK: So the question is about large projects. I mentioned that in my own case, I was the only coder of the TeX system. I was more restrictive in that sense. I mean, Linus Torvalds is still supposed to approve every line of code for the kernel, right? And I was even more of a filter than that. But people *would* suggest code. I’m not even sure how many

¹⁶ Frederick P. Brooks, *The Mythical Man Month: Essays on Software Engineering* (Addison-Wesley, 1975). Second edition, 1995.

lines of code are in \TeX and METAFONT compared to the Linux kernel, but he doesn't have to read through all the drivers, I'm sure.

But the question is, how does this work with the open source movement. Now, I've had nothing but great personal success in the dealings I've had with open source, but other cases where you have a large number of programmers working have tended to be less successful, in my experience. So somehow, open source is doing it better than all these other things that I had met in my life before. And certainly, when Fred Brooks wrote his book, he was reflecting on IBM's experience of trying just to throw more people at the job of writing software without realizing that this might make the project get worse instead of better. So there's something ... Part of it is the pattern that has developed for communicating, but I think still, in the open source world, you still have somebody who's the czar, you know. I met the guy who is `gdb`, and I met the man who's in charge of the C library. And, you know, I've met various people over the years who are really the gurus of individual parts of GNU Linux. So, my experience with open source has been, as I say, it was very good. I would find a problem in one of the programs; one of the examples was—oh, gosh, what's it called—`rtf`, the alternative to `xterm`, and it wasn't behaving the way it said in the man pages, and so on, and so I sent out a message. Within three hours, I had three answers from people who said one of my problems, oh yes, that's already been discovered, and it's fixed in our beta version if you want to try it. Pretty much all my experiences except for the GNU Pascal compiler have been a success. The GNU Pascal compiler has not yet been ... I decided to give up on that for the moment.

It's depressing if we have to come to the conclusion that no large-scale software efforts could be successful. But I found that developing a first-generation system, the fewer people, the better. A system like SIMULA couldn't have been developed—I just refuse to believe that it could have been developed by more than two people—the very first. After it's understood, then it's ready for a larger thing. But the closer something is to the source, and to being a breakthrough idea, the amount of bandwidth that you need to communicate what you're doing will easily swamp things. I can explain, for example, with \TeX .

My very first draft of \TeX I wrote in a long, ... I stayed up late, all night, and I wrote this thing, and I thought I had specified it completely. I gave it to two students, and they were going to implement it while I went to China for a month. So I came back

from China, and they hadn't finished implementing \TeX . I couldn't believe it—what's wrong? They had gotten a subset going, and they could typeset one line on one page, or something like that. But then my sabbatical year began, and immediately after starting to write the implementation myself I realized what the problem was. Because this was breaking new ground, this was a totally different system than had been before. And I found out as I started to write it, when I had written the specifications, I thought it was clear, but there's a million things you don't realize when you write in natural language; you don't understand that you're only answering a small part of the thing, and when you start writing code, then you realize all kind of things come up. So I started to write the system myself—I'm on my sabbatical—and instantly I discover, oh yeah, that question isn't answered. I have to make a decision. Now if my students ... So here's why a lot of projects have failed this way—the students, suppose I had farmed that out to them and I'm not on sabbatical. But I give it to the students, and I'm not in China, so they come to this point where they say, "Oh, what did he mean? What should I do there? Okay, I'll schedule an appointment with Professor Knuth, and I'll see him next Tuesday." So they'll come to my office, and then they'll start to explain the problem to me. And in fifteen minutes they'll tell me, and I'll say "Oh yeah, that's right. I meant to say this." So then I can send them back to work, and they can start again. And then, another five minutes later, they would come up with another question and would start to make another appointment with me. The thing is, these questions are coming up all the time, so if I'm the one who gets to ask the questions and answer them, I'm saving a factor of 10, 20, ... So a first-generation system really needs to have a small number of things, and the people, the lines of authority that they have to make decisions, and then publish them afterwards, has to be chopped up so that there's very little communication needed.

[Q]: In the U.S., you have laws like the DMCA Copyright Act and so on. These laws encourage not full disclosure [*sic*] on bugs and problems, and so on, and you have several researchers in several countries not especially wanting to say what kinds of bugs they have come into their programs. Tell me about these problems.

DEK: Everything I know about the Digital Millennium Copyright Act, it's apparently a total disaster from start to finish. I just hope that, you know, that we recover from it, and come to our senses.

One ray of hope was that they did scrap the initial idea for the Clipper chip and such things for encryption, after they thought about some of the more subtle issues. So what happened then? This Russian man found a very weak encryption in Adobe's book-reading program, or something like that, and then he got thrown in prison and certainly lost a lot of time and had to pay all kind of legal fees and so on—it was horrible. And even after Adobe said we made a mistake, we didn't mean to sue.

There are many parts of life where I like to say well, gosh, I'm so glad I'm only a scientist, and I don't have to think about how to interface with all the lawyers of the world. I had only once so far to do anything associated with the law, and it was because of the public key encryption. Apparently, since I was at Stanford University at the time, I could be forced to spend a lot of time being what they called "deposed".

But I have a standing offer for consulting fee with respect to legal issues. In my own life, I consider the thing that I do best is write *The Art of Computer Programming*. I've got at least 20 years of work to do on that, and if I'm lucky, I'll live another 20 years, so why should I do anything else? Well, I like to get an honorary doctorate, okay, so I come to Norway. But otherwise, I'd be home writing *The Art of Computer Programming*, and so if Bill Gates wants me to be a consultant to Microsoft, I say well, okay, but this costs \$10 million a day, payable to Stanford University. But my fee for lawyers is \$20 million a day. So you have to understand, I don't have this great connection with anything that has to do with suing people and things like that.

And I'm not a big fan of patents, either. I wrote an open letter to the U.S. Patent Office at Richard Stallman's request about ten years ago, and it was published with the CWEB manual; and basically I was saying, look, with software, if everything in the world is patented, then progress starts to go to zero, and almost none of the programs that are used every day would have even existed if they had been created in an environment of patenting. And I especially dislike patents on trivial ideas, that we would expect any student to come up with. I can understand patents for tough stuff, for difficult ideas. So when it comes to intellectual property, I see that the system is bad for reimbursing people. In some professions, people have to rely on getting paid because of royalties, because they aren't well compensated. As a scientist, I have a job where I can be paid, you know, to do research, by the government, because science is good for the community. But if I'm a musician, I can't be paid for practicing my music,

or if I'm an author of novels, I'm not supported—a novelist isn't considered good for the community the way a scientist is. A font designer—we have a National Science Foundation; we don't have a National Font Foundation. So somebody who wants to design the things that we use all the time for reading, and do them better, has to rely on royalties somehow. I think that there are some flaws in the way people are compensated.

Then when we get to royalties, then it becomes one of these things where it's feast or famine. When you get a royalty for something, or you get a patent on something, then all of a sudden you're supposed to be rewarded to such an effect that you don't have to do anything more in your life—the money is supposed to come in for fifty years for an idea you had, for some work that you did in one year. So things are kind of mixed up in that area. I don't have the answers; all I know is that lots of stuff isn't fair and to me, it's better to pay people for what they do and the services they provide, instead of how lucky they were in doing something once.

I don't have all answers to all questions, I guess.

[Q]: What do you think about the Microsoft .Net strategy? Would you support it?

DEK: I don't know what the Microsoft .Net strategy is ... [applause]

I have to limit my ... I don't think I'll need to write about it in the *The Art of Computer Programming*. [laughter] As I say, I only have 20 years to go.

[Q]: What is your thought about quality in today's operating systems? Do you think there can be any huge development in the next 20 years, any breakthrough?

DEK: First of all, I have to say that the main reason why I'm happy with Linux is that my computer hasn't crashed. I haven't had to do anything special since last October, when I think it was because the electricity went down in the building, or something like that. It's very robust for the kind of things that I do. So I tell my friend—we were at a picnic a week ago—and the man's wife said, "okay, why aren't computers easier to use, and more reliable," and things like that. I said, "you should get Linux." "Oh! What's that?" And then we talked a little bit more, and she said, "but how am I going to use ... I have to use Intuit software" (which is a proprietary system I don't think works with Linux). I didn't have an answer for her on that.

But now you're talking really about the operating system, which lies underneath. I think ... I've never been involved with operating system research.

I think in MMIX probably the Achilles' heel at the moment is that with the machine design I have right now, it's not easy to make a virtual operating system, where somebody could have their own operating system and pretend to have authority to do privileged instructions. The machine, it's actually very similar to the Alpha design as far as the protection mechanism goes, but it's enough different that it might . . . , anyway. I have a dream though, that if there's gonna . . .

If a brand new breakthrough is going to come through in operating systems, I think it would be, it would probably come from a small group of people who start from scratch, and just say, let's re-examine all the old assumptions. Is it really true that what we should have is, you know, shared libraries of a certain kind, or something like that, when programs are running. What if we . . . you know, all kind of assumptions. The operating systems we use today are based on the operating systems we used ten years ago, and those were based on the previous ones, and they're based on hardware assumptions, as to what is cheap and what is fast, and so on. So all the parameters are changing so rapidly, I think it would be worthwhile to rethink everything. For example, with MMIX I designed an assembler for it that is very primitive, so that anything can be assembled in one pass. And the assembler goes so fast that it competes with the loader. So what if you kept your program in source form instead of in binary? Because then you could have conditional instructions in there saying, you know, well I can allocate registers differently; I might give myself more global registers in this routine, depending on how many other routines have been loaded with it. All kind of things can be done, if your assembler is fast.

And so it might be a good idea to rethink the whole idea of an operating system, saying, well, let's say we don't have loading routines the way loading routines have worked for 40 years. And if you don't do that, then you probably aren't going to come up with major breakthroughs. Also if you do it, you might find out that, well, the old way is really better. But I think that would be an interesting thing for a group of people who have tenure, so that they don't have to risk their career on it, to spend five or six years just seeing what would happen if they could start over, all the way. Klaus Wirth did that fifteen years ago with his group in Zürich and Utah.

Other questions? Nobody's going to ask me about Volume 4?

Let me preempt the one question that I was thinking somebody would certainly ask. Let's see, here we go. [looks at monitor and starts to type]

Oh, we don't have much here . . . Anyway, let me just say something that's new, because I've actually been working this year on *The Art of Computer Programming* and getting pages out of Volume 4 that are ready for beta test. So anybody who's interested in seeing what might appear in Volume 4 someday is welcome to try looking at these pages. In fact, before coming here I also finished 40 more pages that I haven't announced yet, so if you can figure out how to get at them—there *is* a way—but the thing is, I've put here on my news page, I've put a list of exercises that nobody yet has commented on in the ones that I've put on line. I'm extremely grateful, hundreds of you have taken time to read these drafts, to detect and report the errors that you find, . . . I'm getting . . . the Internet is amazing! Within a week of posting these pages I had mail from all over the world; a fourteen-year-old boy in Nuremberg, Germany, wrote to me and told me I had spelled 'Nuremberg' wrong. [laughter] But some of the parts I worked the hardest on, nobody yet has commented on, so either I got it right the first time, or they're saying, well, this is too hard for me. So I'm soliciting here for people to tell me that they've actually looked at it and they think it's okay. If they have time.

In order to finish my book in 20 years, I'm going to have to be able to write a page a day, and I haven't been able to reach that rate. I'm only getting about 60% of a page in a day. So I have to lower the standard of quality, to do less original work on the pages that I'm writing. But these pages I regard as a fundamental section, so I've put more time into it; I've got original material in there that hasn't appeared in any other publication, and I'd like to have somebody vet it, and check it out.

[Q]: Are you surprised that T_EX is still used, that no new system has come along that has surpassed the quality of T_EX?

DEK: Actually, there are systems that surpass the quality of T_EX but they still haven't apparently taken over. Now the one that's most . . . There's the ε-T_EX, which has greatly improved features for things like combining right-to-left typesetting with left-to-right typesetting. So I imagine people who are doing typesetting in Hebrew or Arabic are making use of these improvements. And there's Omega, which is Unicode-based and it's a work in progress. There's pdfT_EX, I think a lot of you know. But those are still 100% compatible with T_EX if you don't use the extra features. I can understand why it's difficult for any other system to displace it, because a lot of the things that have been improved

over \TeX really don't matter that much to other people. So \TeX only goes 99% of the way. Still, that 1% is like noise, as far as ... So why should I switch to another system if I only have to work a little harder 1% of the time?

I'm totally surprised that \TeX has been used as much as it has in so many different parts of the world. I mean, I downloaded a paper two weeks ago from a new journal in mathematics called the *Moscow Mathematical Journal*. I got this paper, and I printed it out, and I have to admit really feeling a thrill that it looked so good. [laughter] You know, here were people halfway around the world and they were using this system that I had done years ago, and what had come out was something that was aesthetically pleasing to me as well as the mathematics—it was what I wanted to read. And there are fifty chapters of \TeX user groups around so many parts of the world. And I'm getting newsletters that are published, you know, in Greek, in Indian languages, it's a thrill, and that's a big surprise to me. But I'm not surprised that it's been harder to improve on \TeX , because I know how much work there was to get it going.

Excuse me, I haven't been paying attention to people in this part of the world ... [goes to another part of the audience to ask for a question]

[Q]: What do you think is the single most important problem that is left unsolved in computer science?

DEK: Well, computer science has so many sub-fields that it's really hard to ... Now, if we want to compare it to what Abel did, however, since that's timely, well, Abel solved, you know, the major open problem of his time: Can you express the solution to every polynomial equation of fifth degree with plus, minus, times, and taking roots? And he showed no, you can't. And it's very hard to prove that something is *impossible*. After he worked on that, Abel worked on Fermat's Last Theorem, he worked on $x^n + y^n \neq z^n$ when $n > 2$, and he proved that if x and y and z do exist, they are so huge you could never write them down. He worked on that.

So the question is, if you wanted to be Abel now, and solve the most important problem, it has to be the question about whether $P = NP$ or not. And that is whether or not all the things that we can do with a polynomial number of guesses can be done in polynomial time without guessing. Now, I know only one person who is really working on that seriously at the moment and has a chance of succeeding. In my opinion, that's Andy Yao, who's a professor at Princeton, and inspired by another fa-

mous Princeton professor who solved Fermat's Last Theorem. And Andy might very well get it in the next five years. There's a man in Russia who sent me e-mail saying that he has a proof, but, as I say, my life is too short to check everything that comes in.

There was a proof in a Chinese journal a couple of years ago that $P = NP$, and at first I couldn't find a mistake in it. I worked on it for three hours with some students until we found a bug, and I wrote to the author; still, he didn't believe that I'd found any bug.

But the thing is, that experience made me realize that the problem is academic; it's not going to really be practical. He had given an algorithm to solve the clique problem: If you're given a graph and certain pairs of vertices are adjacent, others aren't, you want to find the largest set of vertices that are mutually adjacent to each other. And he had an algorithm that supposedly solved the clique problem in n^{12} steps. And it was very hard to test that algorithm out because, even if $n = 100$ that's 100^{12} and that's way bigger than I could ... And in order to send him a counterexample, to show that he hadn't got it right, I had to send him an example that we couldn't really run on a computer. Still, n^{12} , that's a polynomial of terribly low degree.

Now, it might very well be that the following scenario happens. Somebody proves ... Actually, there was a big questionnaire sent out to all members of SIGACT, the Special Interest Group on Theory in the ACM, and saying, what did people think is going to happen, you know, how long before somebody resolves this problem? And so on. And I encourage you to read it, it was in the current issue of *SIGACT News*, or the second-last issue.¹⁷ And the opinions went all over the map. But my opinion—which was shared by one other person independently—was that probably in the next fifty years, somebody will prove that $P = NP$, because there are only finitely many obstructions that would keep it from being unequal. So that means that there exists a polynomial time way to solve everything, but that we don't know what the polynomial is. We might just know that there's some exponent so that everything can be solved in N to that exponent time. However, we may never know what the algorithm is, or the exponent. All we know is that $P = NP$. Now, that would be the worst possible solution to the problem, because it could never,

¹⁷ William I. Gasarch, "The $P = ?NP$ poll", *SIGACT News* **33**, 2 (June 2002), 34–47.

ever have any use at all, it just would mean that the question was the wrong question to ask.

People make mistakes all the time of misunderstanding the connection between finite things and infinite things. And even computer scientists have trouble understanding that things don't always approach infinity the same way. And when we make judgments about something that's true in the limit, it might have absolutely no connection with our lifetimes.

In the back ...

[Q]: As advice to the young people here, if you were given a second chance and you were starting out fresh in computer science as a graduate student, with all your present experience, what would you do, which things would you go for, and how would you organize your life to achieve that? [laughter]

DEK: That's a great question.

Suppose I'm starting all over completely, how would I reorganize my life? One of the things Abel said, and I guess he got inspired by Lagrange, he said, "Read the masters." He said, in order to learn stuff, don't try to look at everything by yourself, but do look at what other people have done, written in their own words, through the years. But follow your own inclination for sure.

Now, Abel made a bit of a mistake: He didn't take care of having a job, and there is a problem. I was very lucky that I was riding on the crest of waves, so I could get scholarships to go ... I mean, I came from a family where nobody had gotten an advanced degree in college, but I passed exams okay in high school so I was able to get a scholarship to go to a university. Then at the university I could get into computers—computers were just new. So, exciting thing, I found out that I was in this 2% of people who had the ability to think about computers, so I could get jobs, I could get \$200 for converting a computer program. Then in the middle of graduate school I was offered to drop out of college and write compilers for a living at an annual salary of \$100,000 in 1962 dollars. That would correspond to about \$10,000,000 a year now or something like that, not quite as much as a CEO—but anyway I was offered, leave graduate school and get rich, and I decided, you know, I really wanted to spend my life not maximizing the total amount of money that I got. I wanted to do things that were interesting to me, that I thought would be useful to others. And I found that money was a threshold function. If you don't have enough of it, you need it; but after that point, then it's just a problem because you really have to enjoy having money and figuring out what

to do with it responsibly, which is something that I never wanted to do.

So my basic idea is to say, whoever you are, you've got a unique combination of talents that's been given to you. Don't decide ... Your life is kind of like a binary search, you try things and find out you did well in this, you try other things, you find out you didn't do so well in that, you go on and continue discovering what are the best ways to use the abilities that you were born with instead of what you think they ought to have been. And you also read what other people have done, and try to learn from that and extend abilities that you have in ways that appeal to you. And then, ... My experience was, once things start to click, then you can start producing and other people will be able to make use of what you did, and you can help them along in the same way. It sounds idealistic, but that's really been the pattern all the way through my life, and it was based on this sort of idea, saying, well, just learn what you're good at, and what you can do that other people might be able to use, that's fun for you.

[Q]: Why can't I type floating point numbers in my MMIX assembly code?

DEK: The reason was, it was too hard for me to ..., although I do have the subroutine in there that reads them, I wanted to keep the assembler simple. I didn't see any reason to ... Basically, I was over my page budget; I didn't have that much room to explain what the rules were, so I decided not ... I really believe that an assembler should be stripped down, and that it should be considered as a ... Certainly the second system syndrome could affect assemblers, and I didn't want to get caught with that.

[Q]: Do you see the concept of design patterns as important for programming in the future?

DEK: Design patterns ... I think of all these things as ... In my own experience I don't follow them religiously, but I follow them as motivating ideas for things that I do without using the official tools. I react negatively to somebody who presents me with a tool that's supposed to solve all of my problems, because there are too many people doing these tools, and each one is ... I mean, Linux is a great system, but it's also Unix. Another definition of Unix is "200 definitions of regular expressions sitting in one box". Every part of it is slightly different from every other, and I remember when I first got the manuals for XView and Motif, and you know there were all these things in there; and my reaction was, they call these open systems because the only way to use them is to have twelve books open simultaneously. [laughter]

So the mistake that I see people making is that they say, oh, I'm able to find some pattern, and then I provide hooks so people can plug in, and then they'll be able to use this and it'll do everything. And that, I find, is much less useful than it seems. It's like the analogy I made before, where you place too much reliance on a mathematical theorem, you're not supposed to have to know anything about the proof. My experience is the opposite, my experience is that it is a great surprise to me when I come across a mathematical problem that exactly fits the hypothesis of a mathematical theorem. Usually, it matches almost perfectly to something that I find in a book, but then I have to see how to customize it to my problem. And this ... When I see reusable code, it should be reusable with a little bit of editing, is the way it seems to work for me. I realize my opinion is heretical, but I just have to tell you the way it has been in my own experience.

The architect in Berkeley who came up with the inspiration for design patterns,¹⁸ I've read a couple of books of his, and I think they're quite inspiring. But that doesn't mean I just adopt everything he says. I think everybody adapts what they learn to the things that they were destined to do in their life.

[Q]: I see that you have problems, in your texts, with level 50. How many of these problems have you solved.

DEK: In *The Art of Computer Programming* I have some problems listed at level 50—these are research problems. How many have been solved? Well, actually, the only one I can remember is Fermat's Last Theorem, so I've replaced it by another one, which is $w^n + x^n + y^n = z^n$, for $n \geq 5$. So I think that will last for awhile.

But anything more than 45 is an unsolved problem. And some of the 46's and 47's have gone down. And one I just changed two weeks ago; the man in Poland, Marcin Petkovic¹⁹—I can't remember his name for sure—but he ... The question was, the smallest number of comparisons needed to sort 13 elements, and I had conjectured that it could be done in 33 comparisons. No way was known to do it in better than 34; he proved 34 is the best. See, it's kind of a scandal. If you look at optimum sorting, the very fewest comparisons in a comparison-based method of sorting, there's a bound from information theory that says the number of comparisons you need is at least the binary logarithm of $n!$, because if you make binary decisions and you have 13!

possibilities, then you have to make $\log 13!$ comparisons, and that was 33. And the best-known way to do even 16 elements is the following: Pick 10 of them; sort those 10 elements by a way that takes $\log 10!$ steps, and that way is known. Then take the other 6, and one by one, insert them among the 10 by a binary search. So that's four more comparisons for each one. And I didn't believe that by the time you got up to 13 you wouldn't be able to think of something better than to put in numbers 11, 12 and 13 each with four. And I still don't think you have to go all the way to 16 this way, but nobody knows a better way to sort 16 elements than to start with 10 and to plug in the other six one by one. But that was a research problem that recently got upgraded. And this guy was a student, and he won the best student award paper in the ESA—what is it called?—the European Symposium on Algorithms; I think it's being held this week or next week.

Another question is, when do you folks get to go home? [laughter]

[Q]: I'm a person that doesn't have an operating system question to ask. What do you think of the quality of today's programming languages?

DEK: The quality of today's programming languages ...

Well, they always seem to be ... All programming languages have this *nth-system syndrome*, where they take what they know how to do well, and they clean it up, and then they add a new feature, that they *don't* understand. [laughter] And all the time this has been true. Nobody has ever said, no, I'm not going to add any new features in my new language, I'm just going to do everything better. I mean, like if you take the subset of Ada that corresponds to Pascal, you have a beautiful language compared to Pascal. And there's a big problem with Java being inherently unstable, because the library, if I understand it correctly, it's impossible for a user program to create a subroutine that isn't in the main Java library, for which the people to whom you want to distribute this program can use exactly the same syntax they would use if it were in the standard library. They have to use a different syntax for your system than when they're using a real certified subroutine. And so every time somebody finds that there's a gap in the standard library, they have to add to the standard library, and every year the standard library is going to have to get bigger and bigger, unless they ...

I worked hard in METAFONT so that this would never happen, that somebody could add an extension to METAFONT in such a way that a primitive

¹⁸ Christopher Alexander, *A Pattern Language* (1979)

¹⁹ Marcin Peczarski, *Lecture Notes in Computer Science* **2461** (2002), 785-794.

would look as if it had been built in. Apparently the designers of Java did not do this. But I'm not an authority on that because I never read all the details of Java. My experience is that there's never been any stability in the programming language I need to use, so I started writing programs in C--, which is the subset of C that I like. [laughter] And then I discipline myself to keep information private that's supposed to be private, and things like this. But I'm not using ... All of today's languages suffer from some serious problems, and I don't see that the situation will stabilize unless developers come to a situation where they don't want to do something new, they just want to take the things that they really understand ...

With \TeX it was a different ... Why did \TeX become stable? The reason was kind of stupid, but because I *didn't* want \TeX to do everything. You know, I wanted most of all to find the cheapest way, to implement the least I could possibly do, so that I could get out of it without being totally irresponsible, and go back to writing *The Art of Computer Programming*. But quite seriously, I didn't want to spend my life coming out with something better and better and better for typesetting. I wanted to get something that would go only to a certain level, and I spent five years on the endgame, always asking "How am I going to stop, to keep it so that I don't have to do it again?"

I thought I had found a stopping point, and then I learned about Europe. No, not really. [laughter] But I hadn't made it easy to do the Norwegian alphabet, and other things that needed 8-bit codes. I had assumed that nobody would ever design keyboards where it was easy to enter 8-bit codes. And as soon as it became easy to enter such codes, there was pressure from all over the world saying, well, let's be able to do that with \TeX , so I had to spend another six months making \TeX as if it had been designed for 8-bit codes in the beginning. That was a big ... But my attitude was never to wake up in the morning and say, well, how can I improve \TeX to make it typeset better? No, my attitude in the morning was, how can I finish this and get it off my back? [laughter]

That's the only recipe that I have for having a system that's going to become stable.

Tom Lyche: Well, I think we're ready for this to come to an end. I'd like to thank you very much for this lecture, and I won't ask if there are any questions. [DEK laughs, along with the audience]

Professor Knuth will also be here on Tuesday, where I will remind you that he is giving a lecture,

a scientific lecture here on Tuesday. I hope many of you will attend that. So I think we give him a big round of applause.

[extended applause]

Narve Trædal:²⁰ Here is a little symbolic gift from the University: two small wine glasses, with the University logo on them. [DEK: Oh, great!] Thank you for your support today and through many years. We are deeply honored and grateful. Thank you so much. [applause]

²⁰ head of department administration

lengths, different real encodings, and, most substantially, different precisions, so that the same source program could not be guaranteed to have the same results on different platforms, which was of paramount importance for Knuth's goal of portability.

I must admit that even today that choice is strategically perfect and very few people have experienced any drawback caused by this "limitation".

Nevertheless, there are some fields where floating point arithmetic is important, especially if we consider that nowadays there has been a great deal of standardization in the internal representation of floating point numbers, and compilers are much more uniform than twenty years ago. Certainly there is no guarantee that the same input data processed by the same input program on different platforms yields the same results, but the probability of getting the same results is much higher.

2 Fixed radix vs. floating point numbers

Both in \TeX and in METAFONT (and later on, in J. Hobby's METAPOST) Knuth needed fractional quantities and decided to use fixed radix notation.

2.1 Real numbers in \TeX

Lengths, glues, muglues and the like are all intrinsically real quantities consisting of a measure and a unit of measure. To implement fixed radix real numbers in practice, Knuth converted all these quantities into "scaled points"; a scaled point is the fraction $1/2^{16}$ of a printer's point. Since this is the smallest unit of measure used in \TeX , and is such a small quantity (it equals approximately 200 billionths of an inch and 5 millionths of a millimeter, more precisely 53.6 \AA , a small fraction of the visible wavelengths) this unit was correctly taken as an indivisible part. Thus, any other unit, any other length, glue or muglue could be thought of as an integer number of scaled points. This in substance is the idea of a fixed radix number, a number that has a fixed number of fractional bits and is expressed by an integer number which is the multiple of the least significant bit. And this is precisely the reason why a 32-bit word can hold limited quantities which amount to, in our case, a maximum¹ of $(2^{14} - 1)\text{pt} = 16\,383\text{pt}$. Since this huge number of points approximately equals 19 feet or 6 meters, it appears that the fixed radix notation is more than

adequate for representing every useful piece of typographic information.

The transformation between input data (in decimal notation with a variable number of decimal digits) into a fixed radix number is done in the mouth of \TeX , I suppose; that is when input data is being tokenized for further processing in \TeX 's stomach.

The allowed operations that \TeX can perform on numerical data are addition, subtraction, multiplication and division, but with strong limitations; counters can be assigned integer values, including the contents of length registers, where the lengths have already been transformed into fixed radix numbers, that is into integers. Lengths, glues and muglues may be added, subtracted, and may be multiplied or divided by integers; in the input stream they can be "multiplied", or should I say "scaled", by a real number; this means that if you have a length variable named, say, `\foo`, containing some value, you can write something like this in your input stream

```
\foo=0.67\foo
```

or in \LaTeX

```
\setlength{\foo}{0.67\foo}
```

by which the contents of `\foo` is substituted with its previous contents multiplied by 0.67.

All of the graphic extension packages to \TeX and \LaTeX , such as \PCTEX , have to produce some sort of calculation between real numbers; \TeX was not built for that, but by means of some "dirty tricks", such as the ones described in Appendix D of *The \TeX book* [1], the task becomes possible. For example a multiplication is performed as such: the first factor "scales" the fixed length of 1 pt and is assigned to a length variable; this length variable is scaled by the other factor, and finally the result is extracted by means of the primitive `\the` and stripped of the ending pt unit of measure. Divisions are more complicated because of the possibility of dividing by zero and because they might give rise to oversized results (this can happen as well with multiplications). The extension package `calc.sty` helps the user in specifying the calculations to be performed on the fly, but does not extend the possibilities offered by \TeX primitive commands. Close examination of the various other extension packages available in the CTAN archives clearly shows the many efforts spent by several \TeX ies in order to help making calculations a little more comfortable and reliable.

Notice that \LaTeX 's New Font Selection Scheme frequently performs this kind of calculation, including stripping the pt part from `\the` output.

¹ Since 16 bits represent the fractional part, there remain 16 bits available for the integer part; one bit is used for the sign (or equivalent) and the other is for special purposes; there remain 14 bits available for the integer part of a fractional number, hence the maximum value is $2^{14} - 1$.

Another extension that makes heavy use of real numbers is the `graphics` package, which needs to determine the transformation matrix coefficients for rotations by means of the trigonometric functions. By reading the code, I discovered marvels of numerical computation; even considering that the `TeX` community includes many people with varying competence and skill, I must admit that these sorts of numerical procedures are difficult to find even in the specialized literature.

For this reason, writing any extension package that requires any kind of calculation becomes very difficult and error prone.

2.2 Real numbers in METAFONT

METAFONT works with integer and real numbers in a very efficient way; actually all numbers may be thought of as real numbers. It performs any kind of operation and allows the user to specify also some ordinary irrational and transcendental functions such as the trigonometric ones, logarithms and exponentials; it actually deals with complex numbers, whose real and imaginary components are real numbers, but users not skilled in complex analysis need not worry about them. Whoever has done any work with METAFONT cannot but be surprised in discovering its endless resources.

Nevertheless, even METAFONT uses fixed radix real numbers and the largest one, according to *The METAFONTbook*, is 4095.99998, which, taking into account the fractional part, equals $4096 - 2^{-16}$. Now since 4096 equals 2^{12} Knuth reserved four bits of the integer part for the sign and other internal METAFONT requirements. If that number is considered as a number of points, that is almost 5 feet or one meter and a half, certainly more than sufficient for drawing any character. The same limitations hold true for METAPOST.

If the instructions given to METAFONT, or to METAPOST, are correct, very rarely are they forced to issue overflow or underflow messages; nevertheless sometimes it happens. One such instance happened when trying to convert the METAFONT description of very large math delimiters into Type 1 PostScript format, since the program that does this conversion runs METAFONT with a pixel density large enough to fit the 1000×1000 PostScript coordinate space.

3 Floating point numbers

A floating point number, for those who are not familiar with this terminology, is a real number written down in a special way and may be thought of as being made up of two parts: the mantissa and the exponent. For example the decimal number 105.234

may be written in the form $0.105234 \cdot 10^3$; 0.105234 has a null integer part, as any other floating point number, so that the significant information is contained in the mantissa “105234” and in the exponent of 10, “3”, and these two numbers are all that is needed to identify a floating point number. Since the exponent may be very large in absolute value and the mantissa can contain many digits, the floating point representation can deal with an extremely wide range of numbers.

So, what is the difference between fixed radix and floating point numbers, neglecting their range and coding system? The main and fundamental difference is the precision: fixed radix numbers have a fixed number of fractional digits, leading to a fixed *absolute* precision, while floating point numbers have a fixed number of *significant* digits, leading to an approximately fixed *relative* precision. The minimum distance between two consecutive fixed radix numbers is 2^{-16} , while the minimum distance between two consecutive floating point numbers² is 2^{-24} times the base 2 raised to the exponent of the floating point number.

Floating point numbers are essential for engineers, physicists, technologists, etc., whose achievements wouldn't even exist (or at least would be much less impressive) if they had to work with fixed radix numbers. Engineers in particular are the masters of approximation, the “artists” of approximation; engineering theories are based on reasonable simplifications and approximations so as to render the more exact physical theories applicable and useful. As an engineer — did you have any doubt? — I like floating point numbers, even if I appreciate the efforts made by the mathematicians and by Knuth in particular for producing such wonderful masterpieces of the “art of computer programming” given by `TeX` and METAFONT.

4 New typesetting systems

As every reader of *TUGboat* knows well, there are several teams that are working on “upgrades” of

² A word length of 32 bits, the same as with fixed radix numbers, is divided in two parts: 24 bits for the mantissa and 6 bits for the exponent (let me skip the technicalities) plus two bits, one for the sign of the mantissa and one for the sign of the exponent. This means that exponents of the binary base 2 can range from -63 to $+63$. High level languages such as Pascal or C (the two languages that `TeX` is commonly translated to from its original WEB literate programming language) also support double precision floating point numbers (64 bits) and sometimes even quadruple precision (128 bits); such representations allow for increased precision mantissas and an even larger range for the exponents, but I think they are excessive for our purposes with `TeX` and METAFONT.

\TeX . I am not aware of any work going on for upgrading **METAFONT**, but maybe I am wrong.

As every reader knows well, Knuth declared some years ago that \TeX and **METAFONT** are frozen; any further modification will simply correct existing errors, but the programs will not be modified so as to add new functionality. But Knuth himself encourages downward compatible new programs that can do things that the actual \TeX and **METAFONT** cannot do.

The very popular **pdftEX** offers the opportunity of producing portable document formatted documents directly from a **tex** source; this is not the only new functionality, because **pdftEX** improves the already excellent \TeX paragraph line breaking algorithm by implementing a hanging punctuation facility and variable width font justification. *TUGboat* readers may have enjoyed the transcript of H n Th n's PhD thesis on this subject [4].

$\varepsilon\text{-}\text{\TeX}$ [6] is another approach to extending \TeX : it removes the limitation of 256 counters, lengths, glues, ..., it allows bidirectional typesetting, this feature being very useful for mixing languages such as Latin/Cyrillic/Greek scripts with Hebrew and/or Arabic ones, plus many other improvements, too many to go into a detailed description. I directly asked Phil Taylor, one of the authors of $\varepsilon\text{-}\text{\TeX}$ within the \mathcal{NTS} working group, to examine the possibility of extending the program functionality to floating point numbers, but he very humorously refused in a unquestionable way.

Another ongoing project is Ω , by Plaice and Haralambous, together with its companion Λ ; the former extends \TeX while the latter is the Ω extension of **LATEX**. Besides removing the limitation of 256 counters, lengths, etc., Ω can deal with Unicode fonts because it can address glyphs in a set of 2^{15} instead of being limited to the 256 characters per font we are all used to. The many other enhancements offered by Ω are described in [7]. A shorter but extremely good description of Ω and Λ is in [8, chap. 10].

Maybe one day all these project extensions of \TeX will be merged in the successor of \TeX ; but as far as I know none of them deals with floating point numbers. Very nasty situation for a fond appreciator of floating point numbers such as myself.

But what I am asking is nothing special: it sums up to introducing the notion of floating point numbers, with the associated **fpcounters**, and a set of rules for operating on them: (a) the usual arith-

metic operations, (b) the trigonometric functions,³ logarithm, exponential, square root (as they are provided by Pascal and C), (c) a set of rules for mixed mode calculations, and (d) extension of the allowable \TeX expressions as they are defined in $\varepsilon\text{-}\text{\TeX}$, although $\varepsilon\text{-}\text{\TeX}$ now accepts all numerical quantities except real numbers.

I propose the usual operations on numbers with different operand types, such as: any arithmetic operation between floating point and integer numbers, the result being floating point; truncation of a floating point number to an integer; scaling of lengths by floating point numbers, the result being a length; the assignment of a length to a floating point variable, where the implied fractional separator is taken into account; the same operations on glues (with the same limitations); and so on. Most of these operations are already hardwired in the high level languages, so that their implementation is quite simple; the remaining ones imply operations (multiplication and division) between floating point numbers and fixed radix ones, which should not be a problem at all.

Nothing should be changed in the way this suggested extension to \TeX formats paragraphs and pages for at least one obvious reason: it should yield exactly the same results as \TeX , in order to be a downward compatible extension producing the same results for existing documents. But the extension to real numbers could produce simpler and more efficient extension packages, especially those that have to deal with graphics, and with PostScript and pdf output.

5 Real numbers and **METAFONT**

As I explained above, **METAFONT** already uses real numbers, although they are internally represented with fixed radix ones. I believe that the modification of **METAFONT** into a new program that uses floating point real numbers could avoid some of the actual limitations on the size of the quantities dealt with, with negligible impact on the shape of the characters being produced.

When Knuth decided to use fixed radix real numbers in **METAFONT**, computers used to behave in different ways and by the start of the '80s were rather slow compared to modern computers. Back in 1985 my 286 AT computer with an 8 MHz clock needed a few minutes to produce any one of the 128 character fonts of the Computer Modern family.

³ The default angle unit should be the nonagesimal degree, as in **METAFONT**, not the radian.

Now my Pentium based laptop requires a few seconds to produce any one of the 256 character fonts of the EC family, although my laptop is rather “old” and runs with a clock of 233 MHz. Back in the ’80s integer arithmetic (as the fixed radix representation substantially is) was much faster than floating point arithmetic; modern CPUs have special facilities for floating point arithmetic so that the number of floating point operations per second is not dramatically different from the corresponding number of integer operations as it was in the past.

At the same time the implicit rounding involved in any fixed radix multiplication (and division) implies that results are “never” precise, but always approximate; if you ask METAFONT to show the result of the expression $(1/3)*3$ it will display 0.99998,⁴ instead of 1.00000 as shown on page 62 of *The METAFONTbook* [2], together with many other calculations that exhibit the same sort of “error”. With floating point numbers the situation would not be any worse, because with 24 bits available for the mantissa, floating point real numbers have from 6 to 7 “precise” *significant* decimal digits (the relative error ranging approximately from $6 \cdot 10^{-8}$ to 10^{-6}).

But in my opinion the opportunity to change METAFONT to floating point arithmetic arises from the strong arguments connected to section 7.

6 Real numbers and METAPOST

METAPOST also works with fixed radix arithmetic (although nothing is said about in the user manual [3]), but since its purpose is to output graphics of different kinds in PostScript format, and since PostScript uses floating point arithmetic, the arguments discussed in section 7 become even more compelling for extending METAPOST to floating point arithmetic.

7 PostScript and floating point arithmetic

The PostScript language uses both integer and real numbers, the latter being operated upon as floating point numbers.

When Knuth wrote T_EX and METAFONT, PostScript, and in particular PostScript fonts, were in their infancy. Printers that could interpret the PostScript language were expensive and relatively slow, to the point that when Tom Rokicki wrote his excellent translator from dvi to PostScript, he stated that “pk fonts produce better results and run faster

⁴ With floating point numbers having 24 bits for the mantissa the same calculation would yield 0.99999994 with a relative error of $6 \cdot 10^{-8}$; the fixed radix calculation has a relative error of $2 \cdot 10^{-5}$.

on PostScript printers”.⁵ When Rokicki originally wrote his translator, this was true, in the sense that processors and language interpreters mounted on the PostScript printers and phototypesetters of that age were much slower than modern printers and phototypesetters.

At that time there was no Portable Document Format (pdf), so PostScript output was originally used only for printing. Previewing was made possible by Ghostscript (by Peter Deutsch) and GhostView (by Tim Theisen), respectively a PostScript interpreter intended to be resident on the computer instead of the printer and capable of driving a variety of output devices, and a graphic interface for displaying the PostScript output on the screen with the ability of moving within the virtual document.

Nowadays, pdf has become a standard for moving typeset documents from one computer to another along the infinite routes of the Internet, with the assurance that each document will be viewed, and possibly printed, in the same way on any platform, running under any operating system. The T_EX suite has been complemented with the pdft_EX typesetting program [4], with its pdflat_EX variant, with the `hyperref.sty` extension package, so as to make available internal and external cross-linking, and so on.

On the other side the PostScript output produced by dvips may also be converted to pdf format with Adobe Distiller, by Ghostscript itself (by specifying the suitable output driver and the other necessary pieces of information), and other tools.

But this is the real point: viewers for pdf documents display bitmapped fonts very poorly. At the same time, pdf documents are to be read (primarily) on the screen, irrespective of the screen pixel density and size. On large screens the reader may want to keep more than one window open, so that resizing windows implies the simultaneous resizing of what is contained in the window itself. Bitmaps are very poor suited to this task, and bitmapped fonts may become almost unreadable even if originally they were produced with high pixel density settings. Printing the documents does not suffer so much from the poor quality of bitmapped fonts on the screen, because the physical page cannot be resized at will.

In other words T_EX, L_AT_EX, and other T_EX dialects, *have to drop the bitmapped fonts*, as well as pdft_EX, pdflat_EX, etc. All these programs must be capable of using Type 1 PostScript fonts.

⁵ I am quoting by heart, because the last documentation on his translator dvips, [9], does not contain this statement any more.

This explains why recently the CTAN archives have seen a variety of package extensions that make available *complete* PostScript fonts to *TEX* and, especially, to *L^AT_EX*; I am referring myself to the excellent **txfonts** (Times extended) and **pxfonts** (Palatino extended), that are *complete* in the sense that they contain all the glyphs, ligatures, symbols that are used by *L^AT_EX* with the **amsmath** extension, plus many more; if the T1 encoding is specified the font glyphs corresponding to the 256 character *TEX* encoding are all present, and if the **textcomp** extension package is invoked, the Text Companion glyphs become available. All families, series and shapes are available and are designed in such a way that blend together without the difficulties experienced in the past when the standard PostScript fonts were adapted via several patches (and several omissions) for use with *TEX*. Another precious set of Type 1 fonts is the collection of the CM-super fonts; these are Type 1 fonts including the Latin and the Cyrillic alphabets, all the math fonts and the Cork encoded fonts. In a near future it should include also my Greek fonts, at least the author is working on it. The Latin Modern font collection is an even more recent development, not to mention the CM-LGC font collection that contains the PostScript versions of the CM Latin, Greek and Cyrillic fonts. Both available in the CTAN archives.

*But this trend is going to cut off METAFONT from the *TEX* community if it does not upgrade to output PostScript format.*

This is the real point I would like to stress in order to support my request for floating point arithmetics in *TEX*, METAFONT, and METAPost.

8 METAFONT, METAPOST, and Type 1 PostScript fonts

The CTAN archives contain many different fonts designed with METAFONT; such fonts are generally of a very high quality, unless they were just filling the role of patches for supplying certain glyphs not available elsewhere.

It is possible to convert the METAFONT generated fonts to PostScript format in a variety of ways, most of which are also applicable to converting bitmapped fonts obtained by optically scanning original specimens. Some information is available in the paper [18]; more recently Karl Berry has written a detailed paper which appeared in *TUGboat* [10], showing some of what can be done with the software tools available today.

Here I summarize some of these tools and add some of the information I collected either by direct experience or from the documentation. In any case

I must warn the reader that most of these programs are native on UNIX or Linux platforms. However, by means of the environment **cygwin** [11] such programs can be operated also on Win32 platforms (sometimes with minor limitations); therefore, it is not a problem to load and use these programs even if one doesn't have a Unix or Linux platform.

8.1 METAPOST and **roex**

METAPOST can directly read METAFONT programs and produce PostScript files—but they have to be edited quite a bit in order to make PostScript fonts, and then the result is not of Type 1, but of Type 3. The main difference is that every connected part of each glyph in Type 1 fonts is described by just two contours, the external one and the internal one, while METAFONT, and thus METAPOST, produce the glyphs as a superimposition of several strokes that intersect or join superimposing some of their parts, so that they may only be classified as Type 3. But this limitation can (sometimes) be overcome, so read on:

CTAN holds an extension to METAFONT that allows for redefining the **filldraw** and **stroke** METAFONT commands at **shipout** time, so as to join several strokes into a single stroke with one external and one internal contour. This makes the output suitable for further processing by METAPOST to produce PostScript files that conform with Type 1 format, as described above. The resulting files still have to be edited and merged in order to have a single file describing the whole font. The extension package for removing overlaps and expanding strokes in METAFONT fonts is contained in the file **roex.mf** available from CTAN within a zipped file **roex.zip** that contains other valuable information and examples [19]. The drawback is that this procedure is not suitable for processing existing METAFONT fonts because **roex** is not infallible and some editing of the source code may be required.

8.2 Autotrace

The Internet offers a mighty program, **autotrace** [12] by Martin Weber, together with a graphical front end **frontline** (available from the same home page as **autotrace**) that may be used for determining the contours of arbitrary stroked lines, and therefore of character glyphs. You may refer to Karl Berry's paper [10] for a short description of the possibilities of this software.

The essential point is that **autotrace** can determine the third order Bézier splines that make up the internal and the external contours of every connected part of a glyph, so as to output the node and

control points of such splines, which in turn are the only information PostScript requires for drawing the glyph. The input to **autotrace** must be a bitmap in certain specialized formats; the most likely for our purposes is the ‘portable network bitmap’ (.pnm).

8.3 Pfaedit

The superb interactive graphical font editor **pfaedit** by George Williams [13] supports creating and editing PostScript and TrueType fonts. It is capable of importing bitmaps, including the gf format that METAFONT outputs, and of invoking **autotrace** so as to trace them.

By the end of the tracing task the designer already has the contours of each and every glyph already set in its graphical window, so that she can optionally edit the contours and/or simplify the set of nodes; add kerning and ligature information, write the initial PostScript preamble information; and eventually output the font file in **pfa** (printer font ASCII), **pfb** (printer font binary), TrueType, or other formats.

8.4 textrace and mftrace

In spite of the apparent simplicity of the operation with **autotrace** through **frontline** or **pfaedit**, it is even easier and faster to operate through one of two other front-end scripts that make the necessary preliminaries, launch the programs, and output the desired font files.

The Perl script **textrace** [14] can analyze a bitmapped drawing and build up the contours comprising it. It produces the bitmap file in pnm format with the necessary scaling for the 1000×1000 PostScript coordinate space, and launches **autotrace** with suitable parameters. It eventually assembles the PostScript font and outputs its file in the desired format. As far as I know, most of the METAFONT fonts now available on the CTAN in PostScript format have been produced by using **textrace** or similar programs. Also the EC fonts, as produced with **textrace**, have been available on the CTAN for some time (but I can’t give credit to the author because his/her name is encrypted in his/her e-mail address).

The Python script **mftrace** by Han-Wen Nienhuys [15] was formerly known as **pktrace**; the name was changed to the present one after some interaction with its author who was so kind to introduce some specific extensions so as to allow me to use the program in a cygwin environment without resorting to the teTeX distribution (available with cygwin), but by using the MikTeX distribution that I had on my laptop. **mftrace** is a Python script explicitly

designed for translating METAFONT fonts to PostScript format. It starts by generating the **tfm** file, if it’s not already available (this operation must be done by hand under cygwin with MikTeX); from this file it computes the magnification to feed to METAFONT in order to scale the generated bitmap in gf format to suit the 1000×1000 PostScript coordinate space. It then transforms the gf bitmaps to the pnm format and feeds such new bitmaps to **autotrace** with suitable parameters. Next, it feeds the generated code to **pfaedit** with suitable parameters and optionally with the specification of an encoding, so as to optionally simplify the contours and to fill up all the necessary information in the font preamble; the output is finally written to a **pfa** or a **pfb** file. Nothing else should be necessary for using the font, because the TeX metric file was available or was explicitly generated during the process, except for adding the font name to a font map that **dvips**, or **pdftex**, or **dvipdfm** can access and read.

8.5 Metafog

Finally there is **metafog**, a program for doing exactly the direct passage from METAFONT to PostScript. It was written by Richard Kinch, who also wrote a paper on the subject [16]. From what I know, it is not in the public domain, in that it is only available by buying Kinch’s TrueTeX product.

- * -

I have used **mftrace** and obtained good results in very little time; since I have available a cygwin environment, I can draw fonts with **pfaedit**, or I can do the work with METAFONT and then pass the result to **mftrace**. Being accustomed to METAFONT, I find it more comfortable to use the latter method, but I always verify the result with **pfaedit** and possibly make some little corrections.

9 Proposal for a PostScript-enabled METAFONT

But if **metafog**, **mftrace**, and the like already do the conversion, why am I pleading in favor of the introduction of floating point arithmetics in TeX and especially in METAFONT and METAPost? Well, on one side **metafog** is not publicly and freely available as are all the other pieces of software that belong to the TeX distributions. On the other, it’s a philosophical position: I find it stupid to start with a beautiful piece of software such as METAFONT, that has so many facilities for finding the contour points and for describing the third order Bézier arcs in order to draw the glyph strokes, to proceed with rasterization that loses all this information even if it

smoothly handles the arbitrary (convex) pen shape, and to try to recover it from the rasterized drawing when you had all the information from the very beginning.

Unfortunately I can't program in a decent way either in Pascal or in C; my knowledge of METAFONT is greater than the average user's, but it is far from perfect. My geometrical competence is nowhere near that of John Hobby, just to mention the author of METAPOST and the contributor of the algorithm for finding the control points of the third order Bézier splines as described by Knuth on pages 130–131 of *The METAFONTbook*. Therefore I cannot give any suggestion to someone who might want to try to work on a New METAFONT.

Such a New METAFONT should be downward compatible with the “old” METAFONT, of course, just as ε - \TeX and $\text{pdf}\text{\TeX}$ are downward compatible with \TeX . But one of the modes should be a `ps` mode, by which a font `pfb` file is produced together with the usual \TeX font metric file. The files produced this way should be quite a bit smaller and the scalable fonts should be drawn in a simpler way, since the splines are generated directly by METAFONT with the minimum number of points; anybody who has designed a font knows well that rarely does a particular glyph require more than a dozen triplets of z points; some simple strokes require just a couple of triplets, because the control points are determined by METAFONT itself. Altogether, most glyphs may be drawn with less than two dozen points for the inner and respectively the outer contour, which is much less than what you get when using `mftrace` or `textrace` or any other interface to `autotrace`.

Not only that, but the New METAFONT might be capable of producing Multiple Master fonts with little or no effort; in fact the Computer Modern fonts by themselves are a collection of several master fonts, `cmr5`, `cmr6`, `cmr7`, `cmr8`, `cmr9`, `cmr10`, `cmr12`, `cmr14`, `cmr17`; every stroke is described by the same METAFONT statements, only the dimensional parameters are varied from one master to the other, giving rise to a set of fonts that are not obtained one from the other by simple scaling operations.

The EC fonts by Jörg Knappen do even better: the parameters are defined in a single file as vectors of data in univocal correspondence with the design size. EC fonts have master files that contain in their name the design size, so that `ecr1440.mf` is the master file for the serifed upright medium font and its design size of 14.40 pt is set by a statement contained in this master file. But nothing forbids copying and editing that file into another one, say

`ecr3125.mf` where the design size assignment is corrected to 31.25 pt, run it and get the font designed at that very size, without any magnification.

The CB Greek fonts, in this respect, are even simpler, in the sense that the design size is directly determined by the font name, stored in the `jobname` variable, that contains in the last four characters the design size multiplied by 100. Therefore the CB Greek master files have all the same contents, because they need not be edited in order to redefine the design size.

Well, the New METAFONT might be able to read all the font master files starting with the same literal preamble, for example all the `cmr5.mf` ... `cmr17`, get all the parameters and produce a single Multiple Master Type 1 PostScript font, named, say, `cmr`; the same could be done with the EC fonts or the CB Greek ones.

On this subject there is an extension package, together with its documentation, that already sort of simulates Multiple Master fonts with METAFONT; see [17] for further details.

All these capabilities are already embryonically present in METAFONT and METAPOST: why not to try to develop them and bring them to light? METAFONT and METAPOST are powerful tools for font (and graphic) design, much more powerful than most other extant programs for font design. Why should we, the `texmf` community, give them up in favor of less sophisticated programs?

10 Conclusion

The line of thought of introducing floating point numbers in \TeX , METAFONT, and METAPOST carried me further, into discussing the urgent need to think also to a New METAFONT program capable of directly producing Type 1 PostScript fonts, possibly even Multiple Master fonts. In this way it is possible to keep both \TeX and METAFONT up to date with the modern necessities of document production, compatible with the Portable Document Format. In fact, by its very nature, it suggests that documents are directly read from the computer screen, a task that today's METAFONT fonts are not particularly good for.

References

- [1] Knuth D.E., *The $\text{\TeX}book$* , Volume A of the series “Computers and Typesetting”, Addison Wesley, Boston (1986).
- [2] Knuth D.E., *The METAFONTbook*, Volume C of the series “Computers and Typesetting”, Addison Wesley, Boston (1986).
- [3] Hobby J.D., *User manual for METAPOST*, file

- `mpman.pdf`, (1997), in any CTAN repository.
- [4] H n Th  Thành, *Microtypographic extensions to the *TEX* typesetting system*,, TUGboat 21.4 (2000), pages 317–434.
 - [5] H n Th  Thành, *The pdftEX user manual*, file `pdftexman.pdf`, (2000), in any CTAN repository.
 - [6] Breitenlohner P., *The ε-TEX manual*, file `etex_man.pdf`, (2000), in any CTAN repository.
 - [7] Plaice J. and Haralambous Y., *Draft documentation for the Ω system*, file `omega-manual.pdf`, (1999), in any CTAN repository.
 - [8] Syropoulos A., *Digital typography Using LATEX*, Springer Verlag, New York 2002.
 - [9] Rokicki T., *a DVI-to-PostScript translator*, file `dvips.dvi`, (1997), in any CTAN repository.
 - [10] Berry K., “Making outline fonts from bitmap images”, in *TUGboat*, v. 22 (2001), No. 4, p. 281–285.
 - [11] cygwin, a Unix-style environment that allows operating a Unix program on a 32-bit Windows platform; available from <http://sources.redhat.com/cygwin>.
 - [12] Weber M., *autotrace*, available from <http://autotrace.sourceforge.net>.
 - [13] Wilson G., *pfaedit*, available from <http://pfaedit.sourceforge.net>.
 - [14] Szab  P., *textrace*, contained in `textrace-latest.tar.gz` to be found in any CTAN repository or available from <http://textrace.sourceforge.net>.
 - [15] Han-Wen Nienhuys, *mftrace*, available from <http://www.cs.uu.nl/~hanwen/mftrace/index.html>
 - [16] Kinch R., “MetaFog: Converting METAFONT shapes to contours”, *TUGboat* 16.3 (1995), pages 233–243.
 - [17] Berdnikof A.S. and Turtia S.B., *mmf.sty: Computer Modern Typefaces as the multiple master fonts*, file `mff.dvi`, (1997), in any CTAN repository.
 - [18] Pakin S., *mf2pt1 – Produce PostScript Type 1 fonts from METAFONT source*, file `mf2pt1.pdf`, (2001), in any CTAN repository.
 - [19] Jackowski B., Pianowski P. and Ry ko M., “Package for removing overlaps and expanding strokes”, file `roex.zip`, (1998), in `/graphics/MF-PS` in any CTAN repository.

◊ Claudio Beccari
 Politecnico di Torino, Turin, Italy
`claudio.beccari@polito.it`

Table 1: Classical English sheet sizes

Sheet	Broadside (in)	Octavo (mm)	Ratio
Foolscap	13 1/2 x 17	108 x 171	1.588
Crown	15 x 20	127 x 191	1.500
Post	15 1/4 x 19	121 x 194	1.605
Large Post	16 1/2 x 21	133 x 210	1.571
Demy	17 1/2 x 22 1/2	143 x 222	1.556
Medium	18 x 23	146 x 229	1.565
Royal	20 x 25	159 x 254	1.600
Super Royal	21 x 27	171 x 267	1.556
Imperial	22 x 30	191 x 279	1.467

they are folded into signatures. The printed sheets are gathered, folded into signatures, and passed on to a bookbinder who sews the signatures into the book proper. The bound books are finally cut and covered.

As is so often the case with old and venerable crafts, a florid terminology developed which facilitated contacts between the guilds of printers and binders, while at the same time inspiring some sort of grudging respect in the uninitiated, keeping them at bay.

2.1 The paper sheets

The sheets of paper, the broadsides, come in a limited number of more or less standard sizes. Labarre's 'Dictionary and encyclopedia of paper and paper-making' [5] is an absorbing mine of information, listing amongst many other fascinating details, the names used to designate particular sheets of paper. In the Anglo-Saxon world, sheets are known as Foolscap, Crown, Post, Demy, Medium, Royal, Super Royal and Imperial (table 1).

Of course, different countries adopted their own habits and standards. It is well known that typographic standards in France developed in a different, separate way. The Didot point and the Cicero (some 7% larger than their Anglo-Saxon point and pica equivalent), the 'modern' typefaces with their characteristic vertical shading and hairline serifs, continue to exert their influence on European typesetting. It is therefore not surprising that there are also French sheets of paper (table 2).

Western typesetting began with Johannes Gutenberg, and the Germanic tradition is fundamental to the printing of books. The development and interactions with the other European styles of typography is engrossing and complex (McLean sketches some aspects of this history [6]; some informative ar-

Table 2: Classical French sheet sizes

Sheet	Broadside (mm)	Octavo (mm)	Ratio
Cloche	300 x 400	100 x 150	1.500
Tellière	340 x 440	110 x 170	1.545
Coquille	440 x 560	140 x 220	1.571
Carré	450 x 560	140 x 225	1.607
Couronne	460 x 720	180 x 230	1.277
Royal	480 x 630	157 x 240	1.529
Raisin	500 x 650	162 x 250	1.543
Jésus	560 x 760	190 x 280	1.474
Colombier	620 x 850	212 x 310	1.462
Soleil	580 x 800	200 x 290	1.450
Grand Aigle	700 x 1040	260 x 350	1.346

Table 3: Classical German sheet sizes

Sheet	Broadside (mm)	Octavo (mm)	Ratio
Propatria	340 x 430	108 x 170	1.574
Bienenkorb	360 x 450	112 x 180	1.607
Bischof	380 x 480	120 x 190	1.583
Register	420 x 530	132 x 210	1.591
Kl.Median	440 x 560	140 x 220	1.571
Regal	440 x 670	167 x 220	1.317
Lexicon	500 x 650	162 x 250	1.543
Regal	500 x 720	180 x 250	1.388
Super Royal	540 x 680	170 x 270	1.588
Imperial	570 x 780	195 x 285	1.462

ticles on the subject can be found in Klein [2]; and there is also a good discussion in Williamson [14]). It does not come as a surprise then to find that differences and divergences extend to the dimensions of the sheets of paper printed on (table 3).

Folding the sheets adds another set of terms: a single fold yields a folio, a double fold a quarto, a triple fold an octavo and so on (fig. 1). Hence, the combination of a sheet name and the number of folds denotes a particular and precise size of the book [5, 14].

Jan Tschichold devoted his life to typography, and to the design and production of books. Over the years, he wrote extensively about many of his (changing) views and discoveries. Later in life, he assembled his findings in an overview, discussing in a more coherent, concentrated fashion many of the questions which face anyone wishing to make a book

Table 4: Modern sheet sizes

Sheet	(in)	(mm)	Ratio
letter	8 1/2 x 11	216 x 279	1.292
legal	8 1/2 x 14	216 x 356	1.648
executive	7 1/4 x 10 1/2	184 x 267	1.451
A4	8 1/4 x 11 3/4	210 x 297	1.414
A5	5 7/8 x 8 1/4	148 x 210	1.414
B5	7 x 9 7/8	176 x 250	1.420

[12]. Of particular interest is his analysis of size and proportions of page, text and margins [11].

Through his historical investigations, he made a very strong and convincing case against the ‘modern’ tendency of making squarish books. His elegant, eloquent and well illustrated discourse remains valid today and guides us away from the ugliness of the convenient but rather square A format (proportion $1:\sqrt{2}$) back to the more rectangular formats with proportions tending towards the Golden Ratio ($1:1.618$). It is certainly no accident that the octavo-folded sheets of paper traditionally used in the printing of books show proportions which fall very nicely within this range (as you can see in tables 1 to 3). The traditional sheets yield two proportion groups, i.e., broadsheet, quarto, sextodecimo form the first set, while folio, octavo, trigesimo-secundo make up the other set. A crown sheet, for example, will change from a 3:4 proportion to a 2:3 proportion and back. In contrast, the ISO A series has the (dubious) advantage of retaining its proportions no matter how often it is folded. The price for the convenience of the $1:\sqrt{2}$ proportion is the ugliness of its squarish look. Comparing the entries in table 4 with the other tables shows the prevalence of this ‘square’ nature in the modern standard sheets.

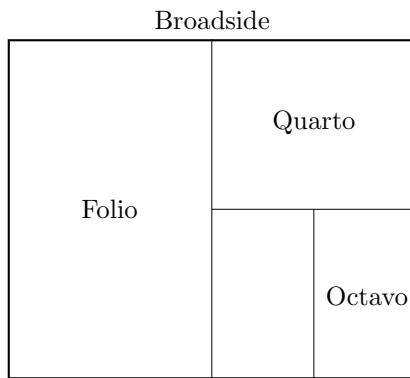


Figure 1: Folding terminology

Table 5: Rosarivo’s page division to text area

text to page length	text area
6:9	44%
7:10	49%
8:11	53%
9:12	56%
10:13	59%

2.2 The spread

The size and the proportions of the typeblock, and their relations to the page, have greatly exercised the minds of the makers of books, and much ink has been spilled arguing in favour of and/or against any design. An excellent illustration of the plethora of designs is found in the second chapter of Wilson’s manual of his memoir class [15]. He shows some 32 (!) different page designs which have been used in actual books, made between 1087 and 1995.

Tschichold realised that the first printers looked for the size and proportions of page and typeblock to the handwritten books. He reasoned that this would have led to the establishing of secret canons known only to the initiated of the workshops. He set out on the laborious task of trying to rediscover these canons through measuring many, many historical books. Tschichold succeeded, and published his findings in 1953. He found two methods, one prevalent in the Middle Ages, and a different, late Gothic one. It turned out that the latter canon had also been arrived at through a different construction by Van de Graaf [13] (fig. 2, on the left), adapting the method of the Picardian architect Villard de Honnecourt, who was active in the twelfth century. This

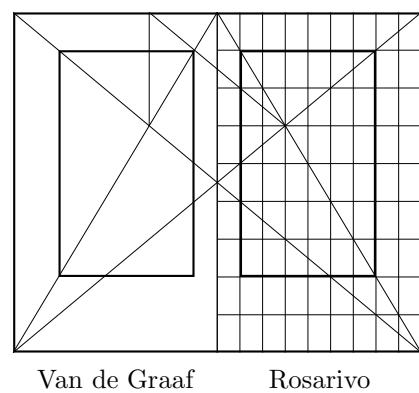


Figure 2: Methods of determining margins

method has been brought back to attention in modern times by Hans Kayser [1].

Rosarivo confirmed this determination through his own, independent discovery of another method which, happily, yielded the same results [8]. Rosarivo's method is very easy to use and implement algorithmically: divide the height and width of the page into an equal number, assign one part to make up the inner and upper margins, and two parts to make up the outer and lower margins, with the remainder defining the area available for printing the body text (fig. 2, on the right).

The area occupied by the text block relative to the total area of the page follows directly from the number of divisions, as shown in table 5. It comes as something of a surprise to notice that only about half the area of a page is occupied with text. And yet, for centuries, the 6:9 division of page dimensions was very much the norm. Both Town and Williamson discuss the matter of margins at some length, and deplore the shrinking of white space with the advent of 'modern, economical' times [10, 14].

2.3 The typeblock

Determining the right amount of leading between lines of print is not a straightforward matter. It depends very much on the nature of the fount, its size, and the length of the line. Founts with very short extenders (of which Times is an example) require more leading than founts with normal or long extenders. The reader benefits considerably from more leading when the text is made up of long lines. But whatever leading is arrived at, it should remain fixed for the entire text, yielding a constant, regular grid of lines. At least, that has been the position of European printers. American printers took a different point of view, and preferred to maintain the unity of the printed block as a whole. Thus they have been prepared to upset the leading between or even within paragraphs in order to maintain the overall size of the printed area. Knuth developed *TeX* to deal especially with books containing mathematics, a notoriously difficult task for typesetters. The ingenuity required to deal algorithmically with this task is truly considerable, and compromises could hardly be avoided. The regularity of the printing grid was one victim of the solutions arrived at.

It should be clear by now that the standard book class included in *LAT_EX* requires a fair amount of adjustment to reflect the design principles just highlighted. The Octavo package is an attempt to provide such a class file.

3 Rationale

For a great many decades, octavo-size books were the most prevalent of all, and it seemed apt to use the name for an attempt to join and hopefully revive this tradition. The Octavo class is a modification of the standard *LAT_EX* book class, with a similar, but limited, number of options and choices. These limitations are built in quite deliberately: it is the very point of this class to assist in the making of books following the particular design principles and guidelines which helped to produce beautiful books during the Renaissance.

The idea behind Octavo is to have a quiet, unobtrusive design which is not meant to shout at the reader, but rather gently helps her along. Three major features characterise Octavo:

- The page sizes are the classical octavo sizes.
- The margins are defined as the late Gothic ones.
- An attempt is made to maintain a uniform grid for the entire text.

Some of the lesser features are the following:

- All display is produced as quiet, unostentatious lines.
- Chapters and lower-order sections are left unnumbered.
- Figures, tables and equations are consecutively numbered throughout the document.
- Tables of contents, of figures, etc., do not use dotted lines.
- A title page is not made but can be requested.

More discussion of these points follows.

3.1 Page sizes

The definition of octavo-sized pages and the calculation of margins in harmony with the dimensions of the page is the necessary first step for the making of a beautiful book. It provides the canvas and the frame that will be with the reader for the entire length of the book. Their beauty, even if only subliminally noticed or appreciated, does contribute to the pleasure of holding and reading the book.

3.2 Margins

The margins are calculated by applying Rosarivo's method and dividing the page dimensions into ten, with seven parts for the text and three for the margins, which yields 49% area of the page with printed text. This is somewhat more generous usage of the page than the medieval tradition of a ninefold division with only 44% of the page covered by text, but such lavish margins would likely startle the modern reader. Headers and footers are placed outside this

block, and thereby help to create a feeling of balance between occupied and white space on the page. Especially the consistent placing of the folio centered at the foot of all pages has such a beneficial effect.

3.3 Title page

A title page is an integral and very necessary part of any book. However, producing a satisfactory title page is not as straightforward a matter as it may seem at first. Even a superficial perusal of the many pages and examples devoted on the subject in the already cited works by Tschichold and Williamson will convince the reader that making a proper title page is not something to be done automatically [12, 14]. From a design perspective, it is neither feasible, nor desirable, to lay down hard and fast rules for the organisation or the placement of the various elements needed on a title page.

Apart from any design considerations, there are also some good practical reasons to make the title pages as a separate document. A great deal is gained by treating title pages as an integral part of the endpapers. When binding (good) books, endpapers fulfil both a functional and an aesthetic role [10]. The ‘W’, or ‘zig-zag’ endpaper in particular is a most useful and efficient kind of endpaper, and one which benefits considerably from the insertion of an extra folded sheet. This is the very place to put a title page sequence.

Ideally, a separate folio should be made, bearing on the first recto page the bastard title, on the verso possibly a frontispiece, or failing something suitable, left blank. The next recto bears title, author, publisher and the like, and the final verso provides space for the biblio or imprint. The equivalent should be done for the endpaper at the back of the book. It provides a good opportunity of bringing the colophon back into use: a folio bearing the colophon ought to be inserted in the back endpaper. By producing these pages separate from the main document, there is no interference with the page numbering, nor does it upset the way in which the signatures are assembled.

4 Implementation

4.1 Class options

The options available in Octavo are largely the same as those of the standard book class. However, their content and default settings are often different.

page size The predefined page sizes are, from smallest to largest: foolscap, crown, post, large-post, demy, medium, royal, superroyal and imperial. The default size is crown.

fount size The size options are the standard 10pt, 11pt and 12pt, with the default size set to 10pt.

The choice of a fount size loads parameters and measures which depend on this size from one of the three class option files.

final/draft Similar to the book class: draft puts black marks where lines are overfull on the copy. The default is final.

(no)titlepage Contrary to the book class, the production of a title page is not encouraged: the default is therefore notitlepage. If you do wish to have a titlepage, you still have to issue the \maketitle command in your text.

open(right/any) New chapters normally start on a recto page, but some kinds of books, e.g., novels, may well be better off with chapters starting on either side. The default is openright.

(one/two)column The option to set text in two column is possibly useful in conjunction with the two or three largest page sizes (imperial, superroyal and royal). The default is onecolumn.

leqno,fleqno These two options deal with the setting of mathematical formulæ. leqno sets the equation number to the left of the formula, instead of the default right position. fleqno redefines the math display environment, setting equations flush with the left margin and indented by a \mathindent amount.

4.2 Printing grid

An attempt is made to maintain a uniform printing grid across the text. This means that ways and means had to be found to counteract the behaviour of some of the ingenious TeX algorithms. Rather than attempt what would amount to a rewriting of part of the TeX code, which really is too daunting a task, it seems possible to subvert the TeX algorithms and make it do our bidding after all with a judicious setting of parameters, lengths and sizes.

The first step taken is to set the textheight to an exact, natural number of \baselineskip. Next, \parskip is set to zero with no ‘glue’ extension, so that TeX will not be tempted to put extra leading between any paragraphs.

Dealing with the space surrounding headings in the text is somewhat more involved. The solution adopted is to use where at all possible multiples (and fractions) of the \baselineskip, and allow for some adjustments by specifying *negative* glue only. Inclusion of stretchable, positive glue may prompt TeX to stretch beyond the measure stated (under protest, but it will happen nevertheless), while TeX

will never shrink glue beyond the measure given. A similar trick is applied to the space surrounding lists. In contrast, the definition of space surrounding floats does contain a modicum of stretchable glue, with the hope that TEX will fill in with a bit of white space around the float but leave the grid of printed text undisturbed.

Nevertheless, there will be times when the actual text, the parameters defined by Octavo, and TEX conspire to produce truly horrible results. Such misfortunes call for direct intervention, or, the user willing, suggestions of improving the definitions of the parameters proposed here for the Octavo class.

4.3 Quiet, unobtrusive design

Octavo yields quiet, gentle-looking texts thanks to a number of design choices. One highly effective decision was to avoid the use of bold founts altogether: none of the display uses it at all. Headers are set above the text in small capitals, following a suggestion by Williamson: it is both effective and discreet.

The headings used by chapters and sections are kept low-key as well, and are differentiated through the surrounding white space and their position away from the margin. Section headings are centered and set in small capitals, using the \Large fount size. Subsection headings are placed 1em from the left margin and set in normal size small capitals. Sub-subsections are likewise placed 1em away from the left margin, but have less white space above and below and are set in normal size italics.

An example of the layout produced by Octavo is probably more eloquent than this verbal description. Figure 3 is a reduced copy of a page spread made with Octavo, showing a chapter opening and various section titles, as well as the relation of the type block to the pages.

Numbering is either simplified or has been done away with altogether. None of the headings bear numbers, as Octavo sets `\secnumdepth` to -2 . This can be reset in the preamble of the text, if numbered chapters and sections are absolutely required. Figures and tables are numbered consecutively throughout the book. The excellent capabilities of `TEX` to make and maintain cross-references leaves little in favour of maintaining a chapter.number.figurenumber type of scheme. With some trepidation, this numbering system is also applied to equations. There seems to be a substantial tradition of having a numbering system for equations which incorporates chapter and section numbers. As far as I can see, there is no particular need for such a system, once again bearing in mind the powerful cross-referencing capabilities of `TEX`, and it would

COLLECTING

wherever water is found. The necessary water spans expanses no more than the thin film of raindrops clinging to some mosses on a wall, over a puddle alongside a grassy bank, to a large lake or an ocean.

PICKING UP

The simplest way of collecting haplessly protist-containing samples is by picking up some likely substrate. Little is needed beyond a knife, a spoon and some containers.

TERRESTRIAL PROTISTS

Moss cushions can be picked up from soil or cased off stones with a knife, leaves decaying in puddles picked up, and a spoon will assist in scooping up some surface mud from the bottom of a pool. The feltly scum covering stems and leaves of submerged water plants can be shaken into a container, or recovered by stripping or cutting lots off.

Dry land

The terrestrial environment, with its for protists highly complex structure and often intense environmental pressures, deserves more attention as a

Before setting out to collect, it is well to pause first and reflect where protists prefer to lead their lives. We are not likely to be any longer deceived to believe that home is where the heart is. Much have gone before us, and much has been learned about the favourite haunts of our organisms. Happily, thanks to the accumulation of records of occurrences we can discuss a number of recurrent themes, such as the general situations which help and guide us in our searches, especially when we leave the already well-trodden paths.

As is the case for any organism, protists require water to live. Because they are single cells, with just a single plasma membrane to shield them from the outside world, they are very sensitive to any kind of dehydration. And so the first generalisation is arrived at: protists are to be found

Figure 3: Example of Octavo output

also be inconsistent with the numbering system used for tables and figures.

A final, minor point is that tables of contents do not display the ugliness of dotted lines separating entries and page numbers. However, these can be brought back if required, by resetting \dotsep to a smaller number.

5 Some hints on the making of books

Thanks to excellent works on bookbinding such as the book by Lawrence Town [10], the physical making of books lies well within reach of anyone caring to make the effort. With the availability of a number of additional programs from CTAN, it is a straightforward matter to organise and print out a text in signatures. Here are some of the means which have proven both useful and effective.

The four smallest paper sizes defined by Octavo, i.e., foolscap, crown, post and largepost, can be made to fit in pairs on an A4 sheet. Hence it is within reach of anyone with a run-of-the-mill printer to produce books in these formats. All that is required is a way in which to organise the pages into a sequence so that signatures can be made. This requires a few careful but simple adjustments.

The problem lies in the imposition of the pages on an A4 sheet. The four formats mentioned are all narrower than an A5 sheet, and hence extra space is left at the outer margins. This is not a problem with the odd-numbered pages, as their inner margin is made to coincide with the inner margin of the A5 sheet. However, even-numbered pages have their outer margin made to coincide with the outer margin of the A5 sheet, the latter of course being too wide. What is required is for the inner margins of the sheets to coincide. The simplest solution is to add the single line to the preamble of the document:

```
\addtolength{\evensidemargin}{\offset}
```

(where `offset` is $148.5 \text{ mm} - \text{\textwidth}$ with A4 paper, and $139.7 \text{ mm} - \text{\textwidth}$ with letter paper) rather than try to reposition the even-numbered pages with `dvidvi`, `dvips`, and the like. Once this is done, the production of signatures is straightforward.

Three steps are needed when using `dvips`:

1. `dvips -ta5 <file>`
2. `psbook -s16 <file>.ps <file>.tmp`
3. `psnup -pa4 -Pa5 -l -2 -s1.0 <file>.tmp <file>.ps`

while `dvihp1j` or similar drivers can be instructed by setting a number of switches, e.g.,

```
prthpljh texfile +columns:2 +rows:1
+section-size:4 +full-last-section
+page-width:148.5mm +page-height:210mm
+landscape-mode +double-sided:odd[even]
```

It may be necessary to instruct the printer to shift each page a little on the sheet to ensure perfect register, and a little experimenting is thus called for. These sets of commands and switches yield signatures made up of 16 pages on 4 sheets, just as one would expect from traditional printing on sheets which would be folded into an octavo format. Such an arrangement is very effective and handy when binding the book.

When sewing these signatures in library style, and particularly when opting for a hollow back, the amount of paper that tends to ‘disappear’ into the binding seldom exceeds 1 mm. It is therefore unnecessary to worry about shifting the pages varying amounts in function of their place in the signature.

6 Availability

Octavo is available from a CTAN near you, in the directory `/macros/latex/contrib/octavo`, under the L^AT_EX Project Public License. The distribution consists of `.dtx` and `.ins` files, which will generate the `octavo.cls`, `oct10.clo`, `oct11.clo` and `oct12.clo` files.

7 Acknowledgements

Octavo is a modification of `classes.dtx` written by Leslie Lamport (1992), Frank Mittelbach (1994–97) and Johannes Braams (1994–97). My own contribution is restricted to tweaking of some parameters and true credit is due to Lamport, Mittelbach and Braams for their monumental efforts.

References

- [1] H. Kayser. *Ein harmonikaler Teilungskanon.* Occident Verlag, Zürich, 1946.
- [2] M. Klein, Y. Schwemer-Scheddin, and E. Spiekermann. *Type & Typographers.* SDU Uitgeverij, 's Gravenhage, 1991.
- [3] D. Knuth. *The T_EXbook.* Addison-Wesley, Reading, 1984.
- [4] D. Knuth. *Digital Typography.* CSLI Publications, Stanford, 1998.
- [5] E. J. Labarre. *Dictionary and Encyclopedia of Paper and Paper-making.* Swets & Zeitlinger, Amsterdam, second edition, 1969.
- [6] R. McLean. *The Thames and Hudson Manual of Typography.* Thames and Hudson, London, 1997.
- [7] F. Neukam, M. Kohm, and A. Kielhorn. Das KOMA-script paket. CTAN, `/macros/latex/contrib/koma-script`, 2002.
- [8] R. Rosario. *Divina proportio typographica.* Scherpe, Krefeld, 1961.
- [9] P. Taylor. Book design for T_EX users, Part 1: Theory. *TUGboat*, 19:65–74, 1998.
- [10] L. Town. *Bookbinding by hand, for students and craftsmen.* Faber & Faber, London, 1951.
- [11] J. Tschichold. Die maßverhältnisse der Buchseite, des Schriftfeldes und der Ränder. *Schweizer Graphische Mitteilungen*, 65:294–305, 1946.
- [12] J. Tschichold. *Ausgewählte Aufsätze über Fragen der Gestalt des Buches und der Typographie.* Birkhäuser Verlag, Basel, 1987.
- [13] J. A. Van de Graaf. Nieuwe berekening voor de vormgeving. *Tété*, 1946:95–100, 1946.
- [14] H. Williamson. *Methods of book design.* Oxford University Press, Oxford, 1966.
- [15] P. Wilson. The memoir class for configurable typesetting. CTAN, `/macros/latex/contrib/memoir`, 2001.

◇ Stefan A. Revets
Thijssenlaan 22
B1780 Wemmel
Belgium
s.revets@tiscali.be

Philology

The **teubner** L^AT_EX package: Typesetting classical Greek philology

Claudio Beccari

Abstract

The **teubner** package provides support for typesetting classical Greek philological texts with L^AT_EX, including textual and rhythmic verse. The special signs and glyphs made available by this package may also be useful for typesetting philological texts with other alphabets.

1 Introduction

In this paper a relatively large package is described that allows the setting into type of philological texts, particularly those written about Greek literature or poetry. Such texts require a large number of special symbols, including the possibility of setting diacritical marks on every letter.

The first section explains my connection to the Greek alphabets and my involvement with this particular project. I then explain why and how I decided to design a new Greek font, inspired by the typefaces used by the printing company Teubner. The name of the package was chosen to honour this small printing company that is so praised by philologists.

We then delve into the package functionality by showing and briefly describing the plethora of diacritical marks and special accents that philologists need.

These accents are not the only signs philologists require; they also need special marks for indicating their interpolations or elisions from the cited text specimens, for marking special pronunciations or special alphabetic glyphs that are not present in the standard alphabets.

Next, the environments for setting poetry are examined; not only those for setting specific verses with numbered lines, but also the *metrae*, verse in which the syllables are substituted by their value for the rhythmic and melodic declamation of poetry.

I conclude with some personal remarks about the fact that the current package is a starting point, not a finished work; feedback from philologists and others will inspire further evolution of **teubner**.

2 Background

My connection to Greek fonts and Greek typesetting dates back several years, when a friend of mine, a

professor of classical Greek in a nearby classical high school, was complaining that she could not typeset her class tests in Greek, as she could do in Latin. I stated that with L^AT_EX she should not have any difficulty, but when I started searching on CTAN, I could not find anything really suitable for her. At that time I found only the excellent Greek fonts designed by Silvio Levy [1] in 1987 but for a variety of reasons I did not find them satisfactory for the New Font Selection Scheme that had been introduced in L^AT_EX in 1994.

Thus, starting from Levy's fonts, I designed many other different families, series, and shapes, and added new glyphs. This eventually resulted in my CB Greek fonts that now have been available on CTAN for some years. Many Greek users and scholars began to use them, giving me valuable feedback regarding corrections some shapes, and, even more important, making them more useful for the community of people who typeset in Greek—both in Greece and in other countries.

Then, one day I received an e-mail from a young man, Paolo Ciacchi, from Trieste, Italy, who was about to start work on his master's thesis in classical Greek philology, and needed some extensions to the available fonts and to the macros provided by the *babel* option for typesetting Greek with multi-accent spelling (i.e., with the *polutoniko* *babel* language attribute when the *greek* language option is selected).

Over time, his requests became quite extensive:

- a different typeface for purposes of emphasizing quoted phrases or sentences;
- a multitude of commands for setting the specific and quite specialized philological marks in the critical text;
- these marks, where applicable, had to be usable with both the Latin and the Greek alphabets;
- special signs for typesetting the *metrae*, the sequences of long and short marks that give the rhythm of Greek and Latin verses;
- environments for typesetting poems with verse numbering—even double numbering;
- new symbols for representing ancient monetary and/or measure entities.

Ciacchi would also have liked to have the facilities offered by the package *edmac*, by John Lavagnino and Dominik Wujastyk [2]. Unfortunately, *edmac* is based on the Plain T_EX format and I was not able to transfer them to the new L^AT_EX 2 _{ε} package I was writing for Ciacchi. Aside from my inexperience at tinkering with the output macros of the L^AT_EX format, I don't think it is a good idea to fiddle with these internal macros, unless one likes

Ἐν ἀρχῇ ἦν ὁ Λόγος, καὶ ὁ Λόγος ἦν πρὸς τὸν Θεόν, καὶ Θεός ἦν ὁ Λόγος. οὗτος ἦν ἐν ἀρχῇ πρὸς Θεόν. πάντα δι' αὐτοῦ ἐγένετο, καὶ χωρὶς αὐτοῦ ἐγένετο οὐδὲ ἐν ὅ γέγονεν. ἐν αὐτῷ ζωὴ ἦν, καὶ ἡ ζωὴ ἦν τὸ φῶς τῶν ἀντρώπων. καὶ τὸ φῶς ἐν τῇ σκοτίᾳ φαίνει, καὶ ἡ σκοτίᾳ αὐτὸν οὐ κατέλαβεν.

Figure 1: Sample text set with the Olga-like Greek typeface

to get into trouble and, if he's lucky, end up with something that is working with the current version of L^AT_EX—but may very well be incompatible with the next version. In my opinion, this sort of operation can be carried out only by the L^AT_EX team.

I must add that recently the package `poemscol` by John Burt [3] was made available to the T_EX community. This package is dedicated to printing anthologies of selected poems with adequate comments and retrieval information of the sources. I find it is a versatile instrument in the hands of a scholar, suitable for publishing well designed poetry books. This package has some of the philological functionality that is missing from `teubner`, but it deals only with poetry with numbered verses and with end-notes; in other words, `poemscol` can set end-notes whose numbers correspond to the verse numbers. This could be implemented also in `teubner` but I have preferred to avoid end-notes, since I consider it unwieldy to look up the notes in a different place from the current page.

`poemscol` provides the structures required to produce a critical edition of the kind specified by the Modern Language Association's Committee on Scholarly Editions, whereas these are not contained in `teubner`; the latter is primarily dedicated to new symbols and to low-level commands for formatting philological text, in particular when the Greek alphabets are used.

3 The new typeface for emphasis

The standard “italic” Greek typeface used with `babel` when the CB fonts are used is an imitation of the rather modern Olga typeface introduced by the Greek Font Society and fully described in the invaluable book *Greek Letters — From Tablets to Pixels* [4]. A sample text composed with such a font appears in figure 1.

However, classical philologists are rather conservative when it comes to typesetting, by and large, and like to see their works typeset in the traditional fonts that have always been used by the best publishing houses for this purpose. In this regard, I discovered that the most authoritative publishing

Ἐν ἀρχῇ ἦν ὁ Λόγος, καὶ ὁ Λόγος ἦν πρὸς τὸν Θεόν, καὶ Θεός ἦν ὁ Λόγος. οὗτος ἦν ἐν ἀρχῇ πρὸς Θεόν. πάντα δι' αὐτοῦ ἐγένετο, καὶ χωρὶς αὐτοῦ ἐγένετο οὐδὲ ἐν ὅ γέγονεν. ἐν αὐτῷ ζωὴ ἦν, καὶ ἡ ζωὴ ἦν τὸ φῶς τῶν ἀντρώπων. καὶ τὸ φῶς ἐν τῇ σκοτίᾳ φαίνει, καὶ ἡ σκοτίᾳ αὐτὸν οὐ κατέλαβεν.

Figure 2: The same sample text, set with the Lipsiakos Greek typeface

house is the B.G. Teubner Verlagsgesellschaft of Lipsia. The Greek typeface used by this publishing house is very elegant and is highly praised by readers; see, for example, [5]. In [4], the Teubner font is thoroughly described, including explanations as to why it is so widely used in scholarly works. This font became so widespread that it received in Greece the definitive name “Lipsiakos”.

Paolo Ciacchi eventually persuaded me to adopt the Lipsiakos as the default typeface for emphasis with `teubner`. As it happened, I had already started remaking the Lipsiakos typeface with METAFONT. By the end of this long process, I had also received the assistance of Dimitrios Filippou who gave me invaluable suggestions for second order corrections of each glyph. Eventually, the typeface design ended up in a “final” version available on CTAN in both METAFONT and PostScript Type 1 format. The sample text of figure 1 reset with the Lipsiakos typeface is shown in figure 2.

The `teubner` package automatically substitutes the default “italic” Olga-like Greek typeface with the more traditional Lipsiakos one, and defines new commands and environments that extend the `babel` options available for typesetting Greek. Specifically, it provides the commands `\textDidot` for typesetting its argument in the “regular” upright Didot Greek typeface, and `\textLipsias` for typesetting its argument with the Lipsiakos typeface, and it redefines `\textlatin` for inserting a phrase in Latin characters within a text typeset in Greek.

All of these commands (and the declarations `\Lipsiakostext` and `\NoLipsiakostext`), as well as selecting a particular font, change the current font encoding and the default language so that when their action is finished the previous default encoding and language are re-established. In this way it is straightforward for scholars to write an article mainly in, say, German, while inserting short (or long!) text samples in Greek within the work.

4 Accents and other diacritical marks

Philologists need to stack several accents and/or diacritical marks over or under alphabetic signs. The

Table 1: Accent macros

Example	Syntax	Example	Syntax
à	\`{{<letter>}}	ø	\ut{{<letters>}}
á	\'{<letter>}	å	\Ab{{<letter>}}
ã	\~{{<letter>}} ¹	ç	\Gb{{<letter>}}
í	\"{{<letter>}}	ڻ	\Arb{{<letter>}}
ڦ	\u{{<letter>}}	ڙ	\Grb{{<letter>}}
ڦ	\U{{<diphthong>}}	ڙ	\Asb{{<letter>}}
ڦ	\= {{<letter>}}	ڦ	\Gsb{{<letter>}}
ڦ	\r{{<letter>}}	ڦ	\Am{{<letter>}}
ڦ	\s{{<letter>}}	ڦ	\Gm{{<letter>}}
ڦ	\Ad{{<letter>}}	ڦ	\Cm{{<letter>}}
ڦ	\Gd{{<letter>}}	ڦ	\Arm{{<letter>}}
ڦ	\Cd{{<letter>}}	ڦ	\Grm{{<letter>}}
ڦ	\Ar{{<letter>}}	ڦ	\Crm{{<letter>}}
ڦ	\Gr{{<letter>}}	ڦ	\Asm{{<letter>}}
ڦ	\Cr{{<letter>}}	ڦ	\Gsm{{<letter>}}
ڦ	\As{{<letter>}}	ڦ	\Csm{{<letter>}}
ڦ	\Gs{{<letter>}}	ڦ	\Sm{{<letter>}}
ڦ	\Cs{{<letter>}}	ڦ	\Rm{{<letter>}}
ڦ	\c{{<letter>}}	ڦ	\iS{{<letter>}}
ڦ	\semiv{{<letter>}} ²	ڦ	\d{{<letter>}}
a	\ring{{<letter>}} ²	ڦ	\bd{{<letter>}}

¹ The circumflex accent may be obtained with `\~` only if the attribute `polotoniko` is specified for the Greek language with `babel v.3.7`.

² Most commands may be used also with Latin letters.

most obvious instance is when they have to typeset poems (or even prose) inserting rhythmic accents and melodic accents simultaneously; for example, a grave accent over a macron. In other situations, it is better (and recommended) to address the specific accented glyphs without resorting to the ligature mechanism embedded into the CB Greek fonts. We will discuss this later.

Thus, teubner defines stacking accent macros that take into account the font slant in skewing the accent stack. Also, end user macros are available so as to allow the typesetter to put multiple accents over any glyph. Philologists must also be able to put accents over consonants, so even simple accents have their own macros that accept any symbol as the alphabetic symbol to mark. It is therefore not particularly difficult to typeset \ddot{a} (`\Ab{a}`) or \ddot{w} (`\Gm{w}`) or even \dot{s} (`\'{s}`).

The sample text displayed in figure 2 has a glitch: compare the words *avτροῦ* and *avτω̄*.¹ They display an evident difference in the initial diphthong; for the first word, the marked epsilon has been ob-

¹ This is the situation referred to earlier where it is best to access accents directly. The glitch in the composition of the sample text was intentionally not corrected in order to show this difference.

Table 2: Accented vowel macros

\aa	á	\ag	á	\ac	ã
\asa	ã	\asg	ã	\asc	ã
\ara	ã	\arg	ã	\arc	ã
\arai	ä	\argi	ä	\arci	ä
\ai	ä	\ar	å	\as	å
\asi	ä	\aai	å		
\ari	ä	\agi	å	\aci	ä
\asai	ä	\asgi	ä	\asci	ä
\ha	ή	\hg	ή	\hc	η̄
\hsa	ῆ	\hsg	ῆ	\hsc	ῆ
\hra	ῆ	\hrg	ῆ	\hrc	ῆ
\hrai	ῆ	\hrgi	ῆ	\hrci	ῆ
\hi	η̄	\hr	η̄	\hs	η̄
\hsi	η̄	\hai	η̄		
\hri	η̄	\hgi	η̄	\hci	η̄
\hsai	η̄	\hsgi	η̄	\hsci	η̄
\wa	ѡ	\wg	ѡ	\wc	օ
\wsa	ա	\wsg	ա	\wsc	ա
\wra	ա	\wrg	ա	\wrc	ա
\wrai	պ	\wrgi	պ	\wrci	պ
\wi	պ	\wr	պ	\ws	պ
\wsi	պ	\wai	պ		
\wri	պ	\wgi	պ	\wci	պ
\wsai	պ	\wsgi	պ	\wsci	պ
\ia	ի	\ig	ի	\ic	ի
\isa	ի	\isg	ի	\isc	ի
\ira	ի	\irg	ի	\irc	ի
\ir	ի	\is	ի	\id	ի
\ida	ի	\idg	ի	\idc	ի
\ua	ւ	\ug	ւ	\uc	ւ
\usa	վ	\usg	վ	\usc	վ
\ura	վ	\urg	վ	\urc	վ
\ur	վ	\us	վ	\ud	վ
\uda	վ	\udg	վ	\fdc	վ
\ea	է	\eg	է	\er	է
\esa	է	\esg	է	\era	է
\es	է	\erg	է		
\oa	օ	\og	օ	\oR ¹	օ
\osa	օ	\osg	օ	\ora	օ
\os	օ	\org	օ		

¹ For homogeneity with the other command definitions the command `\oR` should be spelled `\or`, but then it would produce incompatibilities due to the TeX primitive of the same name.

tained with the accent macro \s{u} (the mnemonic ‘s’ stands for ‘smooth breath’), while for the second word the ligature mechanism was invoked by writing simply >u. With other typefaces you hardly notice the difference, but with the Lipsiakos face the shape of the epsilon makes it very evident that the kerning between the initial alpha and the marked epsilon in the second case is not effective.

This is a drawback of the ligature mechanism, which is otherwise so useful for keying in text: \TeX metric files only support kerning and ligature information for two characters at a time. Therefore, some information is lost when you key in a>u , because it involves three characters, directly processed by \TeX . On the other hand, no information is lost when you key in $\text{a\|s\{u\}}$ because, thanks to the powerful $\text{\LaTeX} 2\epsilon$ commands, $\text{\s\{u\}}$ is mapped directly to the glyph that represents \ddot{v} , without resorting to the ligature $\text{'v}\Rightarrow\ddot{v}$. Of course these refinements can be thought of as second order refinements so as to obtain the very best typesetting. For speeding up the keying process in this particular case, there is also available the command \us , but of course then you must type a\|us_t\~w| in order to obtain $\ddot{a}\ddot{v}\tau\ddot{w}$.

Accent commands and accented vowel macros are collected in tables 1 and 2. The mnemonics are quite transparent; in any case, the package documentation accompanying teubner gives more details. I would like to stress that these accent and accented vowel macros are for fine-tuning the final typesetting; the typesetter should not trouble to use them before reaching the final step, and then only if some irregularities are found in the letter kerning of specific words.

5 Philological signs

It would take us too far afield to describe here each and every command related to the special philological signs. In general, let us recall that philologists insert such signs when they want to stress the point that some letters in the transcription of an ancient manuscript, papyrus, tablet, or the like, has been replaced or interpolated by the scholar, or perhaps that the original spelling is thought to be wrong and the correction is specially marked. All these instances received technical names and the commands to produce them correspond to their full names or are abbreviations; most are Latin names, but sometimes it's difficult to distinguish a real Greek or Latin name from a Latinized version of a Greek name. In any case, philologists know very well their meaning and can distinguish very easily which command to use for producing which sign.

Table 3 contains most, though not all, of the specific commands defined in teubner . Some of them have been already described, others are used for inserting the philological signs, and still others are used for defining special signs used in philology.

In table 4, we have the commands and the ligatures that provide access to all the special symbols that are included among the 256 glyphs of every CB Greek font. Some of these symbols are accessible

Table 4: Greek and general symbols

„	,	((<<))	»
\GEodq	„	\GEcdq	“	:	:
\GEoq	,	\GEcq	‘	?	;
\ENodq	“	\ENcdq	”	;	.
\stigma	ς	\varstigma	ς	\Stigma	Ͳ
\coppa	ϙ	\koppa	ϙ	\Coppa	ϙ
\sampi	ϗ	\Sampi	ϗ	\permill	%
\digamma	Ϝ	\Digamma	Ϝ	\euro	€
\f	F	\F	F	\shwa	ə

through babel commands, so there is no need to resort to the teubner extension package. Nevertheless, it is useful to have them collected in a single table, so as to know the differences between variants. Some of these glyphs are used by the babel commands \greeknumeral and \Greeknumeral : the teubner package redefines both commands in order to represent the value '6' with the digamma sign (which is traditional with philologists), and it defines new variant commands that use the stigma sign for the value '6' (which is more common in ecclesiastical writings). The result can be seen in the following examples:

if you type $\text{\greeknumeral\{1996\}}$ you get $\alpha\gamma\Omega\Gamma$
 if you type $\text{\Greeknumeral\{1996\}}$ you get $\text{\AA}\Omega\Gamma$
 if you type $\text{\greeknumeral*\{1996\}}$ you get $\alpha\gamma\Omega\gamma$
 if you type $\text{\Greeknumeral*\{1996\}}$ you get $\text{\AA}\Omega\Omega\Gamma$

6 Environments for poetry

teubner defines two kinds of environments for typesetting poetry: the first deals with actual verse and has three variants; the second deals with *metrae*, which are the metric patterns describing rhythms of verses and/or stanzas.

6.1 Environments for verse

The first verse environment, with the Latin name of *versi*, typesets verses in block mode, with a separator between each verse. Over the separator is a sequence number. The left margin of the block may be set with a descriptor, generally the overall reference to the group of verses. The result is as follows:

Mer. fr. 4 ἔνθ' ὁ μέν ε[ἰσπλη] ⁶⁸ θὼν Μερόπων κίεν. ἦ
 [δέ δια] ⁶⁹ πρὸ αἰχεμῆι σιθῆτος [ἔλασσεν.]
⁷⁰ ὁ δ' ἐξέχνει· οὐ γάρ [όμοῖαι][ἄ] ⁷¹ θάναται
 θνηταῖσι βολ[αι κατά] ⁷² γαῖαν ἄσιν. πρη[μ] ⁷³ η-
 ζ δ[ο]... ⁷³ τησε. μέλας δὲ περιε[...] ⁷⁴ ρω

Table 3: Extended commands in **teubner**

Example	Syntax	Example	Syntax
<i>Bαχύλιδες</i>	\textLipsias{\text{}}	{ $a\beta\gamma$ }	\lesp{\text{}}
<i>Bαχύλιδες</i>	\textDidot{\text{}}	•	\LitNil
text	\textlatin{\text{}}	\hat{g}	\cap{\text{}}
(<i>Bαχύλιδες</i>)	\frapar{\text{}})—	\Coronis
(\lpar	—	\lmqi
)	\rpar	—	\rmqi
(?)	\qmark	[$a\beta\gamma$]—	\mqi{\text{}}
...	\Dots[<i>number</i>]	—	\lmqs
...	\DOTS[<i>number</i>]	—	\rmqs
— — —	\Dashes[<i>number</i>]	[$a\beta\gamma$]—	\mqs{\text{}}
— — —	\DASHES[<i>number</i>]	[$a\beta\gamma$]	\zeugma{\text{}}
foo	\ap{\text{}}	$\underline{a\beta\gamma}$	\siniz{\text{}}
◊	\sinafia	—	\paragr
:	\colon	—	\dparagr
;	\;:	\otimes	\FinisCarmen
?	\?:	†	\crux
::	\antilabe	' $a\beta\gamma'$	\apici{\text{}}
	\	'	\apex
	\dBar	~	\responsio
	\tBar	\int	\Int
[\lbrk	* a	\star
]	\rbrk	** a	\dstar
[$a\beta\gamma$]	\ladd{\text{}}	*** a	\tstar
[$a\beta\gamma$][$a\beta\gamma$]	\lladd{\text{}}		\,,
$\langle a\beta\gamma \rangle$	\Ladd{\text{}}		\!
$\langle\langle a\beta\gamma \rangle\rangle$	\LLadd{\text{}}	0123456789	\OSN{\text{}}
$\overline{a\beta\gamma}$	\nexus{\text{}}	$\hat{a\beta\gamma}$	\nesso{\text{}}
<u>AB</u>	\Utie{2 letters}	$ah\beta$	\h
$a\beta$	\yod	$a\beta$	\shwa
$a\beta$	\q	AFB	\F
$a\beta$	\f	\dot{a}	\semiv{\text{}}
h^v	\skewstack{\text{}}{\text{}}	\bar{e}	\md{\text{}}
\check{e}	\Ud{\text{}}	\acute{e}	\mO{\text{}}
\check{e}	\UO{\text{}}	\ddot{e}	\Open{\text{}}
\check{e}	\nasal{\text{}}	\ddot{d}	\cut{b d g}
†	\dracma	＊	\denarius
§	\stater	—	\etos
c	\hemiobelion	◦	\tetartemorion
‡	\splus	§	\stimes
k	\kclick		

The second environment, **Versi**, sets verses in display and separates stanzas with a blank line, just as the standard **verse** L^AT_EX environment. The primary change is that **Versi** labels the verses with a numeric sequence which is printed every fifth element. The result is like this:

45 $\tau\alpha\cdot \pi\varrho\sigma\theta\epsilon \chi\epsilon\varrho\omega\nu \beta\acute{\iota}\alpha\nu$
 $\delta\epsilon[\iota]\xi\mu\epsilon\nu \tau\acute{\alpha} \delta' \acute{\epsilon}\pi\acute{\iota}\nu\tau\alpha \delta\alpha[\acute{\iota}\mu\omega]\nu \sigma\acute{\iota}\nu\epsilon\acute{\iota}\nu\cdots$
 $\tau\acute{\alpha}\sigma' \acute{\epsilon}\acute{\iota}\nu\epsilon\nu \acute{\alpha}\acute{\varrho}\acute{\epsilon}\tau\alpha\chi\mu\omega\acute{\varsigma} \acute{\eta}\omega\varsigma\cdots$

τ]<άφον δέ ναυβάται
φ]<ωτός ὑπεράφανον
50 θ]<άρσος· Ἀλίου τε γαμβρῶι χόλωσεν ἥτορ

The third environment, **VERSI**, is similar to **Versi** with the addition that verses can be numbered according to two sequences: the outer one displays every fifth element, as before, while the inner one may be turned on and off, but when it is on displays every element. This composition looks like this:

40 κέλομαι πολύστονον
 18 ἐρύκεν ὑβ' ζων· οὐ γάρ ἀν θέλοι·
 μ' ἄμβροτον ἐρανὸν Αο[ῦς]
 20 ἰδεῖν φάος, ἐπεί τιν' ἡθέ[ων]
 21 σὺ δαμάσειας δέκον·
 45 τα· πρόσθε χειρῶν βίαν
 23 δε[ῆ]ξομεν· τὰ δ' ἐπιόντα δα[ίμων]ν κρυνεῖ.“
 1 τόσ' εἴπεν ἀρέταιχμος ἥρως·
 2 τάφον δὲ ναύθαται
 3 φωτὸς ὑπεράφανον
 50 θάρσος·....

See the *teubner* package documentation for details on the input syntax.

6.2 Environment and commands for *metrae*

Typesetting *metrae* requires both an alphabet—a special font with the necessary symbols—and a series of commands and environments. Table 5 shows most of the symbols that can be used within the metric commands and environments.

Most commands define a single ‘note’, but a few define frequently used rhythmic sequences and the rhythms of common verses. Most important of all, *teubner* provides a defining command `\newmetrics`, similar to `\newcommand`, for defining new metric sequences. (More details are given in the documentation of the *teubner* package, as usual.) The important point is thus not to memorize the multitude of metric commands (although their names are either full names or abbreviations that are commonly used in philological texts), but rather to remember that with the help of `\newmetrics`, one can encapsulate frequently used sequences in a particular verse, so as to save a substantial amount of input keying.

Metrae can generally be typeset within the standard L^AT_EX `verse` environment, but sometimes it is necessary to use special constructs. In particular, a new environment `bracedmetrics` has been defined in order to typeset *metrae* well aligned and grouped together. With the help of this environment, it is not difficult to align various rhythms and to group with vertical braces some rhythmic variants that may appear in particular stanzas. For example, the following *metrae* can be obtained with a bit of work and attention:

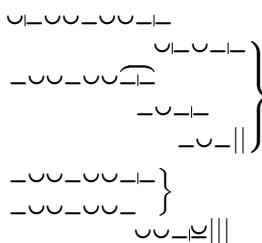


Table 5: Metric symbols

Command	Metric symbol
<code>\longa</code>	—
<code>\bbrevis</code>	∞
<code>\ubarbrevis</code>	∞
<code>\ubarsbrevis</code>	∞∞
<code>\corona</code>	∞
<code>\catal</code>	^
<code>\anceps</code>	×
<code>\ancepsdbrevis</code>	☒
<code>\brevis</code>	◦
<code>\barbrevis</code>	□
<code>\ubarbbrevis</code>	∞□
<code>\coronainv</code>	◦
<code>\ElemInd</code>	∞
<code>\iperccatal</code>	+
<code>\ubrevislonga</code>	□
<code>\banceps</code>	☒
<code>\hiatus</code>	H
<code>\iam</code>	□_◦_
<code>\enopl</code>	◦_◦_◦_◦_
<code>\aeolchorsor</code>	_∞_◦_∞_
<code>\2tr</code>	_◦_x_ _◦_x_
<code>\chor</code>	_◦_◦_
<code>\4MACRO</code>	----
<code>\hexam</code>	◦_◦_◦_◦_◦_◦_◦_◦_
<code>\pentam</code>	◦_∞_◦_◦_◦_ _◦_◦_◦_
<code>\aeolicbii</code>	oo
<code>\aeolicbiii</code>	ooo
<code>\aeolicbiv</code>	oooo

7 Conclusion

The *teubner* package is a complex project whose ultimate usefulness may be evaluated only by the philologists. In my own opinion, some highly useful features are still missing; in particular, integration with the *edmac* Plain T_EX package. As I explained in the introduction, I believe that the particular functionality for setting footnotes in continuous lines with the references tied to the line number of the critical text must be implemented by the L^AT_EX team only; otherwise, sooner or later incompatibilities would certainly show up.

I hope that the special arrangements for switching language and alphabet provided by *teubner*, together with its wealth of special symbols, will be appreciated by scholars who can now typeset their works in their mother tongue or another “language of the trade”, while inserting specimens of philological material written in another language with another alphabet.

As for the future of the `teubner` package, as long as I am able I intend to implement whatever additional features I can, and certainly try to correct any errors found by the users. At this writing, I have had feedback from half a dozen philologists and the actual version 2 of this package is the result of their constructive criticism.

8 Acknowledgements

A deep thank you goes to Paolo Ciacchi who convinced me to work on this project, and helped me very much through his education in classical Greek philology. He patiently taught me the details of the various special signs and their usage. He eventually used them in his master thesis, for which he received his degree with honors (*magna cum laude*). His excellent result was of course primarily due to his special education and the high quality of the contents of his thesis, but I hope that the professional quality of his philological typesetting contributed to the appreciation of his thesis.

Another grateful thank goes to Dimitrios Filippou for his invaluable assistance in helping me to fine tune the Lipsiakos font.

I also want to thank the various philologists who dared to experiment with `teubner` and constructively suggested corrections and additions.

References

- [1] Levy S., *Silvio Levy's Greek fonts*. CTAN, for example in `/tex-archive/macros/latex/contrib/supported/levy`.
- [2] Lavagnino J. and Wujastyk D., "Critical Edition Typesetting: The EDMAC format for Plain T_EX". CTAN, for example in `/tex-archive/macros/plain/contrib/edmac`.
- [3] Burt J., "Typesetting critical editions of poetry" in *TUGboat*, v. 22 (2001), No. 4, p. 353–361.
- [4] Makrakis M.S., ed., *Greek Letters — From Tablets to Pixels*. Oak Knoll Press, New Castle, Delaware, 1996.
- [5] Euripides, *Alcestis*, Garzya A. ed. In "Bibliotheca Scriptorum Graecorum et Romanorum Teubneriana", BSB B.G. Teubner Verlagsgesellschaft, Leipzig, 1980 and 1983.

◊ Claudio Beccari
 Politecnico di Torino, Turin
 Italy
 claudio.beccari@polito.it

Typesetting in Bengali script using T_EX

Palash B. Pal

Abstract

I will discuss various problems regarding typesetting in Indian languages, with special reference to the Bangla script. These problems include: conjunct vowel symbols; conjunct consonants; non-linear placement of glyphs; minimum glyph set not unanimously agreed upon. I will also discuss how an almost phonetic transcription in the input file can produce true Bangla output.

1 Introduction

T_EX was originally designed [1] to be an efficient typesetter for scientific documents. The basic, or the default, font was Roman, and mathematical symbols appeared as control sequences. Some diacritical marks and a few special symbols were also included in the original T_EX, making it possible to typeset documents in most major languages of Western Europe. The macro packages derived from T_EX, like the versatile L^AT_EX or more specialized packages to meet the needs for a specific society or a specific journal, also inherit these characteristics of T_EX. Henceforth, whenever I mention T_EX, it will mean the original, unadorned T_EX in addition with macro packages like L^AT_EX.

The design of T_EX fonts is done through METAFONT [2]. Once one masters this language, one can create any font containing any design, or shape, of "letters". Therefore, it was straightforward to create Cyrillic fonts, Gothic fonts etc., as well as many different font-shapes for the Roman font, such as the blackboard bold fonts, the calligraphic fonts, etc.

Apart from the easy typesetting of mathematical formulas, T_EX has many useful features. I need not go through all of them, only indicate a few. T_EX files do not have any hidden commands (what you see is what you have ordered); one can use macros to make typesetting simple; the contents, the running page headings etc can be programmed, and so on.

For many of these reasons, it was desirable to create fonts so that the languages using other scripts can also take advantage of T_EX—not only to write mathematical articles in these scripts, but also for simple typesetting of the plain text of a novel, for example. In addition, several macro packages were necessary so that the default fonts, styles, etc., for T_EX can be set for these scripts.

অ	আ	ই	ঈ
উ	উ	ঝ	
এ	়	ও	ও
ক	খ	গ	ঘ
চ	ছ	জ	ঝ
ট	ঢ	ড	ঢ
ত	থ	দ	ধ
প	ফ	ব	ভ
য	ৱ	ল	৷
ম		হ	ড়
ঝ		ঃ	ঃ

Figure 1: The Bangla alphabet. The vowels first, the consonants second.

Recently, I have developed such a package for our language Bangla* [3]. The purpose of this article is to outline the development of this package, and to discuss some issues arising from it.

2 The writing system

2.1 The alphabet

The Bangla alphabet derives from the Brahmi and Kharoshthi alphabets that were in vogue more than two thousand years ago. (The same can be said about most other alphabets used in India today.) The arrangement of letters in these alphabets follow a very logical pattern—the vowels and the consonants are kept separately, and the consonants are arranged according to the point of articulation of the corresponding phonemes. In Fig. 1, I show the Bangla alphabet.

To create the fonts, one might start with coding the letters in the alphabet, but troubles arrive even at this stage. Some people might want to say that in the list of vowels, there should be one more, indicated by the suspicious gap in the second row of vowels. Indeed, in older primers of the language, this space used to be occupied by another “letter” which stood for the vocalic /l/ sound as in the final syllable of the English word *little*. This was a legacy of the Vedic language, and neither the letter nor the sound was ever used in Bangla. Even those who would like to include it in the alphabet would agree that it has no function in writing Bangla. So most

primers and dictionaries now omit this letter from the alphabet, and we find it reasonable to do so.

Once this decision is taken, one can design the letters in a straightforward manner. However, the story just starts there. There are two important issues to be addressed before the font design can be complete. First, we see that the number of vowels and consonants are much larger than that in the Roman alphabet. So one has to decide how to access the letters from the keyboard. Second, a font does not contain only letters. There are the punctuation marks, the numerical symbols, possibly some diacritical marks, and so on. All such symbols, together with the letters, are called *glyphs*. Before constructing a font for a certain script, one thus has to determine what is the minimum glyph set. This is a non-trivial task for an Indian script like Bangla, as we will discuss in detail below. Most of these issues are not specific to Bangla, but are common to other Indian scripts.

We will address the second issue first, and then come back to the first one.

2.2 Conjunct vowel symbols

We have already shown the vowels in Fig. 1. However, these symbols are used only if the vowel sound occurs alone in a syllable, or at the beginning of a syllable, i.e., if the syllables are of the form [V] or [VC].† Examples:

আম	<i>ām</i>	(mango)
উট	<i>ut</i>	(camel)

However, if the syllable has a vowel elsewhere, e.g., has the form [CV] or [CVC], the vowel symbol attaches itself to the preceding consonant symbol, and takes a different form. For example:

মা	<i>mā</i>	(mother)
টুপ	<i>tup</i>	(sound of dripping water)

Notice the differences. The /ā/ sound is written as ‘আ’ in a [V] or [VC] syllable, but only as ‘়’ attached to the preceding consonant symbol in [CV] or [CVC] syllables. The sound /u/ appears as a little attachment to the bottom of a consonant if this consonant is in the same syllable. And so on for other vowels as well.

So far, this means that we need to double the number of vowel glyphs to accommodate the conjunct vowel symbols as well as the pure vowel symbols. It need not be as simple as that, as we will see.

* Bangla (or “Bengali”, as it is often written in English) is a language with more than 200 million native speakers. It is the national language of Bangladesh, and the state language of two Eastern states of India—West Bengal and Tripura. The Bangla script, with minor variations on a few letters, is used to write other languages of northeastern India, e.g., Assamese.

† We will denote syllables by square brackets, where V will denote a generic vowel sound and C a consonant sound.

2.3 Conjunct consonant symbols

The consonants also often do not appear in their simplest form in writing. If two consonant sounds occur within a word which are not separated by any vocalic sound, the two are often represented by a conjunct symbol. Let me provide a few examples:

$$\text{ন}+\text{দ} = \text{ন্দ}, \quad \text{প}+\text{প} = \text{ঞ}, \quad \text{স}+\text{ম} = \text{শ্ম}$$

Roughly speaking, the first consonant takes an abbreviated form which attaches itself to the top or to the left of the second consonant symbol.

Although not all possible combinations of two consonants are used in writing, the number of allowed combinations is huge. Moreover, there can be conjuncts of three consonants as well, although in this case the third one must be the member of a few select consonants: /y/, /r/, /l/, /w/. For example:

$$\text{স}+\text{ত}+\text{র} = \text{ষ্ট}, \quad \text{ত}+\text{ত}+\text{র} = \text{ঙ্গ}, \quad \text{ন}+\text{দ}+\text{র} = \text{ণ্ড}$$

All such combinations need to be accommodated in the font in some way or other, and it is no easy task.

2.4 A short history of the Bangla glyph set

Large scale printing started in Bengal in the 19th century. From the beginning to about the middle of the 20th century, printing was done exclusively on letter presses. In these machines, one used a rectangular matrix mold for each character. These matrices were arranged one after another to produce a line of text. The matrix molds for one line of text were then held between two metal plates, usually called “lead”. Such lines were then arranged vertically to form a page. All lines in a page were finally held together by a piece of string and the impression of the page was taken on paper.[‡]

The matrix molds for single characters could not be placed vertically on one another without a lead separating them. This posed a problem. Conjunct consonants are stacked vertically in most cases. It was impossible to use the matrices for single consonants and stack them vertically to obtain the conjunct consonants. Therefore, one had to have matrices not only for the simple consonants, but separate ones for each possible conjunct.

This list included not only the conjunct consonants, but some conjunct vowels as well. For example, consider the symbol for the sound /u/ in the non-initial positions in a syllable, which will be represented by the symbol /+u/. The symbol is called *u-kar*. It looks like a little knot below the consonant. It was impossible to produce this effect by using a separate matrix for the *u-kar*. One had

[‡] In fact, one did not take an impression of a single page but of several pages together, but this detail is irrelevant for us.

to have separate matrices for /ku/, /bu/, /mu/, and many more. What was worse, one needed separate matrix molds for the *u-kar* attached to conjunct consonants such as /s+t/, /s+t+r/ and so on. The same problem existed for other conjunct vowels, and in each case one had to have a complete set of matrices for all consonants attached to this vowel. In the middle part of the 19th century when Ishwar Chandra Vidyasagar standardized the glyph set, about 1000 glyphs were necessary for typesetting Bangla.

The situation improved a bit with kerned molds, but the real revolution in Bangla printing occurred in the 1930s, when lino printing was introduced. In a lino machine, one uses mirror images of the matrices for different glyphs, arrange them to form a line of text, and pour molten metal on it to produce a matrix mold for the entire line. The problem was that lino machines had room for only about 200 glyphs. Hence it was necessary to devise new strategies for typesetting Bangla in lino machines.

Two main strategies were employed. First, symbols like *u-kar* appeared not vertically to the bottom of a consonant but slightly to the bottom right corner of it. This eliminated a few dozens of matrix molds for *u-kar* attached to various consonants and conjunct consonants. If you add the reduction from other conjunct vowel symbols, the reduction amounted to several hundreds.

Second, the conjunct consonants themselves were typeset a little differently. Rather than the two consonant symbols joining in the vertical direction, the first consonant was typeset in a somewhat midget form to the left of the second one. For example,

$$\text{ন}+\text{ত} : \text{ন} \rightarrow \text{ন্ত}, \quad \text{স}+\text{ক} : \text{স} \rightarrow \text{স্ক}$$

The advantage was that, the same glyph for a “broken” /n/ could now be used in front of any other consonant. So, instead of all glyphs of the form /n/ plus some other consonant(s), one could make do with just a single glyph for /n/.

With these two strategies, the number of necessary glyphs were reduced to the number that a lino machine could support, and Bangla typesetting entered into a new era.

2.5 Non-linear arrangement of glyphs

There was another big problem, which was not solved by lino printing.

Consider the conjunct vowel symbol for the vowel sounds /+e/ and /+i/. We show how they

appear when they attach themselves to the consonant କ, representing the sound /k/:

କ+ୟ = କେ, କ+ଇ = କି

The problem now shows. The conjunct vowel symbols for these two vowels appear *before* the consonants, although phonetically the vowel sounds appear later. In other words, the glyphs cannot be typeset in the order in which the sounds appear in the spoken word.

The problem continues with some other symbols for conjunct vowels. What happens with the /+o/ sound is curious:[§]

କ+ଓ = କୋ

Notice that the symbol for /+o/ appears in two parts—one part (which looks like the symbol for /+e/) goes to the left of the consonant, whereas the other (which looks like the symbol for /+ā/) goes to the right. There are two special conjunct symbols for the diphthongs /+oi/ and /+ou/, which also have this problem of non-linearity.

3 Matters of outlook

When one starts to design a typeface for any script, one has to face several questions; and if the typeface is to be constructed with a new technology, the questions become more intriguing.

Take an example from the Roman typeface. Typewriters had monospaced fonts, i.e., all characters had equal width. Consequently, the symbols for the number zero (0) and the capital letter O looked pretty similar, sometimes identical. This created a problem for computers, because substituting one for the other would send incorrect signals to the computer. However, looking at the file, it would be difficult to spot the error. So, in the early days of line printers, the number zero started to appear with a cross mark through it. This was a welcome change, because it reduced confusion.

Then came laser printers. Try to imagine a person trying to develop a font for this new technology. The print quality of laser printers is much better than that of typewriters or line printers, and the fonts need not be mono-spaced. So maybe, even without a cross mark through zero, the two symbols could be distinguishable. The font-maker would now face the question: should he go back to the traditional design where the difference would now be appreciable, or should he maintain the cross mark running through zero in order to make the distinction more pronounced, something that will be obvious even in handwriting? To my knowledge, all

[§] By /o/, we represent the vowel sound of the word ‘role’, not that of ‘rock’.

laser fonts have taken the former option, and I think it is a pity. There are similar issues regarding ‘2’ and ‘z’ (where the second one is often hand-written with a horizontal cross mark in the middle in order to avoid confusion), or regarding the number one (1) and the lowercase letter ‘ell’ (l), which had a single key on most typewriters.

There will be many such questions for Bangla. One class of questions would involve the shapes of the glyphs. As I already mentioned, lino printing simplified the shapes a lot. The change in that case was prompted by the restriction on the total number of glyphs. In a computer font, one can accommodate much larger number of glyphs. Therefore the first question is: should we forget about the reforms implemented by lino printing, or should we go back to the traditional designs?

The answer need not be a clear yes or no. In some cases, it is easy and convenient to go back to the traditional design. Consider for example the issue about the *u-kar*. It is easy for METAFONT to define a glyph with zero width where the character appears to the left of the cursor. In other words, while processing the *u-kar*, the cursor does not move further to the right, so that the symbol for *u-kar* appears below the preceding consonant.

METAFONT output 2002.01.06:0722 Page 15 Character 117 "hraswa-u-kar"

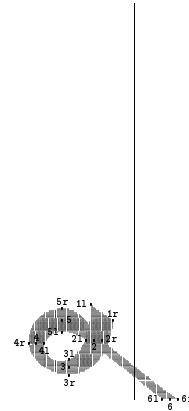


Figure 2: Design for the conjunct vowel symbol for /+u/. The vertical line denotes the cursor.

In some other cases, however, I felt it would be preferable to keep the lino reform. There were some conjunct consonants which, in the traditional method, were *non-transparent*. This means that, looking at the glyph for the conjunct consonant, one could not make out which consonant sounds they contain, because the glyph for the conjunct did not

resemble a superposition of the glyphs of the component consonants. In the lino style, such glyphs were discarded in favor of two different glyphs, a “broken” one for the initial consonant and the full one for the later one. They were therefore *transparent*, and it would make more sense to keep these reformed shapes.

In fact, in some cases we can do better than the lino. Consider for example the symbol for the sound /r/ when it appears as the final component of a conjunct consonant cluster (we will call it the symbol for /+r/). It then shows as a curvy line at the bottom of the rest of the cluster, like this:

$$\text{প}+\text{র} = \text{প}, \quad \text{গ}+\text{র} = \text{গ}$$

Lino fonts had to have separate glyphs for each such conjunct. However, in our computer fonts, only one curvy line, extended to the left of the cursor in a zero-width character, can do most of the job. Notice that consonants like প (/p/) or গ (/g/) have a vertical bar near the right end. If the distance between this vertical bar and the right end is kept the same for all such consonants, it is trivial to adjust the glyph for /+r/ so that it attaches exactly to the bar.

An important comment has to be made here. It is to be understood that we are trying to reduce the number of glyphs, but *not* in order to avoid more work in the design stage. Neither are we doing it as a desperate solution for accommodating all glyphs into a standard font which can contain 256 characters. The point is that one doesn’t need unnecessary complication. A more advanced technology should be capable of making more adjustments, and so a fewer number of glyphs might be able to produce the same (or even a better) output than what could have been done by a less advanced technology with a larger number of glyphs.

4 Input of characters

4.1 Posing the question

We now come to a question which was raised earlier but was not addressed. After we create the font, how do we input the characters into a file?

The question is serious for Bangla and other Indian languages firstly for the sheer number of glyphs. Even after cutting down on the number through various techniques as described above, it is not possible to fit all glyphs on a keyboard unless one is ready to deal with a much larger keyboard. This is the main reason why good typewriters did not develop for the Indian scripts. As a result, the input procedure of characters remains unsettled.

However, the number of glyphs is not the only problem. There are others. Take, for example, the letters corresponding to the sibilant sounds. In the Roman alphabet, there are two of these: /s/ and /z/—the first one unvoiced and the second voiced. In Bangla, there is no voiced sibilant consonant. There are two unvoiced sibilant phonemes, a palatal sound (like /sh/ in English ‘shot’) and an alveolar one (like /s/ in ‘sit’), and two letters for them. However, Sanskrit had also a retroflex sibilant sound and a letter for it, as do some modern South-Indian languages. Bangla and other northern Indian languages do not really have the retroflex sound but carry the letter as a legacy to Sanskrit. So, in total, there are three sibilant consonants. Add with them the ‘broken’ forms which can be used as the initial component in a conjunct consonant, and you have six. Let us now ask the simple question: how are we going to input these six different things into a computer file?

This is less of a problem in WYSIWIG word processors. You could have the six things bound to certain keys so that pressing those keys would give you these characters on the screen. But for TeX, the issue is different. Here, the input file contains ASCII characters only. Thus it is desirable to have the input file in such a way that the phonetic correspondences are recognizable. In other words, if we choose the upper-case ‘Y’ to represent the letter for the alveolar /s/, it would not be very nice because the source file will not be readable. It would be extremely difficult to edit such a file.

4.2 Use of control sequences

To my knowledge, there are two alternatives for producing a TeX input file which is at least roughly phonetic. One is to use control sequences.

This was indeed the strategy taken by an earlier Bangla font designed by Abhijit Das [4]. To use his font, one can type \kh to obtain the consonant ‘ঁ’, or \th to obtain ‘ঁ’. Indeed, the consonant ‘ঁ’ pronounces like an aspirated /k/, and ‘h’ can be taken as the indication for aspiration. The usual rules for the Roman transcription of Bangla (and other Indian) names uses ‘kh’ to denote this letter.

So this looks like a pleasant solution, until you realize that TeX has to know where the control sequence ends, so you need to leave a space in the input file right after using this sequence. The space has to appear even if the intended letter appears at the beginning or in the middle of a word. Consequently, the words will not appear together in the input file, and it will be a constant problem while trying to edit the file. If, on the other hand,

one protects the control sequence within braces, it will mean too much typing for producing just one letter—five to be precise, including the open and closed braces, the backslash, and the letters ‘kh’.

The use of control sequences terminating with a non-letter character could have been a more economic solution. For example, the macro for the intended character could have been `\kh.` instead of `\kh` without the period. While typing, four keystrokes would have been needed, and words could have been kept together as well. But this solution was not taken by the package created by Das [4].

Of course it is not mandatory to input the letter as a control sequence; instead, one may input the letter directly. In this case, the letter in question occupies position number 75 in his font, the number which is taken by the capital letter ‘K’ in ASCII. Thus, within the font, if one types ‘K’ (i.e., shift+k), one will also obtain the same letter. One can then take capitalization (in the input file) as a sign of aspirated consonants.

Unfortunately, this approach would not take us very far. In fact, the previously cited `\th`, which is an aspirated form of the consonantal soft /t/ sound and pronounces almost like the consonant in the English word ‘thaw’, occupies position number 122, which in ASCII is the place for lower case ‘z’. If we have to write ‘z’ in place of the sound /th/ in the input file, we will be far from a phonetic input file. Moreover, since the sheer number of characters exceeds the number of keys on the keyboard, there will always be some characters which could not be typed directly from the keyboard.

4.3 Use of ligatures

For the reasons stated above, I have not used control characters at all. Instead, I have used ligatures. In the input file, if we have ‘k’, TeX gets ready to typeset the letter for the /k/ sound (ঃ), but also receives information from the ligature table not to do it quite yet. If the next input character happens to be ‘h’, it forgets about the letter it picked up, and picks up the letter for the /kh/ sound (ঃ) instead.

The method is used for the Roman fonts as well, and is described in the METAFONT book [2]. However, the number of ligatures necessary in a Roman font is very few—‘fi’, ‘ff’, ‘ffi’, ‘fl’ and ‘fff’. On the other hand, for my Bangla font, there are ligatures starting with about 80 characters, and in most cases there are multiple possibilities for the completion of the ligature.

Of course, when there are so many ligatures, one has to be very careful not to create *too* many. For example, if you want to use ‘kh’ as a ligature,

you have to be careful that the need for typesetting ‘k’ and ‘h’ separately will either not arise at all or will arise only very rarely, in which case the ligature must be broken manually, by enclosing one of the letters within a pair of braces. This, in my opinion, was the hardest part in the design of the fonts, but so far I have not heard any complaint about it. For the most part, I have been able to use the standard combinations which are used in Romanized transcription of Bangla words.

Things would have been easier if METAFONT had one additional feature. In the ligature file, you can replace any combination *AB* (of course the letters are symbolic here) by only *A*, or by only *B* (i.e., tell TeX to ignore one of the components of a ligature), or by *C* (i.e., tell TeX to forget both and use a third one). However, I found no way of replacing *AB* by *CD*, i.e., by two (or possibly more) new characters. It could have helped very much, although I managed somehow without it.

4.4 The missing vowel symbol

There is another interesting aspect of the Indian scripts that we have not touched before. The first vowel of the alphabet, whose sound will be denote by /a/ (in Bangla, it sounds roughly like the vowel sound in the English word ‘rock’), is left out of the written representation when attached after a consonant in a syllable. In other words, just the consonant letter written by itself implies that the vowel sound /a/ is attached to it. Thus, for example, if you want to typeset ‘rock’ in Bangla (there is such a word, although it means something completely different), it would be sufficient to typeset ‘rk’. But obviously that will not look very phonetic.

I found a solution to this problem when I realized that no special sign needs to be created for /+o/. As already mentioned, the symbol for /+o/ is a combination of the symbols /+e/ and /+ā/. Thus, the space for ‘o’ is “free” in some sense, and so I have put a null letter in its place. It means that if you type an ‘o’ in your source file, it will not produce anything in the output. Therefore, you can type ‘rok’ to obtain the word which sounds like ‘rok’, and certainly it looks phonetic.

I have to make a special comment about other Indian languages here. In most other major languages of India, the first vowel is not pronounced as it is in Bangla. Rather, it is pronounced like a short /ā/, something like the vowel sound of the English word ‘mud’. For these languages therefore, my solution will not be very attractive. For Bangla, however, it seems quite acceptable.

4.5 The question of non-linearity

Finally, the question of non-linear arrangements of characters has to be addressed. As I mentioned earlier, the conjunct vowel signs for /e/ and /i/ appear before the related consonant, and the sign for conjunct /o/ appears on both sides. For example, suppose we want to write the word ‘pen’ in Bangla. (This word has been borrowed from English, so it is a Bangla word as well.) How would we have to typeset it? Not as pen, but as epn. Needless to say, this is not phonetic.

I have used a control sequence to get rid of this trouble. I have made the following definition:

```
\def\*#1#2{o\null{#2}{#1}}
```

In short, it reverses the order of two things. Thus, the word ‘pen’ can now be typeset as ‘*p*en’, because the symbol after the second star sign would be typeset first, as needs to happen for Bangla. The advantage is that one can disregard the star signs while reading from the screen, thus obtaining a phonetic transcription.

5 Afterthoughts

The quality of a font is of course ultimately determined by its users. It is not easy to foresee a time when commercial editions of Bangla novels would be typeset in *T_EX*, just as it is difficult to find many English novels which have been typeset in *T_EX*. However, there are many academic books in English and other European languages which are typeset in *T_EX*. If this can be done in Bangla as well, it will be reap great benefits. The publishers will love it if the authors write their books in *L^AT_EX* which they won’t have to typeset. In a limited market like that for Bangla books, the resulting savings in the typesetting cost can prove to be very valuable.

The first Bangla book that came out in my fonts is a book on Bangla phonetics authored by me [5]. It is easy to say that it is the first, because it came out before I made the fonts public. It might even be the first book in any Indian language typeset in *L^AT_EX*. I am happy to say that since then, a few other books have come out as well, including some which are not written by me. Let’s see what lies in the future.

References

- [1] D. E. Knuth, *The T_EXbook*, Addison-Wesley Publishing Company, 1984.
- [2] D. E. Knuth, *The METAFONTbook*, Addison-Wesley Publishing Company, 7th printing, 1992.
- [3] Palash B. Pal, *Bangtex: A package for*

typesetting documents in Bangla using the T_EX/L^AT_EXsystems, available primarily from <http://tnp.saha.ernet.in/~pbpal/bangtex/bangtex.html>.

- [4] Abhijit Das, *Bengali Writer T_EX Interface*, available from <http://www2.csa.iisc.ernet.in/~abhij/bwti/>.
- [5] Palash B. Pal, *ধ্বনিমালা বর্ণমালা* (Sounds and Letters), Papyrus, 2001.

◇ Palash B. Pal
Theory Group, Saha Institute of
Nuclear Physics
1/AF Bidhan-Nagar, Calcutta
700064
India
pbpal@theory.saha.ernet.in

Electronic Documents

interactiveworkbook: L^AT_EX-based interactive PDF on the Web

Jonathan Kuhn

Abstract

The package **interactiveworkbook** gives the user the ability to write L^AT_EX documents which, ultimately, create interactive question-and-answer Portable Document Format (PDF) tutorials meant to be used by Internet distance learning students (Kuhn, 1999).

1 Introduction

Combining interactivity with the display of mathematical notation is important for online distance learning. The interactive PDF (Adobe, 1999a; Adobe, 1999b) tutorials created using **interactiveworkbook** are meant to mimic a classroom discussion situation. The interactive PDF tutorial program responds in either a positive or negative way to an Internet student’s answers to a variety of questions, including multiple choice (multiple and single answer), fill-in-the-blank, true/false and matching type problems. The fact that the interactive PDF tutorial is L^AT_EX-based allows both the program questions and responses to student answers to be very

explanatory and, in particular, to freely use mathematical notation.

Achieving interactivity with the display of mathematical notation is difficult on the Internet. The Internet is presently based on the Hypertext Markup Language (HTML) which essentially does not allow for mathematical notation. Not surprisingly, a fair bit of effort has been devoted in the last few years to finding a solution to this problem. The World Wide Web Consortium, which manages HTML, has prepared a Mathematical Markup Language (MathML) (Sutor and Dooley, 1998; W3C, 2001). The computer company IBM has produced a plug-in for Netscape Navigator and Microsoft Internet Explorer called *techexplorer Hypermedia Browser* (Sutor and Dooley, 1998). The interactive PDF approach used in this paper has been used by others, in particular, D.P. Story (1999) and T. Merz (1998). An initial version of **interactive-workbook**, called *pdfflash*, was written by A. Montgomery (1998).

2 Creating an interactive PDF question

A simple example of the use of **interactive-workbook**, to create an interactive PDF multiple choice question with multiple answers using check boxes, is given below.

```
\documentclass[dvips]{article}
\usepackage{interactiveworkbook}

\begin{document}

\questionandresponses{check}
{ques1.pdf}{questionindex.pdf}{ques3.pdf}
{Question 2. $\\$ This is a multiple
check box question where more than one
answer can be correct.\\}
\begin{center}
\checkone {\boldmath $0.94$}
\answercheckone{Off} $\\$;
\checktwo {\boldmath $0.95$}
\answerchecktwo{Off}$\\$;
\checkthree {\boldmath $0.96$}
\answercheckthree{On} $\\$;
\checkfour {\boldmath $0.97$}
\answercheckfour{Off} $\\$;
\checkfive {\boldmath $0.98$}
\answercheckfive{On}
\end{center}
}

{Yes, correct.}
{No, try again.}

\end{document}
```

Using a L^AT_EX → DVI → PS → PDF process, three PDF pages (which, together, are called an interactive PDF question), are generated by this sample L^AT_EX file. These three pages include a question (first) page, correct response (second) page and incorrect response (third) page. In addition to the question text and check boxes, there are a number of navigational buttons appearing at the bottom of all of these pages. Depending on the navigational button clicked, the user is passed either between the three pages of this interactive PDF question, or outside of this interactive PDF question, to another interactive PDF question.

2.1 The \questionandresponses command

The command \questionandresponses used in the sample L^AT_EX file above has seven arguments.

The first argument, “check”, tells **interactive-workbook** that check boxes are used in this particular question. This first argument may take on one of four values: check, popup, field or radio.

The second, third and fourth arguments name the PDF files that a user passes to when clicking on the Previous, Index and Next navigational buttons, respectively. In this case, a user passes back to the first question, ques1.pdf, when clicking on the Previous question, ques3.pdf, when clicking on the Next button.

The question asked is given in the fifth argument. Five check boxes, \checkone, ..., \checkfive, are used, where the five possible choices are 0.94, ..., 0.99. The third and fifth choices, in this case, are correct, because \answercheckthree and \answercheckfive are “On” and the other three are “Off”. The correct answers are not displayed to the user on any of the three pages of the interactive PDF question, and are only shown on the two response pages after clicking on an Answer button.

Correct response and incorrect response comments are given in the sixth and seventh arguments, respectively. The correct response comment, in this case, is “Yes, correct.”, whereas the incorrect response comment is “No, try again.”. These comments appear below the question on the correct response and incorrect response pages, respectively, of the interactive PDF file.

2.2 Interactive index PDF page

In addition to the interactive PDF question, there is one other kind of interactive PDF document generated by **interactive-workbook**, called an interactive PDF index page. The interactive PDF index page can be used to tie a collection of interactive

PDF questions into an *exercise* of related questions. Consider the following example.

```
\documentclass[dvips]{article}
\usepackage{interactiveworkbook}

\begin{document}

\exerciseintroduction{Index. This is an
index of four interactive PDF questions.}

\vspace{.2in}
\exerquessetupone{ques1.pdf}
\exerquessetuptwo{ques2.pdf}
\exerquessetupthree{ques3.pdf}
\exerquessetupfour{ques4.pdf}

\begin{center}
\begin{tabular}{|l|} \hline
Question 1. & \exerquesone \\ \hline
Question 2. & \exerquestwo \\ \hline
Question 3. & \exerquesthree \\ \hline
Question 4. & \exerquesfour \\ \hline
\end{tabular}
\end{center}
\end{document}
```

The introduction to the exercise, beginning “Index. This is an ...”, is given as the argument of `\exerciseintroduction`. Also, `\exerquessetupone`, for example, specifies the user passes to the first interactive PDF question, `ques1.pdf`, when clicking on a rectangular button defined by `\exerquesone`.

3 interactiveworkbook

The two sample L^AT_EX files given above were written using the WinEdt editor. They have been executed using the MiK^TE_X version of L^AT_EX on a Pentium II 32 Meg RAM, 6 Gig hard disk, 266 MHz Aptiva PC. One style file, `interactiveworkbook.sty`, was located in the `/tex/latex/graphics` directory and a number of supporting Encapsulated PostScript (EPS) files (Adobe, 1999c; McGilton and Campione, 1992; Flanagan, 1998) were located in the same directory as the two sample L^AT_EX files.

The package `interactiveworkbook.sty`, EPS files, some sample interactive PDF files, and a readme are contained in the self-extracting zip file, `interactiveworkbookzip.exe`, which can be downloaded from

<http://www.pnc.edu/faculty/jkuhn/research/research.html>

or from CTAN at
<http://www.ctan.org/tex-archive/macros/latex/contrib/interactiveworkbook>.

After making sure Adobe Acrobat 4.0 or later is installed and setting the File | General Preferences | Magnification | Default Zoom preferences to “Fit in Window”, click on one of the PDF files, such as `ndex.pdf`, say, to begin the interactive session.

4 Acknowledgements

This work was funded by the Indiana Higher Education Telecommunications System grant 652 9395–2956.

References

- Adobe. “pdfmark Reference Manual, Technical Note 5150”. 1999a. Available at <http://www.adobe.com>.
- Adobe. “Portable Document Format Reference Manual, Version 1.3”. 1999b. Available at <http://www.adobe.com>.
- Adobe. “PostScript Language Reference, 3rd ed.”. 1999c. Available at <http://www.adobe.com>.
- Flanagan, David. *JavaScript, The Definitive Guide, 3rd ed.* O'Reilly, Sebastopol, California, 1998.
- Kuhn, Jonathan. “An Interactive Workbook For Internet and Classroom Students”. Available at <http://www.pnc.edu/faculty/jkuhn/research/research.html>.
- McGilton, Henry and M. Campione. *PostScript by Example*. Addison-Wesley, Reading, Massachusetts, 1992.
- Merz, Thomas. *Web Publishing with Acrobat/PDF*. Springer, New York, New York, 1998.
- Montgomery, Aaron. “pdfflash”. 1998. Software package available at <http://mac69108.math.cwu.edu/code/latex.html>.
- Story, Donald P. 1999. Interactive PDF software available at <http://www.math.uakron.edu/dpstory/acrotex.html>.
- Sutor, Robert R. and S. S. Dooley. “T_EX and L^AT_EX on the Web Via IBM techexplorer”. *TUGboat* 19(2), 157–161, 1998.
- W3C. “Mathematical Markup Language (MathML) Version 2.0”. 2001. Available at <http://www.w3.org/TR/MathML2>.

◊ Jonathan Kuhn
Purdue University North Central
Westville, Indiana 46391–4197
USA
jkuhn@pnc.edu
<http://www.pnc.edu/faculty/jkuhn>

Font Forum

Multiple Master math extension fonts

D. Men'shikov, A. Kostin, and M. Vulis

Abstract

From its inception, \TeX has relied on the concept of extensible (composite) characters to implement large glyphs that occur in mathematical typesetting. While thousands of papers have been successfully written using this technology, it nevertheless has obvious shortcomings. This article demonstrates them, as well as an alternative solution, free from the problems.

1 Problems with the traditional approach

The extensible glyphs in \TeX are composed out of several smaller characters, and, in some cases, rules. The construction itself is carried either by code inside the \TeX compiler itself, using the special information in the $.tfm$ files, or by macros, implemented as part of the format. We will consider each in turn.

1.1 Extensions done by \TeX

An example of extensible glyphs constructed within \TeX is the construction of large delimiters, such as brackets. In this case, a vertical segment is inserted to stretch the delimiters vertically:

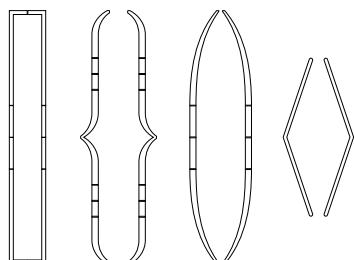


Figure 1: Vertical delimiters

One of the shortcomings of the extensible characters is that some character shapes do not lend themselves to the insertion of extension pieces; for example, angle brackets are not extensible. Thus, when used to encompass a large formula, they would not scale to the required size. For example, the input:¹

¹ `\emptybox` is a macro which makes a frame box of the specified width and height, used to avoid printing large black squares in this journal.

```
$\left< \emptybox{1cm}{1cm} \over \emptybox{1cm}{1cm} \right>$
```

results in .

A variant of the situation is \TeX 's treatment of the radical symbol: while the vertical extension is driven by the $.tfm$ information, the horizontal bar is supplied by the \TeX program itself—and in this case, the bar is a rule, rather than an extension character.

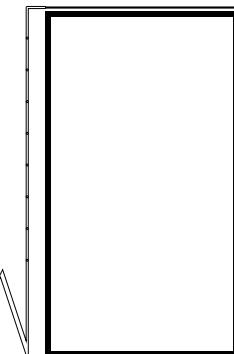


Figure 2: Radical delimiters

In the above picture, the rule component is shown filled.

The problem with the use of rule components in building a composite symbol is that rules are subjected to different roundoff rules than characters. With DVI files and bitmap fonts, these roundoff errors could in principle be handled by careful calculations within the \TeX program and the DVI driver; but in the modern world of scalable fonts and PS/PDF targeting this becomes an impossibility.

For instance, the picture below shows a snapshot of a square root constructed by Pdf \TeX :

The underlying \TeX code is simply

```
$ \sqrt{\emptybox{1cm}{2cm}} $
```

Depending on factors such as the magnification used in the PDF previewer and the resolution of the output device, the rule may be thinner, or thicker, or above, or below the part of the `sqrt` glyph it is supposed to connect to smoothly. Sometimes it might even fit correctly.

The root of the problem is that glyph rounding is subject to font hinting, while the thickness and positioning of rules is not precisely controllable by the PS/PDF rendering.

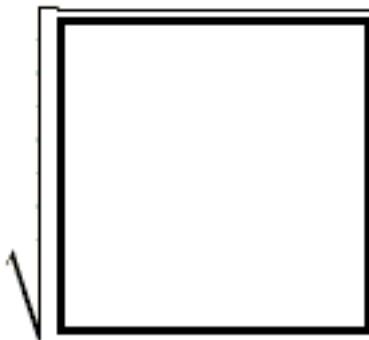


Figure 3: Incorrect radical

1.2 Extensions done by macros

The second type of extension mechanism is via \TeX macros. This is typically used to construct long horizontal symbols, as done by commands similar to `\underbrace`:

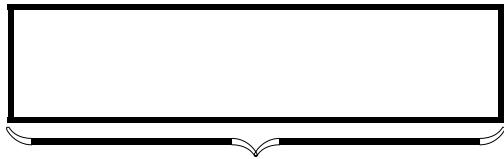


Figure 4: Underbrace extension

Once again, we get mismatches in the width and positioning of the connecting rules comparing to that of the symbols used in the middle and at the ends of the long brace. A snapshot:



Figure 5: Incorrect extensible braces

1.3 Shading characters

A third problem with extensible characters is that sometimes (usually, in slides) it is desirable to outline, or shade characters; this cannot be done correctly with composite characters. In fact, the diagrams above that show the structure of extensible characters, are *unintended output* of an attempt to stroke extensible symbols, making them suitable for this article, but not for a use within a slide.

Summarizing, we have identified three separate problems with the \TeX approach to the extensible symbols:

- Some delimiters (angle brackets, for example) cannot be correctly scaled.
- Rules and characters together cannot be correctly aligned.
- Composite symbols cannot be shaded or correctly stroked.

All these problems can be solved, however, if we switch to a different mechanism of constructing large symbols: Multiple Master fonts.

2 Multiple Master fonts

For those unfamiliar with MM fonts, they represent a PostScript world answer to METAFONT. Just as METAFONT allows creation of different designs from the same font program, so does the MM technology. While MM fonts are less flexible than METAFONT, they are easy to use. The output of MM font instancing is a Type 1 outline font, whereas a generic METAFONT emits bitmapped fonts. (See [3,6] for additional information on MM fonts.)

MM technology was introduced by Adobe in 1991; the first description appeared in PC Week [4]. The first MM font was Myriad (1992), with two axes: weight and width.

Since 1992 Adobe has designed at least 36 MM font families with about 100 fonts. Perhaps the most successful is the Minion family.

While Adobe originally intended to include MM in the OpenType specifications, this effort has been abandoned, and Adobe has stopped making new MM fonts. The last Adobe MM font, VerveMM, was designed in 1998; Adobe had announced that it was giving up on MM technology at the 1999 ATypI Congress.

Following the Adobe announcement, an article in *MacObserver* lamented [5]:

Was it necessary for Multiple Masters to die? Probably not. Several factors contributed: inexpert and uninteresting designs; a purportedly “open” technology that was in fact proprietary; and inadequate interface support early on to Aldus, Quark and Macromedia. None of this had to be. Pity, pity, pity!

Despite its abandonment by Adobe, MM fonts still represent a very convenient technology for use in typesetting applications like \TeX . While MM functionality is not supported within PDF documents, instances of MM fonts are, and the entire MM font can be made available to the \TeX document.

VTEX has supported MM fonts since 2001, and the use of MM fonts in math extension fonts builds on the existing MM support; we will start by outlining the current support.

VTEX recognizes MM fonts by the name of the font; a font name that includes the bracket character is taken as an instance of a Multiple Master font. This does not represent much of a restriction on the font name selection, since use of brackets in file names is not common and is in fact invalid on some operating systems.

The left bracket in the font name is followed by the instancing parameters. For example:

```
\font\sm=xo_____
\font\mm=xo_____ [300,10]
```

The `\sm` font declaration defines the default shape of the Multiple Master CrononMM font; this is the “normal” way to use the typeface. The second declaration, however, defines an instance of this MM font to be constructed dynamically.

In the above example, the first definition makes VTEX to load the usual TFM metrics file, namely `xo_____tfm`; the second declaration causes VTEX to load the multiple font metrics file `xo_____mfm` and generate the instance metrics on the fly. Because of the use of `.mfm` files, providing metrics for each instance becomes unnecessary.

In some MM fonts, all the instances of the font have the same metrics; if so, it is possible to use the ordinary `.tfm` file. In most MM fonts, however, the metrics of each instance are different, and this made the development of the new metrics format necessary.

Upon seeing a MM font used in the document, VTEX automatically uses its built-in PostScript interpreter GEX [1] to instance it. Since the instancing is done within PostScript, the instance font is always built correctly. The TeX compiler then automatically packs it into the output PDF file.

Additional details on the MM support can be found in the `mmsupp.pdf` document in VTEX distributions.

3 Math extension Multiple Master fonts

Supporting math extension MM fonts requires several additional steps.

First, of course such fonts need to be developed. At this writing, two such fonts exist: `cmex10mm` and `paex10mm`. The first is intended for use with Computer Modern, the second for use with the alternative PaMATH fonts, available from MicroPress.

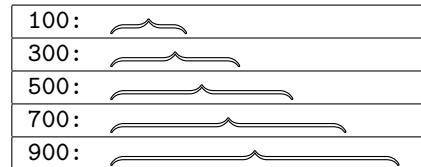
Math extension MM fonts include symbols for vertical delimiters, long horizontal symbols (like the

ones constructed by the `\underbrace` macro), and the radical symbol. Since the radical requires two MM axes, the entire font is a two-axes MM font, even though the majority of symbols are actually “one-dimensional”.

Specific sizes of delimiters can be constructed by loading the font with different instancing parameters. For example, for the `\overbrace` symbol, we can use font commands such as

```
\font\f=cmex10mm[30,100] \f \char"7A
\font\f=cmex10mm[30,300] \f \char"7A
\font\f=cmex10mm[30,500] \f \char"7A
\font\f=cmex10mm[30,700] \f \char"7A
\font\f=cmex10mm[30,900] \f \char"7A
```

obtaining the shapes



(Only the second instancing parameter is used in long horizontal symbols.)

Since the symbols are built from single glyphs, they do not suffer from the problems listed at the beginning of this paper.

Since TeX supports `\overbrace` and other long horizontal symbols entirely through macros, switching TeX to supporting MM instances requires only changes to the macros. One of the tasks of the `mathexmm` style is to therefore redefine these macros.

Supporting vertical delimiters and radicals requires more, well, radical changes: the `.tfm` mechanism of extensible characters needs to be replaced by an MM alternative. We accomplish this as follows:

First, we prepare an alternative metrics file, `cmex10m`, which is mostly the same as `cmex10`, except that

- The TFM CODINGScheme is MMEXTENSION; this is the signal to the TeX compiler that what normally would be interpreted as `exten` instructions in the `.tfm` instead should be seen as pointers to MM glyphs. (It is unfortunate that the `.tfm` syntax does not allow for new flags; this is what forces us to use the CODINGScheme field.) In a `.PL` file, this appears as:

`(CODINGScheme MMEXTENSION)`

- Each `exten` specification in the `.tfm` is replaced by the glyph number of the character to use in the `cmex10mm` MM font. For example, the original `cmex10.pl` listing contains

```
(CHARACTER C 0
  (CHARWD R 0.875003)
  (CHARHT R 0.039999)
  (CHARDP R 1.760019)
  (VARCHAR
    (TOP C 0)
    (BOT O 100)
    (REP C B)
  )
)
```

specifying that the character 0 is to be built from glyphs 0, ‘100 and B. In `cmex10m.pl`, however, we have

```
(CHARACTER C 0
  (CHARWD R 0.875003)
  (CHARHT R 0.039999)
  (CHARDP R 1.760019)
  (VARCHAR
    (REP O 303)
  )
)
```

which means that the character 0 is to be built as an instance of the glyph ‘0303 in the corresponding MM font (`cmex10mm`).

- `cmex10` and `cmex10m` are otherwise identical.

The `mathexmm` L^AT_EX style, when used, loads the font `cmex10m` instead of `cmex10`. Unless extensible characters are involved, it functions in exactly the same way as `cmex10`; but when T_EX is about to build an extensible character, it instead builds an appropriately-sized instance from the `cmex10mm` MM font.

From the user’s point of view, this is all invisible, and no action is required except for adding `\usepackage{mathexmm}` in the document preamble for L^AT_EX 2 _{ϵ} , or `\input{mathexmm}` for plain T_EX and $\mathcal{M}\mathcal{T}$ -T_EX.

4 Acknowledgement

Related ideas previously appeared in [2]; however, they did not add up a practically usable implementation, which has been our contribution.

5 Bibliography

- [1] A. Kostin & M. Vulis, “Mixing T_EX and PostScript: The G_EX Model”, in *TUGboat*, Vol. 23(3), 251–264 (Sep 2000, proceedings of the TUG 2000 conference).
- [2] J. André & I. Vatton, “Dynamic optical scaling and variable-sized characters”, in *Electronic Publishing*, Vol. 7(4), 231–250 (Dec 1994).
- [3] Adobe Systems Inc., “*Type 1 Font Format Supplement*”. Adobe Technical Specification 5015.

- [4] PC Week, Vol. 8(10), pg. 1 (11 Mar 1991).
- [5] Bill Troop, “A giant step backwards for Adobe?”, in *MacObserver*, Oct 6, 1999. <http://www.macobserver.com/columns/troop/99/october/991006.shtml>
- [6] John D. Berry, “*dot-font: Avant Garde, Then and Now*”. <http://www.creativepro.com/story/feature/19432.html>

◊ D. Men’shikov, A. Kostin, and
M. Vulis
MicroPress, Inc.
68-30 Harrow Street
Forest Hills, New York 11375
USA
phone: +1 (718) 575 1818
fax: +1 (718) 575 8038
support@micropress-inc.com
<http://www.micropress-inc.com>

TrueType fonts in PostScript

Thomas H. Barton

1 Introduction

Many TrueType fonts are available at no cost; for example, 328 fonts came with my copy of Windows 2000. Unfortunately, TrueType fonts cannot be directly employed with the versions of PostScript currently in use, levels 1 through 3. With Adobe’s introduction of Open Type fonts, this situation may change in future versions of PostScript but at the moment TrueType remains a resource that cannot be used directly in PostScript. This problem has been addressed by various authors by programs that convert TrueType to PostScript Type 1 fonts. At the time of writing, three, `ttf2pfb`, `tfmpk`, and `ttf2pt1` are currently listed on CTAN in the directory `fonts/utilities`.

Early in the life of level 2 PostScript, which was introduced in 1994, with version 2.013, Adobe transformed the situation with regard to TrueType fonts by introducing a new PostScript font specification, Type 42. The specification is contained in Adobe Technical Note 5012, which can be obtained from: <http://partners.adobe.com/asn/developer/technotes/fonts.html>.

In its opening paragraph, the technical note states that conversion to Type 42 *yields better print quality than can be achieved by converting a TrueType font to a Type 1 or Type 3 font*.

Note 5012 indicates that the font conversion is a fairly complex process but fortunately Ghostscript has already done it via its internal procedure `.loadttfont`. Pointed to the TrueType font file in the form of a file object as its sole argument, this procedure loads on the user stack the dictionary of the font's Type 42 equivalent. This dictionary can be used directly within Ghostscript, but if the aim is to create a program to run under Adobe PostScript, information must be extracted from it.

This article describes a PostScript program which does that and outputs two PostScript files, one a PFA file containing the Type 42 font and another which will print a list of the names and appearances of all the glyphs defined by the font on any PostScript printer capable of using Type 42 fonts. The second file is useful because many TrueType fonts define far more glyphs than are usual in a Type 1 PostScript font, e.g., the TrueType font `arial.ttf` defines 1296 glyphs.

While there may well be others, I am only aware of one other program which accomplishes this task, `To42.ps` by Christof Labouisse. This program is not easy to find; for example, at the time of this writing a Google search using the keyword `to42` produces nothing. I obtained my copy via an appeal on the PostScript newsgroup by the kindness of Michael Piotrowski.

2 The program

The program is contained in a PostScript file called `TrueTypeToType42.ps` which is intended to be run under Ghostscript. It will not run under Ghostview. Because it uses the Ghostscript internal procedure `.loadttfont`, it will not run under a standard PostScript interpreter, whether in a printer or RIP. The program is available on CTAN in `fonts/utilities/TrueTypeToType42`.

Apart from using some Ghostscript internal procedures, it is a standard PostScript program, and so should run on any platform provided a recent version of Ghostscript is installed. I am not sure of the precise meaning of *recent* but I have run it under Ghostscript versions 6.5 and 7.0. The program is extensively commented, so much so that for regular use it would be advantageous to remove the comments.

2.1 The Ghostscript command line

The invocation of this program from the command line uses the -- (double hyphen) option:

```
<Path to \GS{} executable>
-I<Path to \GS{} main folder>
-- <Path to conversion program>
```

`<Path to \TT{} font>`

(This is intended to be typed on a single command line, it has been folded only to fit the text width.)

I am currently using Ghostscript 7.00 on a PC under Windows 2000. The full path name of my Ghostscript Windows executable is:

`C:\GS\GS7.00\bin\gswin32.exe`

All my TrueType fonts are in the directory `C:\TrueTypeFonts`, and `TrueTypeToType42.ps` is in the directory `D:\Fonts\TrueType\Utilities`. So, my command line is:

```
C:\GS\GS7.00\bin\gswin32
-IC:C:\GS\GS7.00
-- D:\Fonts\TrueType\Utilities
    \TrueTypeToType42.ps
C:\TrueTypeFonts\<\TT{} font file>
```

2.2 Output

While running, the program writes extensive commentary to the Ghostscript standard output, and creates three files in a directory of your choice. These three files are:

1. The Type 42 equivalent of the TrueType font.
2. A PostScript file which, when downloaded to a PostScript printer, will produce a hard copy of all the glyphs defined by the font together with their names in ASCII sort order.
3. A text file which is a permanent record of the comments made to standard output.

The root name of the Type 42 file is the PostScript name of the font and its extension is `.pfa`. For example, if you are creating the Type 42 equivalent of `times.ttf`, whose PostScript name is `Times NewRomanPSMT`, the file's name will be `TimesNew RomanPSMT.pfa`. The translation from the binary TrueType file to the ASCII pfa file results in an approximate doubling of file length.

All the usual pfa operations can be carried out with the Type 42 file, e.g.,

- included in the prologue of a PostScript program;
- stored on a printer hard disk;
- re-encoded by changing the encoding vector;
- renamed by changing the `/FontName` key.

The glyph file is called `ShowAllGlyphs.ps`. It is self-contained, having a copy of the Type 42 file in its prologue, and will produce hard copy when downloaded to a printer containing an Adobe PostScript interpreter version 2.013 or later. It can also be viewed in recent versions of Ghostview. The output is a list of all the glyphs defined by the font together with their names in ASCII sort order.

This is useful since many TrueType fonts define very large numbers of glyphs; e.g., my copy of `times.ttf` defines 1296 glyphs, and my copy of `garamond.ttf` defines 663 glyphs.

The text file is called `TrueTypeToType42.txt` and provides a permanent record of the progress of the program as signalled on the Ghostscript standard output. This file is useful if things go wrong.

2.3 Program defaults

The program needs to know where to put its output files. The file names are as just described and the path for them is defined in the program as the value of the key `/Path`. In the program, this is set to the string `(D:\\Temp\\)`. This must be reset to whatever you wish to use. In your editor, search for '`\Path`' and change the contents of the string to whatever suits your needs. '`\Path`' will be found in the `setup` section of the program. Note that Ghostscript requires that the backslash in Windows paths must be replaced by double backslashes. Note also that the terminating path separator must be included.

The program is set up for simplex printing from paper tray 3. If you wish to change this, open the program in your editor, search for '`setpagedevice`' and change the values of `/Duplex` and `/Media Position` to suit your needs. The value of `/Duplex` is the boolean `true` or `false`. The value of `/Media Position` is an integer, the number of the printer tray from which paper will be taken.

2.4 Running under Windows

I run the program on a PC under Windows 2000. It can be run from the Windows command prompt with the command line given above but I prefer to work in Windows Explorer. I have a Windows batch file called `TTF2T42WE.bat`, which is included with the program on CTAN. Place a copy of this file in your `SendTo` folder. Mine is in:

```
C:\Documents and Settings\Thomas H. Barton
                           \SendTo
```

(Again, this is folded over two lines only because of the text width.)

To use this batch file:

- Start Windows Explorer.
- Move to the folder containing the TrueType file.
- Right click on the TrueType file. This displays a list of activities.
- On this list, click on `SendTo` to display another list of activities, one of which is `TTF2T42WE`.
- Left click on `TTF2T42WE`. This invokes the conversion program. The three output files are

placed in the folder specified by the `/Path` key in program `TrueTypeToType42.ps`.

The TrueType fonts which came with my copy of Windows 2000 are automatically stored in the folder:

`C:\Winnt\Fonts`

This folder has special properties for working with the Windows font software which make it unsuitable for conversion. I therefore copied the font files to the folder `C:\TrueTypeFonts` and convert them from there.

2.5 Obtaining the files

The CTAN directory

`/fonts/utilities/TrueTypeToType42`

contains four files:

- 1) the program itself;
- 2) `TrueTypeToType42.ps`;
- 3) the Windows batch file, `TTF2T42WE.bat`;
- 4) and `ReadMe` files in text and HTML format.

◊ Thomas H. Barton
 Emeritus Professor of Electrical
 Engineering
 University of Calgary
 Canada
 thbarton@shaw.ca

The **Kerkis** font family

Antonis Tsolomitis

History

The history of the Greek language in **TeX** and “its friends” starts with the Greek fonts for the mathematics mode created by Donald Knuth. Many people (mainly Greek) being unable to typeset *essentially* in Greek they used the mathematics mode for short (and sometimes long) Greek passages. The first serious attempt for a Hellenized version of **TeX** was made by Silvio Levy and his Greek font based on Fermin Didot’s Greek. Since then, several attempts have been made, not to forget among them Kostas Dryllerakis’ and Yannis Haralambous’ fonts. The latest and most widely used method for typesetting Greek is based on Babel and uses the fonts by Claudio Beccari. The fonts are again based on Didot Greek and the language support macros in Babel were created by Apostolos Syropoulos.

All of these attempts were lacking in the same regard: they are all based on METAFONT fonts. And although METAFONT is, to my opinion, by far the best program for font design, sticking to it leaves excellent font families unavailable to the TeX world. In addition, users of the Greek language were not able to create a “decent” PDF file since METAFONT fonts render poorly on Acrobat Reader. There is more to say here; many colleagues in several mathematics departments around Greece were complaining that the Didot design, although excellent for philological work, was too cursive for mathematics.

This was the framework in which the *Kerkis* font family appeared. Since our main expertise is not in font design we did not concentrate on how beautiful the new font will be. The targets were

1. to provide a free font in the Type 1 format for L^AT_EX supporting the Greek language through Babel,
2. to provide a design less cursive than the Didot Greek,
3. to provide the necessary tools for installing any Type 1 font that includes Greek (e.g., encoding vectors),
4. to enable users of the Greek language to create decent PDF files that include Greek (via ps2pdf or pdfL^AT_EX),
5. to provide information on how to install fonts in other formats like TrueType or OpenType.

One could argue that it would be simpler to trace METAFONT fonts with programs such as Autotrace [5] and use the resulting Type 1 fonts for PDF generation. This may be true but such programs were not available when Kerkis started (at least in the open source community). More than that, I believe that the cb fonts (cb for Claudio Beccari) and the cmr fonts are very light for screen previewing. So we must add one more target to the above list

6. the new font must be heavier than cmr and cb and thus be optically compatible (from the weight point of view) with other Type 1 families, such as Times.

For context in the following detailed discussion, here are examples with both mathematics and text, in both serif and sans serif variants, of the Kerkis font in its current form:

The font’s name

Kerkis (Κέρκης) is the name of the highest mountain of the Aegean sea. Its altitude is 1497 meters and it is located in West Samos, the island of Pythagoras, Aristarchus, Epicurus, Aesop and others. Samos is

If

$$f(q, n, r) = \sum_{r_1 + \dots + r_n = r} \frac{\Gamma(qr_1 + 1) \cdots \Gamma(qr_n + 1)}{r_1! \cdots r_n!} \quad (1)$$

then

$$\int \|x_1 + \dots + x_s\|_q^{qr} d\mu \leq c(n, r, s, q) f(n, q, r).$$

Figure 1: Kerkis serif example.

If

$$f(q, n, r) = \sum_{r_1 + \dots + r_n = r} \frac{\Gamma(qr_1 + 1) \cdots \Gamma(qr_n + 1)}{r_1! \cdots r_n!} \quad (1)$$

then

$$\int \|x_1 + \dots + x_s\|_q^{qr} d\mu \leq c(n, r, s, q) f(n, q, r).$$

Figure 2: Kerkis sans serif example.

situated in the East Aegean Sea (just off the Turkish coast) and it is part of the Greek state. The island hosts the School of Sciences of the University of the Aegean.

Choice of base font

We looked around to see what Latin and freely distributed fonts were available. We thought that URW Bookman was a good choice, since we had seen Greek versions in print that looked compatible with the Latin ones, and satisfied items 2 and 6 above. Since we did not have the resources to redesign everything from scratch, we were pleased to find a suitable font with the Latin glyphs already done. We immediately contacted URW and they kindly gave us the permission to redistribute their Bookman inside Kerkis. Thus we could immediately start working on the Greek part.

The font’s structure

Kerkis is a purely neoclassical font:

Its stroke is modulated that is, of uneven width.

The axis is rationalist that is, vertical.

The serifs are adnate that is, they stem out of the penstroke in a gradual way (look at the letters b, f in Figure 3 and α in Figure 4). This leads the reader’s eye smoothly on the text line.

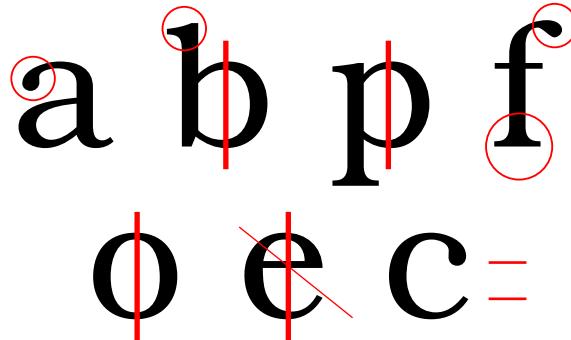


Figure 3: Sample of Latin letters: modulated penstroke (all letters), neoclassical (vertical) primary axis (axis of symmetry of the penstroke width), humanized (oblique) secondary axis for some letters (here the letter e), lachrymal terminals (here the letters a and top of f), adnate serifs (here the letters b and the bottom of f), moderate aperture (letter e and c)

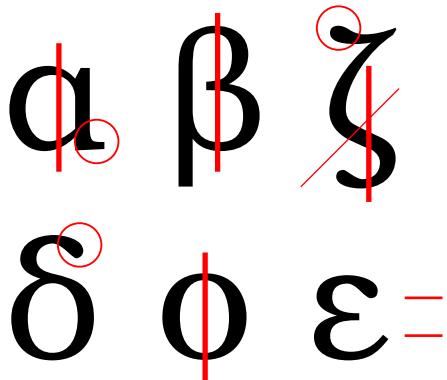


Figure 4: Sample of greek letters: modulated penstroke (all letters), neoclassical (vertical) primary axis (axis of symmetry of the penstroke width), humanized (oblique) secondary axis for some letters (here the letter ζ), lachrymal terminals (here the letters ζ and δ), adnate serifs (here the letters α), moderate aperture (letter ε)

The terminals are lachrymal that is, with tear drop shapes (look at the letters a, f in Figure 3 and δ, ζ in Figure 4).

The aperture is moderate (look at the letters c and e in Figure 3 and ε in Figure 4).

The italic is fully compatible with the roman. The serifs are on the baseline and at the x-height. Kerkis (as well as its predecessor Bookman) does not have serifs below the baseline with the only exceptions being the Greek letter chi (χ) and the Latin p and q.

The tools we used

Several Greek characters are the same as the Latin ones or close to some of them. For example the

Greek omicron ο is identical to the Latin o o and thus there was no need to be redesigned. Similarly, the Latin k can be transferred into the Greek kappa κ by lowering the top ascender of k. On the other hand letters like lambda λ or xi ξ have no counterpart in the Latin glyphs and had to be newly designed. All these letters were first drawn on paper and scanned with sane on a Linux machine. For templates we did use parts of the Latin font; for example the lachrymal terminals of the Latin part were used again for some of the Greek letters. Likewise with the serifs. The output was imported to the excellent program PfaEdit by George Williams [6] and traced there, either by hand or by the Autotrace program through the interface that PfaEdit provides. Then the splines were corrected by hand.

All of the functions one needs to create a font are provided by PfaEdit except one: the “change weight” function provided by commercial programs like Fontographer [7]. This is certainly a complex function, but absolutely useful for the creation of the bold and the small caps. For the small capitals we first apply a scaling with respect to the origin of the capitals at 75% (this can be done with PfaEdit). Then we apply the change weight function of Fontographer but we check the boxes “do not change the character’s width” and “do not change the character’s height”. With these restrictions many characters acquire the wider penstroke required by small capitals. For example, think of the letter O. This is constructed with two elliptical curves. The effect of the above method is that the inner ellipse becomes smaller (smaller axes) and the outer ellipse remains the same. Other letters, such as ones with serifs need manual tuning after these transformations. Finally there are cases where the above fails. For example, the sans serif letter I. The above procedure will leave such characters unaltered, and then intervention by hand is required.

All the bold glyphs (except the ones already provided by URW) and the full set of small caps were edited by hand, glyph by glyph, to bring them to the right look. However, one must be careful with Fontographer, as it fails to save kerning information for glyphs after position 256.¹ This means that the

¹ If you buy a commercial font that includes Greek you should check if the company uses Fontographer for the font generation. If true, ask for both the TrueType and the Type 1 fonts. Fontographer saves the kerning pairs in the TrueType format and you will be able to extract this information with PfaEdit. The resulting afm file is the correct one that must be used with the supplied pfb file. It is better to stay with the provided pfb file, since transforming the TrueType to Type 1 results in information loss for the glyphs.

program was used only for the splines and we had to preserve the **afm** files that PfaEdit provided.

Many kerning pairs are defined in the Kerkis fonts and this was done with the metrics dialog of PfaEdit. This is for example the case for Greek capital letters like A, Y, Δ and Λ. Compare

ΑΥΛΟΣ, ΛΥΔΙΑ (kerned) with
ΑΥΛΟΣ, ΛΥΔΙΑ (unkerned).

More important are the kernings for accents in polytonic Greek. For example, although Kerkis contains the letter 'A' pre-composed, Babel cannot use this pre-composed character (there is not enough space in the virtual font table) and so composes it by placing the letter A after the accent. At this point we again need to define a kerning pair otherwise the result is bad. Compare the kered 'A' with the unkerned 'A'.

The provided glyphs

Let us begin with the Latin part. URW Bookman is by no means complete for the Latin part. For example, it does not contain standard ligatures like ff, ffi and ffl. These were added and provided by Kerkis. Even further, Kerkis provides fj, tj and the more exotic fjj (for example “fiji”). In addition, Bookman does not provide true small caps (if you use the package `bookman.sty` you will get fake small capitals). Kerkis provides true small caps for the Latin characters too:

TRUE SMALL CAPITALS
FAKE SMALL CAPITALS
ΑΛΗΘΙΝΑ ΠΕΖΟΚΕΦΑΛΑΙΑ
ΨΕΥΤΙΚΑ ΠΕΖΟΚΕΦΑΛΑΙΑ

(However, the metrics of the small capitals of Kerkis need to be loosened. They are too tight in the current release.)

Finally, the Latin part of Kerkis contains many pre-composed accented letters that do not exist in URW Bookman. All fonts are provided in three weights: normal, bold and semi-bold.

Let us go now to the Greek part. This part supports the Greek language fully. Both monotonic and polytonic support is provided for all shapes and series. Here is an example:

“Ηρης, Θυμίαμα ἀρώματα.

Κυανέοις κόλποισιν ἐνημένη, ἀερόμορφε,
“Ἡρα παμβασίλεια, Διός σύλλεκτρε μάκαιρα,
ψυχοτρόφους αὐρας θνητοῖς παρέχουσα προσηγεῖς,
ὅμβρων μὲν μήτηρ, ἀνέμων τροφέ, παντογένεθλε
χωρὶς γάρ σέθεν οὐδὲν ὅλως ζωῆς φύσιν ἔγνω
κοινωνεῖς γάρ ἄπασι κεκραμένη ἡέρι σεμνῷ
πάντων γάρ κρατέεις μούνη πάντεσσι τ' ἀνάσσεις
ἡερίοις ροίζοισι τινασσομένη κατὰ χεῦμα.
ἄλλα, μάκαιρα θεά, πολυσώνυμε, παμβασίλεια,
ἔλθοις εύμενέουσα καλῶι γήθοντι προσώπῳ.

The fonts inside comply to the third version of the Unicode standard. Moreover, there are some special features that need to be mentioned:

Greek numerals Kerkis supports all of Babel's commands for Greek numerals, such as \greeknumeral and \Greeknumeral. For example, \greeknumeral{9999} gives $\text{\greeknumeral{9999}}$ and \Greeknumeral{9999} gives $\text{\Greeknumeral{9999}}$. Of course, these commands produce the numbers as they were written by the people in the Hellenistic era and not by ancient Greeks. Ancient Greeks wrote only in capitals using special glyphs—which Kerkis provides. They are accessible with the \athnum command provided by the package athnum² by A. Syropoulos; athnum is part of Babel. For example, the numbers 1, 2, 5, 10, 50, 100, 500, 1000, 5000, 10000 and 50000 are I, II, Π, Δ, Η, ΙΙΙ, X, Σ, M, Μ., while \athnum{9999} produces $\text{\athnum{9999}}$.

For the numerals used after the Alexandrian grammarians, Kerkis tries to clean up a little bit. The letter 'ῃ' from the cb fonts was added to Greek by the christian church (Hebrew letter “koph”) and has nothing to do with the ancient Greek language. So this design was removed from Kerkis and replaced by the true qoppa which is 'ϙ' and accessed with \qoppa (this glyph is also available by the cb fonts for the number 90, although not by default). The situation is similar for the number 6. cb fonts give 'Ϛ'. Again this symbol is non-existent in ancient Greek. The correct symbol for 6 is digamma and Kerkis provides this as the default: \greeknumeral{6} gives 'Ϛ'.

Glyph variations This is a unique feature of the Kerkis family. Greek typography has some letters whose shape varies, depending upon their position in the word. One such is sigma. When a sigma is at

² There has been a confusion about the Greek numerals. For reasons not to be explained here (see [8]), the true Greek numbers are known as Athenian numbers and these are the ones accessed with the *athnum* package.

the end of a word it changes from σ to ς . The same is true for several other letters when they appear at the beginning of a word. Thus we have $\beta\theta$, $\zeta\theta$, $\theta\theta$, $\rho\phi$, $\phi\phi$ plus two forms for π and ε . For example, compare

$\beta\iota\beta\lambda\iota\circ$, $\zeta\zeta\acute{\alpha}\nu\circ$, $\vartheta\mu\acute{\eta}\theta\kappa\alpha$, $\rho\acute{\o}\pi\tau\rho$, $\varphi\alpha\varphi\lambda\atilde{\alpha}\circ$
with

$\beta\iota\beta\lambda\iota\circ$, $\zeta\zeta\acute{\alpha}\nu\circ$, $\vartheta\mu\acute{\eta}\theta\kappa\alpha$, $\rho\acute{\o}\pi\tau\rho$, $\varphi\alpha\varphi\lambda\atilde{\alpha}\circ$
The second form of ρ (the initial one) will be reworked as the difference with the first form is not clear in small sizes. The second form of π is under construction.

Finally, the second form of ε poses a “virtual font” problem that is worth being mentioned. This form is ϵ (compare with ε). This second form is used when epsilon was followed by iota or iota-tonos (iota-dashed). So we would like the combinations $\varepsilon\iota$ and $\varepsilon\acute{\iota}$ to have the second form of epsilon, that is $\epsilon\iota$ and $\epsilon\acute{\iota}$. However, this appears to be impossible for the $\varepsilon\iota$ combination (in the previous lines ϵ is accessed directly with a `\symbol` command). The ligature mechanism provided by virtual fonts provides only the `=|>` and `=|>>` operators which make the mechanism skip letters forward. But we need to rescan for ligatures skipping to the left! An operator (denoted, say, by `=|<` and `=|<<`) would enable the ligature mechanism after constructing iota-tonos (as the ligature of tonos and iota) to skip backwards(!) and recognize $\varepsilon\iota$ as a new ligature to act upon. We think that this addition would help the installation of complex typefaces in TeX. We would welcome such an addition to the virtual font mechanism.

The small capitals series

As said before, Kerkis provides true small capitals for both the Latin alphabet and the Greek alphabet. When writing in capitals we do not write accents in Greek. However, Kerkis provides accented small capitals as a stylistic alternative: $\text{A E H I O Y \ddot{O}}$ and $\text{Á É Í Ó Ý \ddot{Á}}$.

The small capitals font includes the old style numbers. Thus `0123456789` are accessed with

`\textsc{0123456789}`.

Small capitals are available in oblique form as well: `0123456789\textit{AB}`. `kerkis.sty` provides the commands `\scslshape` and `\textscscl{}` for accessing these glyphs.

The semi-bold series

The semi-bold series is as complete as the normal weight series. It is accessed by the commands

`\sbseries` and `\textsb{}`. Here's an example showing the different weights:

'Εκ Διός αρχόμεσθα	(normal)
'Εκ Διός αρχόμεσθα	(semi-bold)
'Εκ Διός αρχόμεσθα	(bold)

The italic shape is also available:

'Εκ Διός αρχόμεσθα	(normal)
'Εκ Διός αρχόμεσθα	(semi-bold)
'Εκ Διός αρχόμεσθα	(bold)

The italic shape

Kerkis provides a true italic (not just the roman slanted). The Latin part is again based on URW Bookman but completed with missing glyphs such as the f-ligatures, as before. The Greek is also a true italic with the exception of $\omega\phi\phi\omega$ which are essentially the roman slanted (this was easily done with PfaEdit). Nonetheless, those letters with two forms have a true italic for the second form: $\rho\varphi$.

The upright italic shape

An upright italic shape is available through skewing. The shape can be called with the commands `\textui{}` and `\uistyle`. It looks like this:

The brown fox jumps	(n/ui)
<i>The brown fox jumps</i>	(n/it)
The brown fox jumps	(sb/ui)
The brown fox jumps	(sb/it)
The brown fox jumps	(b/ui)
The brown fox jumps	(b/it)

The sans font

Kerkis Sans is the companion sans serif face that comes with Kerkis. The font is based on Avant Garde. The Latin part comes from a free font found on the Internet. We improved it considerably by simplifying curves and adding missing ligatures. The Greek part was again newly designed to match the Latin part. Again PfaEdit was the main tool plus Fontographer for its “change weight” function.

The choice was made on the basis of common elements in the structure of Avant Garde and Bookman. They are similar in spirit, aperture, eye size and, based on personal judgment, they go nicely together.

Kerkis Sans fully supports the Greek language through Babel. Here is an example:

Ἐκ Διος ἀρχώμεσθα, τὸν οὐδέποτ’ ἄνδρες ἔῶμεν
ἄρρητον: μεσταὶ δὲ Διὸς πᾶσαι μὲν ἀγυιά,
πᾶσαι δ’ ἀνθρώπων ἀγοραί, μεστὴ τὲ θάλασσα
καὶ λιμένες: πάντη δὲ Διὸς κεχρήμεθα πάντες.

The euro

The symbol for the Euro is provided with the \euro command while in Greek text (in the LGR encoding):

Roman	€	€	€	€
Sans	€	€	€	€

Usage tips

This section is set with Kerkis Roman and Kerkis Sans as a sample, using both fonts. If one uses Kerkis for mathematical texts s/he can use the Times math fonts (`mathptm.sty`) and still draw the math alphabet from the Kerkis Italic font (`kmath.sty`). For slides one can use the Kerkis Sans with mathematics from the `cmbright.sty` package. In both cases `kerkis.sty` and `kmath.sty` must be loaded *after* the above packages. Here is a sample with Greek:

Οι περιλαμπτοί Ναοί, οι εκπληκτικοί κοινωνικο-πολιτικοί θεσμοί των Αθηνών και της Σπάρτης, τα θεϊκά αγάλματα, η παιδευτική τραγωδία, η εξυφωτική Φιλοσοφία, ήσαν οι καρποί μιάς μακράς συνειδησιακής διεργασίας που επιτελέσθηκε στην προχριστιανική Ελλάδα. Όμως ο απλός θαυμασμός, η βαριά νοοταλγία, ή η διαλεκτική μόνο ενατένιστη του παρελθόντος δεν οδηγούν παρά σε στείρα προγονολατρεία.

Αυτό που απορροφά το δικό μας ενδιαφέρον, είναι οι ρίζες των συλλήψεων που εδημιούργησαν το μεγαλείο του ΕΛΛΗΝΙΚΟΥ ΕΘΝΟΥΣ. Μας ενδιαφέρει ο τρόπος που ο κάθε «πολίτης» ή «όμοιος» αντιλαμβάνετο τη ΦΥΣΗ και τον εαυτό του, μας ενδιαφέρουν τα συναισθήματα, τα όνειρα και οι προθέσεις που οδήγησαν στη μυθοπλασία, τη λογική σύλληψη του Κόσμου και στο μεγαλείο της φιλοσοφικής διανοήσεως. (The text is from <http://www.diipetes.gr>.)

Conclusion

Kerkis was, and remains, an experiment of how a Type 1 font with Greek glyphs may be used with L^AT_EX. Of course there are still many design issues but all the necessary information for installing and using a Type 1 font is available to users of the Greek language. This also includes the encoding vectors. The project triggered several articles written in the journal of the Greek T_EX users group ΕҮΤҮΠӨΝ [1, 2, 3].

ACKNOWLEDGEMENT: I want to thank Kostas Lionakis who helped me at the beginning of the project, and Vassilis Metaftsis, a colleague at the

Department of Mathematics, University of the Aegean, who proposed and supported the project in several ways. Special thanks go to Apostolos Syropoulos who helped me very much with the kerkis style file.

Many thanks to numerous people from Greece and abroad who have used the fonts and mailed me bug reports.

References

- [1] A. Syropoulos and A. Tsolomitis, TrueType fonts and L^AT_EX 2_ε, *Τὸ ΕҮΤҮΠӨΝ*, No. 2, p17–22, (1999) (through METAFONT) (in Greek).
- [2] A. Syropoulos and A. Tsolomitis, Installing new fonts for L^AT_EX 2_ε, *Τὸ ΕҮΤҮΠӨΝ*, No. 3, p57–68, (1999) (in Greek).
- [3] A. Tsolomitis, Installing TrueType fonts for L^AT_EX 2_ε, *Τὸ ΕҮΤҮΠӨΝ*, to appear (in greek).
- [4] A. Syropoulos, A. Tsolomitis and N. Sofroniou, Digital Typography using L^AT_EX, Springer Professional Computing, Springer-Verlag, 2002.
- [5] <http://autotrace.sourceforge.net>
- [6] <http://pfaedit.sourceforge.net>
- [7] <http://www.macromedia.com/software/fontographer>
- [8] Blasis Rassias, Adventures of the Greek alphabet and the Greek numbering system, Diipetes 44, 2003.

◊ Antonis Tsolomitis
University of the Aegean
Department of Mathematics
832 00 Karlovasi, Samos
Greece
atsol@aegean.gr
<http://iris.math.aegean.gr/~atsol>

Euler-VM: Generic math fonts for use with L^AT_EX

Walter Schmidt

Abstract

The Euler math fonts are suitable for math typesetting in conjunction with a variety of popular text fonts which do not provide math character sets of their own. Euler-VM is a set of virtual fonts based on Euler and Computer Modern, accompanied by a macro package for easy use with L^AT_EX.

The Euler math fonts

“With Donald Knuth’s assistance and encouragement, Hermann Zapf, one of the premier font designers of this century, was commissioned to create designs for Fraktur and script, and for a somewhat experimental, upright cursive alphabet that would represent a mathematician’s handwriting on a blackboard and that could be used in place of italic. The designs that resulted were named Euler, in honor of Leonhard Euler, a prominent mathematician of the eighteenth century. Zapf’s designs were rendered in METAFONT code by graduate students at Stanford, working under Knuth’s direction. [...] Knuth also noticed that the style of some symbols in the Computer Modern extension font, in particular the integral sign, was too slanted to be attractive with Euler, and consequently he prepared a new (partial) extension font for use with Euler.” [3]

Knuth’s book *Concrete Mathematics* [1] was typeset using the Concrete font family for text and the Euler fonts for the formulas. With L^AT_EX, the particular math font setup of this book can be mimicked through the package `euler.sty` [2]. Later it became obvious that the Euler math fonts match other text font families equally well.

Unfortunately, the Euler fonts do not comprise all symbols required for mathematical typesetting with L^AT_EX. As a result, the `euler` package needs to redefine most of L^AT_EX’s math font setup, so that certain symbols are taken from Euler, whereas others come from Computer Modern. This has resulted in many problems and prevented the widespread use of the package beyond its initial purpose.

A new interface to the Euler fonts

Euler Virtual Math (Euler-VM) is a set of *virtual* fonts based primarily on the Euler fonts. The missing symbols are “stolen” from Computer Modern through the virtual font mechanism, rather than at T_EX level, and the encoding follows that of the classical Computer Modern math fonts as far as possible. This approach has several advantages over immediately using the *real* Euler fonts, as implemented in the `euler` package. Most noticeably, less T_EX resources are consumed, the appearance of various math symbols is improved, and there are (almost) no more compatibility problems with other packages. Thus, Euler-VM constitutes, together with the related macro package `eulervm`, a “generic” math font set for L^AT_EX.

The initial reason for creating Euler-VM was the fact that the `euler` package does not provide a usable `\hbar` or `\hslash`, and the `\hslash` from

the `amssymb` package cannot be used, because the latter follows the CM Roman style (rather than matching Euler). This made the beautiful Euler fonts essentially unusable for physics and related fields. The only way to fix this was to provide a “faked” Euler-style `\hslash` through the virtual font mechanism. As a beneficial side effect, it was possible to make the layout of the new virtual fonts compatible with Computer Modern Math to a large extent, and—since the style file had to be rewritten anyway—many further improvements were introduced.

Typographical considerations

From a technical point of view, the Euler fonts can be used together with arbitrary text fonts. From a typographical point of view, however, it is obvious that this will not always yield an attractive result. In particular, using Euler with Computer Modern is not a good idea at all!

Beside the above-mentioned “Concrete”, the typeface families Palatino, Aldus and Melior blend well with Euler. This is not surprising, since they were also designed by H. Zapf. One might think of using Optima with Euler, but the typefaces are too similar, thus making formulas hard to read.

The Euler math fonts have also proven to go sufficiently well with other typefaces, such as Minion, that exhibit a similar weight (stroke width) and x-height.

Unfortunately, the popular “Garamond-style” typefaces Adobe Garamond and Stempel Garamond do *not* go well with Euler, because their x-height is too small. A Garamond descendant that yields a more pleasing result together with Euler is Sabon, because of its somewhat larger x-height.

Using the Euler math fonts in conjunction with a *sans serif* typeface for text is possible, too. The only sans serif typeface which I know so far to work well is “Syntax”; of course there may be others.

The L^AT_EX macro package `eulervm`

Loading the `eulervm` package redefines L^AT_EX’s math font setup, so that the Euler-VM fonts and the default body font are substituted for CM Math and CM Roman. Roughly:

- CM Math Italic is replaced with Euler Roman.
- CM Calligraphic is replaced with Euler Script.
- “Large” operators and delimiters are replaced with alternative symbols matching the Euler style.
- In numbers and operator names, CM Roman is replaced with the default text font.

The model has the parameters f_i , E_i , C and r ($0 \leq r \leq 1$). The oscillator strengths f_i and the atomic level energies E_i should satisfy the constraints

$$f_1 + f_2 = 1 \quad (1)$$

$$f_1 \ln E_1 + f_2 \ln E_2 = \ln I \quad (2)$$

The parameter C can be defined with the help of the mean energy loss dE/dx in the following way: The numbers of collisions (n_i , $i = 1, 2$ for the excitation and 3 for the ionisation) follow the Poisson distribution with a mean number $\langle n_i \rangle$. In a step Δx the mean number of collisions is

$$\langle n_i \rangle = \Sigma_i \Delta x \quad (3)$$

The mean energy loss dE/dx in a step is the sum of the excitation and ionisation contributions

$$\frac{dE}{dx} = \left[\Sigma_1 E_1 + \Sigma_2 E_2 + \Sigma_3 \int_1^{E_{\max}+I} E g(E) dE \right] \quad (4)$$

Figure 1: Using the Palatino typeface for text and the Euler fonts for the formulas via the package `eulervm`.

For instance, to use Palatino for text and Euler for the formulas, you would need only the following commands:

```
\renewcommand{\rmdefault}{ppl}
\usepackage{eulervm}
```

Figure 1 shows an example, using these typefaces.

The packages `amssymb` or `eufrak` can be loaded to provide extra math symbols and the Euler Fraktur letters, and `eulervm` is also fully compatible with with AMS-L^AT_EX, i.e., `amsmath.sty`.

The package provides a number of options and non-standard facilities which are described in detail in the accompanying documentation. The present article will focus only on two features:

Scaling Loading the package with the option `small` causes the Euler fonts to be loaded at 95% of their nominal size, thus blending better with certain text font families, for instance Aldus or Minion. The option acts also on the AMS symbols and Euler Fraktur fonts; any other math fonts used in the document remain unaffected.

Numbers and punctuation The normal behavior of the `eulervm` package is to take the digits, the comma and the period in math mode from the default text font family. Text fonts are, however, not always suitable for typesetting math: The digit “1” may not be distinguishable clearly enough from the letter “l”, or the style of the digits may not mesh well with the Euler letters. Furthermore, most text fonts are scaled linearly, so that the digits may become too thin when used in super- or subscripts.

As a workaround the `eulervm` package provides the option `euler-digits`, which makes the digits, the comma and the period come from Euler Roman in math mode. Since the Euler fonts are available with designs sizes of 10 pt, 7 pt and 5 pt, no problems are to be expected in super- and subscripts! The option should be used with care, because the input `1.23` will then yield a different result than `1.23`, and thus one will in each case have to decide whether an input fragment is a mathematical or a non-mathematical entity. The example shown as figure 1 was created using this option.

Availability

The Euler fonts are supplied with all modern T_EX systems, both in METAFONT and in Type 1 (Post-Script) format. Most likely, also the virtual Euler-VM fonts, the package `eulervm`, and the related documentation are already part of your T_EX distribution. On CTAN, they are available from the directory `fonts/eulervm/`.

References

- [1] Ronald Graham, Donald Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison-Wesley Publishing Company, Reading, 1989.
- [2] Frank Jensen. The `euler` package, 1995. CTAN: `macros/latex/contrib/euler/euler.dtx`.
- [3] American Mathematical Society. User’s guide to AMSFonts. CTAN: `fonts/amsfonts/doc/amsfndoc.ps`.

◊ Walter Schmidt
Nürnberg Straße 76
Erlangen
Germany
`w.a.schmidt@gmx.net`
`http://home.vr-web.de/was`

Software & Tools

Rambutan: Literate programming in Java

Prasenjit Saha

Introduction. RAMBUTAN is a literate programming system for Java with TeX, closely resembling CWEB and the original WEB system.* I developed it using Norman Ramsey's Spidery WEB.

This article is also the manual, as well as an example of a RAMBUTAN literate program; that is to say, the file `Manual.w` consists of code and documentation written together in the RAMBUTAN idiom. From this common source, the RAMBUTAN system does two things:

`javatangle Manual`

extracts a compilable Java applet to compute the first N primes, and

`javaweave Manual`

produces a TeX file laying out code and documentation together, including these words.

Actually, the above is a slight oversimplification: `Manual.w` could have contained the whole source, but in fact I have distributed the source between `Manual.w`, `Primes.w`, and `Manual.ch`, in order to illustrate multiple source files—but more on that later.

The example code follows this preamble, and introduces the main ideas of literate programming, as relevant to RAMBUTAN. (The reader is assumed to be reasonably familiar with Java and TeX.) After the program there are short explanations of all of RAMBUTAN's features. The important features are few and simple and explained first; the arcana for literate-programming experts come later. A brief annotated bibliography concludes.

1. Computing primes. This is a Java applet that takes two numbers N_1, N_2 and prints out the N_1 -th prime to the N_2 -th prime.

Like all literate programs, this one consists of a series of numbered **sections**. We are currently in section 1. (Any material before section 1 is

* In other words, what you would expect to be called JavaWEB. But since JavaWEB sounds too much like a Sun trademark and is a clumsy word anyway, the system as a whole is called RAMBUTAN. But inside RAMBUTAN the usual naming conventions apply: the preprocessors are called `javatangle` and `javaweave` and the TeX macro file is called `javaweb.tex`. (A rambutan, by the way, is a delicious fruit, not unlike a lychee, widely enjoyed in Java and elsewhere.)

called **limbo**; in this case, the introduction.) Most sections consist of a short **text** part followed by a short **code** part. Some sections (such as this one) contain only text, some others contain only code.

Section 1 is always a **starred section**. That just means it has a title: ‘Computing primes’ in this case. The title is supposed to describe a large group of consecutive sections, and gets printed at the start and on the page headline. Long programs have many starred sections, which behave like chapter headings.

The source for this section begins

@* Computing primes. This is...

In the source, @* begins a starred section, and any text up to the first period makes up the title.

2. This is an ordinary (i.e., unstarred) section, and its source begins

@ This is an ordinary...

In the source, @ followed by space or tab or newline begins an ordinary section.

In the next section things get more interesting.

3. ⟨ Imported packages 3 ⟩ ≡

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
```

This code is used in section 5.

4. The source for section 3 begins

```
@ @<Imported packages@>=
import java.applet...
```

The result is to make @<Imported packages@> an abbreviation for four Java statements—note the = in the source.

The bit ⟨ Imported packages 3 ⟩ is called the **section name**, not to be confused with the title of a starred section. Notice how RAMBUTAN has attached the number 3 and inserted a forward reference to section 5.

5. Now we have a whole Java class in abbreviated form. The section ⟨ Imported packages 3 ⟩ is used here, as promised; so are other sections that haven't been defined yet.

(`Primes.java` 5) ≡

```
⟨ Imported packages 3 ⟩
public class Primes extends Applet
    implements ActionListener
{ ⟨ Fields 12 ⟩
    ⟨ Code for initializing 13 ⟩
```

```
< The event handler 18 >
}
```

6. The source for section 5 is

```
@ Now we have...
@(Primes.java@)=
<Imported packages@=
public class Primes extends Applet
    implements ActionListener
{ @Fields@
    @Code for initializing@
    @The event handler@
}
```

Note the left parenthesis in `(Primes.java 5)`, in contrast with the angle brackets used for other section names. The source for the section name `(Primes.java 5)` is

```
@(Primes.java@)=
```

rather than

```
@<Primes.java@=
```

Because of this, section 5 is an **output section**: its expansion is output to the specified file, `Primes.java`.

7. That's it for the really essential features of a literate programming system: `javatangle` collects the code fragments into a compilable program and `javaweave` cross-references the sections. The remaining features of RAMBUTAN are basically refinements. This example will illustrate a few more features, but the full list can wait till the next chapter of this manual. Meanwhile we'll get on with explaining the program.

8. The algorithmic job of this program is to produce a list of primes, which it does inductively.

First, note that testing p for primeness is easy if we know all the primes $< p$. We set $pmul[j]$ to consecutive odd multiples of $prime[j]$ and check whether we ever hit p . It is enough to try multiples of primes $\leq \sqrt{p}$.

```
< Set factor ← true if p is a multiple of a
  prime 8 >=
for (int j ← 2; psqr[j] ≤ p; j++)
{ while (pmul[j] < p) pmul[j] ← 2 * prime[j];
  if (pmul[j] ≡ p) factor ← true;
}
```

This code is used in section 9.

9. Now suppose we have found $prime[1]$ through $prime[k - 1]$. We then try successive odd numbers $p > prime[k - 1]$ until we find a prime p .

```
< Compute prime[k] 9 >=
```

```
if (k ≡ 1) prime[k] ← 2;
else if (k ≡ 2) prime[k] ← 3;
else
    for (int p ← prime[k - 1] + 2; ; p ← 2)
        { boolean factor ← false;
          < Set factor ← true if p is a multiple of a
            prime 8 >
          if (¬factor)
              { prime[k] ← p; break;
          }
        }
    pmul[k] ← prime[k];
    psqr[k] ← prime[k] * prime[k];
```

This code is used in section 20.

10. `< Arrays for computing primes 10 >≡`

```
int[] prime ← new int[N2 + 1];
int[] pmul ← new int[N2 + 1];
int[] psqr ← new int[N2 + 1];
```

This code is used in section 20.

11. When we use the code from section 8 in section 9, the source actually gives the section name as

```
<Set |factor=true| if...@=
```

with the three dots. Once a section name has appeared in the source RAMBUTAN can complete it from this kind of three-dot shorthand. (And by the way, RAMBUTAN sensibly collapses extra spaces or newlines in section names.)

Another feature is the usage `|factor=true|` which tells `javaweave` to typeset the enclosed text in code-style.

12. The rest of this program is the GUI. Here are the elements for it. (We restrict ourselves to Java 1.1, which more people's browsers will interpret than Java 2.)

The code here includes some comments; literate programs usually need comparatively few comments. RAMBUTAN knows about the `//` comment syntax in Java but not about `/*...*/` comments.

If you need to include strings in the `.java` file that RAMBUTAN can't parse, enclose them in `@...@`. A `@/** javadoc comment */@` can be inserted in this way.

```
< Fields 12 >≡
int N1 ← 0, N2 ← 0; TextField N1_txt, N2_txt;
Button run; Panel panel; ... for input
TextArea disp; ... for output
```

This code is used in section 5.

13. This method makes a labelled `TextField` and attaches it to `panel`.

```

⟨ Code for initializing 13 ⟩ ≡
  TextField new_tf(String str, int n)
  { Panel p ← new Panel();
    TextField t ← new TextField(n);
    p.add(new Label(str, Label.CENTER));
    p.add(t);
    p.add(new Label("✉", Label.CENTER));
    panel.add(p);
    return t;
  }
}

```

See also section 15.

This code is used in section 5.

14. Section 15 has the same section name as section 13. When two or more sections have the same name, RAMBUTAN automatically concatenates them. Note the forward reference in section 13 and the continuation mark ‘+ ≡’ in section 15.

15. The applet's *init()* method. Because *disp* here is **Center** in a *BorderLayout*, it will take up any spare space.

```
< Code for initializing 13 >+ ≡  
public void init()  
{ panel ← new Panel();  
    N1.txt ← new_tf("N1", 4);  
    N2.txt ← new_tf("N2", 4);  
    run ← new Button("run");  
    panel.add(run);  
    run.addActionListener(this);  
    disp ← new TextArea();  
    disp.setEditable(false);  
    setLayout(new BorderLayout());  
    add("North", panel); add("Center", disp);  
}
```

16. Some (very rare) sections have a **definitions** part.

```
define intN(i) ≡  
    Integer.parseInt( N@& i@& _txt.getText() )
```

17. In section 16 we have a macro. The `\&` removes any space between its neighbors in the `java` file. Accordingly, `intN(1)` will do something with the variable `N1.txt`, and so on.

18. \langle The event handler 18 $\rangle \equiv$

```
public void actionPerformed(ActionEvent event)
    { run.setEnabled(false);
        try
            { int n1  $\leftarrow$  intN(1); int n2  $\leftarrow$  intN(2);
                if (n1  $\geq$  1  $\wedge$  n2  $\geq$  n1)
                    { N1  $\leftarrow$  n1; N2  $\leftarrow$  n2;
```

```

    < Compute and display primes 20 >
}
else
{
    < Restore old values of N1, N2 19 >
}
}
catch (NumberFormatException ex)
{
    < Restore old values of N1, N2 19 >
}
run.setEnabled(true);

```

This code is used in section 5.

```

19. < Restore old values of N1 , N2 19> ≡
    if (N1 ≡ 0)
        { N1_txt.setText(""); N1_txt.setText("");
        }
    else
        { N1_txt.setText(Integer.toString(N1));
          N2_txt.setText(Integer.toString(N2));
        }
    }

```

This code is used in section 18.

20. \langle Compute and display primes 20 $\rangle \equiv$
 \langle If too extravagant **return** 21 † \rangle
StringBuffer lyne \leftarrow **new** *StringBuffer()*;
disp.setText("");
 \langle Arrays for computing primes 10 \rangle
for (**int** *k* \leftarrow 1; *k* \leq N2; *k*++)
{ \langle Compute prime[*k*] 9 \rangle
String num \leftarrow
new *String*(*Integer.toString*(prime[*k*]) +
" ");
if (*k* \geq N1)
{ *lyne.append*(*num*);
if (*lyne.length()* < 64)
disp.append(*num*);
else
{ *disp.append*("\\n" + *num*);
lyne \leftarrow **new** *StringBuffer*(*num*);
}
}
}

This code is used in section 18.

```

21†. < If too extravagant return 21† > ≡
  if ( $N2 - N1 \geq 2000$ )
    { disp.setText("Printing more than ");
      disp.append("2000 primes");
      disp.append("is too boring\n");
      disp.append("Try increasing N1");
      run.setEnabled(true);
      return;
    }
  
```

This code is used in section 20.

22[†]. The source of this program is actually in the file `Primes.w`, while `Manual.w` says

`@i Primes.w`

to include that file.

If you look in `Primes.w`, you will find that it considers printing > 1000 primes as already too boring, rather than > 2000 primes. The relevant lines of code have been overridden by the **change file** `Manual.ch`. This last file contains

```
@x
if (N2-N1 >= 1000)
{ disp.setText("Printing more than ");
{ disp.setText("1000 primes ");
@y
if (N2-N1 >= 2000)
{ disp.setText("Printing more than ");
{ disp.setText("2000 primes ";
@z
```

and continues with a similar construction containing this section. The section numbers 21 and 22 have daggers attached to indicate that a change file is involved.

A change file consists of constructions of the type

```
@x
<Lines quoted from the source file>
@y
<Replacement lines>
@z
```

The change-file name is an optional second input parameter on the command line. Thus

`javatangle Manual.w Manual.ch`

or simply

`javatangle Manual Manual`

and similarly for `javaweave`.

23. Control codes. Following are the complete set of control codes understood by RAMBUTAN. Only the first two sections are really important.

24. Basic controls. These cover the essentials of a literate programming system.

`@<space>` Begins a new section. (A tab or newline is also read as *space* here.)

`@*<group title>`. Begins a starred section.

`@<section name>@>=` Section definition, which is really the code-part definition. A section can have at most one such definition. The code can be continued in later sections (see examples in sections 13 and 15).

`@<section name>@>` Code-part of the named section used. A section can have any number of these.

After a section name has first appeared (whether as definition or use) it can be abbreviated using three trailing dots. (See example in section 11).

`@<(filename)>@>=` Output-section definition. Written to the named file.

This feature allows `javatangle` to write multiple `.java` files from a single source, which is useful if you have many short public classes. `javaweave` still generates a single `.tex` file.

`@u` “Unnamed” output-section; the filename is inferred by replacing the main source file’s extension with `java`. For this program, that default file would be `Manual.java` (but it is not in fact used).

In `WEB` and `CWEB` a feature analogous to `@u` is the only way to output code, but in `RAMBUTAN` `@u` is less important.

25. File controls. These are for using multiple `RAMBUTAN` source files:

`@i <filename>` Includes the file. Must be followed by a newline.

`@x...@y...@z` Valid only in change files. The control codes `@x`, `@y`, `@z`, must appear at the beginning of a line, and the rest of such a line is ignored. Any material outside the blocks `@x...@y...@z` is also ignored.

26. Special tangle controls. These are for getting special effects in the output `java` file. We have met the first three in the prime-numbers example.

`@d <name> = <defn>` Defines a macro. [`@D` is equivalent.]

`<token1> @& <token2>` `javatangle` outputs the two tokens without intervening space.

`@=<code text>@>` `javatangle` passes the `<code text>` verbatim into the `java` file.

`@'<digits>` An octal constant (must be positive). For example, `@'100` tangles to `64` and weaves to `'100`.

`@"<digits>` A hexadecimal constant. For example, `@"DODO` tangles to `53456` and weaves to `"DODO`.

27. Special weave controls. These are for fine-tuning the typesetting. We have met the first one in the prime-numbers example.

`|<code fragment>|` Used in text, or section names, to format a code fragment in code-style. The `<code fragment>` must not contain section names. [This is the only RAMBUTAN control code not involving `\@`.]

`\@t<text>\@>` The `<text>` is put into a `\TeX \hbox`. For example, `\size < \@t$2^{15}\$@\@>` produces `size < 215`. The `<text>` must not contain newlines.

`\@f <id1> <id2>` Format definition; an optional comment enclosed in braces can follow. [`\@F` is equivalent.] Makes `javaweave` treat `<id1>` as it currently treats `<id2>`. Format definitions appear between the text part and the code part of a section, together with `\@d` macros (in any order).

`\@/` Produces a line break. [Should not be used inside expressions.]

`\@#` Like `\@/` but adds some extra vertical space.

`\@-` Like `\@/` but indents the next line, to show that it is a continuation line.

`\@|` Recommends a line break, but does not force one. [Can be used inside expressions.]

`\@+` Cancels a line break that might otherwise be inserted by `javaweave`.

`\@,` A thin space.

`\@;` Formats code as if there were a semicolon there.

`\@@` `javaweave` outputs a single `\@`. This cannot be used inside `\@<text>\@>` or similar contexts. An alternative is `\AT!` in text.

28. Index controls. These are for fine-tuning the index, and ignored by `javatangle`.

A reserved word or an identifier of length one will not be indexed except for underlined entries.

`\@^<text>\@>` The `<text>` will appear in the index in roman type.

`\@.<text>\@>` The `<text>` will appear in the index in typewriter type.

`\@:<text>\@>` In the index, the `\TeX` file will have `\@{<text>}` and the user can define `\@` freely in `\TeX`.

`\@!<token>` In the index entry for `<token>` the section number will be underlined.

29. Other information. The input syntax for `javatangle` is

`javatangle <source file> <change file> -I<path>`

The `<source file>` has default extension `.w` while the optional `<change file>` has default extension `.ch` and the default `<path>` is the current directory.

The input syntax for `javaweave` is similar:

`javaweave <source file> <change file> -x -I<path>`

The additional `-x` option omits the index.

Both programs also implement the `--version` and `--help` options.

30. If you use `pdftex` on the output of `javaweave`, section-number cross-references will be clickable. Using `\LP{<section number>}` in text will also give you a clickable link.

31[†]. `\TeX` macros are in `javaweb.tex`, which is based on the original `webmac.tex` but considerably modified and reorganized. The default format is a standalone Plain `\TeX` document, but if you want to use `LATEX`, or embed within a larger document (such as this article in `TUGboat` style) minimal changes are necessary.

32. To get a table of contents (listing the starred sections), put

`\contents`

at the very top of the input file. Unlike in `WEB` and `CWEB`, the table of contents comes first. So you will have to run `\TeX` twice to get an up-to-date list.

If you use `pdftex` the contents will also appear as bookmarks.

33. If you are using a change file and want to view only the changed sections, put

`\let\maybe\iffalse`

in the source file or the change file, in the limbo part.

Using this option with `pdftex` will generally produce a lot of clickable links to absent sections, but such links will still behave sensibly.

34. Bibliography. The basic introductory reference on literate programming in general is Knuth's article:

Literate Programming, in *The Computer Journal* **27**, 97-111 (1984).

which is also reprinted in Knuth's anthology of the same title. (The prime-numbers example in this manual is adapted from the Knuth article.)

For reviews and links on all aspects of literate programming, see Daniel Mall's literate programming web site:

www.literateprogramming.com

Normal Ramsey's Spidery WEB (a generator for **tangle** and **weave** programs) is described in:

Literate programming: Weaving a language-independent WEB, Communications of the ACM, **32**, 1051–1055 (1989)

and archived on CTAN. I made a few modifications (such as adding hyperlinks) to the Spidery WEB system itself; such modifications are through change files, so Ramsey's original code is untouched. The change files are included in the RAMBUTAN distribution. Ramsey himself now deprecates Spidery WEB and favors the simpler **noweb** system:

Literate programming simplified, IEEE Software, **11**, 97–105 (1994)

which is language independent but sacrifices many features, including automatic cross-referencing. See also Ramsey's web site:

www.eecs.harvard.edu/~nr

I use **noweb** too, but I think Spidery WEB still has a place.

Finally, the RAMBUTAN distribution is available from my web site:

ankh-morpork.maths.qmul.ac.uk/~saha

and is also archived on CTAN.

I thank Karl Berry for several improvements to this article.

- ◊ Prasenjit Saha
Astronomy Unit
Queen Mary and Westfield College
University of London
London E1 4NS
United Kingdom
p.saha@qmul.ac.uk

Graphics

Eukleides: A geometry drawing language

Christian Obrecht

As a mathematics teacher in a French high school, I have to compose a rather large number of documents for my students, containing both text and formulas. In my point of view, L^AT_EX is the best

Editor's note: This article is a reiteration of the article by the same title in *TUGboat* **22**:4, pp. 334–337. Unfortunately, owing to an editorial glitch, the figures in that version were not properly displayed. We regret the mixup.

tool in such a situation, combining efficiency and high quality. Very often, these documents should be illustrated with geometric figures. I first used the excellent *PSTricks* package to draw them. I didn't want to use WYSIWYG software instead, because I wanted to keep following L^AT_EX's philosophy, that is: What You Mean Is What You Get. Unfortunately, *PSTricks* isn't designed for geometry at all and is rather inappropriate in many situations.

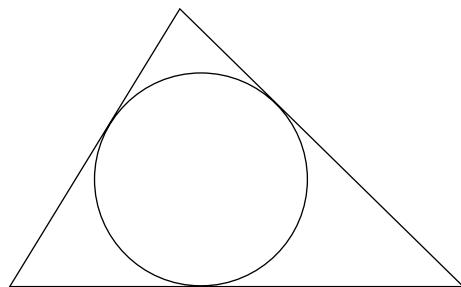
One night, I wanted to draw a triangle with an inscribed circle, so I had to compute by hand the coordinates of the center and the radius of this circle, which is quite boring. During these calculations, I realized that they could easily be done by a computer, and that gave me the idea to create Eukleides, a geometry drawing language. My goal was to make it as close as possible to what mathematics teachers would say to describe geometric figures. For instance, the former problem, written as an exercise, could be:

Let *ABC* be a triangle and *I* its inscribed circle. Draw *ABC* and *I*.

In Eukleides, it gives:

```
A B C triangle
I = incircle(A,B,C)
draw(A,B,C) ; draw(I)
```

Which leads to the following graphical result:



Once the design of the language was done, I wrote *eukleides*,¹ a compiler which translates Eukleides code into *PSTricks* macros. This program can run as a filter. That is, it can take a L^AT_EX source containing Eukleides code, and replace this code with *PSTricks* macros, producing a ready-to-T_EX file.

There's also a graphical interface to the language, with additional interactive features, named

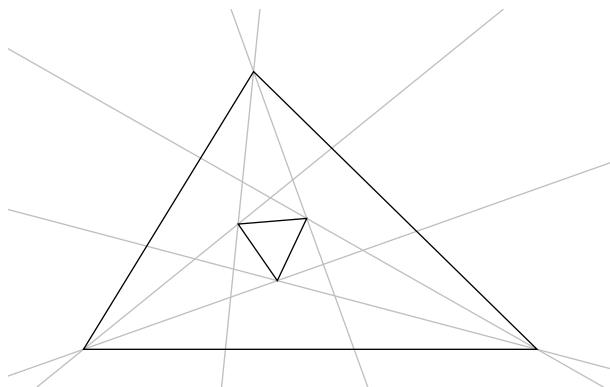
¹ It was formerly named *euklides*, but this name was already given to other geometry software.

eukleides. It was first meant for classroom presentations, but it can also be seen as a tool to compose and tune some Eukleides code for later inclusion in a L^AT_EX source.

Both programs are released under the GNU Public License. They were developed on a GNU/Linux system, and were ported to several operating systems: NetBSD, FreeBSD, Mac OS X, MS Windows. Their source code is available from CTAN² or the Eukleides home page³ (which also offers GNU/Linux and Win32 executables).

Around Morley's triangle

As a first introduction to the Eukleides language, we'll study the source code which gives the following figure. It illustrates Morley's theorem: *The points of intersection of the adjacent trisectors of the angles of any triangle are the vertices of an equilateral triangle.*



Here is the corresponding code⁴.

```

1  A B C triangle
2  a = angle(B,A,C)
3  b = angle(C,B,A)
4  c = angle(A,C,B)
5  ab = angle(vector(A,B))
6  bc = angle(vector(B,C))
7  ca = angle(vector(C,A))
8  l1 = line(A,(ab + a/3):)
9  l2 = line(A,(ab + 2*a/3):)
10 l3 = line(B,(bc + b/3):)
11 l4 = line(B,(bc + 2*b/3):)
12 l5 = line(C,(ca + c/3):)
13 l6 = line(C,(ca + 2*c/3):)
14 D = intersection(l1,l4)
15 E = intersection(l3,l6)

```

² In /tex-archive/support/eukleides/.

³ At <http://perso.wanadoo.fr/obrecht/>.

⁴ The numbers at the beginning of each line are not part of it.

```

16 F = intersection(l2,l5)
17 color(lightgray)
18 draw(l1) ; draw(l2)
19 draw(l3) ; draw(l4)
20 draw(l5) ; draw(l6)
21 color(black)
22 draw(A,B,C) ; draw(D,E,F)

```

In Eukleides source code, a line can contain several commands (in that case, they have to be separated by semicolons). Commands are of two kinds: variable assignments and graphical commands. Among variable assignments are single assignments (see lines 2–16) and multiple assignments (line 1). A variable can store a wide variety of objects used in elementary geometry: numbers, vectors, points, lines, segments, circles, conics.

Multiple assignments are used for definitions of polygons and for some intersection determinations. The statement in line 1 defines an optimal scalene triangle such that segment *AB* is horizontal and 6 cm long. All these characteristics can be modified by adding some optional parameters to the keyword ‘triangle’. For instance ‘A B C triangle(4,5,6)’ would define a triangle *ABC* such that *AB* = 4 cm, *BC* = 5 cm and *AC* = 6 cm.

If the desired triangle has to be of a specific kind, the simplest way is to replace ‘triangle’ with ‘right’, ‘isosceles’ or ‘equilateral’. For instance ‘A B C right(5,30:,10:)’ would define a triangle *ABC* with an angle of 30° in *A*, a right angle in *B* and such that segment *AB* measures 5 cm and makes an angle of 10° with the horizontal direction. The colon character is used to distinguish angular parameters from others (like lengths).

On lines 2–7, one can see two possible usages of the function ‘angle’. In the first case (lines 2–4), it simply gives the measures of the angles in triangle *ABC*. In the second case (lines 5–7), it gives the argument of some vectors. As with many functions in Eukleides, ‘angle’ can handle several kinds of arguments.

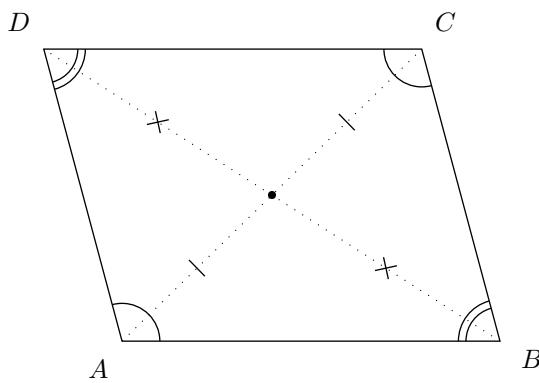
On lines 8–13 are the definitions of the trisectors of the triangle *ABC*. Since trisectors (unlike bisectors) aren't very common objects, there's no built-in function to define them. The function ‘line’ is used instead. Here, the second argument is the angle that the line makes with the horizontal direction. This is not the only way to define a line: the second argument could have been a point or a vector.

Graphical commands are of two kinds: setting commands (see lines 17 and 21) and drawing commands (see lines 18–20 and 22). To draw an object,

one simply has to use the function ‘`draw`’. This function can take additional arguments in order to modify the aspect of the drawn object (such as ‘`dotted`’ or ‘`dashed`’ for lines). On line 22, the arguments of ‘`draw`’ are a list of points: it’s the way to draw polygons. Since polygons are not considered as specific objects in Eukleides, there’s no need to declare DEF as a triangle before drawing it.

More graphical commands

Usually, a geometric figure doesn’t contain only straight and curved lines, but also letters and some conventional marks (used to make some properties obvious). Below is a classical example of such a figure representing a parallelogram.



Here is the corresponding code.

```

1  A B C D parallelogram(5,4,105:)
2  O = barycenter(A,B,C,D)
3  frame(-2,-1,6,4.5)
4  draw(A,B,C,D) ; draw(O)
5  draw("$A$",A,-130:)
6  draw("$B$",B,-30:)
7  draw("$C$",C,50:)
8  draw("$D$",D,130:)
9  draw(segment(A,C),dotted)
10 draw(segment(B,D),dotted)
11 mark(segment(A,O))
12 mark(segment(O,C))
13 mark(segment(B,O),cross)
14 mark(segment(O,D),cross)
15 mark(B,A,D)
16 mark(D,C,B)
17 mark(C,B,A,double)
18 mark(A,D,C,double)

```

Since this figure is rather simple (from a geometrical point of view) only two assignments are needed. On line 1 is a multiple assignment which defines a parallelogram $ABCD$ such that $AB = 5$ cm, $AD = 4$ cm and $\widehat{BAD} = 105^\circ$. On line 2, a single

assignment defines O as the center of parallelogram $ABCD$.

Even though Eukleides is designed in order to use as few coordinates as possible, the internal representation of the geometrical objects is based on them. By default, figures are drawn in a frame such that the lower left corner has coordinates $(-2; -2)$ and the upper right corner $(8; 6)$. The function ‘`frame`’ enables one to change these settings.

As one can see on line 4, the function ‘`draw`’ is useful to represent single points (the default shape is a dot, but it can also be a square or a cross). This function can also be used to give names to points,⁵ as in lines 5–8. Here, the first argument is a string, the second a point and the third an angular argument specifying the position of the label. This string can contain TeX code⁶ such as mathematical formulas.

On lines 11–18 are the marking commands. It is possible to mark, in various ways, either segments (lines 11–14) or angles (lines 15–18).

A classical locus problem

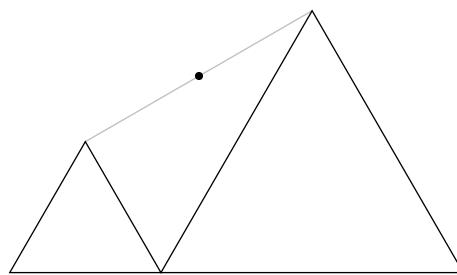
In some situations, a computer screen can be very useful to teach geometry. For instance, a locus problem becomes much easier if one can see several states of the figure. The program `xeukleides` has been developed for this. At startup, it appears as a text editor. If you type the lines below:

```

1  x interactive(2,.1,0,6,"A",right)
2  A M I equilateral(x)
3  M B J equilateral(6-x)
4  color(lightgray)
5  draw(segment(I,J))
6  color(black)
7  draw(A,M,I) ; draw(M,B,J)
8  draw(barycenter(I,J))

```

and press the escape key, the text area will be replaced by a graphical area containing the following figure:



⁵ Or to put any kind of text in a specific place.

⁶ This code will only be interpreted if you run `eukleides` and `latex`. With `xeukleides` it is displayed verbatim.

If you now press the right arrow key, you'll see the left triangle becoming bigger and the right one smaller. Pressing the left arrow key performs the opposite transformation. Pressing the escape key again switches back to the text editor.

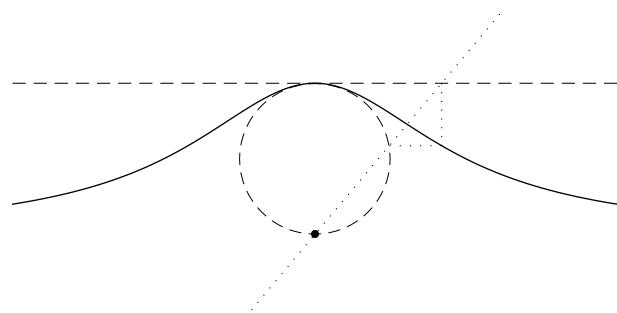
On line 1 of the source code is an interactive assignment: it allows to modify the value of the numerical variable x (and consequently the figure) by pressing the arrow keys. The first argument is the initial value of x , the second the increment which is added to (subtracted from) x every time the right (left) arrow key is pressed. The third and fourth are the optional lower and upper bound. The fifth argument has to be a string containing a single letter. It indicates the key that has to be pressed before modifying the variable.⁷ This is useful when more than two variables have to be bound to the keyboard. The sixth argument is either 'right' or 'up'. It indicates which pair of arrow keys (right/left or up/down) is bound to the variable.

In an interactive assignment, the initial value can be modified while viewing. If you press the F1 key, the program replaces the original initial value in the source code by the last value of the variable and switches back to the text editor.

The first multiple assignment on line 2 defines an equilateral triangle AMI such that segment AM is horizontal and x cm long. The second assignment defines an equilateral triangle MBJ such that segment MB is horizontal and $6-x$ cm long. A specific feature of polygonal assignments is used here: if the first variable is already in use (and contains a point) its content remains the same (if not, the variable is set to the origin). This implies that segment AB has a constant length of 6 cm and that M belongs to AB .

Drawing curves

In elementary geometry, the most usual curves are conics. Eukleides provides a large number of function to define and handle these objects. For less common curves, there's the 'trace' command. For instance, the figure below illustrates the geometrical definition of a cubic curve known as the *Witch of Agnesi*.



This curve is obtained by drawing a line from the origin through the dashed circle, then picking the point with the x coordinate of the intersection with the dashed line and the y coordinate of the intersection with the circle. Here is the corresponding code:

```

1  frame(-4,-1,4,3)
2  O = point(0,0)
3  c = circle(point(0,1),1)
4  l = line(point(0,2),0:)
5  trace(t,.1,179.9){
6  L = line(0,t:)
7  O M intersection(L,c)
8  P = intersection(L,l)
9  point(abscissa(P),ordinate(M))} 
10 t = 50
11 L = line(0,t:)
12 O M intersection(L,c)
13 P = intersection(L,l)
14 N = point(abscissa(P),ordinate(M))
15 draw(O)
16 style(dotted)
17 draw(segment(M,N))
18 draw(segment(P,N))
19 draw(L)
20 thickness(.5) ; style(dashed)
21 draw(c) ; draw(l)

```

On lines 2–4 we create the objects which are needed to define the curve. Lines 5–9 are related to the 'trace' command. They are based on the geometrical definition above. Line 5 tells variable t to scan the numbers between⁸ 0.1 and 179.9. A line-circle intersection can lead to two points, hence in Eukleides a multiple assignment (like the one at line 7) is used to obtain these points. Since lines are implicitly directed, in the present case the first assigned point will always be O and the second, the wanted point. The last line (line 9) contains a point

⁷ Since the program starts viewing in state "A", there's no need here to press this key.

⁸ These bounds are chosen in order to avoid 0 and 180, which are invalid and may cause spurious lines to appear.

valued expression. This is the point which will be drawn for each value of t .

To draw this curve, it would also be possible to use parametric representation. Nevertheless, in the present case the geometrical definition is more appropriate because the same piece of code can be used again (in lines 11–14) to produce an example of the construction.

The last part (lines 15–21) contains the drawing commands. The setting command ‘`style`’ changes the default aspect of drawn objects. This may sometimes shorten the code.

Conclusion

In my humble opinion, Eukleides is now mature enough to be considered by \TeX users as an effective way to create geometric figures. As a matter of fact, the language is sufficiently powerful to describe almost any figure which can be seen in an elementary geometry textbook.

My aim is now to enhance Eukleides with features such as tests, loops, and user-defined functions. Since I did not anticipate this when I started the project, I'll have to rewrite large parts of the programs. This is a long-term undertaking, so I'll soon stop working on the present versions of `eukleides` and `xeukleides`.

◊ Christian Obrecht
 Le Monsard
 71960 Bussières
 France
`christian.obrecht@wanadoo.fr`
`http://perso.wanadoo.fr/obrecht/`

METATEX

Ramón Casares

Abstract

METATEX is a set of plain \TeX and METAFONT macros that you can use to define both the text and the figures in a single source file. Because METATEX sets up two way communication, from \TeX to METAFONT and back from METAFONT to \TeX , drawing dimensions can be controlled by \TeX and labels can be located by METAFONT. Only standard features of \TeX and METAFONT are used, but two runs of \TeX and one of METAFONT are needed.

Overview

Together, \TeX and METAFONT define the page layout to the pixel. This means that nothing more is needed, not even a means of including figures in a \TeX document. To prove this is the aim of this paper.

To split the typesetting process in two parts, one to define and draw the characters and the other to arrange the characters in paragraphs and pages, is surely the best way to reduce the complexity of the typesetting task, provided it needs simplification (see Figure 1). But this method makes it difficult, for example, to integrate labels with graphics in figures, because while \TeX is best suited to typeset the labels, METAFONT is the appropriate tool to draw the graphics. And, of course, labels should be located in accordance with the graphics.

Therefore, the true successor of \TeX has to include in a single program both the capabilities of \TeX and METAFONT. Then the typesetting engine would include a powerful graphic tool, a grid in which to typeset if required, and it could take into account the shapes of the characters to determine, for example, kernings or italic corrections. The other way around is also possible. It could be seen as a graphic engine with a powerful typesetting tool. From this point of view, the page would be a graphic object that could contain paragraphs of different shapes requested from the typesetting tool.

METATEX, although it does not fulfill the requirements of such a successor, can be seen as an early sign of its possibilities. For the moment, METATEX takes advantage of METAFONT's equation solving capabilities to locate objects, including the labels, which are typeset by \TeX . The cost of this nice feature is that two \TeX passes are required.

During the first \TeX pass a METAFONT file is written. As it is \TeX itself who writes the METAFONT file, any dimension controlled by \TeX can be used and incorporated in METAFONT's calculations. For example, the label sizes, as they will be typeset by \TeX , are made known to METAFONT.

After the first \TeX pass, METAFONT draws the graphic figures and writes the label locations in its log file. So it is METAFONT's responsibility to locate the labels. Note that, depending on the style of METAFONT programming, this can be completely determined from \TeX . In other words, you can relate the label location to the location and size of other METAFONT objects, or not.

When \TeX executes its second pass, it takes the graphics from the new font, reads the location

of labels from the METAFONT log file, and then everything is complete.

Because labels are just `\hboxes` typeset by `TEX`, every macro currently defined for text automatically applies also to figures. For example, if a macro `\person` is defined to write its argument in a small caps font and save it to an index file, the same happens whenever it is used inside a figure label.

Methods

`METATEX` allows the source file to include, in addition to the customary `TEX` macros to control the text, other commands to generate figures with METAFONT.

Steps. In order to use `METATEX` the following three steps are to be executed:

1. The `METATEX` file, suppose it is `filename.ext`, is first processed by `TEX`, with the `plain` format, during which a METAFONT file named `auxiliar.mf` is created. This METAFONT file contains information provided by `TEX` concerning the size of the labels, so the MF program can delete this area from the figure if requested. If the output file `filename.dvi` were typeset now, it would have blanks in place of the figures, but otherwise be the same as the final document.
2. Then MF, with the `plain` base, is run on `auxiliar.mf`. As a result, information specifying where to typeset the labels is written in the log file, `auxiliar.log`. In addition, the metric file, `auxiliar.tfm`, and the generic format bitmap font, `auxiliar.329gf`, are created. On my system I have to process this `gf` file to get a `pk` file that my drivers can read, so I execute the program `GFToPK` on it, getting the packed bitmap font `auxiliar.329pk`. Please note three points. i) The number 329, referring to the resolution, varies according to the METAFONT mode. ii) The `tfm` and `pk` files must be in or moved to directories where programs can find them. iii) `METATEX` sets the METAFONT mode to `localfont`, thus assuming that `localfont` is assigned the appropriate name.
3. Lastly, `filename.ext` is again run through `TEX`. During this second run, both the font `auxiliar` containing the figures and the information explaining where to locate the labels are available, so the document is complete.

The figures fill exactly the same area in both the first and second `TEX` program runs, so indices, tables of contents, and other references that also

need two passes to be resolved can take advantage of the two runs needed by `METATEX`.

Use. To use the `METATEX` macros, they must be imported by writing in the source file:

```
\input metatex
```

This has to be written after `\mag` has been given its final value. When `metatex.tex` is read, `METATEX` checks whether the file `auxiliar.mf` exists. If it does not exist, then things are set up for the first pass; for example, `auxiliar.mf` is opened for writing. If it does exist, then things are set for the second pass; for example, `auxiliar.log` is opened for reading. This means that if `auxiliar.mf` is not deleted, then step 3, the second `TEX` program pass, is executed directly. This saves time when only the text in file `filename.ext`, but not the figures, were modified.

User macros. The `METATEX` user macros are:

- `\MTbeginchar(wd,ht,dp)`; states that a figure sized as given (width `wd`, height `ht`, depth `dp`) will be created. These values should be known both by `TEX` and by METAFONT, so for example `12pt, 6cm, \the\hsize` or `\the\dimen0`, always without #, are allowed. During the *first pass*, `TEX` writes in `auxiliar.mf` the METAFONT macro `beginchar` assigning character codes sequentially, and box `\MTbox` is made empty but sized as specified by the arguments of this macro. During the *second pass*, `TEX` puts the corresponding character of the font `auxiliar` in box `\MTbox`. The size of `\MTbox` is that specified and not affected by the character dimensions.
- `\MTendchar`; finishes the figure definition. During the *first pass*, `TEX` writes the METAFONT macro `endchar`; in file `auxiliar.mf`. During the *second pass*, box `\MTbox` contains the complete figure, including labels. Something like `\box\MTbox` is used to typeset the figure.
- `\MTlabel*(s)cc"Text"`; adds a label to the current figure. The parameter between quotes, `Text` in the example, is the label content; it will be put inside an `\hbox` and therefore could be anything that `TEX` allows inside an `\hbox`. The optional asterisk after `\MTlabel` instructs `METATEX` to erase the area of the figure already drawn that it is under the label.

The label will be located at METAFONT point `z.s`, where `s` is the parameter between parentheses. The reference point is further specified by the optional parameter after the

right parenthesis, `cc` in the example. This parameter is composed of exactly two letters: the first can be `t` meaning top, `c` meaning center or `b` meaning bottom; and the second letter can be `l` meaning left, `c` meaning center or `r` meaning right. So, for example, `t1` means that the label reference point is its top left corner. The default value for the reference point is `cc`, that is, its center.

`\MTlabel` should only be used between `\MTbeginchar` and `\MTendchar`. During the *first pass*, it writes the following three elements in `auxiliar.mf`: i) the METAFONT macros which in turn cause MF to write the label reference point location to its log file, `auxiliar.log`; ii) the four label sides, which are by this means made available to the following METAFONT code for the figure, notated as `y.s.t` for the top side, `y.s.b` for the bottom side, `x.s.l` for the left side and `x.s.r` for the right side; and iii) the code to delete, if requested, the figure area already drawn that is under the rectangle occupied by the label. During the *second pass*, it adds the label to the box `\MTbox` in the place that reads from file `auxiliar.log`, making no modification to the dimensions of `\MTbox`, even if the label is typeset outside the box.

There are three more macros for passing information to METAFONT, that is, for writing general text in `auxiliar.mf`: `\MT:`, `\MTcode` and `\MTline`. This happens only during the first pass; during the second pass, these macros do nothing.

- `\MT:` writes in file `auxiliar.mf` everything till the end of line. It writes verbatim except for the character `\`, which keeps its normal TeX `\catcode` of 0. Spaces are *not* ignored after macros. The sequence `\\"` writes a single `\` in file `auxiliar.mf`.
- `\MTcode` writes in file `auxiliar.mf` everything until it finds a line equal (including `\catcodes`) to the current value of `\MTendmark`. By default, this is a blank line, thus, `\def\MTendmark{}`. As with `\MT:`, it writes verbatim except for `\`, which still operates as an escape character. The control sequence `\\"` writes a single `\` in file `auxiliar.mf`.
- `\MTline{text}` writes its parameter to `auxiliar.mf`, `text` in the example. It does not change the `\catcodes` in the argument, so it does not perform verbatim writing. But all plain special characters can be written prefixing them by the escape character `\`. The plain

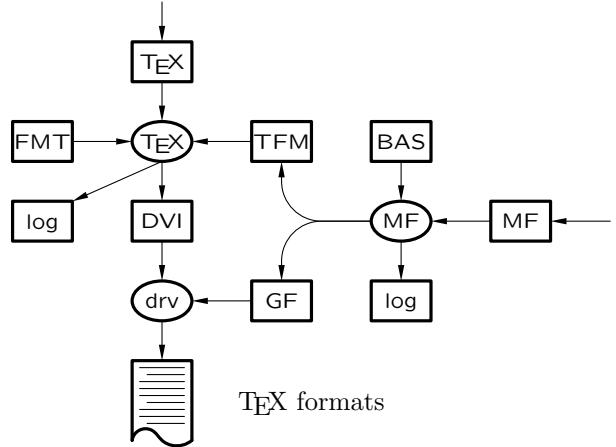


Figure 1: One column diagram

special characters are (not including the first colon nor the final period): `\{}$#^_~%`. For example, `\#` results in `#`.

When defining TeX macros that write to `auxiliar.mf`, `\MTline` should generally be used in preference to `\MT:` or `\MTcode`, because the latter two use the end of line in a special way that is not usually available when TeX is reading a macro.

TeX dimensions can be included using any of these three writing macros. For example, `\the\hsize` will be expanded to `225.0pt` (for the present article), and written as such to the METAFONT file `auxiliar.mf`. Note that the character `\` keeps its escape `\catcode` in all three writing macros. In the case of `\MTline`, braces `{}` also keep their `\catcodes` and therefore macros with parameters can be used normally.

Examples

Diagram. First a typical example of METATEX usage, showing the file formats, programs, and their relationships. The figure width is exactly `\hsize`, but what is more important is that the same code will adapt itself to any value for the measure. Well, of course, not to *any* width but to any width between, let's say, 8 cm and 25 cm.

Figure 1 is the one column version, and Figure 2 (above the appendix) is the two column version, generated by the same source.

Shadowing. Both TeX and METAFONT are ill suited to creating shadows. In TeX, one straightforward technique is double use of `\leaders`, but in practice this results in huge dvi files. In METAFONT, drawing lots of tiny points easily exceeds the capacity of the program. The solution is to coordinate the work of both programs.

To create a large rectangular shadow we divide it into an array of $n \times m$ smaller rectangles. The smaller rectangles are all identical, so it is enough for METAFONT to draw one shadow character and then for T_EX to typeset a solid area repeating it.

To simplify the tasks of both T_EX and METAFONT, the size of the shadow character should be similar to that of normal characters, because neither program was designed to work well with extraordinarily large (or small) characters. So a good approach is to make the shadow character as big as possible but never wider nor higher than 16 pt.

For METATEX, each figure is a character. This causes problems with METAFONT when the figure is big and the resolution is high, because it cannot draw areas bigger than 4095×4095 pixels. This is not usually a problem working at 300 dpi. (It is never a problem with METAPOST, see the following section on PostScript. Another advantage of using PostScript is that you get a shadow simply by drawing a grey rule, and none of the above machinations are necessary.)

Keys. After the following METATEX macros:

```

1. \MTcode
2. def keybox =
3. pickup pencircle scaled 0.8pt;
4. x1 = x3 = 1pt;
5. x2 = x4 = w - 1pt;
6. x5 = 0; x6 = w;
7. y1 = y2 = -d;
8. y3 = y4 = h;
9. y5 = y6 = (h - d)/2;
10. draw z1 -- z2 .. z6{up} ...
11.      z4 -- z3 .. z5{down} .. cycle;
12. z0 = (x1,0);
13. enddef;
14.
15. \def\defkey#1#2{\setbox0=\hbox{\sf#2}%
16. \dimen0=\wd0\advance\dimen0 by 2pt
17. \dimen2=\ht0\advance\dimen2 by 1pt
18. \dimen4=\dp0\advance\dimen4 by 1pt
19. \MTbeginchar(\the\dimen0,%
20.             \the\dimen2,%
21.             \the\dimen4);%
22.   \MTline{keybox;}%
23.   \MTlabel(0)bl"\sf #2";%
24. \MTendchar;%
25. \expandafter\newbox
26.   \csname\string\box\endcsname
27. \expandafter\setbox
28.   \csname\string\box\endcsname
29. =\vtop{\unvbox\MTbox}%

```

```

30. \def#1{\expandafter\copy
31.   \csname\string#1box\endcsname}%
32.
33. \def\makekey#1{\expandafter\defkey%
34.   \csname#1\endcsname{#1}}%

```

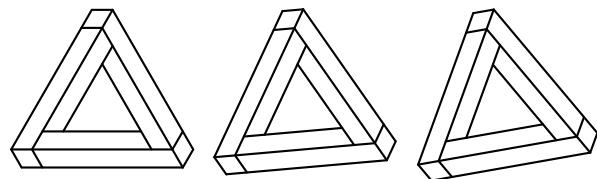
Then, we can declare `\makekey{Alt}` to typeset `[Alt]` simply via `\Alt`. It is also possible to declare `\defkey\escape{\tt\char92}` and then `\escape` results in `\`.

Baroque tables. Baroque periods are the result of new technical achievements providing unexplored possibilities and hence the urgent need to experiment with them, frequently far away from what discretion might recommend. This explains the time of baroque software that we live in, and increases the value of METATEX, because it provides the means to easily draw baroque tables. I am not a baroque man, so my baroque table example is not baroque but, and this is the point, it is at least not built with straight lines.

This is not a straight table
but it's only an example
and therefore not so ample
of what's METATEX-able!

Ornate paragraphs. Only if you are truly baroque can you get the most from METATEX. If, for example, you like ornate paragraphs, you are in your element. Just put the material in a `\vbox` to get the height and the depth, and pass these dimensions to METAFONT to draw a right sized embellishment. Ah!, but be aware that Computer Modern is a neoclassical font, so it won't mix well with your elaborations.

Exercise.* Take three equal bars and build as shown. Ask Roger Penrose if you don't find the solution.



* The real exercise is to fill the rest of the page with tribars. A clue: let T_EX to calculate how many are needed, so you can concentrate your efforts in drawing the figure.

PostScript

By taking advantage of METAPOST, we can make PostScript versions of the METATEX files. Simply execute `mptopost &plain auxiliar.mf` instead of METAFONT and, after TeX's second pass, execute `dvips` (METATEX uses `dvips` specials). This works because `auxiliar.mf` is valid code for both METAFONT and METAPOST, and because, in the second TeX pass, METATEX checks which one was used and adapts itself to the situation.

And thanks to PDFTeX and the ConTeXt files `supp-mis.tex` and `supp-pdf.tex`, it is also possible to get PDF output. Just execute PDFTeX twice, instead of TeX, and once METAPOST, instead of METAFONT. (In practice, METATEX uses its own macro file `mptopdf.tex` for this, instead of the ConTeXt files. I extracted all that METATEX needs from the ConTeXt files into `mptopdf.tex`.)

This works because, if METATEX determines that METAPOST was employed to draw the figures, it then checks which program, TeX or PDFTeX, is executing. If TeX, then it includes the files produced by METAPOST using the `dvips` specials. If PDFTeX, it translates from `ps` to `pdf`.

Therefore, the same METATEX source file generates at will any of the three output formats—`dvi`, `ps`, or `pdf`—just by running the appropriate programs. In the appendix, as an example, there is a DOS batch file that shows how to get the `pdf` version of a file `filename.ext`.

Other graphical tools

There are other tools to include pictures in a TeX document. LATEX's `picture` environment, PICTeX, and `mfpic`, are three of them. METATEX is similar to `mfpic`, in that both use METAFONT to draw.

The aim of `mfpic` is to overcome the difficulties of LATEX's `picture` environment and of PICTeX. LATEX's `picture` environment uses four pre-cooked special fonts, and its drawings are just compositions of these characters (as well as TeX's builtin `\hrule`s and `\vrule`s). PICTeX only uses a tiny point to compose the pictures, so it is more general. But letting TeX to draw the figures setting point after point is painful, as noted above. The `mfpic` solution uses a better tool for drawing: METAFONT.

With these origins, for `mfpic`, METAFONT is a hidden back-end processor, and `mfpic` imposes only

two requirements on its users: to know TeX, and to know `mfpic`. On the other hand, METATEX's approach is minimalist, at the cost of being more demanding with its users. A METATEX user has to know TeX, METAFONT, and METATEX—although this last requirement is small, because METATEX only builds the necessary bridges to use TeX and METAFONT in a cooperative way.

A feature that shows the different strategies employed in designing `mfpic` and METATEX is label positioning. METATEX labels are located by METAFONT, so TeX has to read the METAFONT log file to learn where to typeset them. For `mfpic`, TeX itself locates the labels, but by doing so `mfpic` has to give up some nice METAFONT characteristics, such as its equation solving capabilities.

In summary, `mfpic`'s aim is to draw pictures in TeX documents in a better way than using LATEX's `picture` environment or PICTeX; while METATEX's intention is to coordinate the work of TeX and METAFONT. In this way, METATEX provides the full raw power of TeX and METAFONT, and it is up to you to harness them.

Final remarks

I have been using METATEX for some years. The first version was dated 1994, but it has been used only for personal purposes. For this reason, it is not truly a straightforward end-user tool, as for example LATEX packages should be. It has to be used knowledgeably and with care. And though most tasks can be automated, by chaining TeX and METAFONT errors are even more difficult to pinpoint than in TeX or METAFONT alone.

Nevertheless, METATEX serves to validate the feasibility of a closer collaboration between TeX and METAFONT and to appraise the interest of such a collaboration. And, of course, if you dare, you can get lots of fun, and at least an equal amount of frustration, using METATEX. Try it!

The METATEX package is available from CTAN in CTAN:/macros/plain/contrib/metatex.

Happy METATEXing!

◊ Ramón Casares
Telefónica de España
r.casares@computer.org

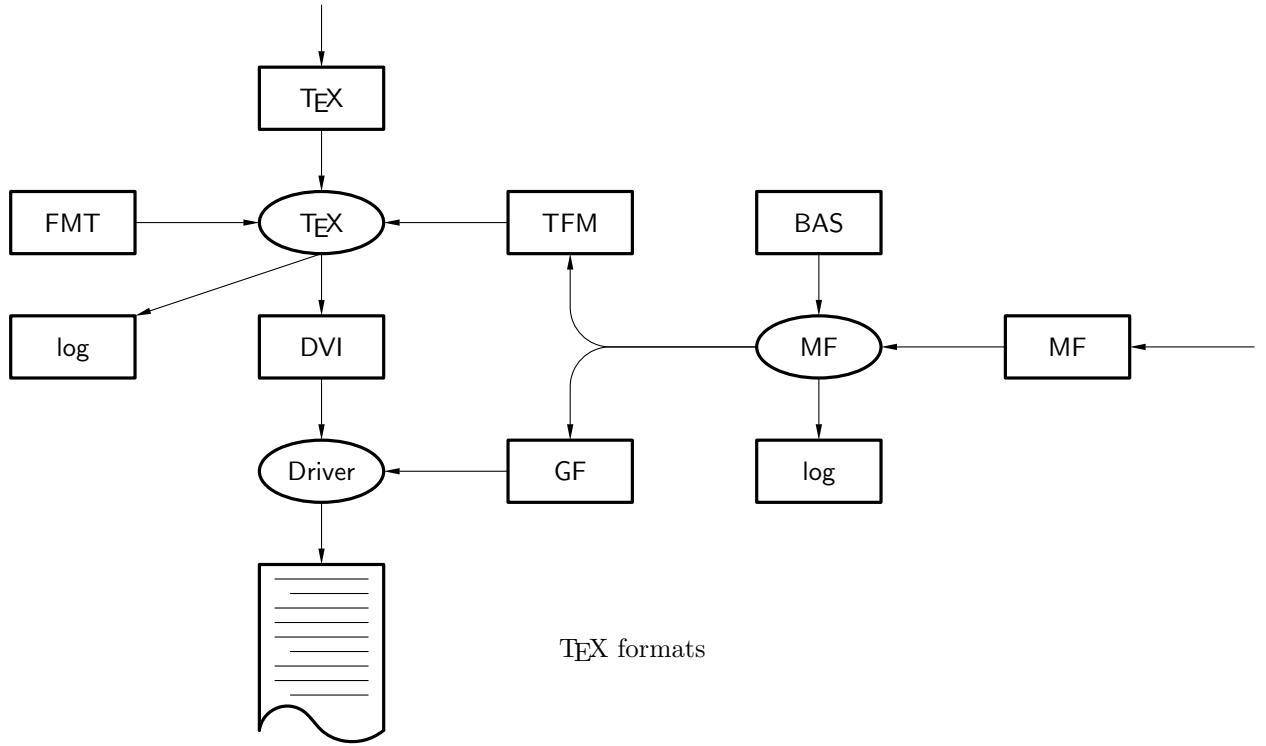


Figure 2: Two-column diagram

Appendix: Pseudo-batch example

This is an example based on DOS batch files that can be adapted for other operating systems. It processes the file `filename.ext`, generating `filename.pdf`. We will comment each line of pseudo-code.

1. We go to the directory where our working files are.

```
cd c:\dir\ subdir\mydir
```

2. We set environment variables (if necessary). In this case we are using a `web2c` system, so it is enough to set one. In `web2c`, by default, all programs look for files in the current dir, “.”, and that is enough for us.

```
set TEXMFNF=c:\tex\texmf.local\web2c;c:\tex\texmf\web2c;d:\texmf\web2c
```

3. We tell the operating system where to find the programs.

```
path=$path$c:\tex\bin\dos
```

4. After the settings, we force the first TeX pass.

```
if exist auxiliar.mf del auxiliar.mf
```

5. Then, we execute the first TeX pass (in this case, it is PDFTeX).

```
pdftex &plain filename.ext
```

6. If METATEX was not used, and no `auxiliar.mf` was written, then we are done.

```
if not exist auxiliar.mf goto end
```

7. Otherwise we run METAPost with its `&plain` memory (format), also known as `&mpost`.

```
mpost &plain auxiliar.mf
```

8. Finally, we execute the second TeX pass.

```
pdftex &plain filename.ext
```

9. We now have the complete `filename.pdf` file.

```
:end
```



The Treasure Chest

Packages posted to CTAN

Did you know that CTAN has two different packages for balancing columns? (At least two: `balance` in `.../other/preprint` and `flushend.sty` in `.../sttools`.) That there are packages for drawing graphical clock faces and dice? (`clock` and `dice3d` in `fonts/dice`.) That there is a tool for determining if all files needed by a given package are installed? (Appropriately enough, named `diagnose`.) That there is a package for typesetting exams, along with their answer keys? (`examdesign`) Or that there is even a “toolbox” of macros “for various tasks often needed in `TEX` programming”? (`toolbox` of course.) A veritable “Treasure Chest”, no?

This is a list of packages posted to CTAN from July 2001 through June 2002, with descriptive text pulled from the announcement or researched at need and edited for brevity. All errors are ours.

Each month’s list of packages are in alphabetic order; a package is only listed in the last month in which it was updated. Leading slashes are omitted from URLs, since the prefix will depend on the particular CTAN server you use. To save space, we have omitted the directory name `macros/latex/contrib` for packages residing there (a large proportion), and those in subdirectories of that branch are denoted by `...`; for example, `url.sty` (in `macros/latex/contrib/misc`) would be shown as being in `.../misc`. Furthermore, when we list an entire directory, we don’t repeat the directory name.

This edition of Treasure Chest was split between the authors: Mark wrote the entries for July–December 2001, and Will for January–June 2002. Will is bowing out as of this column—a big thank you from the editors. Mark will continue for future installments.

Hopefully this column and those which follow will help to make CTAN a more accessible resource to the `TEX` community.

July 2001

`2in1` in `.../misc`

`LATEX` style file for printing two pages on a single landscape page.

`alphanum` in `biblio/bibtex/contrib`

Bibliography style that omits the year, and uses a number instead of a letter.

ascelike Unofficial `LATEX` class and bibliography stylefiles for ASCE documents (American Society of Civil Engineers).

`bibfind` in `biblio/bibtex/utils`

Reads a `.bib` file and prints those references matching a search string. It works a bit like `grep`, only it prints the whole reference (paragraph) and not just the line that matches.

`comprehensive` in `info/symbols`

This is a major update to the Comprehensive `LATEX` Symbol List. Not only does it contain hundreds of new symbols, but the symbol tables have been reordered into a more logical structure; the document begins with a “frequently requested symbols” list, and there is a new, quick reference table for Latin 1. The result is that `LATEX` symbols should be easier to find than ever before.

`esvect` Write vectors with an arrow different than the Computer Modern one.

`gellmu` in `support`

`LATEX`-like markup for writing XML documents.

`hypernat` Allows use of `hyperref` and `natbib`.

`iahyphen` in `language/hyphenation`

Hyphenation patterns for Interlingua.

`jadetex` Processes the output from Jade/OpenJade in `TEX` (-t) mode. Development has moved to <http://jadetex.sourceforge.net>.

`latable` in `support`

A near-WYSIWYG editor for `LATEX` tables.

`lineno` Adds line numbers to selected paragraphs with references possible via the standard `LATEX` `\ref` and `\pageref` cross reference mechanism.

`MiKTeX-WinEdt-TrueType-Anleitung` in `info/german` German information about MiKTeX, WinEdt, and TrueType.

`mpattern` in `graphics/metapost/macros` Patterns in METAPOST.

`neuron` in `bibliography/bibtex/contrib`

A bibliography style for use in submitting manuscripts to the journal `Neuron` (www.neuron.org).

`pdfcrypt.sty` in `.../oberdiek`

Allows the setting of PDF encryption options for `pdfTEX` and `VTEX`.

`pkfix` in `support`

Replace pk fonts with Type 1 in PostScript files.

`SIunits` Typeset physical units following the rules of the International System of Units (SI).

`t-angles` Draws tangles, trees, Hopf algebra operations, and other pictures.

August 2001

`amstex` in `macros`

`AMSTEX` version 2.2.

`arraymaker` in `support`

This Tcl/Tk program is designed to take some of the tedium out of constructing arrays, diagrams

and tables. L^AT_EX code is entered into an array of blank boxes and arrows can be added with a mouse click. The program then outputs an array that can be pasted into a L^AT_EX file.

binhex in *macros/generic/kastrup*

Convert numbers into binary, octal and hex.

bitfield Draws bitfield diagrams.

biocon Aids typesetting of biological species names, and of taxa below the species level.

booklet Provides some aids for printing simple booklets or signatures for longer books.

bundledoc in *support*

This post-processor for the **snapshot** package bundles together all the classes, packages, and files for building a given L^AT_EX document.

As an added bonus, this distribution includes a small script called **arlatex**, a L^AT_EX-specific archiving program that combines a bunch of files into a single **.tex** file. When the **.tex** file is run through **latex**, all of the original files are recreated and the L^AT_EX document is built.

crossreference Cross-referencing package to help provide traceability for technical documents, especially Software Requirements Specifications.

dashbox Draws dashed boxes.

datenumber Provides commands to convert a date into a number and vice versa.

ebezier An extension of the (old) package **bezier.sty** which is currently part of L^AT_EX 2_E. It defines linear and cubic Bernstein polynomials together with some plotting macros for arcs.

examples in *info/metapost*

A few hundred example pictures drawn with METAPOST, ranging from very simple (lines and circles) to rather intricate (uncommon geometric transformations, fractals, bitmaps, etc.).

filecontents The environments **filecontents** and **filecontents*** enable a L^AT_EX source file to generate external files as it runs. However, these environments have two limitations: they refuse to overwrite existing files; and they can be used only before the **\documentclass** declaration. This package removes these limitations.

float Improves the interface for defining floating objects, such as figures and tables. Introduces the boxed float, the ruled float, and the plaintop float.

framed in .../misc

Creates framed or shaded regions that can break across pages.

greenpoint in *fonts*

METAFONT implementation of the logo commonly known as “Der Gruene Punkt” (“The Green Point”). In Austria, it can be found on nearly every bottle. It should not be confused with the “Recycle” logo, implemented by Ian Green.

musixps in *fonts/musixtex/ps-type1*

This package provides PostScript (Type 1) fonts (PFB format), and **dvips** and **dvipdfm** map files for MusiX_TE_X.

ps2eps in *support*

A tool to produce EPS/EPSF files from typical one-paged PostScript documents. Requirements: Perl, Ghostscript and an ANSI C compiler if your platform is not Linux, Solaris, Digital UNIX or Windows 2000/9x/NT (binaries included).

revtex Includes styles for the journals of the American Physical Society, American Institute of Physics, and Optical Society of America.

ruhyphen in *language/hyphenation*

Updates to this package for Russian hyphenation.

slideshow This is a set of METAPOST macros which simplify creating a PDF presentation using METAPOST and Ghostscript.

uiucthesis University of Illinois at Urbana-Champaign thesis style.

warning This package provides a command that generates a list of warnings that are printed out at the very end of the logfile. This is useful for notes such as “Rerun for this or that reason” or “This is draft, change this before the final run”.

xl2latex in *support*

Convert Excel (97 and above) tables to L^AT_EX.

September 2001

boites Defines environments that allow page breaks inside framed boxes. A few examples are included (shaded box, box with a wavy line on its side, etc.).

brushscr in *fonts*

This Type 1 font, named BrushScriptX-Italic, simulates hand-written characters.

circuit_macros in *graphics*

A set of macros for drawing high-quality electric circuit diagrams containing fundamental elements, amplifiers, transistors, and basic logic gates to include in T_EX, L^AT_EX, or similar documents. Some tools and examples for other types of diagrams also included.

combine The **combine** class lets you bundle individual documents into a single document, such as when preparing a conference proceedings.

frankenbundle in *support*

Allows an author to maintain and distribute a bundle of one or more L^AT_EX packages, classes, BIBT_EX bibliography styles, documentation, etc.

frankenstein A collection of L^AT_EX/BIBT_EX macros.

ifvtex.sty in .../oberdiek

Looks for V_TE_X, defining the conditionals **\ifvtex**, **\ifvtex<mode>**, and **\ifvtexge**.

JavaBib in *biblio/bibtex/utils*

A BIBT_EX reference manager written in Java.

register Typesets the programmable elements in digital hardware, i.e., registers.

TkTeXCAD in graphics

A Tk-Python script which provides a graphical user interface to the `picture` environment.

October 2001**belleek in fonts**

The Belleek fonts described in Richard Kinch's paper at TUG'98.

bibweb in biblio/bibtex/utils

A utility for automatically retrieving bibliographical information from the American Mathematical Society's MathSciNet program.

clock Provides various graphical and textual clocks for TeX and L^AT_EX 2_&. Renamed from `texclock`.**cm-super in fonts/ps-type1**

Update to the huge CM-Super font package.

eps2pdf in support

Converts EPS files to PDF files.

euro Converts arbitrary national currency amounts using the Euro as a base unit.**frenchle in language/french**

Better support for `hyperref` and `pdfscreen`.

hitec This document class for technical papers in the "hi-tec" industry aims to escape from the academic look of the well-known L^AT_EX document classes.**mathtimepro in .../psnfssx**

For using Y&Y's new math font set MathTimeProfessional with L^AT_EX.

montexiv in language/mongolian

MonTeX Implementation Level IV offers vertical texts in Mongolian and Manju, intermixed with Latin texts.

onlyamsmath This package inhibits the usage of plain TeX and standard L^AT_EX math environments. Useful for class writers who want to force clients to use the environments provided by the `amsmath` package.**pdftricks** Support for `pstricks` in pdfTeX.**psgo in graphics/pstricks/contrib/psgo**

Draws Go diagrams with L^AT_EX.

repeatingindex Repeats item of an index if a page or column break occurs within a list of subitems.**swline in macros/generic/swrule.sty**

Typesets swelled rules: lines that are thicker in the middle than at the ends.

technics For writing technical documents. Uses the `fancyhdr` package to build headers containing document name, prepared by, approved by, etc.**TeXnicCenter in systems/win32**

An integrated development environment (IDE) for developing L^AT_EX documents on Windows.

tipa in fonts/tipa

Updates for Concrete, Palatino, CM Bright, and CM Type 1 TIPA (TeX International Phonetic Association) fonts.

tmmath in .../psnfssx

L^AT_EX packages for using MicroPress's TM-Math fonts.

tt2001 in fonts/ps-type1

This is a TeX .pfb font collection that contains almost all the EC (European Computer Modern) and TC (Text Companion) fonts in all possible design sizes, all the AMS fonts in all possible design sizes, plus some more.

underscore Package update adds a `[nohyphen]` option and compatibility with `babel`'s active characters.**November 2001****adrconv** Converts BIBTeX databases of addresses to the address format used in `scrletter` and `akletter`. It includes a TeX file to print the addresses in different layouts.**anysize** A simple package that can be used to set up the margins of a document. This package is obsolete. Use the package `typearea` to define your margins typographically. Use the `geometry` or `vmargin` package for everything else.**coordsys** Provides commands for typesetting number lines (coordinate axes) and coordinate systems in the `picture` environment.**em in .../psnfssx**

The PSNFSS support files for the "European Modern" (EM) fonts.

findhyph A Perl program to find all words hyphenated by TeX in a document.**flshort in info/lshort/french**

French translation of "The Not Short Introduction to L^AT_EX 2_&".

frenchpro in nonfree/languages/french

Update to French Pro.

ithyph in languages/hyphenation

Update of the hyphenation patterns for Italian.

jkthesis Just another L^AT_EX document class for formatting your thesis.**lahyph in languages/hyphenation**

Update of the hyphenation patterns for Latin.

latexmk in support

Given the source files for a document, `latexmk` issues the appropriate sequence of commands to generate a .dvi, .ps, .pdf or hardcopy.

minutes Package for writing minutes of meetings.**mxd in language/mongolian**

Package for typesetting Xewtee Dorwoljin, a Mongolian script conceived in the 17th century.

ocherrokee in languages/cherokee

For typesetting the Cherokee language, including a PostScript version of the official Cherokee font.

pdfcprot A package to activate pdfTeX's character protruding feature.**pdftex_for_oztex in nonfree/systems/mac/pdftex**

pdfTeX for OzTeX is a shareware package. This update brings the pdfTeX programs to the 20010806 source code.

pictex2 An extension to standard PiCTEX.

preview A standalone L^AT_EX style to extract selected elements from a L^AT_EX source into separate pages of a DVI file.

psfixbb in support

A program which sets or fixes the bounding box of a PostScript file, using Ghostscript, `pnmfile` and `pnmcrop`. This version handles files which have no `showpage` command, landscape files, and files produced on a Mac.

spie SPIE Proceedings format.

timetable in `macros/plain/contrib/timetable`

A T_EX macro for generating timetables.

usergrps in `usergrps`

HTML pages for T_EX usergroups.

December 2001

bold-extra in `.../misc`

Makes NFSS tables for the `fonts/cm/mf-extra/bold` fonts (bold `cmtt` and `cmcsc`).

bpchem docstrip-documented package for chemical typesetting.

epstopdf in support

Converts EPS files to encapsulated PDF files for use with pdfL^AT_EX.

invoice For writing invoices. Supports English, German, Dutch and French.

keystroke For typesetting the graphical representation of the keys on a computer keyboard.

latex2man in support

Translates UNIX manual pages written with L^AT_EX into a format understood by the UNIX `man(1)` command. HTML, Texinfo, or L^AT_EX code can be produced too.

LaTeX-Pack in support

Macros written in the NEdit scripting language to turn NEdit, the Nirvana text editor, into an advanced L^AT_EX editor. NEdit version 5.2 or later is required.

mattens Vector, matrix and tensor symbols.

metre Provides classicists with some of the tools needed to typeset scholarly publications dealing with Greek and Latin texts; special emphasis on Greek verse.

parskip in `.../misc`

Simply changing `\parskip` and `\parindent` leaves a layout that is untidy; this package helps alleviate this untidiness.

printlen in `.../misc`

Print L^AT_EX lengths in a variety of units.

venn in `graphics/metapost/contrib/macros`

Macros for Venn diagrams with METAPOST.

January 2002

acroread_new in support

(V.1.0) A Perl script to run Acroread under a different name, allowing several instances to be run independently.

antt in `fonts/psfonts/polish`

Antykwa Toruńska, a traditional Polish typeface.

bahyph in `language/basque` Hyphenation patterns for the Basque language. Compatible with the support for Basque in Babel 3.7.

cd in `.../other`

Print CD covers, featuring easy batch printing with crop marks, automatic track numbering and more.

cd-cover in `.../other`

Class for typesetting various forms of CD covers. V.1.4 adds automatic printing in landscape mode. Some minor documentation changes.

ctie in `web/c_cpp`

A CWEB-aware variant of TIE.

dk-bib in `biblio/bibtex/contrib`

A translation of the four standard BibT_EX style files (`abbrv`, `alpha`, `plain` and `unsrt`) into Danish. From the Norwegian translation by Dag Langmyhr extended with ISBN and ISSN fields.

epsfx.tex in `macros/generic/TEx-PS`

A T_EX macro package for including EPS graphics.

eurosans (V.3.0) Supports scaling of the Adobe Euro symbol fonts.

frhyph.tex in `language/hyphenation`

(V.2.11) Corrects the hyphenation of some words like cognitive, ignifuger, ignition, inexpugnable, etc.

gatech-thesis Georgia Institute of Technology theses.

LaTeX_WIDE in `nonfree/systems/win32` A demonstration version (1.3) of an integrated editor and shell for T_EX is free for noncommercial use, but without registration customization is disabled.

makor in `language/hebrew`

Fonts and macros to typeset Hebrew in a natural manner with a software switch to turn vowels off or on, and examples.

nath Natural math notation, a style to separate presentation and content in mathematical typography. Delivers a particular context-dependent presentation on the basis of a rather coarse context-independent notation—aims for producing traditional math typography.

polynom Implements macros for manipulating polynomials, for example it can typeset long polynomial divisions. V.0.14 adds a new `style=C`.

procIAGssymp Package for the preparation of a paper in the style of the proceeding of symposia sponsored by the International Association of Geodesy (IAG) published by Springer-Verlag.

qfonts in `fonts/psfonts/polish`

A collection of fonts based on URW's clones in the QX encoding layout.

rectopma Allows the reuse of the main content of `\title` and `\author` anywhere in the document. Includes `TestTitle.tex` (combination manual/example of use).

scientificviewer in *nonfree/systems/win32*

Scientific Viewer 4.0 is a free program for reading and printing (read-only) documents created with Scientific Notebook, Scientific Word or Scientific WorkPlace by MacKichan Software, Inc. Scientific Viewer can also be used to view many native L^AT_EX documents.

sihyph22.tex in *language/slovene*

Corrected Slovenian hyphenation patterns, as found on <ftp://sizif.mf.uni-lj.si/pub/i18n/tex>. Made by Matjaz Vrecko in 1990, maintained mostly by Leon Zlajpah in the '90s and uploaded on behalf of Primoz Peterlin.

srcctx Inserts source specials in .dvi files, enabling switching from a particular point in a previewer which supports them (e.g., Yap or xdvi-22.38) to the matching source in an editor which supports external direction to a specific point. Updated with a simpler implementation of input file tracking, using macro expansion instead of a stack. L^AT_EX's \input{...} command is now overloaded as well.

stack This class implements a stack mechanism without using T_EX's stack, so that your stacks are not related to the T_EX-grouping mechanism (useful for package developers). **stack.dtx** also defines a class **relinput** which implements a chdir-like mechanism so that \input can be done relative to the directory of any file.

tables in *macros/generic/tables*

An easy T_EX macro package for typesetting complex tables.

vtex in *systems*

Release 7.530 of MicroPress's T_EX for OS/2 and Linux (x86). Supports PDF security options and xpdf source specials.

February 2002**AcroTeX** in *.../webeq*

Formerly the Web/Exerquiz Packages, now bundled together as the "AcroT_EX Education Bundle." A series of packages for the creation of dynamic and interactive education (or other) materials in PDF format. Includes a library of JavaScript functions.

AlProTex.sty in *web/protex*

Minor modifications to this literate programming system, for compatibility with TeX4ht.

bibtool in *biblio/bibtex/utils*

Manipulation of BibT_EX files including: sorting and merging, pretty-printing, syntax checks with error recovery, semantic checks, generation of uniform reference keys, controlled rewriting with regular expressions, collection of statistics, and more. Includes documentation. C source only (no binary).

cropbox.sty in *.../ncctools*

Crop marks with customizable parameters.

cropmark.sty in *.../ncctools*

Produces a crop box around the page text area.

dcounter.dtx in *.../ncctools*

Dynamic counters.

desclist.sty in *.../ncctools*

An improved description list environment.

dcpic in *macros/generic/diagrams*

Macros for drawing Commutative Diagrams in a T_EX (including L^AT_EX and ConT_EXt) document.

draftcopy Print the (translatable) word 'DRAFT' on selected pages.

dvispell-dvistats in *support*

A Linux or MSDOS/djgpp version of dvispell (no longer limited to 8.3 file names). A comprehensive set of dictionaries is included.

eqlist Allows description-like lists to have equal indentation. Requires **eqparbox**.

esint Provides several alternate integral symbols (like \oiint) using the Computer Modern font.

extdash.dtx in *.../ncctools*

Flexible hyphenation of compound words; can conform to Russian typesetting conventions.

g-brief (V.3.0) A document class for L^AT_EX 2 ϵ for formless letters in German or English.

guitar (L^A)T_EX package for typesetting guitar chords over song texts. c.f. **gchords.sty** to typeset the chords graphically (not only by name). With references to other related packages. Requires **toolbox**.

labels.sty in *.../labels*

Provides a new package option **newdimens** which allows more flexible specification of the label layout.

latex4wp in *info*

A guide for word processor users designed to help convert knowledge and techniques of word processing into the L^AT_EX environment.

manyfoot.sty in *.../ncctools*

Adds footnote levels to the standard footnote mechanism includes an option for run-in paragraph footnotes (useful for critical editions).

moresize in *.../taupin*

Provides three larger font sizes: 'giant', 'Giant' and 'GIANT'.

nth.sty in *.../generic*

Ordinal numbering style: 1st 2nd 3rd 4th 5th, ...

nccbbb.sty in *.../ncctools*

Implementation of poor Black Board Bold symbols. Not needed with contemporary L^AT_EX systems.

nccboxes.sty in *.../ncctools*

Adjustable boxes useful for setting tables.

nccfancyhdr.sty in *.../ncctools*

New implementation of the functionality of the **fancyhdr** package for NCC-L^AT_EX.

nccfoots.sty in *.../ncctools*

Commands for generating footnotes with manual marks. For example, the command to mark a footnote with an asterisk is \Footnote{*} {text}.

nccmath.sty in .../ncctools

Extension of the `amsmath` package to combine its typesetting of display equations with NCC-L^AT_EX, especially for arrays.

ncsect.sty in .../ncctools

Extension of L^AT_EX's section, caption, and toc-entries generation technique. Improvements include: simple declaring of sections of any level, user-controlled typeout, customizing of section or caption tag, simple declaring of toc-entries, `\numberline` command never overlaps the following text, automatic calculation of right margin in table of contents, different captions for different float types and simplified handling of new types of floats.

nccthdm.sty in .../ncctools

Extension to the `\newtheorem` command. Four properties of theorems are used: numbering mode, theorem type, indent style and break mode. Easy customization including spacing and commands inserted after headers.

osa.bst in biblio/bibtex/contrib

BibT_EX style file for Optical Society of America (OSA) journals.

rotfloat This package bridges the `float` and `rotating` packages, extending them with rotated versions of new floats.

rotpages Allows pages to be rotated by 180 degrees and rearranged, so that they can be read by turning the printed copy upside-down.

smallcap in .../taupin

Make small caps a font family rather than a shape, thus allowing bold and slanted small caps.

smartref Expand capabilities of `\label` and `\ref` by adding commands similar to `\pageref` and other references to user-chosen counters. V.1.9 fixes a bug where equation labels were not used with A^MS-L^AT_EX, and makes `\smartref` robust.

TeX4ht in support

An enhanced version of a system for translating (L^A)T_EX sources into hypertext. Annual upgrade.

tif2eps in support/pstools

A PostScript program for converting TIFF to EPS.

titling Provides control over the typesetting of the `\maketitle` command, and makes the `\title`, `\author` and `\date` information permanently available. New examples added to the documentation.

toccenter.sty in .../ncctools

Provides commands `\ToCenter` and `\FromMargins`, which simplify the customization of page layout, either by specifying text element area dimensions, or by setting margins from the left, right, top, and bottom.

toolbox A package for (L^A)T_EX which provides some macros which are convenient for writing indices, glossaries, or other macros. V.4.2 keeps '#' in certain arguments unexpanded (as pointed out and solved by David Kastrup).

txt2latex in support

Perl script to facilitate batch conversion of largely unformatted ASCII text for use with L^AT_EX. Its sole purpose is to escape special characters and to fix single and double quotation marks.

varsects.sty in taupin

Redefines `\chapter`, `\section`, `\subsection` etc. with optional parameters changing vertical space before and after the title, and the font size, series, shape or family.

watermark.sty in .../ncctools

Provides watermarks, with options for left and right pages, temporary, `\thiswatermark` for current page only.

webomints in fonts

Provides support files for the border and ornament font Web-O-Mints freely available from Galapagos Design. Created by George Ryan, Web-O-Mints contains a rich assortment of typographic decorations inspired by historical sources.

york-thesis A L^AT_EX 2 ϵ thesis class definition for York University (Toronto, Canada).

March 2002**ama.bst** in biblio/bibtex/contrib

A modification of the IEEE BibT_EX style file to conform to the *American Medical Association Style Guide* guidelines for references. This is the style required by numerous biomedical journals (including JAMA, Cancer, NEJM).

bangtex in language

Font and class files for typesetting in Bangla (Bengali) and Assamese. Includes sample files, which and manual.

biblio in support

(V.1.2) A perl program for preprocessing your bibliographic references. After preprocessing the input `.tex` file it creates an output file with references included according to user-specified formatting.

bmeeps in support

A program to convert from PNG, TIFF, JPEG, NetPBM to EPS. V.1.0.2 fixes a bug related to malformed DSC comments.

CWEBbin in web/c_cpp

(V.3.64 [p20]) A set of change files (to be applied with TIE) that make the original sources usable with ANSI-C/C++ compilers on UNIX/Linux, MS Windows, and Amiga. Additional functionality, like macros, and macros for international documentation of CWEB programs, is introduced.

dvpn in language/devanagari/contrib/fonts

PostScript Type 1 renderings of Frans Velthuis' Devanagari METAFONT fonts for T_EX.

easy A collection of "easy" to use macros, including macros for writing tables, matrices, vectors, equations and custom bibliographies.

emTeXTDS in systems/os2/emtex-contrib

(Fixpak 8) The emTeX/TDS distribution for OS/2, featuring the latest releases of L^AT_EX, A_MS-L^AT_EX and other packages, as well as several bug fixes.

eulervm in fonts

This is a set of *virtual* math fonts, based on Euler and CM. Included is a L^AT_EX package, which makes them easy to use, particularly in conjunction with Type 1 PostScript text fonts. V.2.8 now works with sans serif text fonts.

icomma in .../was

An ‘intelligent’ comma, which yields proper spacing in decimal numbers as well as in mathematical expressions which does not depend on a specific encoding, and works with decimal numbers of arbitrary length. V.2.0 changes the behavior of the comma in math mode, so that the comma can be used both as a punctuation character and as a decimal separator.

iTeXMac in systems/mac

All in one Cocoa application for T_EX on Mac OS X.

javadvi in dviware

A dvi viewer and printer coded in Java. Needs Java RTE 1.3.

letrine (V.1.2)

Package designed to typeset various sorts of dropped capitals. Corrects a bug due to incorrect handling of rubber lengths; the bug occurred with `seminar.cls` for instance.

ly1-min.zip in .../psnfssx/ly1

A minimum set of support files for using the LY1 encoding with L^AT_EX. Fixes `texnansi.enc`, which had a wrong glyph name for the Euro symbol.

mathdots.sty in generic

Provides the command `\iddots` for diagonal dots (slanting the opposite of `\ddots`). Also modifies the `\ddots` and `\vdots` so that they change size with L^AT_EX’s size changing commands and in sub- and superscripts.

memoir

Peter Wilson’s flexible L^AT_EX document class for typesetting of books such as novels, biographies, histories, etc., with options for trim marks, draft appearance, various sizes and much more. V.1.1 adds bug fixes, native support for subfigures, additional facilities for verse typesetting, experimental marginpar-like extensions, and a new edition of the user manual.

ncclatex

The NCC class and NCCLATEX package. It works together with NCCTOOLS, release 2.03 or later. Has many customization features and service commands making it more powerful than standard L^AT_EX classes.

newlfm

For creating letters, faxes and memos; integrates the `letter` class with `fancyhdr` and `geometry` and includes support for an address database, languages, Avery labels and has full documentation.

paralist

This style file provides some new list environments. Itemized and enumerated lists can be

typeset within paragraphs, as paragraphs and in a compact version. Most environments have optional arguments to format the labels. Additionally, the L^AT_EX environments `itemize` and `enumerate` can be extended to use a similar optional argument. V.2.3b fixes bugs in `\setdefaultenum` caused by changes made in V.2.3a (which provided improved documentation and bug fixes).

s2latex-win32.exe in support/s2latex/win32

An MS-Windows (Win32) executable for s2latex, a utility which converts Scribe documents to L^AT_EX.

ssfms

For formatting science fiction and fantasy manuscripts in the format standard for the genre—12pt monospaced font with double spacing, etc. It is based on a previous style file called `sfms`. Full documentation is provided.

snapshot

Provides a snapshot of the external dependencies of a document insofar as they can be determined from inside L^AT_EX. V.1.13 gives suitable warnings for files listed in an `\RequireVersions` statement or that do not appear in `\@filelist`, and adds a `test` option.

texaide in support

A special version of Design Science’s MathType Equation Editor that generates T_EX and L^AT_EX.

TeXmacs in systems/unix

(V.0.3.5.11) A text editor inspired by the popular T_EX typesetting system and the Emacs editor. Runs on PCs under Linux and on Sun computers: “It is reasonable to expect that it will run on most UNIX/X-Windows systems in the near future.”

texshade (V.1.6)

For typesetting nucleotide and protein alignments in L^AT_EX.

TrueTypeToType42 in fonts/utilities

Creates a PostScript Type 42 font file from a True-Type font file.

verse

Aids in typesetting simple verse (poems). Update adds optional numbering, repeating patterns, `\|` as a linebreaking command and fixes `\label` when used with the first line of a poem.

webfiles in web

Improved documentation and code for PDF links.

yhmath in fonts

Type 1 font for the `yhmath` package.

April 2002**captcont**

Retain a figure or table number across several float environments—usually over several pages. It also allows control over the contents of the List of Figures and the List of Tables pages.

ccaption

Provides: continuation captions, unnumbered captions and legends, captions outside float environments, bilingual captions, etc. It also enables the definition of new float environments and their captions. V.3.1a changes the `subfigure` option to support version 2.1.2 of the `subfigure` package.

checkend.sty in .../bezos

Improves L^AT_EX's error messages for unclosed environments to make it easier to fix “`\end occurred inside a group at level N.`”

curve A class for making curriculum vitae (in distinct versions). Provides commands to create rubrics, entries in these rubrics, etc. V.1.2. adds: more translations, a starred form for the `\entry` macro, with a simplified syntax, and most important, support for standard bibliography inclusion.

datetime Changes format of `\today`. Provides commands for current time (12 or 24 hour forms), printing ordinal number forms (e.g., 3rd), and as a string (e.g., `\numberstring{3}` would print three).

dtk (V.1.6) Document class and style for *Die T_EXnische Komoedie*, the communications of Dante e.V., the German speaking T_EX users' group.

feyn in **fonts**

Can produce relatively simple Feynman diagrams within equations or text in a L^AT_EX document.

figsize Dynamic and automatic sizing of graphics and other elements relative to page.

fixme Provides insertion of “fixme” notes in draft documents, either in the margin of the document, as index entries, in the log file and/or as warnings on stdout or to summarize them in a list. Has support for AUC-T_EX. V.2.0 has many new features, including: a new layout option (`inline`), notes counters, a usage summary at the end of processing, more translations, and most important, three additional macros for including non-critical notes of different importance levels.

gauss Package for typesetting matrix operations, now compatible with *AMS*-T_EX.

gloss Allows creation of glossaries via BIBT_EX.

jurabib Supports various forms of short and long citations; now does Oxford-style formatting and includes support for Spanish. Many other changes and improvements.

kerkis in **fonts/greek**

The first freely available Type 1 font to fully support Greek (polytonic included) through Babel. An extension of the Bookman Old Style design. Also supports the Latin alphabet very well, with support for OT1 and T1 encodings as well as many missing ligatures and true small caps.

lettre (V.2.345) Designed to write letters with L^AT_EX in French, German, English. The letter aspect is entirely configurable, through a number of dimensions and macros, that can be put also in a so-called “institute package” called at initialization. Includes documentation and examples in all three languages.

mathabx in **fonts**

Large series of mathematical symbols. Initial release, “merely for evaluation”.

msc Draws Message Sequence Charts. Version 1.10 is a major update since the previous version, supporting the ITU standard MSC2000.

oztex in **nonfree/systems/mac**

V.5 of a shareware implementation of T_EX and associated programs for Apple Macintosh systems.

pdfpages Package to enable the inclusion of pages of external PDF documents. V.0.2d adds a `fitpaper` option; V.0.2g the option `addtotoc` which adds entries to the Table of Contents (e.g., for putting together conference proceedings).

plates Configurable `float` environment that is gathered to end of the document for special printing (e.g., plates, photos, color diagrams)

preview-latex A system for displaying inline images of selected parts of a file in Emacs source buffers. The style file is independently useful for extraction of selected text elements as images.

protocol Typeset meeting protocols (in German).

purifyeps in **support**

Makes `.eps` files work with both `dvips` and `pdflatex` by converting PostScript code into the “stylized format that METAPOST outputs.”

rcsinfo Use RCS (Revision Control System) information in a L^AT_EX document. For L^AT_EX2HTML users, `rcsinfo.perl` adds the functionality to that tool. With V.1.9, the initial `\$` of `\$Id` may be omitted.

shorttoc Adds the macro `\anothertableofcontents` (or `\anothertoc`) to import and display the table of contents of another document. It can also import and display the list of figures (or similar things) from another document.

showlabels Puts the names of `\labels` (or other macro arguments) into the margins of a draft document. V.1.4 adds an `[inline]` option and is more compatible with `wrapfig`.

skull in **fonts**

METAFONT version of a skull-and-crossbones symbol, with L^AT_EX bindings.

songbook (V.4.0) An all-purpose songbook style. Allows for three types of output from a single input file: words and chords books for musicians to play from, words-only songbooks for a congregation to sing from, and overhead transparency masters. Will also print a table of contents, an index sorted by title and first line, and an index sorted by key. It attempts to handle songs in multiple keys, as well as songs in multiple languages.

subfigure Package for providing support for the inclusion of small, ‘sub’, figures and tables. It simplifies the positioning, captioning and labeling of them within a single figure or table environment. In addition, this package allows such subcaptions to be written to the List of Figures or List of Tables if desired. V.2.1.3 fixes an incompatibility with several packages including the `tocloft` and the `ccaption` packages when used with the `hyperref` package and 2.1 includes bug fixes, second optional argument to allow control over the list-of-figures entry separately from the subcaption, a `tight` option for a

more compact arrangement, a `\subref` command to allow references to the subfigure, new `raggedright` option for the subcaption, and font formatting options only affect the label portion of the subcaption.

textpos Places boxes (containing text, or graphics, or a table, etc.) at absolute positions on the page. V.1.2 adds `[verbose]` and `[quiet]` options.

textopo Plots topology data of membrane proteins derived from PHD predictions, SwissProt database files, or manually entered data; and transmembrane domains as seen from above or beneath the cell membrane. Fully compatible with `\TeXshade`—this allows calculated shading based on sequence conservation and functional aspects of the residue side-chains. V.1.3 introduces two more helical wheel styles and the indication of the hydrophobic moment. Further, the `dvi` driver is not mandatory anymore as is `rotating.sty`.

titlesec Essentially a replacement—partial or total—for the L^AT_EX macros related to sections; namely, titles, headers, and contents (upgrade to V.2.5). Now `calc-savvy`, shorthand in lengths now accepts non-integer numbers and the unit is not hardcoded, more meaningful error messages, bug fixes.

tocbibind A package to add document elements like a bibliography or an index to the Table of Contents. The “List of ...” headings can also be put into the ToC. V.1.5c fixes a problem with `\tocchapter` in non-chaptered documents and a problem with `\tocetcmark`.

urlbst in `biblio/bibtex/contrib`
Perl script to adds a ‘webpage’ entry type, and support for general ‘url’ and ‘lastchecked’ fields, to arbitrary BIBL^EX style files.

WinShell in `systems/win32`
(V.2.2) A graphical user interface for easily working with T_EX. It is *not* a T_EX system so requires a system such as Mik^TE_X or T_EX Live.

wp2latex in `support`
Conversion from Wordperfect 6–8 into L^AT_EX. Update adds many new features: support for style `epsfig.sty`, fixed problem with filenames, working RTF parser, small attempt to do handle raster images, Windows GUI, ability to read WP3.x Macintosh files, etc.

May 2002

BSR2dvi in `dviware`
Alpha release. Utility for extraction of a standard `.dvi` file from Textures’ “all-in-one” file format.

cases.sty in `.../other/misc`
(V.2.5) Fixes formatting of over-full displays.

cmbright in `fonts`
Updates the CM Bright typefaces to V1.1.0g; corrects the fonts metrics of the text companion fonts, and improved documentation of the macro package. License changed to LPPL making possible the creation of “scalable vectorized versions...including Adobe Type 1.”

catdvi in `dviware`

The program is a `.dvi`-to-plain text translator capable of generating ASCII, Latin-1 and UTF-8 (Unicode) output. It aims to become a superior replacement for the `dvi2tty` utility. The current version is quite usable for previewing (on character-cell displays) `.dvi` files that contain mostly linear text. V.0.13 adds new or improved T_EX font encodings, European currency symbol support, improved math layout, usability enhancements and fixes.

crop The package provides different forms of cropmarks for trimming paper stacks, for camera alignment, and for visualizing the page dimensions, as well as options for reflecting and inverting the whole document. V.1.7 adds new options, the ability to set paper dimensions explicitly, and to suppress marks when printing duplex and improved compatibility.

cite Sync with T_EX Live; improve `babel` support.

dtxtut in `info`

“How to Package Your L^AT_EX Package”—a tutorial for advanced L^AT_EX 2_E users who want to learn how to create `.dtx` and `.ins` files for distributing their homebrewed classes and style files.

euro-ce in `fonts`

(V.3.0) METAFONT source of the official Euro currency symbol and CE logo including variants, in both solid and outline.

eurosym in `fonts`

METAFONT source for the official Euro currency symbol and CE logo including variants, in both solid and outline with a L^AT_EX style to access it. V.1.3 adds Type 1 fonts generated by `textrace` and modified by `pfaedit`.

fontinst-prerelease in `fonts/utilities`

Update for the font installation package which includes up-to-date `8y.etx` and `8r.etx` files, with the Euro symbol.

gradback A small addition to `pstricks` to generate gradient backgrounds.

jasthesis Conforms to thesis requirements of the University of Bristol, UK. It also approximates the BS4821:1990 standard.

logfilter in `nonfree/support`

A Java program that monitors the logfiles produced during a typesetting run and presents them in an interactive environment with the option when encountering an error in a file to open that file in an editor at the proper point in the file.

lshort in `info/lshort/slovak`

A translation of `lshort` (*The Not So Short Introduction to L^AT_EX 2_E*) into Slovak.

manuscript Package for recreating the appearance of a typewritten (monospaced with double linespacing) article manuscript.

mathpazo in `fonts`

Package (fonts and L^AT_EX style file) for mathematical typesetting with the Palatino fonts. V.1.003 for

use with version 9 of PSNFSS; also fills out Math Blackboard Bold, re-designed Φ , ς , ξ and ζ .

mn2eguide in *nonfree/macros/latex/contrib/mnras*

A guide for authors who are preparing papers for *Monthly Notices of the Royal Astronomical Society* (MNRAS), together with a sample article.

multicap Formatting captions inside *multicol*s.

mwcls Standard L^AT_EX classes, tailored to Polish printing customs.

ntheorem (V.1.203) Extensions for handling theorem-like environments.

piechartMP in *graphics/metapost/macros*

Enables even users with not much experience to draw their pie-charts with MetaPost. It supports basic 2D pie-charts as well pie-charts in 3D projection. The chief purpose for the piechartMP development was the support for presentations. The user can make segments invisible or hide a segment completely, in order to draw multiple different charts from one data-set.

plateX (V.1.2.3) A set of tools for typesetting in Polish. Some of its general purpose macros are used in other packages hence the present location (with a link from the old *languages/polish/plateX*).

proof in *support*

A Bash (Bourne Again Shell) based complement to an ordinary word processing program, able to coordinate the processing and viewing of TeX, L^AT_EX, METAFONT and METAPOST-sources.

psnfss in *macros/latex/required*

Version 9 of the PSNFSS system, supporting the use of the base 35 PostScript fonts and the Charter, Utopia and Pazo fonts with L^AT_EX. The TeXBase1 encoding now includes the Euro symbol, \hbar in **mathptmx.sty** improved, missing **scriptratio** redefinitions added. Documentation updated.

psnfss-source in *fonts/psfonts*

Sources of the latest PSNFSS version 9.

soul Provides flexible, hyphenatable letterspacing, underlining, overstriking, and highlighting. Now allows natural input syntax for accents, dashes etc., font switching commands in the argument, bug fixes, provides some additional features: highlighting, **\sloppyword**, improved documentation and programming interface. V.2.3 allows natural input syntax for accents, dashes etc., font switching commands in the argument, bug fixes, provides some additional features: highlighting, **\sloppyword**, improved documentation and programming interface. Compatible with **ccfonts.sty**, **beton.sty**, and **cmbright.sty**.

SQLTeX in *support*

A well documented Perl preprocessor to enable the use of SQL statements in L^AT_EX. V.1.4 adds an option to replace values in the database results using configurable replace files.

swimgraph Produces color graphs of a swimmer's performances, from data recorded in a text file, specified by arguments for the desired date range. Records and qualifying times may be included.

ttf2tex in *support*

Bash script to create all files necessary to use TrueType fonts with t_eT_EX. V.0.5 has preliminary support for OpenType fonts (TT based only), support for real small caps, oldstyle figures, support for **-n** and **-N** switches of **ttf2tfm**, an option to disable hyphenation, more robust and flexible and an updated and extended manual, and more.

ziffer Formats numbers according to the conventions used in Germany and some other countries (even in math mode). Update adds a switch to disable conversion of **--** for compatibility with newer versions of **amsmath**.

June 2002

astro in *fonts*

(V.2.00) METAFONT source for astronomical and astrological symbols: planets, signs of the Zodiac, and others, all with variant shapes, in three styles, plus six symbols for the phases of the moon.

bakoma in *systems/win32*

Upgrade of BaKoMa T_EX system to V.4.60.

CJK in *language/chinese*

(V.4.5.1) L^AT_EX support for Asian scripts: Chinese (both traditional and simplified), Japanese, Korean and Thai, in many encodings (including Unicode). Adds support for BG5+ and GBK character sets, Thai support for Babel, multifile documents, and much more.

ctable in *biblio/bibtex/contrib*

Provides commands to easily typeset centered table and (multiple-)figure floats, with footnotes.

decsci.bst in *biblio/bibtex/contrib*

Style for the journal *Decision Sciences* for use with the **natbib** package (author-year format citations).

EPIX in *graphics*

Command-driven (non-GUI) preprocessor for creating flexible, mathematically accurate line figures containing mathematical typography. Should run on any UNIX-like operating system; a C++ compiler and GNU Bash are required.

koma-script Reimplementation of the L^AT_EX classes (**article**, **report**, **book**, **letter**), "implementing European rules of typography and paper formats as documented in Tschichold (*Selected Papers on Book Design and Typography*). New version includes a completely rewritten letter class (**scrlltr2**), enhanced control over captions, support for date and time in Dutch and Croatian, and new developer interface for styling text elements.

layouts Enables display of various elements of a document's layout including: the general page layout; disposition of floats; layouts of paragraphs, lists,

footnotes, table of contents, and sectional headings; font boxes. Facilities are provided for document designers to experiment with the layout parameters. V.2.6 adds expanded page layout options.

memoir Peter Wilson's flexible L^AT_EX `documentclass` for typesetting of books such as novels, biographies, histories, etc., with options for trim marks, draft appearance, various sizes and much more. V.1.1 adds bug fixes, native support for subfigures, additional facilities for verse typesetting, experimental marginpar-like extensions, and a new edition of the user manual.

mpp in `web/noweb/kostas`

A multilingual pretty-printer for `noweb`.

newfile Provides convenient user level commands for reading and writing new files during a L^AT_EX run.

oletex in `systems/win32`

Allows embedding of pictures and other OLE objects into L^AT_EX.

placeins.sty in `.../misc`

(V.2.0) Now prevents floats from floating up past a `\FloatBarrier`, with an option to restore the original behavior.

poemscol Numbers lines of verse, makes separate end-note sections for emendations, collations, and explanatory notes, with notes tied to the line numbers, marks cases where stanza breaks fall on page breaks. Provides most of the features necessary for a critical edition of a poet's works, such as described by the MLA Committee on Scholarly Editions.

skak in `fonts`

Package for typesetting chess. V.1.0 has bug fixes and updated documentation: a short user guide, an example and a reference manual listing all available commands and the package options.

subfiles Allows individual typesetting of files that are `\input` and of the main file without making any changes to them.

thumbpdf in `support`

The package provides support for thumbnails with pdfL^AT_EX, and plain/L^AT_EX formats. Requirements: Perl5, Ghostscript, pdfL^AT_EX. V.3.0 adds: support for ghostscript anti-alias parameters, VT_EX's PostScript mode, option `--greek` for Greek style mode and signal handlers added for cleanup.

tocloft Provides control over the typography of the Table of Contents, List of Figures, List of Tables and user defined "List of ..." lists. Version 2.3 is compatible with the `koma` classes and adds control over the page style.

uk-tex-faq in `help`

Updated version also available via the Web at <http://www.tex.ac.uk/faq>. Includes a new answer from David Kastrup summarizing recent work on the T_EX document preparation environment.

xmldtex in `macros`

`xmldtex` is a system for typesetting XML files with

T_EX. It may be used on its own or in conjunction with another T_EX format (L^AT_EX is assumed for most examples). Update to sync with the enhanced version bundled with passiveT_EX.

- ◊ Mark LaPlante
109 Turnbrook Drive
Huntsville, AL 35824
laplante@mac.com
- ◊ William F. Adams
75 Utley Drive, Ste. 110
Mechanicsburg, PA 17055
willadams@aol.com

Tutorials

Introduction to pdfT_EX*

Thomas Feuerstack

Abstract

The following article is aimed at beginners and/or those interested in quickly creating PDF documents using pdfT_EX, without having to work intensively to gain background knowledge.

We will point out problems that frequently occur while creating PDF documents, as well as their solutions — of course, we don't claim to solve everything.

1 Introduction

If you look around in T_EX newsgroups or discussion lists, you will quickly see that a large number of the questions posed deal with bringing together T_EX and Adobe's Portable Document Format (PDF).

Beginners in particular seem to get caught on the same rough edges with amazing regularity. Often the first problem is just how, for example, you can convert a T_EX document into PDF. Once this hurdle is jumped, there is likely to be difficulties with using graphics, and fonts in particular.

* This article originally appeared in *Die T_EXnische Komödie* 2/2001, in German, with the title "Einführung in pdfT_EX"; it was translated by Steve Peter, and is published here with permission.

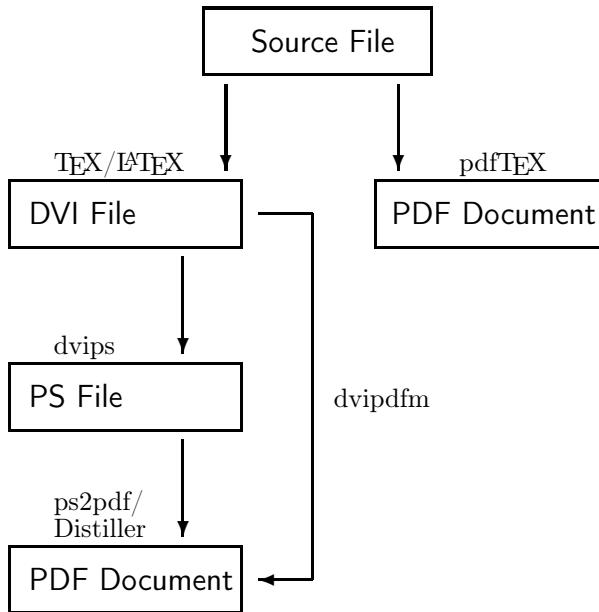


Figure 1: Even if not all roads lead to PDF, at least there are several.

In brief: most of these “rough edges” are easy to master with simple techniques—as long as you know what to do. This article is for those who aren’t (yet) in the know.

2 From TeX/LaTeX to PDF

It is well-known that all roads lead to Rome, nothing is impossible, and only you can make yourself happy. It therefore shouldn’t be surprising that there isn’t just one way to put your TeX file on the road to PDF, but more than a half dozen ways, given a bit of creativity; Figure 1 gives a roadmap.

In principle, we can differentiate roughly the two following ways.

The classic way: The document is TeXed as usual and converted to PDF with the aid of a special driver (e.g., `dvipdfm`) or via PostScript (`dvips` followed by `ps2pdf` or `Distiller`).

The elegant way: pdflatex creates the PDF format you want directly from your document—and without further detours.

Since the second alternative is usually not only faster to describe, but also in fact faster to use, let’s concentrate on it.

3 12 characters to make your document “portable”

Simply put `\pdfoutput=1` in the preamble of your document, and instead of the usual DVI file, you’ll get PDF. If however, you get neither PDF nor DVI,

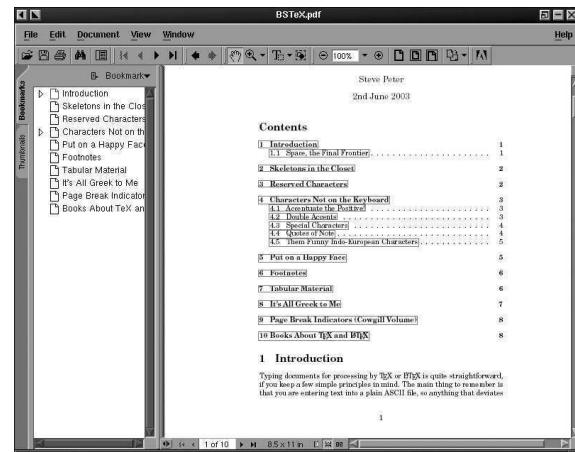


Figure 2: The result of the `hyperref` package.

but an error message on your screen: `Undefined control sequence...`, then you’ve tried to run (as you previously did) `latex` instead of `pdflatex`. You should change this right away.

Just by adding `\pdfoutput=1` you got the desired effect; you can find the whole set of specific pdflatex sequences in [6]. With just a little more effort (29 characters to be exact) you can drastically increase the effectiveness of your PDF output.

Including `\usepackage[pdftex]{hyperref}` as the last package in the preamble of your TeX file not only causes PDF output, but also gives you the following functionality for free:

- All tables, such as table of contents, table of figures, etc., will automatically be linked to the text. Clickable text is shown surrounded by a colored frame.
- A list of PDF bookmarks is created from the structure of the table of contents. Using these bookmarks in a manner similar to the table of contents itself, you can quickly navigate to various parts of the text.
- Cross references within the text, like footnotes, indices, etc., are hyperlinked to the text.

If you now load your freshly-made PDF document into Adobe Reader (or any other PDF reader, for that matter), it might already look like Figure 2.

4 Of course `hyperref` can do much more

By using a selective set of `hyperref` options you can tweak the appearance of your PDF document to almost the smallest detail. The options, as is usual in declaring L^AT_EX packages, are given in square brackets before the actual package name. We can distinguish between parameters that are simply passed

in (as in the example above with the parameter `pdftex`), and those where the parameter requires a value, such as `pdfstartview=Fit`.

Thus, a typical declaration of `hyperref` might look like the following.

```
\documentclass{article}
...
\usepackage[pdftex,a5paper,%
            pdftitle={The Ducks},%
            pdfauthor={Mother Goose},%
            colorlinks=true,%
            linkcolor=blue%
            ]{hyperref}
...
\begin{document}
...
```

The complete range of all specifiable options can be found in the documentation by Sebastian Rahtz [5], with a somewhat more complete description in [4]. The possibilities described in the manuals may seem to be impossibly many, and often just as hard to understand. Therefore, let's now look at a few commonly used options more closely.

4.1 General options

The ‘backend driver’, in our case `pdftex`, informs the `hyperref` package how you want to create your PDF document. If, in spite of this article, you choose some other way to create PDFs (cf. Figure 1), you will need to change this option correspondingly.

Since there may be other L^AT_EX packages in the document that depend on a backend driver, for example the `graphicx` package, you can specify the driver as an option to `\documentclass`; the option is then passed to all packages that need it, and doesn't have to be given each time you call `\usepackage`.

The `paper size` is set to `a4paper` by default, but you can change it to `a5paper`, `b5paper`, `letterpaper`, `legalpaper` or `executivepaper`.

The two options `draft` and `debug` play a special role. While `draft` turns off all `hyperref` functionality, `debug` places into your log file additional diagnostic information.

4.2 Screen display and document information

Normally your document will be displayed by Acrobat Reader as in Figure 2. You can change this with the help of the following options:

`pdfpagemode` determines how Acrobat Reader is opened. Possible values are `UseOutlines` (default), to show the bookmarks in the left frame; `UseThumbs`, to show the individual pages as thumb-

nails in the left frame; `None` and `FullScreen` (synonyms), to show the document without menubar, bookmarks, thumbnails, or a left frame in general.

The *zoom factor* of the document can furthermore be configured by the following:

`pdfstartview` sets the size of the page first displayed (normally page 1, but that can be changed with `pdfstartpage`). Possible values are `Fit`, to show the whole page; `FitH`, to fit the width of the page in the window; or `FitB`, to fit the width of the contents to the window.

`pdfview` determines the viewing size of a page that is opened via hyperlink. The possible values are the same as those for `pdfstartview`; by default it is the same viewing size as the previous page.

The *additional information* about the document, accessed in Acrobat Reader with the menu command `File – Document Properties – Summary`, typically includes the document title, the author's name, as well as keywords to enable automated searches.

You can set these values with options such as `pdftitle`, `pdfauthor`, `pdfkeywords`, ... Our list of options is getting slowly but surely longer.

```
\documentclass{article}
...
\usepackage[pdftex,a5paper,%
            pdftitle={The Ducks},%
            pdfauthor={Mother Goose},%
            pdfkeywords={Ducks Fowl},%
            pdfpagemode=FullScreen,%
            pdfstartview=FitB%
            ]{hyperref}
...
\begin{document}
...
```

4.3 Configuring bookmarks

You can turn off automatic creation of bookmarks with `bookmarks=false`, which (as you learned in the last section) means that the option `pdfpagemode` is set to `None`.

Only top level bookmarks are typically shown; to see lower bookmarks, you have to manually “expand” the hierarchy. You can change that with the `bookmarksopen=true` option, which will cause the entire bookmark tree to be displayed.

If the bookmarks become a bit too prominent when you do that, you can reach a usable compromise with the `bookmarksopenlevel` option.

4.4 Colors

As you will quickly see, links created by `hyperref` are colored. Links are colored differently depending on the type of reference—for example, internal links (like in the table of contents) are red, blue for web links or email addresses, green for links to the bibliography, and so forth.

The options `citebordercolor` (bibliography), `linkbordercolor` (“normal” links), `urlbordercolor` (web links and email), etc., override the default color values. You enter the color you want as three RGB values, which must be $0 \leq \text{RGB} \leq 1$. For example, `linkbordercolor={0 0 1}` will change the standard red of “normal” links to blue.

The width and style of the link box can be configured via `pdfborder`, which also requires a numeric triplet. `pdfborder={0 0 5}` changes the line from 1 point to 5 points, with the result that you can hardly see the text.

Especially interesting effects can be achieved by adding an optional parameter to the numeric triplet. For instance, `pdfborder={0 0 1 [3]}` creates a box with a 1 point border that is broken—the higher the value of the optional parameter, the longer the breaks in the line.

`colorlinks=true` removes the box around the link (which `pdfborder={0 0 0}` will also do), but the text of the link is now colored. The color scheme for boxes is also valid for colored text links, and can be changed with the options `linkcolor`, `citecolor`, `urlcolor`, etc.

What’s good about this is that for changing colors used for text, you can use the color names in the `color` package; in other words, a declaration such as `linkcolor=blue` works just fine.

What’s not so nice about this is that when you print, colored text is of course printed in color, which means the table of contents is red, or gray on a black-and-white printer.

4.5 Global settings

Settings that are the same for several documents need not be entered each time via the `\usepackage` declaration. They can instead be placed into a systemwide `hyperref.cfg` file.

For example, if all of our documents use `colorlinks` and the “normal” link color should be blue, the following will accomplish it:

```
%% hyperref.cfg: Global Settings
\hypersetup{colorlinks=true,
            linkcolor=blue,
            pdfproducer={Birds-of-a-Feather},
            pdfstartview=FitB}
```

Then the list of options for the `hyperref` package can be reduced to only those needed just for this document.

```
\documentclass{article}
...
\usepackage[pdfTEX,a5paper,%
           pdftitle={The Ducks},%
           pdfauthor={Mother Goose},%
           pdfkeywords={Ducks Fowl}%
           ]{hyperref}
...
\begin{document}
...
```

Options passed in the `\usepackage` declaration override options from the `hyperref.cfg` file that have the same name.

5 Your own links

Although the `hyperref` package already automatically links for you almost everything in a `TEX` document that ought to be, you may actually want to add your own links to it. Links to URLs (i.e., to Web or email addresses) and document-internal links should be differentiated.

Links to URLs can be made quite simply with the command `\href`.

```
A new day dawns in
\href{http://www.fburg.com}{Featherburg},
home of the richest
\href{mailto:d.duck@fburg.com}{duck}
in the world.
```

In the example above, the words `Featherburg` and `duck` are colored according to the setting given by `colorlinks`. By clicking on `Featherburg` we will probably land on the home page of the city; by clicking on `duck` you can send D. Duck an email—but don’t bother asking for money!

For *hyperlinks within a document* you need to specify both the `\hypertarget` and `\hyperlink`. With `\hypertarget` you define a jump location (also known as an anchor), which `\hyperlink` will then target.

```
In order to protect his financial
interests D. Duck has lived in a
\hypertarget{trove}{safe} for decades.
```

```
...
“Let’s go”, the safecrackers shouted,
“Today it’s the
\hyperlink{trove}{tightwad}’s turn!”.
```

In this example, the word `tightwad` is made clickable—using the mouse button will send us directly to the `safe`. The connection between link and target is made by the freely-chosen word `trove`.

6 Inclusion of graphics

Conventional wisdom has it that a picture is worth a thousand words—which has lead many an author to the conclusion that the value of their work would increase according to the number of graphics it contained.

The most complicated graphic that a pdftEX user puts into their document is generally the first one. So let's start at the beginning.

If you have already put graphics into your “normal” L^AT_EX documents, ideally you should find there the following constructs:

```
\documentclass{article}
...
\usepackage{graphicx}
...
\begin{document}
...
If we want to use examples from the world
of comics, we can include pictures.

\begin{figure}
\includegraphics{entenhausen.eps}
\end{figure}
...
```

After you put your document on the road to PDF by including the `hyperref` package, your first `\includegraphics` command [1] will likely cause an error. pdftEX directly supports the inclusion of JPEG, PDF and PNG graphic files—but not EPS!

I started a new paragraph here to allow you some more time to let the shock sink in. For years, you've struggled to get all the necessary graphics into Encapsulated PostScript, and now this.

Not to try your patience further, but I want to offer here an explanation of why this is, and not unnecessarily; note quite simply that pdftEX has considerable difficulty (to put it kindly) with PostScript constructs of any sort, including the beloved package `pstricks`.

But keep your chin up, and to close this section, consider the following:

- In the future, you won't have to spend any more time converting to EPS, since most graphics programs directly support the formats given above.
- For those cases where you already have EPS pictures included, but don't have copies in any of the above-mentioned formats, in most cases the `epstopdf` converter, which ought to be included in every T_EX installation, will handle the conversion.

7 Fonts and typefaces

Finally! You're using the `hyperref` package, and you've got it all configured just as you like it, all your graphics are cleanly converted for pdftEX, and your document doesn't have any errors when you T_EX it. We're almost to the holy grail.

All too often though, you're coming down the home stretch, and then Acrobat Reader offers something frightful: the typeface looks as if instead of the usual T_EX fonts, the font **Horror** is being used.

If you didn't really choose **Horror** as your base font (thereby being quite happy with the typeface), then the cause is probably the following:

T_EX normally uses the so-called packed or bitmap fonts (e.g., `cmm10.pk`), which cause Acrobat no small amount of difficulty for screen display. pdftEX automatically avoids this issue, in that instead of bitmaps, it uses outline-based PostScript varieties of Computer Modern (fonts are the principal case where the PDF needs the PostScript construct). You can recognize this in the log file near the end of a pdftEX run when you see messages like:

```
...<cmm10.pfb><cmbx12.pfb>...
```

To come back to the chewed-up screen representation of your document, the cause is usually in the preamble of your document, more specifically in the declaration

```
\usepackage[T1]{fontenc}
```

with which you want, among other things, to improve hyphenation¹, but instead of Computer Modern it embeds the EC fonts which are not included in PostScript form in most T_EX installations, by default. A sure confirmation of this hypothesis is when pdftEX writes to the log file messages like `<...ecrm1000.pk>`.

Considering the question of which is preferable, better hyphenation or beautiful screen appearance, the answer must of course be “both!” We have several possibilities to choose from:²

- Instead of the usual T_EX fonts, use the standard PostScript fonts (Times, Helvetica, etc.), via `\usepackage{times}` for example. Possibly also use the `txfonts` (Times) or `pxfonts` (Palatino) to get companion symbols.
- If you (as I) are dependent on a typical T_EX layout, invest \$150 (including shipping) and get the European Modern Fonts from Y&Y.

¹ This is useful for continental hyphenation, but is not necessary for hyphenation in English.

² Two other notable possibilities have been developed since this article was originally written: The `cm-super` and Latin Modern PostScript font collections.

- If that's a bit too rich for you (like one of the frequently-occurring characters in this article), get the (free) `ae` package and include that in your document. With `ae` you get a modern font encoding, but the actual glyphs come from the Computer Modern PostScript fonts.
- Use the freely available Latin Modern or CM-super font collections now available from CTAN.

8 Additional PostScript fonts

Even if using the standard PostScript fonts is quite problem-free, including other, more exotic fonts can present a very labor-intensive task. This may be necessary since many firms and institutions, thanks to corporate design, insist on typefaces such as Frutiger, Futura, Garamond, etc.

Including such Type 1 fonts, for the most part commercially-sold, is certainly not trivial. However, as is well-known, with `TEX` almost nothing is impossible; consider the following:

- If you've purchased a PostScript font, its distribution should include files with the extensions `.pfb`, `.afm`, and/or `.pfm`.
- For use with `pdftEX` you now still need the accompanying files with the extensions `.vf`, `.tfm`, `.map`, and `.fd`, and it is quite useful to have a style file to use the whole lot as a package in your document.

As a rule, these files are *not* on the disk when you buy fonts! With a bit of luck, you might find them on CTAN in the `fonts/psfonts` directory.

- Look in the directory structure of your `TEX` installation, and try to find out where other font packages place files with the above-named extensions. Play around with putting the new font files in the same places.
- Look for the `pdftex.cfg` configuration file and add your `.map` file to it. For example, for a map file named `newfont.map`, you would add the line `map +newfont.map`.

If you did everything correctly, the new fonts should now be usable with `pdftEX`. If you have difficulties, you might consult [2] and/or [3] for detailed information.

If you want to know a bit more about what you've just actually done, and what those files with the weird extension names are good for, I have a tip for you.

Come to the next meeting of Dante and ask Walter Schmidt, or better yet: come to one of Walter Schmidt's talks.

The only thing I have left now is the →

9 Conclusion

← that will close this overly-long article. If this introduction to `pdftEX` still seems complicated, don't worry, just jump right in.

When creating new documents, using `pdftEX` directly causes no problems in general, since you don't have to do anything extra aside from calling the `hyperref` package. Instead of the usual previewer or print driver, Acrobat Reader can be used without difficulty for both tasks.

It can be somewhat more complicated to repurpose existing `TEX` files for `pdftEX`. Especially in cases where many (EPS) graphics need to be converted, it is frequently more efficient to choose the traditional way and use tools like `pstopdf` for PDF creation.

References

- [1] D.P. Carlisle. *Packages in the graphics bundle*, Jan. 1999. <http://www.ctan.org/tex-archive/macros/latex/required/graphics/grfguide%.ps>.
- [2] Peter Flynn. "Installing PostScript fonts" in *Formatting Information*. <http://www.ctan.org/tex-archive/documentation/beginlatex/html/chapter8.%html#instfonts>.
- [3] Philipp Lehman. The font installation guide. <http://www.ctan.org/tex-archive/info/Type1fonts/fontinstallationguide.pdf>.
- [4] Heiko Oberdiek. PDF information and navigation elements with `hyperref`, `pdftEX` and `thumbpdf`. In *EuroT_EX'99 Proceedings*, 1999. <http://www.tug.org/applications/hyperref/ftp/doc/paper.pdf>.
- [5] Sebastian Rahtz. *Hypertext marks in L_AT_EX: the hyperref package*, Jun. 1998. <http://www.tug.org/applications/hyperref/>.
- [6] H_àn Th_é Th_{anh}, Sebastian Rahtz, and Hans Hagen. *The pdftEX user manual*, Jan. 2000. <http://www.tug.org/applications/pdftex/pdftex-a.pdf>.

◊ Thomas Feuerstack
FernUniversität in Hagen
58084 Hagen
Germany
`Thomas.Feuerstack@fernuni-hagen.de`

LaTeX

Constructing circuit diagrams with *pst-circ*^{*}

Herbert Voß

Abstract

In this article the group of packages collectively referred to as *pstricks* is extended with a package which adds the ability to generate electronic circuit diagrams. These diagrams follow standard conventions in engineering and can be particularly handy when it comes to simple construction of circuit diagrams for publication, without having to come to grips with vector drawing programs.

1 The basic concept

The package *pst-circ* described here is in principle based on *pst-node* in which a graphical object is placed between reference points or “nodes” (Figure 1), while the underlying details of the object *per se* are secondary.

One can define such a coordinate pair two ways:

```
resistor(-1.5,0)(+1.5,0){$R$}
```

or

```
\pnode(-1.5,0){A}
\pnode(1.5,0){B}
resistor(A)(B){$R$}
```

where, in the second case, the location of A and B has been defined separately. When there is no need to explicitly define a coordinate pair as a node, you can simply use the (x, y) -form.

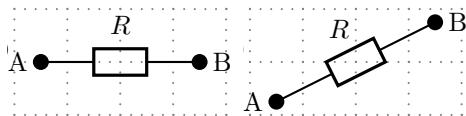


Figure 1: Example displays

The text labels are displayed horizontally by default but can be rotated by invoking a specific parameter. Listing 1 illustrates the coding for Figure 1 (without the code which generated the coordinate axes and labels on the nodes).

Listing 1: Coding for example displays

¹ \begin{pspicture}(-2,-1)(2,1)

* Translated from *Die TEXnische Komödie* 3/2003, p. 33-49, with permission. Translation by Douglas Waud.

```

2   \pnode(-1.5,0){A}
3   \pnode(1.5,0){B}
4   \resistor(A)(B){$R$}
5 \end{pspicture}\hspace{0.5cm}
6 \begin{pspicture}(-2,-1)(2,1)
7   \pnode(-1.5,-0.75){A}
8   \pnode(1.5,0.75){B}
9   \resistor(A)(B){$R$}
10 \end{pspicture}
```

2 Circuit components

Tables 1 through 3 give summaries of the circuit components presently available (version 1.2). For purposes of description these are divided into

Dipoles: Two terminal circuit elements.

Multidipoles: Combinations of two terminal circuit elements.

Tripoles: Three terminal circuit elements.

Quadrupoles: Four terminal circuit elements.

So the tables can focus on the key features, possible options are not included. These are covered later in this article. Macros are called generally in the format

```
<Object name>(<node 1>)(<node 2>)
... (<Label>)
```

The last argument (Label) can be empty. Some of the tripole do not make this argument available, so a label must be generated separately.

2.1 Two terminal circuit elements

See Table 1 for the dipole elements available. Here we have the greatest number of available objects. All but *wire* can have a label specified.

The symbol *circledipole* can be used, in particular, for display of current and voltage sources as variations on the battery symbol. With the *labeloffset=0* option, one can place the label in the center of the circle as, for example, in Figure 11.

2.2 Multidipole

This is nothing more than a linear chain of dipoles which can lead to a simplification of the coding. Thus one can, for example, represent a real inductance simply by defining a new macro as a multipole as in Figure 2.

Listing 2: Code for the multidipole of figure 2

```

1 \begin{pspicture}(-2.75,-0.5)(2.75,1)
2   \pnode(-2.75,0){A}
3   \pnode(2.75,0){B}
4   \multidipole(A)(B)%
5     \coil{$L$}%
6     \resistor{$R$}.
7 \end{pspicture}
```

Table 1: Predefined two terminal circuit elements

Name	Macro	Output
Battery	\battery	U_B
Voltage source	\Ucc	U
Current source	\Icc	I
Resistance	\resistor	R
Capacitance	\capacitor	C
Inductance	\coil	L
Diode	\diode	D
Zener-Diode	\Zener	D
LED	\LED	D
Lamp	\lamp	
Switch	\switch	C
Wire	\wire	
Arrow	\tension	u
Circle	\circledipole	

Note that the period at the end of line 6 ends the definition of the multipole. The number of dipoles has no hard limit. The size of the drawing space presents the practical limit.

2.3 Three terminal circuit elements

In Table 2 the three terminals have the names of the labels added to the sample displays. This makes it clearer what the order of the (A), (B) and (C) calls in the coding refers to. A label is available only for switches.



Figure 2: Code defining a multidipole and the result

For the other two, one must use the \uput command (see Listing 1).

Table 2: Predefined three terminal circuit elements

Name	Macro	Graphic
Op Amp	\OA	
pnp Transistor	\transistor	
Potentiometer	\potentiometer	
SPDT switch	\Tswitch	

2.4 Four terminal circuit elements

In Table 3 the names of the nodes are again superimposed on the example diagrams as (A), (B), (C), and (D) to facilitate matching the coding. The order of these nodes is meaningful since the calls all involve (A)(B)(C)(D) in a specific order.

3 Options

The package `pst-circ` itself has no options of any kind; in contrast, the macros have a considerable number, including options for color displays which will be particularly useful in PDF output.

3.1 Circuit direction arrows

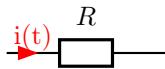
Each object can be provided with an arrow indicating direction of current. The example in Figure 3 illustrates use of all the available options for the characterization

Table 3: Predefined four terminal circuit element

Name	Macro	Graphic
Transformer	\transformer	
Opto-coupler	\optoCoupler	

of the arrow. The default values for these options can be obtained from the `pst-circ` documentation.

The direction of current is determined by the order of the nodes—in the previous example of coding, from (A) towards (B). This can, however, be changed with the option `\directconvention=false`.



```

1 \begin{pspicture}(-2,-1)(2,1)
2   \pnode(-1.5,0){A}
3   \pnode(1.5,0){B}
4   \resistor[%]
5     intensity=true,%
6     intensitycolor=red,%
7     intensitylabel=i(t),%
8     intensitylabelcolor=red,%
9     intensitylabeloffset=0.3,%
10    intensitywidth=2pt](A)(B){$R$}
11 \end{pspicture}

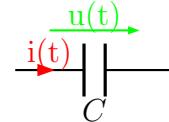
```

Figure 3: Indicating current direction

3.2 Arrows indicating potential differences

Analogous to an arrow showing current direction, each object can have an arrow to indicate a potential difference. The example in Figure 4 illustrates use of all the available options for the characterization of the arrow. The default values for these options can be obtained from the `pst-circ` documentation. The listing shows only the options for the potential difference arrow; those for the current arrow would be the same as in the preceding example.

This example also shows that the label, “C”, which normally appears above the object, can also appear below. Furthermore the default direction of the arrow



```

1 \begin{pspicture}(-2,-1)(2,1.5)
2   \pnode(-1.5,0){A}
3   \pnode(1.5,0){B}
4   \capacitor[%]
5     labeloffset=-0.75,%
6     tension=true,%
7     tensioncolor=green,%
8     tensionoffset=0.75,%
9     tensionlabel=u(t),%
10    tensionlabelcolor=green,%
11    tensionlabeloffset=1,%
12    tensionwidth=\pslinewidth](A)(B){$C$}
13 \end{pspicture}

```

Figure 4: Displaying a potential difference

indicating potential is, like that for current, directed from B to A.

If, for example, an alternative convention is desired, arrows can be easily reversed with the options

`dipoleconvention=generator|receptor`

where `load` is the default. Also both arrows can be jointly reversed with the option

`directconvention=false|true`

3.3 Parallel circuits

This case comes up frequently, for example, in the equivalent circuit for a real capacitor, so `pst-circ` provides extra options for it. Figure 5 shows a simple example with all the options including those in the preceding examples (but without repetition of the code already presented).

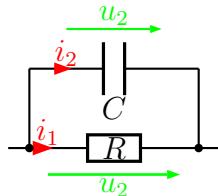
The idea is that both objects themselves have nodes but one of the two, with the option `parallel`, is placed above or below the other. A negative value for `parallelarm` would place the object under the other; for example, with the specific value -2 , the separation would be 2 units. Similarly, parallel connections with three objects are possible.

3.4 Alternative display forms

Different conventions for presentation frequently exist, particularly between European and American symbols. Most can be handled by the options summarized in Table 4.

3.5 Variable elements

With the option `variable=true`, diagonal arrows are added to indicate the component (resistor, inductance, or capacitor) is variable, as illustrated in Figure 6:



```

1 \begin{pspicture}(-2,-1)(2,3)
2   \pnode(-2,0){A}
3   \pnode(2,0){B}
4   \resistor[%]
5     labeloffset=0,% [ ... ]
6     tensionwidth=1pt](A)(B){$R\$}
7   \capacitor[%]
8     labeloffset=0,% parallel=true,% parallelnode=true,% parallelsep=0.2,% parallelarm=1.5,% [ ... ]
9     tensionwidth=\pslinewidth](A)(B){$C\$}
10 \end{pspicture}

```

Figure 5: A parallel circuit

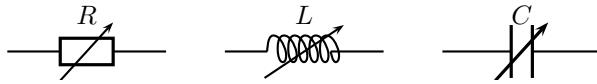


Figure 6: Adjustable objects

3.6 Transistors

The transistor in Table 1 was presented in bare-bones fashion. Figure 7 shows further options.

- The boolean option `Transistorinvert` reverses the emitter and collector connections.
- The higher-level option `intensity=true` sets all three current arrows for base/emitter/collector to `true`.
- The colors for the current arrows can be set globally with the option `intensitycolor`, and for labels with `intensitylabelcolor`.

3.7 Operational amplifier

Table 1 illustrated a standard operational amplifier with infinite gain. Figure 8 illustrates some of the possible options. Here, as with the transistor, one can simplify coding with the higher level option `intensity` to control all three current arrows.

With the option `tripolestyle=french` one can also invoke the alternative presentation illustrated in Figure 9.

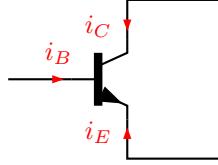
Table 4: Alternative formats

Macro	dipolestyle=...	Graphic
\resistor -		
	zigzag	
\coil -		
	rectangle	
	curved	
	elektor	
	elektorcurved	
\capacitor -		
	chemical	
	elektor	
	elektorchemical	
\diode -		
	thyristor	
	GTO	
	triac	

A final option, `OAinvert`, makes it possible to reverse the connections of the two inputs.

3.8 Transformers

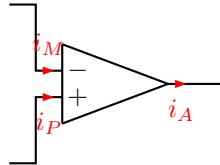
Finally Figure 10 illustrates the options available for display of transformers. Again the labels on the current



```

1 \begin{pspicture}(4,3)
2   \pnode(0,1.5){A}\uput[-45](A){A}
3   \pnode(4,3){B}\uput[-135](B){B}
4   \pnode(4,0){C}\uput[135](C){C}
5   \transistor[%]
6     transistorcircle=false,%
7     transistortype=NPN,%
8     transistorbase=true,%
9     transistorcollector=true,%
10    transistoremitter=true,%
11    transistorbaselabel={\red \$i_B\$},%
12    transistorcollectorlabel=$i_C$,%
13    transistoremitterlabel=$i_E$,%
14    intensitycolor=red,%
15    intensitylabelcolor=red](A)(B)(C)
16 \end{pspicture}

```

Figure 7: Transistor options

```

1 \begin{pspicture}(4,3.5)
2   \pnode(0,3){A}
3   \pnode(0,0){B}
4   \pnode(4,1.5){C}
5   \OA[%]
6     OAperfect=false,%
7     OAiplus=true,%
8     OAiminus=true,%
9     OAiout=true,%
10    OAipluslabel=$i_P$,%
11    OAiminuslabel=$i_M$,%
12    OAioutlabel=$i_A$,%
13    intensitycolor=red,%
14    intensitylabelcolor=red](A)
15 \end{pspicture}

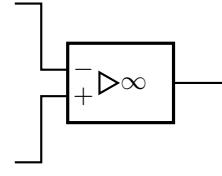
```

Figure 8: Operational amplifier options

arrows can be controlled jointly with the higher level option `intensity`. Furthermore, the German convention of using rectangles for the windings can be chosen with the option `dipole-style=rectangle` to match the parallel usage shown earlier with inductances.

4 Example of use for a complex diagram

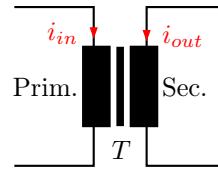
With `pst-circ` one cannot readily design extremely complex circuits. Still one can easily display small circuits or equivalent circuits for real devices if one works in the framework of a coordinate system such as that



```

1 \begin{pspicture}(4,3.5)
2   \pnode(0,3){A}
3   \pnode(0,0){B}
4   \pnode(4,1.5){C}
5   \OA[triplepolestyle=french](A)(B)(C)
6 \end{pspicture}

```

Figure 9: Alternative format for an operational amplifier

```

1 \begin{pspicture}(4,4)
2   \pnode(0,3){A}
3   \pnode(0,0){B}
4   \pnode(4,3){C}
5   \pnode(4,0){D}
6   \transformer[%]
7     dipolestyle=rectangle,%
8     primarylabel={Prim.},%
9     secondarylabel={Sec.},%
10    transformerprimary=true,%
11    transformerisecondary=true,%
12    transformeriprimarylabel=$i_{in}$,%
13    transformerisecondarylabel=$i_{out}$,%
14    intensitycolor=red,%
15    intensitylabelcolor=red](A)(B)(C)(D){$T$}
16 \end{pspicture}

```

Figure 10: Transformer options

provided by `psgrid`. Figure 11 displays the equivalent circuit for a constant current generator as produced by the code in Listing 3.

Listing 3: Code for Figure 11

```

1 \psset{intensitycolor=red,%
2   intensitylabelcolor=red,%
3   tensioncolor=green,%
4   tensionlabelcolor=green,%
5   intensitywidth=3pt}%
6 \begin{pspicture}(-1.5,-0.2)(13.5,9)

```

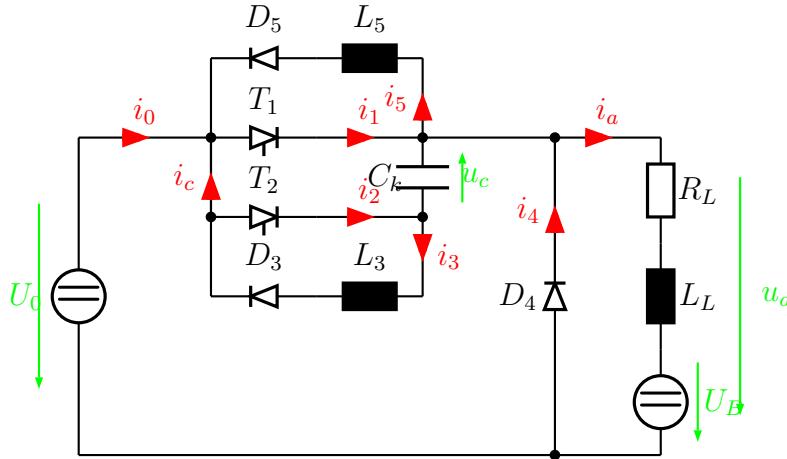


Figure 11: Example of capabilities of `pst-circ`

```

7 \psgrid[griddots=5,gridlabels=7pt,subgriddiv=0]
8 \circledipole[
9 tension,%
10 tensionlabel=$U_0$,%
11 tensionoffset=0.75,%
12 labeloffset=0](0,0)(0,6){\LARGE\textrm{bf}{=}}
13 \wire[intensity,intensitylabel=$i_0$](0,6)
    (2.5,6)
14 \diode[dipolestyle=thyristor](2.5,6)(4.5,6){$T
    _1$}
15 \wire[intensity,intensitylabel=$i_1$](4.5,6)
    (6.5,6)
16 \multidipole(6.5,7.5)(2.5,7.5)%
17 \coil[dipolestyle=rectangle,labeloffset
    =-0.75]{$L_5$}%
18 \diode[labeloffset=-0.75]{$D_5$}.
19 \wire[intensity,intensitylabel=$i_5$](6.5,6)
    (6.5,7.5)
20 \wire(2.5,7.5)(2.5,3)
21 \wire[intensity,intensitylabel=$i_c$](2.5,4.5)
    (2.5,6)
22 \qdisk(2.5,6){2pt}\qdisk(6.5,6){2pt}
23 \diode[dipolestyle=thyristor](2.5,4.5)(4.5,4.5)
    {$T_2$}
24 \wire[intensity,intensitylabel=$i_2$](4.5,4.5)
    (6.5,4.5)
25 \capacitor[tension,tensionlabel=$u_c$,%
26 tensionoffset=-0.75,tensionlabeloffset
    =-1](6.5,4.5)(6.5,6){$C_k$}
27 \qdisk(2.5,4.5){2pt}\qdisk(6.5,4.5){2pt}
28 \wire[intensity,intensitylabel=$i_3$](6.5,4.5)
    (6.5,3)
29 \multidipole(6.5,3)(2.5,3)%
30 \coil[dipolestyle=rectangle,labeloffset
    =-0.75]{$L_3$}%
31 \diode[labeloffset=-0.75]{$D_3$}.
32 \wire(6.5,6)(9,6)\qdisk(9,6){2pt}
33 \diode(9,0)(9,6){$D_4$}

```

The package **pst-circ** is a great help with switching circuit diagrams of reasonable complexity, in the same way as noted earlier with equivalent circuits.

5 PDF output

Since the package **pst-circ** [1] is based on PostScript [3] (like all **pstricks** [8] packages), it cannot be used directly with pdfTEX. However, there are a number of alternative ways to get PDF output:

- The **pdftricks** package [7]; unfortunately, this very frequently, because of the use of PostScript, leads to problems with bounding boxes.
- The **ps4pdf** package [5, 6], which requires installation of the L^AT_EX package **preview** [2].
- The program VTEX[4] (free for Linux and OS/2).
- The program **ps2pdf** with the sequential conversions dvi → ps → pdf.

ps4pdf also offers the possibility of storing figures generated with **pstricks** in a single PDF or EPS file.

6 Summary

This article describes most of the common features of the

The package ***pst-circ*** is a great help with switching circuit diagrams of reasonable complexity, in the same way as noted earlier with equivalent circuits.

5 PDF output

Since the package `pst-circ` [1] is based on PostScript [3] (like all `pstricks` [8] packages), it cannot be used directly with pd_TE_X. However, there are a number of alternative ways to get PDF output:

- The `pdftricks` package [7]; unfortunately, this very frequently, because of the use of PostScript, leads to problems with bounding boxes.
 - The `ps4pdf` package [5, 6], which requires installation of the L^AT_EX package `preview` [2].
 - The program V^TE_X[4] (free for Linux and OS/2).
 - The program `ps2pdf` with the sequential conversions `dvi → ps → pdf`.

`ps4pdf` also offers the possibility of storing figures generated with `pstricks` in a single PDF or EPS file.

6 Summary

This article describes most of the common features of the `pst-circ` package. Features not mentioned, for example

crossing of connections and use of switches, can be found in the package documentation.

References

- [1] Christophe Jorssen and Herbert Voß. *pst-circ: PostScript macros for drawing electronic circuits*. <http://ctan.tug.org/tex-archive/graphics/pstricks/contrib/pst-circ/>, 2003.
- [2] David Kastrup. *preview-latex*. <http://ctan.tug.org/tex-archive/support/preview-latex/>, 2003.
- [3] Nikolai G. Kollock. *PostScript richtig eingesetzt: vom Konzept zum praktischen Einsatz*. IWT, Vaterstetten, 1989.
- [4] Micropress. *VTEX/Lnx*. <http://www.micropress-inc.com/linux/>, 2003.
- [5] Rolf Niepraschk. *ps4pdf*. <http://ctan.tug.org/tex-archive/macros/latex/contrib/ps4pdf/>, 2003.
- [6] Rolf Niepraschk and Herbert Voß. The package *ps4pdf*: from PostScript to PDF. *TUGboat*, 22:290–292, December 2001.
- [7] Herbert Voß. *PSTricks Support for PDF*. <http://www.pstricks.de/pdf/pdftricks.phtml>, 2002.
- [8] Timothy Van Zandt. *pstricks: PostScript macros for generic T_EX*. <http://www.tug.org/applications/PSTricks/>, 1993.

◊ Herbert Voß
 Wasgenstr. 21
 14129 Berlin
 Germany
 voss@perce.de
<http://www.perce.de>

Absolute positioning with **textpos**

Norman Gray

Abstract

I describe the **textpos** package, which allows you to place blocks of text at arbitrary positions on the page. I give an overview of its functionality, and discuss a few points of T_EXnical interest.

1 Introduction

T_EX has many wonderful features, but the one most often, and most forcefully, advanced to potential converts is that they need no longer fret about layout. ‘Concentrate on the text; the layout is not your concern’, we say, ‘Produce your deathless text in a golden stream and let L^AT_EX handle the minutiae of fonts and linebreaking and spacing’.

This is true, but in those few cases where the layout actually is part of the point—this block of text must go *here*, the logo *just there*—authors find themselves embroiled in an unseemly struggle with an application which suddenly seems officious rather than helpful, resorting to increasingly shrill demands (`\newline` dammit!), and more or less desperate hackery. This is where **textpos** can help.

In 1998 I was involved in just such a struggle with L^AT_EX, laying out text on an A0 sheet to produce a conference poster. The package I produced I released as **textpos**, and it has become a common way to achieve this sort of positioning control. The package is available from CTAN, at *macros/latex/contrib/textpos*, and also from the **textpos** home page, <http://www.astro.gla.ac.uk/users/norman/distrib/latex/>.

In this article I’ll give a quick overview of the functionality of **textpos**. Then I’ll make a few observations on the way that **textpos** is implemented, and on its relationship with the **everyshi** package.

2 Using **textpos**

The **textpos** package is not a complicated one, since in outline it consists of only a single environment, plus a starred variant and a few configuration parameters. The manual distributed with the package [1] gives the details; I need only outline the principles here.

The package defines an environment **textblock** which contains the text (or other material) which is to be placed in a block, and takes parameters which specify the width of the block and its position relative to a reference point. The syntax is as follows:

```
\begin{textblock}{\width}{[\hx,\hy]}(\x,\y)
...
\end{textblock}
```

The `\width` gives the width the block is to have, and the `(x)` and `(y)` parameters give the position of the block’s ‘handle’ relative to the ‘reference point’. The handle is by default the top-left corner of the block, but may be moved using the optional argument (in square brackets as usual); the reference point I will return to in a moment. Notice that the positioning arguments for the **textblock** environment—the coordinates `(x), (y)`—are in parentheses rather than curly braces, in slight imitation of the **picture** environment.

The salient features of this syntax and its effects are illustrated in figure 1. In this illustration, the rules around the boxes are there because I included the `[showboxes]` option when I loaded the **textpos** package at the beginning of this article—this is

```
In my beginning
\begin{textblock}{2.5}(0.5,2)
\raggedright
Work is of two kinds: first, altering
the position of matter at or near the
earth's surface relatively to other such
matter; second, telling other people
to do so.
\end{textblock}
\begin{textblock}{2}[0.5,0.5](4,2)
\raggedleft
The first kind is unpleasant and ill paid;
the second is pleasant and highly paid
\emph{[Russell]}.
\end{textblock}
is my end.
```

In my beginning
is my end.

Work is of two kinds:
first, altering the position
of matter at or near the
earth's surface relatively
to other such matter;
second, telling other
people to do so.

The first kind is
unpleasant and ill
paid; the second is
pleasant and highly
paid [Russell].

Figure 1: An example of using the `textblock` environment.

useful when laying out the boxes, but it is off by default.

In a `textblock` environment, you specify the widths of the textblocks, but not their heights, which are as large as they have to be to enclose their content.

Here, I have shown the content as simple text, but the contents can be anything which can go into a `\vbox`.

The positioning parameters specify the position of a notional handle, which by default is located at the top left corner of the block. However, you can move this handle to any part of the block by using the arguments $[\langle hx \rangle, \langle hy \rangle]$, as I did in the second block in figure 1. The coordinates of the handle are given as multiples of the horizontal and (final) vertical size of the block, so that $[0,0]$ is the top left (the default), $[1,0]$ is the top right, and $[0.5,0.5]$ is the centre. The fractions don't have to be restricted to the range $0 \leq f \leq 1$.

Each of the environments takes up zero space, so that the ‘reference point’—the point relative

to which the boxes are positioned—is the same for each of the environments, as long as there is no material between them (or more precisely, no material with a vertical extent greater than 0 pt). This is why the text ‘is my end’ appears close to the text in the top left; observe, however, that they are not immediately adjacent, and the presence of the `textblock` environments has inserted a single paragraph break. If a `textblock` appears when TeX is typesetting the text of a paragraph (that is, it is in horizontal mode), then the environment ends the paragraph, as if you had typed `\par` at that point, or inserted a blank line. While I’m talking about spacing, note that there is nothing inhibiting you from (or defending you against!) overlapping the text of the boxes, so that there is necessarily an element of visual layout involved in using the environment.

2.1 Lengths

The widths and positions which are arguments to the `textblock` environment are given in units of `\TPHorizModule` for the horizontal lengths, and `\TPVertModule` for the vertical ones. These are TeX `<dimen>`s, so you can set their values in the usual way, with `\setlength`. You usually will want to do this, since the default values, of $1/16$ of the `\paperwidth` and `\paperheight` respectively, are not likely to be particularly useful. For example, the figures above were preceded by

```
\setlength{\TPHorizModule}{\columnwidth}
\divide\TPHorizModule by 5
\setlength{\TPVertModule}{\baselineskip}
```

to adapt the positioning to the typographical environment. Alternatively, you could use the `calc` package and write this more straightforwardly as

```
\setlength{\TPHorizModule}{\columnwidth/5}
```

`textpos` is compatible with `calc`, thanks to code from Rolf Niepraschk. You can also use these dimensions directly, in `\hspace{2 \TPHorizModule}` for example, if that helps to give your document more consistency.

Using these modules makes your arrangement of blocks easily rescalable, and it helps fit the blocks neatly into a larger structure; however, they can also help your layout look better. Although you can give fractional positions (as I illustrated in figure 1), your layout will tend to look more coherent if you pick a suitable module and try to restrict yourself to integer multiples of it. This can make the difference between a layout which looks busy and cluttered, and one which is elegantly restrained.

If your layout requirements are more specific than this, then you may want to use the starred variant of the `textblock` environment. This is just like the unstarred version, except that the block width and positioning parameters are given as absolute lengths, in any `TEX` units such as `pt` or `em`, rather than as multiples of the horizontal and vertical modules (the optional arguments remain relative to the size of the final block).

2.2 Absolute positioning

As I explained above, the position arguments are by default relative to a reference point which is identified as the ‘current position’ on the page, which is unchanged by the presence of the `textblock`. For some applications, such as laying out a conference poster, it is most useful if the reference point can be guaranteed to be in a particular location, and guaranteed not to move, and it is for this reason that `textpos` also has an ‘absolute’ mode.

You put `textpos` in this mode with

```
\usepackage[absolute]{textpos}
```

after which all `textblocks` are positioned relative to a single origin on the page, irrespective of any material that separates them. This origin is by default located at the top left corner of the paper (that is, 25.4 mm (ahem!) leftwards and upwards of `TEX`’s usual nominal reference point), but you can adjust it with the `\textblockorigin{<x>}{<y>}` command, which takes two arguments, giving the horizontal and vertical position of the origin, relative to the top left corner of the paper. The dimensions `<x>` and `<y>` must have units; they are not multiples of any module.

The command

```
\TPGrid[<bh>,<bv>]{<nh>}{<nv>}
```

is an alternative way of setting the `\TPHorizModule` and `\TPVertModule` lengths, particularly useful in absolute mode. This firstly sets the modules so that

$$\langle nh \rangle \times \text{\TPHorizModule} + 2\langle bh \rangle = \text{\paperwidth}$$

$$\langle nv \rangle \times \text{\TPVertModule} + 2\langle bv \rangle = \text{\paperheight}$$

and secondly calls `\textblockorigin{<bh>}{<bv>}`, so that the modules form a $\langle nh \rangle \times \langle nv \rangle$ grid on the paper, with a border `<bh>` wide and `<bv>` deep around it. If the optional border argument is absent, it defaults to `[0pt,0pt]`. The `\textblockorigin` command is available only in ‘absolute’ mode, but the `\TPGrid` command is available in relative mode also.

For other hints on formatting posters, see [2].

There are a few other `textpos` details, to do with colouring in boxes, and overlaying them or not; for those, see the `textpos` documentation.

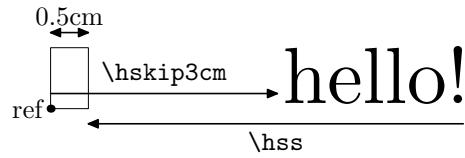


Figure 2: Positioning text in and out of boxes.

3 Implementation

The `textpos` package is not a complicated one, and is at heart a wrapper round two simple but powerful techniques, namely positioning with glue and boxes, and hooking into the output routine using `everyshi`. I will describe these here, in case they are of wider `TEX`nical interest.

3.1 The basics of positioning

At one level, a `TEX` page is no more than a sequence of boxes, glue, and a few more exotic things that need not concern us here. Each box has a height, a depth, a width, and at this level nothing else. The material inside these boxes, consisting of characters, rules, and other boxes, is what we ultimately wish to see on the page, but there is no requirement for this material to lie strictly within the boundaries of the box it is associated with; furthermore, glue can be negative in extent as well as positive, and these two observations together give us a technique for positioning things.

Consider the `TEX` box

```
\hbox to 0.5cm{\hskip 3cm hello!\hss}
```

which is illustrated in figure 2. That creates an `\hbox` which is exactly 0.5 cm wide, and puts into it a skip and some text which are together substantially larger than the box. This would create an overfull `\hbox` were it not for the ‘infinitely shrinkable glue’, the `\hss`, at the end. This inserts whatever glue is required to make the whole construction have zero badness. The end result is that we have placed the text ‘hello!’ at a point 3 cm to the right of the reference point of the `\hbox` which, as far as `TEX` is concerned, is only 0.5 cm wide. We can do exactly the same thing with `\vbox`s and `\vss` glue and, putting these together, get the result in figure 3.

That — plus a little bit of syntactic sugar¹ and some spacing magic — is `textpos`.

3.2 Absolute positioning and `\shipout`

Well, *almost* all there is to it. While the technique in the last section is enough for the default ‘relative’

¹ The fact that `textpos.sty` has ended up over 250 lines long, shows that sugar can be very fattening indeed.

```
I am here
\vbox to 0pt{%
 \vskip 1cm
 \hbox to 0pt{\hskip 2cm In my beginning.\hss}%
 \vss}Or there, or elsewhere.
```

I am here Or there, or elsewhere.

In my beginning.

Figure 3: Boxes in boxes: how `textpos` works.

mode of `textpos`, it does not address the problem of pinning the reference point for absolute mode to a fixed position on the page. The most elegant way to do this was suggested by Olaf Maibaum, using Martin Schröder's `everyshi` package (also at CTAN, of course).

The 'shipout' is the very final stage of `TeX`'s handling of a page. When it has found an optimal page break, `TeX` puts the page contents into the special box register `\box255` and calls the `\output` routine. That routine's job is to do the final assembly of a page, handling footnotes, marginal notes, and the rest, and when it is finished, it wraps it all up into a final box which it passes to the `TeX` primitive `\shipout`. The `everyshi` package gives you a last chance to tinker with the output, by letting you register a sequence of commands which will be invoked at precisely this point, with the contents of `\shipout`'s argument available in `\box255` (though this is almost certainly not the same `\box255` which was originally prepared for the `\output` routine). It is the content of this box *after* you've made any adjustments that is passed to the primitive `\shipout`. This is a tremendously powerful mechanism.

In absolute mode, `textpos` converts each text block into a zero-height `\vbox` as usual (in fact, into the temporary `\box0`), but instead of contributing them to the current page, it accumulates them in a holding box:

```
\global\setbox\TP@holdbox=\vbox{%
 \box0
 \unvbox\TP@holdbox}
```

However, in this mode `textpos` has also registered some commands with the `everyshi` hook:

```
\EveryShipout{%
 \global\setbox255=\vbox{%
 \unvbox\TP@holdbox
 \unvbox255}}
```

Thus, whenever the output routine is called, either because a page has filled up or because the input file has come to an end, it constructs its final box and

calls `\shipout`. At this last moment this fragment of code prepends the `textpos` hold box to the everyshi `\box255` and lets this enlarged box be the one shipped out.

3.3 So ...

`textpos` has pulled off the rather un-`TeX`-like trick of supporting the *non*-automatic layout of text, and it has done so without outrageous trickery, by simply exploiting the core functionality of `TeX`'s page-layout algorithm—constrained gluing together of boxes of given sizes—in an unexpected way. As a result, understanding `textpos` requires, or prompts, an understanding of that algorithm, with its parameters `\baselineskip`, `\prevdepth` and friends, and this, together with the use of other common techniques such as `TeX` arithmetic and token lists, means that it manages simultaneously to solve a non-trivial problem usefully, and to be instructive.

And that's a good position to be in.

References

- [1] Norman Gray. *textpos User Manual, version 1.4*, 2003. Distributed with the `textpos` package.
- [2] Norman Gray. Using L^AT_EX to produce conference posters, 2003. <http://www.astro.gla.ac.uk/users/norman/docs/posters/>.

◊ Norman Gray
Department of Physics and
Astronomy
University of Glasgow
Glasgow, UK
norman@astro.gla.ac.uk
<http://www.astro.gla.ac.uk/users/norman/>

Multilingual bibliographies: Using and extending the `babelbib` package

Harald Harders

1 Introduction

When generating bibliographies using BIB^LA_TE_X, style files (file extension `.bst`) are used to determine the appearance of the bibliographies. Most of the available BIB^LA_TE_X styles are hardcoded to a specific language, often English. This is unsatisfactory in many cases. If you, for example, write a German document and use one of the standard `bst` files, English keywords as "edition", "page", etc. are used instead of their German translations »Auflage«, »Seite« etc.

Another limitation of most BIBTEX styles is the fact that even small changes of the bibliography's layout are not possible without creating a new `bst` file. Since the syntax of `bst` files is completely different from that of LATEX this is difficult for most LATEX users.

The package `babelbib` provides solutions for both problems. It is available from the CTAN network in the directory `CTAN:biblio/bibtex/contrib/babelbib/`.

2 Multilingual bibliographies

2.1 Available packages

The restriction to one bibliography language is avoided by the packages `bibgerm` and `babelbib`. Both use following approach: Their BIBTEX styles use T_EX macros instead of hardcoded strings, e.g., the command `\btxeditorlong` instead of the string "editor". These commands are defined within the packages for different languages differently, e.g., "editor" in English, »Herausgeber« in German, or «editore» in Italian.

`bibgerm` [4] is the older package and has served as basis for `babelbib`. It is restricted to English and German and works together with the `babel` [1], `german`, and `ngerman` [3] packages. `bibgerm` works both with plain T_EX and LATEX. It does not provide commands to change the typography of bibliographies.

`babelbib` [2] has been developed in order to be extendable to more languages in cooperation with the `babel` package. Thus, it needs the `babel` package to be loaded, too. Version 0.40 of the package `babelbib` supports Afrikaans, Danish, Dutch, English, French, German, Italian, Portuguese, Swedish, and Spanish.¹ The author would be grateful for any offers of assistance with adding more languages. How this can be done is described in section 4. The current version only runs with LATEX 2_E. `babelbib` provides commands to change the typography of a bibliography within the LATEX source, described in section 3.2.

With `babelbib`, replacements for the standard BIBTEX styles (`bababbrv`, `babalpha`, `babplain`, and `babunsrt`) as well as a replacement for the *AMS* BIBTEX style `amsplain` (`babamspl`) are shipped. In addition, two styles `bababbr3` and `babplai3` are included that convert a list of more than three authors to "first author *et al.*". All styles have multilingual support, include additional field types (de-

scribed in section 3.1), and allow easy layout changes (section 3.2).

2.2 Different approaches for multilingual bibliographies

Two approaches are possible for bibliographies with flexible languages:

Each citation can use the language of the cited document. Then, the keywords vary within one bibliography. This approach is used by `bibgerm`. The following example shows this behaviour:

References

- [1] Beitz, W. und K.-H. Küttner (Herausgeber): *Dubbel – Taschenbuch für den Maschinenbau*, Kapitel Werkstofftechnik, Seiten E 1–E 120. Springer-Verlag, Berlin, 17. Auflage, 1990, ISBN 3-540-52381-2.
- [2] Dieter, George E. *et al.* (editors): *Materials Selection and Design*, volume 20 of *ASM Handbook*, chapter Effects of Composition, Processing, and Structure on Properties of Engineering Plastics, pages 434–456. ASM International, 1997, ISBN 0-87170-386-6.

The second approach uses the document's main language for the whole bibliography. Thus, the keywords are uniform. This means for example, that the "edition" of a cited document in a German text is named »Auflage«, even if the cited document is not German. Nevertheless, the data fields (title, authors, etc.) are typeset in the citation language given for the cited document in order to use the correct hyphenation patterns. For example, the above bibliography looks in an English text like this:

References

- [1] Beitz, W. and K.-H. Küttner (editors): *Dubbel – Taschenbuch für den Maschinenbau*, chapter Werkstofftechnik, pages E 1–E 120. Springer-Verlag, Berlin, 17. edition, 1990, ISBN 3-540-52381-2.
- [2] Dieter, George E. *et al.* (editors): *Materials Selection and Design*, volume 20 of *ASM Handbook*, chapter Effects of Composition, Processing, and Structure on Properties of Engineering Plastics, pages 434–456. ASM International, 1997, ISBN 0-87170-386-6.

And in a German text like this:

¹ In some languages, all names for other languages are not yet present. For instance, the French name for new-norwegian (Nynorsk) is not defined.

Literatur

- [1] Beitz, W. und K.-H. Küttner (Herausgeber): *Dubbel – Taschenbuch für den Maschinenbau*, Kapitel Werkstofftechnik, Seiten E 1–E 120. Springer-Verlag, Berlin, 17. Auflage, 1990, ISBN 3-540-52381-2.
- [2] Dieter, George E. et al. (Herausgeber): *Materials Selection and Design*, Band 20 der Reihe *ASM Handbook*, Kapitel Effects of Composition, Processing, and Structure on Properties of Engineering Plastics, Seiten 434–456. ASM International, 1997, ISBN 0-87170-386-6.

Both approaches can be typeset with `babelbib`, described in the following sections.

2.3 Using the `babelbib` package

To use the features of `babelbib`, first load it with:

```
\usepackage{babelbib}
```

The default behaviour is to change the language settings depending on the cited document (first approach in section 2.2). If you want a unique language over the whole bibliography, use the option `fixlanguage`:

```
\usepackage[fixlanguage]{babelbib}
```

Then, the language is fixed to the main document language. If you want to use another language, you may change it by

```
\selectbiblanguage{\{language\}}
```

The following BIBTeX styles are available for `babelbib`: `bababbr3`, `bababrv`, `babalpha`, `babamspl`, `babplai3`, `babplain`, and `babunsrt`. You select one of them by using:

```
\bibliographystyle{\{style\}}
```

If you use `babamspl`, you have to load the package with the option `languagenames` because the *AMS* BIBTeX styles typeset the language of the cited document and `babelbib` then has to define the names of the languages. This is not done by default to save memory.

You can also use the BIBTeX styles of the package `bibgerm` (`gerabbrv`, `geralpha`, `gerapali`, `gerplain`, `gerunsrt`) but with fixed typography.

2.4 BIBTeX database files (*.bib)

The BIBTeX database files (extension `.bib`) for usage with `babelbib` don't differ much from standard files. All document types have the additional field `language` which should be given for *each* cited document. The examples, given above, were generated using following `bib` file:

```
@InBook{dubbel1990a,
  editor = {Beitz, W. and K."uttner, K.-H.},
  title = {Dubbel--- Taschenbuch f"ur den
           Maschinenbau},
  chapter = {Werkstofftechnik},
  publisher = {Springer"=Verlag},
  year = 1990,
  address = {Berlin},
  edition = {17.},
  pages = {E~1--E~120},
  isbn = {3-540-52381-2},
  language = {ngerman}
}

@InBook{dieter1997a,
  editor = {Dieter, George~E. and others},
  title = {Materials Selection and Design},
  chapter = {Effects of Composition,
             Processing, and Structure on
             Properties of Engineering
             Plastics},
  publisher = {\acro{ASM} International},
  year = 1997,
  volume = 20,
  series = {\acro{ASM} Handbook},
  pages = {434--456},
  isbn = {0-87170-386-6},
  language = {english}
}
```

Note that all extensions and shortcuts provided by `babel` for the different languages, e.g., "u instead of \"u for "ü" in German items, can be used in the fields of a document, if { and } are used as delimiters in the `bib` file.

If you leave out the specification of the language for a citation item this item is typeset in the document's main language, when you use one of the `bab*.bst` styles. In addition, a warning is generated for an omitted `language` field. If you use a `bibgerm` style (`ger*.bst`), no warning is produced, and—what is more important—the new item is typeset in the language of the preceding citation in the bibliography which may not be wanted.

3 Other extensions

The `babelbib` package as well as the associated `bst` files contain additional extensions that don't concern the multilingual support.

3.1 More data fields supported by the bst files

The `bab*.bst` styles support three additional fields for most of the document types.

You are able to specify the ISBN resp. ISSN of the documents, using fields of the same names, as can be seen in the example above.

Element	Fields	Default value	
	<code>bababbr3, bababbrv</code>	<code>babamspl</code>	
	<code>babalpha, babplai3</code>		
	<code>babplain, babunsrt</code>		
name	authors, editors		
title	title, series	<code>\emph</code>	<code>\emph</code>
etal	“ <i>et al.</i> ”	<code>\emph</code>	
journal	journal title		
volume	volume (journal)		<code>\textbf</code>
ISBN	ISBN	<code>\MakeUppercase</code>	<code>\MakeUppercase</code>
ISSN	ISSN	<code>\MakeUppercase</code>	<code>\MakeUppercase</code>
url	net address	<code>\url</code>	<code>\url</code>

Table 1: Default values of the fonts in bibliographies. A missing value means that the standard font of the document is used.

Using the field `url`, URLs can be given. If the command `\url` is available, e.g., by loading `url.sty` or `hyperref.sty`, they are printed using it. If not, `babelbib` defines a `\url` command that produces an error message when using the field `URL`.

3.2 Flexible typography of the bibliography

The standard `.bst` files have a fixed typography of the bibliography. Even small changes (e.g., changing the font for author names to small caps) need to change the `.bst` file. To avoid that, the `bab*.bst` files use user definable font commands for some elements of the bibliography.

The approach works as follows: If the user does not specify the font for an element of the bibliography, the `BIBTEX` style includes a default font that it uses. If, in contrast, the user specifies a font, this value is taken instead of the default.

Fonts of the bibliography are changed with the command:

```
\setbibliographyfont{<element>}{{<font command>}}
```

The possible elements and font commands are listed in Table 1.

The font command has to be a `LATEX` command with exactly one argument, e.g., `\emph`, `\textbf`, and `\textsc`. You can also use commands as `\mbox`, that do not change a font but, for example, inhibit line breaks within one element. This can be useful for ISBN and ISSN.

If you want to switch to a font for which no command is available that accords to the above rule, you have to define a new command, e.g.,

```
\newcommand{\textitbf}[1]{%
  {\bfseries\itshape #1\%}}
\setbibliographyfont{title}{\textitbf}%
```

In the argument of `\setbibliographyfont`, the font command is given without an argument, as shown in this example.

If you try to define a font for an element not listed in Table 1, an error message is generated. To define a font for a new element that is not known by `babelbib`, e.g., for a custom `BIBTEX` style, the `\setbibliographyfont*` command is available.

Internally, the `\setbibliographyfont` and `\setbibliographyfont*` commands define a command with a name built by `\btx{element}font`, e.g., `\btxtitlefont` for titles. This command can be used by the `.bst` files.

The strings “ISBN” and “ISSN” are generated by the commands `\btxISBN` and `\btxISSN`. They don’t take an argument. By default, these commands just write the corresponding strings without a change of the font. In this article, they have been changed as follows:

```
\renewcommand{\btxISBN}{\text{\small \acro{ISBN}}}
\renewcommand{\btxISSN}{\text{\small \acro{ISSN}}}
```

where `\acro` prints the text slightly smaller. Another possibility could be to use small caps:

```
\renewcommand{\btxISBN}{\textsc{isbn}}
\renewcommand{\btxISSN}{\textsc{issn}}
```

3.3 Changing keywords

If you don’t like some of the keywords provided by `babelbib` you are able to change them using `\declarebtxcommands`. For example, it is possible to call Ph.D. theses »Dissertation« or »Doktorarbeit« in German, where the first name is used by `babelbib`, as follows:

```
\declarebtxcommands{german}{%
  \def\btxphdthesis{\%}
  \foreignlanguage{german}{Doktorarbeit}\%}
}
```

As it can be seen in the example, the command changes the keyword for the language specified in the first argument, while the second argument gives the (re)definition of the command. You may change more than one command within one call of `\declarebtxcommands`, but you have to avoid the insertion of unwanted spaces. Which `\btx...` command you have to change can be determined by searching for the unwanted keyword in the language-dependent `bdf` file (see the next section).

`\declarebtxcommands` can also be used to add new keyword commands, e.g., for newly developed BIBTEX styles.

4 Adding new languages to `babelbib`

The package `babelbib` includes a list of known languages. It determines automatically which of these have been loaded by `babel`. It then defines the bibliographic keywords for them. This is done by loading special files (extension `.bdf`) that provide the keyword definitions, similarly to the language definition files (extension `.ldf`) of `babel`.

If the user defines a new `bdf` file the package `babelbib` does not know about it and thus cannot load it automatically. Thus, the user has to specify it as option when loading `babelbib`. For instance, say you have provided `norsk.bdf`. Then you have to load `babelbib` as follows:

```
\usepackage[norsk]{babelbib}
```

If you have generated a new `bdf` file or if you have extended one of the other files, please email them to me so I can include your changes into the distribution.

4.1 Writing new bdf files

The `bdf` files do two things. First, they provide the commands that contain the keywords for bibliographies. And second, they append the call of these commands to the `\extras<language>` command of the loaded languages, if the option `fixlanguage` is not used. In the further text, the organization of `bdf` files is described for the example of `english.bdf`.

The commands for the bibliographic keywords are called `\btx<keyword>` for keywords in the middle of a sentence (often starting with lowercase letters) resp. `\Btx<keyword>` for keywords at the beginning of a sentence (starting with uppercase letters). Many of these commands provide a long and a short (abbreviated) version, for which `long` resp. `short` is appended to the command name, e.g., `\btxeditorlong` for “editor” and `\btxeditorshort` for “ed”.

The keyword definitions are put into a command `\bibs<language>`, e.g., `\bibsenglish`, which is called when the document language is changed by `\selectlanguage`, if `fixlanguage` is not set, or at `\begin{document}`, if `fixlanguage` is set.

A part of the command `\bibsenglish` looks like this:

```
\newcommand\bibsenglish[1][english]{%
  \def\biblanguagename{\#1}%
  \def\btxtetalshort##1{%
    \foreignlanguage{\#1}{et~al##1{}{}}%
  }%
  \def\btxeditorshort##1{%
    \foreignlanguage{\#1}{ed##1{}{}}%
  }%
  \def\btxeditorlong##1{%
    \foreignlanguage{\#1}{editor}%
  }%
  \def\btxeditorsshort##1{%
    \foreignlanguage{\#1}{eds##1{}{}}%
  }%
  \def\btxeditorslong##1{%
    \foreignlanguage{\#1}{editors}%
  }%
  \def\Btxeditorshort##1{%
    \foreignlanguage{\#1}{Ed##1{}{}}%
  }%
  \def\Btxeditorlong##1{%
    \foreignlanguage{\#1}{Editor}%
  }%
  \def\ifbbbbblanguagenames{%
    \def\btxlanguagenameamerican{%
      \foreignlanguage{\#1}{english}%
    }%
    \def\btxlanguagenameaustrian{%
      \foreignlanguage{\#1}{german}%
    }%
  }%
  \def\btxlanguagenamefrenchb{%
    \foreignlanguage{\#1}{french}%
  }%
  \def\btxlanguagenamegerman{%
    \foreignlanguage{\#1}{german}%
  }%
  \def\btxlanguagenameUKenglish{%
    \foreignlanguage{\#1}{english}%
  }%
  \def\btxlanguagenameUSenglish{%
    \foreignlanguage{\#1}{english}%
  }%
}
\fi
}
```

The `\btxlanguagename...` commands typeset different language names in the keyword language of citations. This is necessary if the BIBTEX style writes the language of the citation into the bibliography, as `babamspl` does. In order to save memory, the language names are only defined if the option `languages` is set when loading the `babelbib` package.

The `\bibsenglish` command takes one optional argument which specifies the language of the keywords. By default, it is `english`. This optional

argument is useful for defining English dialects that mostly use the same keywords. For example, American is defined like this:

```
\newcommand{\bibsamerican}{%
  \bibsenglish[american]}
```

\bibsamerican simply calls \bibsenglish with the keyword language changed to american. If, for example, there was an English dialect “myeng” where a Master’s thesis was called “Diploma thesis”, the definition could look like this:

```
\newcommand{\bibsmyeng}{%
  \bibsenglish[myeng]%
  \def\btxmastthesis##1{%
    \foreignlanguage{myeng}{Diploma thesis}}%
}
```

This first would set \btxmastthesis to “Master’s thesis” and then redefine it to “Diploma thesis”. This approach wastes some time, but it avoids repeating identical entries in the source code.

All commands defined by \bibsenglish take one argument ##1, whose content is appended to the keyword text in some cases. This can be used by the bst files to append the dot for abbreviations. For uniformity, all commands take this argument even if they don’t need it.² All \btx... and \Btx... commands switch to the keyword language using \foreignlanguage and typeset the keyword as specified. Thus, the keywords are hyphenated correctly.

The second part of the bdf file appends the macro \bibs⟨language⟩ to the \extras⟨language⟩ command for all languages that are loaded by babel, if fixlanguage is not used. This is done by the command \bbbbaddto{⟨language⟩} which is called at \begin{document} for all dialects defined in the bdf file (which are American, British, Canadian, English, UK English, and US English³ for english.bdf):

```
\AtBeginDocument{%
  \ifbbbbfixlanguage
  \else
    \bbbbaddto{american}{bibsamerican}
    \bbbbaddto{british}{bibsbritish}
    \bbbbaddto{canadian}{bibscanadian}
    \bbbbaddto{english}{bibsenglish}
    \bbbbaddto{UKenglish}{bibsUKenglish}
    \bbbbaddto{USenglish}{bibsUSenglish}
  \fi
  \bbbbaddto{american}{btxifchangecaseon}}
```

² This is due to the fact that bibgerm does it this way. The two packages are intended to stay compatible to some extent.

³ For some dialects, different names are available (e.g., American and US English), since babel also supports different names for some dialects.

```
\bbbbaddto{british}{btxifchangecaseon}
\bbbbaddto{canadian}{btxifchangecaseon}
\bbbbaddto{english}{btxifchangecaseon}
\bbbbaddto{UKenglish}{btxifchangecaseon}
\bbbbaddto{USenglish}{btxifchangecaseon}
}
```

The switch \ifbbbbfixlanguage ensures that this is only done if fixlanguage is not set.

The second part of this code snippet, after the \fi, is necessary, because the case of titles is changed in some languages and preserved in others. For example, in English, titles are printed lowercase, while in German, titles are printed as given. This is achieved via the following approach: The BIBTEX style prints the title twice as arguments of the \btxifcasechange commands. The first one is lowercase, the second with preserved case. The LATEX code then decides based on the language which version is typeset. There are two commands \btxifchangecaseon and \btxifchangecaseoff that switch between the two behaviours. Since in all English dialects the case of titles is changed, \btxifchangecaseon is appended to \extras⟨language⟩. If a language does not change the case, you have to append \btxifchangecaseoff instead.

Finally, if you want to create a bdf file for a new language, you should copy an existing one to a new file and then change it. To test the new language, babelbib.sty does not have to be changed; instead, specify the name of the new bdf file without extension as an option to the \usepackage[⟨filename⟩]{babelbib} command.

4.2 Extending the babelbib package

The package file babelbib.sty provides the common commands for all languages and loads the necessary bdf files. Therefore, it contains a list of all known languages and dialects. babelbib version 0.40 knows about the following languages and dialects: afrikaans, american, austrian, brazil, brazilian, british, canadian, canadien, danish, dutch, english, franceis, french, frenchb, german, germanb, italian, mexican, naustrian, ngerman, portuges, portuguese, UKenglish, USenglish, spanish, and swedish.

The language definitions are loaded by the command

```
\inputbdf{⟨language⟩}{⟨filename⟩},
```

where ⟨language⟩ is the dialect and ⟨filename⟩ is the name of the bdf file without the extension. If you add a new language, just add a new line containing

an `\inputbdf` command to the list of `\inputbdf` commands.⁴

5 Adapting other BIBTEX styles to babelbib

Using `amsplain bst` as an example, we show how other BIBTEX styles can be adapted to `babelbib`. (The resulting `bst` file is included in the `babelbib` distribution as `babamspl bst`.)

In the source code snippets, newly inserted, changed, and important lines are marked by “←”.

5.1 Multilingual support

The *AMS* BIBTEX styles are different from the standard styles in one aspect: They print the language of the citation for some document types. Thus, they already have the BIBTEX field `language`. This can be seen in the list of supported fields:

```
ENTRY
  address
  :
  key
  language ←
  month
  :
  year
}
{
{ label bysame }
```

If the field `language` is missing from a particular BIBTEX style, it must be inserted.

As described in section 4.1, `bst` files print titles twice—with changed case and with preserved case—in order to enable the LATEX code to decide which version will be typeset, using the macro `\btxifcasechange`. This is done by the function `language.change.case`:

```
FUNCTION {language.change.case} ←
{
  'change.temp := ←
  't := ←
  "\btxifchangecase{" ←
  t change.temp change.case$ * ←
  "}" * ←
  t * ←
  "}" * ←
}
```

In order for this to work, the string variables have to be defined beforehand. Therefore, the line `STRINGS { s t }` in the original is changed to

```
STRINGS { s t language.state ←
          change.temp } ←
```

At the beginning of each citation, the language has to be switched to the citation language, if it is different from the preceding citation. Therefore, some code is integrated into the function `output.bibitem`:

```
FUNCTION {output.bibitem}
{ newline$ ←
  language empty$ ←
  { "empty language in " cite$ * warning$ ←
    language.state "nolanguage" = ←
    'skip$ ←
    { ←
      "\expandafter\btxselectlanguage" ←
      "\expandafter{" * ←
      "\btxfallbacklanguage}" * write$ ←
      newline$ ←
    }
  }
  if$ ←
  "nolanguage" 'language.state := ←
}
{ language.state language = ←
  'skip$ ←
  { "\btxselectlanguage{" ←
    language * "}" *
    write$ newline$ ←
  }
  if$ ←
  language 'language.state := ←
}
if$ ←
"\bibitem{" write$ ←
cite$ write$ ←
"}" write$ ←
newline$ ←
"
before.all 'output.state := ←
}
```

This function also generates a warning if the language is omitted. In addition, the language is changed to a fall-back language which is the document’s main language.

Since this BIBTEX style prints the citation language, a function `format.language` is defined that typesets the language name in brackets. Many styles don’t need this function. Since it would not be good if non-English texts used the English names of languages, e.g., “german” and “french” instead of »deutsch« and »französisch« in a German text, LATEX macros are used instead of the language names of `babel`. These macros print the language name in the correct language. Therefore, the function `format.language` is used:

```
FUNCTION {format.language}
{ language empty$ ←
  { "" } ←
  { " (\btxlanguagename{" ←
```

⁴ If you do this, you have to rename your style file.

```

language * "})" * }
if$ ←
}

```

The command `\btxlanguagename` prints the language name using the keyword `language` of the citation. This only works if the option `languagenames` is used when loading `babelbib`:

```
\usepackage[languagenames]{babelbib}
```

If a language name is not available, an error message is generated and the name in the source code is used instead.

Since the change of case is only used in some languages, the call of `change.case$` has to be replaced by `language.change.case`, e.g.,

```

FUNCTION {format.title}
{ title empty$ ←
  { "" } ←
  { title "t" language.change.case ←
    emphasize } ←
  if$ ←
}

```

This has to be done for all occurrences.

5.2 Flexible typography

In this section, we describe how the typography of BIBTeX styles is made flexible.

Some functions are defined to allow switching fonts easily. They are similar to the existing `emphasize` function:

```

FUNCTION {emphasize}
{ duplicate$ empty$ ←
  { pop$ "" } ←
  { "\emph{" swap$ * "}" * } ←
  if$ ←
}

```

```

FUNCTION {namefont}
{ duplicate$ empty$ ←
  { pop$ "" } ←
  { "\btxnamefont{" swap$ * "}" * } ←
  if$ ←
}

```

```

FUNCTION {titlefont}
{ duplicate$ empty$ ←
  { pop$ "" } ←
  { "\btxtitlefont{" swap$ * "}" * } ←
  if$ ←
}

```

```

FUNCTION {journalfont}
{ duplicate$ empty$ ←
  { pop$ "" } ←
  { "\btxjournalfont{" swap$ * "}" * } ←
  if$ ←
}

```

```

FUNCTION {volumefont} ←
{ duplicate$ empty$ ←
  { pop$ "" } ←
  { "\btxvolumefont{" swap$ * "}" * } ←
  if$ ←
}

```



```

FUNCTION {etalfont} ←
{ duplicate$ empty$ ←
  { pop$ "" } ←
  { "\btxetalfont{" swap$ * "}" * } ←
  if$ ←
}

```

More font-switching commands can be defined analogously. Since `babelbib` would not know about such, however, you must call `\setbibliographyfont*` instead of `\setbibliographyfont` in the L^AT_EX file, to make use of them. Also, the `bst` file has to use `\providebibliographyfont*` instead of `\providebibliographyfont`, as described below. (Please tell me if you add a new font command, so I can add it to the package.)

The font functions are called in later functions in the `bst` file. For example, names (authors, editors) are typeset by `format.names`, which is defined as follows:

```

FUNCTION {format.names}
{ 's := ←
  #1 'nameptr := ←
  s num.names$ 'numnames := ←
  numnames 'namesleft := ←
  { namesleft #0 > } ←
  { s nameptr "{ff~}{vv~}{ll}{, jj}" ←
    format.name$ 't := ←
    nameptr #1 > ←
    { namesleft #1 > ←
      { " " * t namefont * } ←
      { numnames #2 > ←
        { "\btxandcomma{}" * } ←
        'skip$ ←
      if$ ←
      t "others" = ←
        { " " * "\btxetalshort{.}" ←
          etalfont * } ←
        { " \btxandlong{}" * } ←
        t namefont * } ←
      if$ ←
    }
    if$ ←
  }
  { t nameptr "{ff~}{vv~}{ll}{, jj}" ←
    format.name$ namefont } ←
  if$ ←
  nameptr #1 + 'nameptr := ←
  namesleft #1 - 'namesleft := ←
}

```

```

        }
    while$}

Additionally, the LATEX macros \btxandcomma, \btxetalshort, and \btxandlong have been added, to print language-dependent keywords.

For titles, emphasize is replaced by titlefont:
FUNCTION {format.title}
{ title empty$
  { "" }
  { title "t" language.change.case
    titlefont } ←
  if$ ←
}

Similarly the title and volume of journals:
FUNCTION {format.journal.vol.year}
{ journal empty$
  { "journal name" missing.warning ""}
  { journal journalfont } ←
  if$ ←
  volume empty$
  'skip$ ←
  { " " * volume volumefont * } ←
  if$ ←
  year empty$
  { "year" missing.warning }
  { " (" * year * ")" * }
  if$ ←
}

```

In format.incoll.inproc.crossref, the original use of \emph is replaced by titlefont; in format.article.crossref, journalfont is added.

As shown for format.names, the hardcoded keywords have to be replaced by the corresponding L^AT_EX macros. Table 2 shows important replacements. Sometimes, identical keywords have to be replaced by different macros, depending on the context.

If you want to use new keywords that are not included in the existing bdf files, you have to define them in the L^AT_EX document using \declarebtxcommands, as described in section 3.3.

The fonts used for the data fields have to be initialized at the beginning of the bibliography. This is done by the \providebibliographyfont command, that only does an initialization if the author has not done it before. Since the function begin.bib starts the bibliography, the initializations are added here (a few lines are omitted):

```

FUNCTION {begin.bib}
{ preamble$ empty$
  'skip$ ←
  { preamble$ write$ newline$ } ←
  if$ ←
  "\providecommand{\bysame}{\leavevmode\hbox "
}
```

Text	Macro
and	\btxandlong{}
ch.	\btxchaptershort{.}
ed.	\btxeditorshort{.}
ed.	\btxeditionshort{.}
eds.	\btxeditorsshort{.}
et al.	\btxetalshort{.}
in	\btxinlong{}
in	\btxinserieslong{}
Master's thesis	\btxmastthesis{}
no.	\btxnumbershort{.}
of	\btxofserieslong{.}
p.	\btxpageshort{.}
Ph.D. thesis	\btxphdthesis{}
pp.	\btxpageshorth{.}
Tech. Report	\Btxtechrepshort{.}
vol.	\btxvolumeshort{.}
January	\btxmonjanlong{}
February	\btxmonfeblong{}
:	:

Table 2: Replacements for BIBL^EX styles.

```

"to3em{\hrulefill}\thinspace}" *
write$ newline$ ←
:
"\providecommand{\href}2{#2}" ←
write$ newline$ ←
"\begin{thebibliography}" ←
  longest.label * "}" * ←
  write$ newline$ ←
"  \providebibliographyfont{name}{}%" ←
  write$ newline$ ←
"  \providebibliographyfont{title}%" ←
  "\emph}%" * ←
  write$ newline$ ←
"  \providebibliographyfont{journal}{}%" ←
  write$ newline$ ←
"  \providebibliographyfont{etal}{}%" ←
  write$ newline$ ←
"  \providebibliographyfont{volume}%" ←
  "\textbf}%" * ←
  write$ newline$ ←
"  \providebibliographyfont{ISBN}%" ←
  "\MakeUppercase}%" * ←
  write$ newline$ ←
"  \providebibliographyfont{ISSN}%" ←
  "\MakeUppercase}%" * ←
  write$ newline$ ←
"  \providebibliographyfont{url}{{\url}%" ←
  write$ newline$ ←
}
```

The lines containing \providebibliographyfont are output as the first lines of the **thebibliography** environment by the BIBTEX style file instead of defining them in the definition of the **thebibliography** environment for two reasons: first, it is then possible to use different default fonts with different BIBTEX styles; second, other packages may redefine the **thebibliography** environment without problems.

5.3 Additional data fields

As mentioned earlier, the `babelbib` BIBTEX styles support the additional data fields `isbn`, `issn`, and `url`. Now, we add them to the new `bst` file.

Like the field `language`, the names `isbn`, `issn`, and `url` have to be added to the `ENTRY` definition at the beginning of the `.bst` file (see section 5.1).

The new fields are formatted by the following functions:

```

FUNCTION {format.edition}
{ edition empty$
  { " " }
  { output.state mid.sentence =
      { edition "l" language.change.case "
          \btxeditionsshort{.}" * }
      { edition "t" language.change.case "
          \btxeditionsshort{.}" * }
    if$ }
  if$ }
}

FUNCTION {format.isbn}
{ isbn empty$
  { " " }
  { "\btxISBN~\btxISBNfont{" isbn *
    "}" * }
  if$ }
}

FUNCTION {format.issn}
{ issn empty$
  { " " }
  { "\btxISSN~\btxISSNfont{" issn *
    "}" * }
  if$ }
}

FUNCTION {format.url}
{ url empty$
  { " " }
  { "\btxurlfont{" url * "}" * }
  if$ }
}

```

The new fields have to be printed for all citations, where they are useful. For example, for books, an ISBN and maybe a URL is useful, while an ISSN is senseless. Thus, the function `book` looks like this:

```
FUNCTION {book}
{ output.bibitem
  :
  format.date "year" output.check
  format.isbn output
  format.url output
  format.language *
  note output
  fin.entry
}
```

In a similar way, we extend the functions `booklet`, `inbook`, `incollection`, `inproceedings`, `manual`, `masterthesis`, `misc`, `phdthesis`, `proceedings`, `techreport`, and `unpublished`.

6 Conclusion

This article has described how the `babelbib` package can be used to generate multilingual and flexible bibliographies. In addition, it has shown how the `babelbib` system can be extended to more languages and BIBTEX styles.

Since the package is still young, the number of supported languages and BIBTEX styles is somewhat limited. Thus, there are two main future topics: Both the number of languages and BIBTEX styles has to be increased. But I need help for both tasks.

I hope the package is already useful for generating bibliographies in many multilingual environments.

References

- [1] Braams, Johannes: *Babel, a multilingual package for use with L^AT_EX's standard document classes*, 2002. CTAN:macros/latex/required/babel/.
 - [2] Harders, Harald: *The babelbib package*, 2003. CTAN:biblio/bibtex/contrib/babelbib/.
 - [3] Raichle, Bernd: *Kurzbeschreibung german.sty und ngerman.sty*, 1998. CTAN:language/german/.
 - [4] Wallmeier, M., A. Scherer, and H. Harders: *Macros for german BIBTEXing*, 2000. CTAN: biblio/bibtex/contrib/germbib/.

◊ Harald Harders
Nußbergstraße 48
38102 Braunschweig
Germany
h.harders@tu-bs.de

Abstracts

Les Cahiers GUTenberg
Contents of Thematic Issue 41
(November 2001)

PIERRE FOURNIER, Éditorial : METAPOST le sessin sous *TeX/LaTeX* [Editorial: METAPOST, designing with *TeX/LaTeX*]; pp. 3–4

After presenting PSTricks as a set of tools designed to help integrate graphics into *LATEX* documents, a subject explored in *Cahier* issue no. 16, the editor introduces the current *Cahier* with a similar object, METAPOST.

This issue of the *Cahiers* includes not only a translation (done in collaboration with Jean-Côme Charpentier) of the original METAPOST manual, but also two additional contributions, one philosophical, the other practical. To better appreciate its value, Denis Roegel has a short piece which positions METAPOST in the more general context of graphic design software. The graph package, an extension to METAPOST, is also presented. The translations for the texts on both METAPOST and graph were authorised by John Hobby, the creator of METAPOST. The last article, by Fabrice Popineau, is a hands-on demonstration of METAPOST in action in a *LATEX* document, complete with verbatim code samples.

The editor hopes that the flow of the articles, from theory to application, will allow users to begin incorporating METAPOST into their own documents. Further information and help, along with real samples, can be found on various websites. Two such are cited:

[http://www.math.jussieu.fr/~zoonek/TeX/
Metapost/metapost.html](http://www.math.jussieu.fr/~zoonek/TeX/Metapost/metapost.html)

<http://melusine.eu.org/syracuse/metapost>

Indeed, the editor suggests that the end of the year provides an excellent excuse to apply METAPOST to one's greeting cards!

DENIS ROEGEL, METAPOST, l'intelligence graphique [METAPOST, graphical intelligence]; pp. 5–16

In this article, we position METAPOST with respect to various types of drawings. We argue in favor of a text-based format and we develop the concept of “graphical intelligence.” We illustrate how this intelligence appears in an example constructing a graph.

[Author's abstract (edited)]

JOHN HOBBY, Un manuel de l'utilisateur pour METAPOST [A User's manual for METAPOST]; pp. 17–139

The METAPOST system implements a picture-drawing language very much like Knuth's METAFONT except that it outputs PostScript commands instead of run-length-encoded bitmaps. METAPOST

is a powerful language for producing figures for documents to be printed on PostScript printers. It provides easy access to all the features of PostScript and its facilities for integrating text and graphics.

This document serves as an introductory user's manual.¹ It does not require knowledge of METAFONT or access to *The METAFONTbook*, but both are beneficial. An appendix explains the differences between METAPOST and METAFONT.

[Author's abstract (edited)]

JOHN HOBBY, Tracer les graphes avec METAPOST [Drawing graphs with METAPOST]; pp. 140–166

This paper² describes a graph-drawing package that has been implemented as an extension to the METAPOST graphics language. METAPOST has a powerful macro facility for implementing such extensions. There are also some new language features that support the graph macros. Existing features for generating and manipulating pictures allow the user to do things that would be difficult to achieve in a standalone graph package.

[Author's abstract]

FABRICE POPINEAU, METAPOST pratique [METAPOST, a practical exercise]; pp. 167–175

In this article, I will explain how to use METAPOST in practice. This program is very different from the usual drawing programs, but it fits very well into a *TeX*-based typesetting system.

[Author's abstract (edited)]

— * —

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

[http://www.gutenberg.eu.org/pub/gut/
publications](http://www.gutenberg.eu.org/pub/gut/publications)

[Compiled by Christina Thiele]

¹ This manual originally appeared in English, *A User's manual for METAPOST*. The French translation was done by Pierre Fournier and Jean-Côme Charpentier.

² This document originally appeared in English, *Drawing Graphs With METAPOST*. The French translation was done by Pierre Fournier and Jean-Côme Charpentier.

Calendar

2003

- Apr 2–4 DANTE 2003, 28th meeting, Universität Bremen, Germany. For information, visit <http://www.dante.de/dante2003/>.
- May 1–3 BachoTeX 2003, 11th annual meeting of the Polish TeX Users' Group (GUST), Bachotek, Brodnica Lake District, Poland. For information, visit <http://www.gust.org.pl/BachoTeX/2003/>.
- May 22 NTG 31st meeting, Hogeschool Helicon, Zeist, Netherlands. For information, visit <http://www.ntg.nl/bijeen/bijeen31.html>.
- May 28–30 Society for Scholarly Publishing, 25th annual meeting, "Navigating Change", Baltimore, Maryland. For information, visit <http://www.sspnet.org>.
- May 29–Jun 2 ACH/ALLC 2003: Joint International Conference of the Association for Computers and the Humanities, and Association for Literary and Linguistic Computing, "Web X: A Decade of the World Wide Web", University of Georgia, Athens, Georgia. For information, visit <http://www.english.uga.edu/webx/> or the organization web site at <http://www.ach.org>.
- Jun 11–13 Seybold Seminars PDF Summit, Amsterdam, Netherlands. For information, visit http://www.seybold365.com/pdf_summit/.
- Jun 24–27 EuroTeX 2003, "Back to Typography", Brest (Brittany), France. For information, visit <http://omega.enstb.org/eurotex2003/>. (The EuroTeX 2003 proceedings will be published in *TUGboat*.)

- Jul 7– Aug 8 Rare Book School Summer Session, University of Virginia, Charlottesville, Virginia. A series of one-week courses on topics concerning rare books, manuscripts, the history of books and printing, and special collections. For information, visit <http://www.virginia.edu/oldbooks>.

TUG 2003
**Outrigger Waikoloa Beach Resort,
Big Island, Hawai'i.**

- Jul 15–18 Beginning/Intermediate LATEX, at the University of Hawaii at Hilo. For information, visit <http://www.tug.org/tug2003/latexclass.html>.
- Jul 20–24 The 24th annual meeting of the TeX Users Group, "Silver Anniversary—25 years!—of TeX". For information, visit <http://www.tug.org/tug2003/>.
- Jul 17–20 TypeCon2003, "Counter Culture", Minneapolis, Minnesota. For information, visit <http://www.typecon2003.com/>.
- Jul 27– Aug 1 SIGGRAPH 2003, San Diego, California. For information, visit <http://www.siggraph.org/calendar/>.
- Aug 3 Web Document Analysis workshop, Edinburgh, Scotland, UK. For information, visit <http://www.csc.liv.ac.uk/~wda2003>.
- Aug 3–6 ICDAR 2003, International Conference on Document Analysis and Recognition, Edinburgh, Scotland, UK. For information, visit <http://www.essex.ac.uk/ese/icdar2003/>.
- Aug 4–8 Extreme Markup Languages 2003, Montréal, Québec, Canada. For information, visit <http://www.extrememarkup.com/extreme/>.

Status as of 1 December 2003

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

An updated version of this calendar is online at <http://www.tug.org/calendar/>.

Additional type-related events are listed in the Typophile calendar, at

<http://www.icalx.com/html/typophile/month.php?cal=Typophile>.

Owing to the lateness of this issue, please consider that events shown for 2003 are included only "for the record".

- Sep 3–5 IUC24, The 24th Internationalization and Unicode Conference: “Unicode, Internationalization, the Web: Powering Global Business”, Atlanta, Georgia. For information, visit <http://www.unicode.org/iuc/iuc24/>.
- Sep 8–9 DANTE 29th meeting, Universität Giessen, Germany. For information, visit <http://www.dante.de/events/>.
- Sep 8–12 Seybold San Francisco, San Francisco, California. For information, visit <http://www.seybold365.com/sf2003/>.
- Sep 11–12 Journées GUTenberg, Mâcon, France. For information, visit <http://www.gutenberg.eu.org/manifestations/gut2003/>.
- Sep 25–26 EGUTH 2003, CervanTeX meeting, Universidad de Murcia, Murcia, Spain. For information, visit <http://www.latex.um.es/CervanTeX/eguth03/>.
- Sep 25–29 Association Typographique Internationale (ATypI) annual conference, “Between Text and Reader”, Vancouver, Canada. For information, visit http://www.atypi.org/40_conferences.
- Oct 20–21 Second Annual St. Bride Conference, “Hidden Typography”, London, England. For information, visit <http://www.stbride.org/conference2003/>.
- Nov 9 NTG program on Open Standards and Open Source Software, Utrecht, Netherlands. For information, visit <http://www.ntg.nl/bijeen/bijeenOSOSS.html>.
- Nov 22 UK TUG Autumn meeting, Nottingham University. For information, visit <http://uk.tug.org/>.
- Nov 13 NTG 32nd meeting, Arnhem, Netherlands. For information, visit <http://www.ntg.nl/bijeen/bijeen32.html>.
- Nov 20–22 ACM Symposium on Document Engineering, Grenoble, France. For information, visit <http://www.documentengineering.org>.
- Nov 24–26 6^e Colloque International sur le Document Électronique, Caen, France. For information, visit <http://infodoc.unicaen.fr/cide/cide.6/>.
- Dec 7–12 XML 2003, Philadelphia, Pennsylvania. For information, visit http://www.idealliance.org/events_upcoming.asp.
- Dec 10–12 The Printing Press and its Variants, International Conference on Bibliography and the History of the Book, Florence, Italy. For information, visit <http://web.uniud.it/poliphilo/>.
-
- ## 2004
- Jan 5–6 Printing and the worlds of learning, Colloquium organized by the Printing Historical Society, Cambridge, UK. For information, visit <http://www.printinghistoricalsociety.org.uk/phs.html>.
- Jan 5–9 Rare Book School, University of Virginia, Charlottesville, Virginia. Several one-week courses on topics concerning typography, bookbinding, book illustration and electronic texts. For information, visit <http://www.virginia.edu/oldbooks>.
- Jan 16– Mar 5 In Flight: A traveling juried exhibition of books by members of the Guild of Book Workers. University of Utah, Salt Lake City, Utah. Sites and dates are listed at <http://palimpsest.stanford.edu/byorg/gbw>.
- Mar 3–5 DANTE 2004, 30th meeting, “15 Jahre DANTE”, Technisches Universität Darmstadt, Germany. For information, visit <http://www.dante.de/dante2004/>.
- Mar 22– May 7 In Flight: A traveling juried exhibition of books by members of the Guild of Book Workers. University of Washington, Seattle, Washington. Sites and dates are listed at <http://palimpsest.stanford.edu/byorg/gbw>.
- Mar 31– Apr 2 IUC25, The 25th Internationalization and Unicode Conference: “Unicode in Government: Building a Multilingual Infrastructure”, Washington, DC. For information, visit <http://www.unicode.org/iuc/iuc25/>.
- Apr 5–8 Book History Workshop, Institut d'histoire du livre, Lyon, France. For information, visit <http://ihl.enssib.fr/>.
- Apr 19–21 Seybold Seminars Amsterdam 2004, Netherlands. For information, visit <http://www.seybold365.com/ams2004/>.

Apr 30 – May 3	BachoTeX 2004, 12 th annual meeting of the Polish TeX Users' Group (GUST), Bachotek, Brodnica Lake District, Poland. For information, visit http://www.gust.org.pl/BachoTeX/2004/ . Proposals for papers are due by 16 February 2004.	Jul 20 – 24	SHARP Conference (Society for the History of Authorship, Reading and Publishing), Lyon, France For information, visit http://sharpweb.org/ .
Jun 11 – 16	ALLC/ACH-2004, Joint International Conference of the Association for Computers and the Humanities, and Association for Literary and Linguistic Computing, "Computing and Multilingual, Multicultural Heritage", Göteborg University, Sweden. For information, visit http://www.hum.gu.se/allcach2004/ or the organization web site at http://www.ach.org .	Jul 22 – 25	TypeCon2004, "Type High", San Francisco, California. For information, visit http://www.typecon2004.com/ .
Jun 24 – 29	2 nd International Conference on Typography and Visual Communication: Communication and new technologies, Thessaloniki, Greece. For information, visit http://www.uom.gr/uompress/ .	Aug 8 – 12	SIGGRAPH 2004, Los Angeles, California. For information, visit http://www.siggraph.org/calendar/ .

TUG 2004**Democritus University of Thrace,
Xanthi, Greece.**

Aug 30 – Sep 3	The 25 th annual meeting of the TeX Users Group, "XML and Digital Typography". For information, visit http://www.tug.org/tug2004/ .
-------------------	--

Oct 18 – 19	Third Annual St. Bride Conference, "Bad Type", London, England. For information, visit http://www.stbride.org/conference.html .
-------------	--

Institutional Members

American Mathematical Society,
Providence, Rhode Island

Banca d'Italia,
Roma, Italy

Center for Computing Science,
Bowie, Maryland

Cessna Aircraft Company,
Wichita, Kansas

The Clarinda Company,
Clarinda, Iowa

CNRS - IDRIS,
Orsay, France

CSTUG, *Praha, Czech Republic*

Florida State University,
School of Computational Science
and Information Technology,
Tallahassee, Florida

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

Institute for Advanced Study,
Princeton, New Jersey

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

Kluwer Academic Publishers,
Dordrecht, The Netherlands

KTH Royal Institute of
Technology, *Stockholm, Sweden*

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

Siemens Corporate Research,
Princeton, New Jersey

Springer-Verlag Heidelberg,
Heidelberg, Germany

Stanford Linear Accelerator
Center (SLAC),
Stanford, California

Stanford University,
Computer Science Department,
Stanford, California

Stockholm University,
Department of Mathematics,
Stockholm, Sweden

University College, Cork,
Computer Centre,
Cork, Ireland

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

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

Vanderbilt University,
Nashville, Tennessee

Workshops and Presentations on L^AT_EX, T_EX, and A_MS-L^AT_EX

Practical T_EX 2004: Training and Techniques

**Holiday Inn Fisherman's Wharf
San Francisco, California
July 19–22, 2004**

<http://tug.org/practicaltex2004>
conferences@tug.org

Who should attend?

Mathematicians	University & corporate (I)T _E X documentation staff
Students	Publishing company production staff
Scientists	Researchers

... and anyone who uses or is considering using the L^AT_EX and T_EX technical documentation system.

Further information

This three-day conference will have user-oriented presentations and workshops on L^AT_EX, T_EX, A_MS-L^AT_EX, ConT_EXt and their use in document production. Parallel tracks will be offered to accommodate a wide range of users, from beginning to advanced. Expert speakers and lecturers will present experiences and techniques useful to all.

Conference attendees will enjoy an opening night reception and a Chinatown banquet. Coffee and lunch will be served each day of the meeting. Located in the Fisherman's Wharf area, with easy access to many of San Francisco's colorful sights.

Post-conference workshops

On the fourth day, July 22, courses will be offered focusing on specific areas, such as Intermediate and Advanced L^AT_EX training, and A_MS-L^AT_EX techniques for mathematicians.

Contact: conferences@tug.org

If you would like to present a paper or hold a workshop, please email us. Registration information and hotel reservations will be available soon through the web site.

Hope to see you there!

(Sponsored by the T_EX Users Group.)

T_EX Collection

T_EX Live + CTAN

2CDs+DVD
Edition 9/2003

dante e.V.
Postfach 10 18 40
69008 Heidelberg
dante@dante.de
www.dante.de

LEHMANNS
FACHBUCHHANDLUNG



Editor of T_EX Live: Sebastian Rahtz – <http://www.tug.org/texlive>

Editor of CTAN snapshot: Manfred Lotz – <http://www.ctan.org>

AsTeX – CervantTeX – CSTUG – CTUG – CyrTUG – DK-TUG – Estonian User Group – εφτ – GUIT – GUST – GUTenberg – GUTpt – ITALIC – KTUG – Lietuvos T_EX'o Vartotojų Grupė – MaTeX – Nordic T_EX Group – NTG – T_EXCeH – T_EX México – Tirant lo T_EX – TUG – TUGIndia – TUG-Philippines – UK TUG – ViêtTUG

The latest T_EX Live software distribution was produced in September 2003. It's available by joining TUG (see membership application in this issue, or <http://tug.org/join.html>), or by separate purchase from the TUG store (<http://tug.org/store>). Most other T_EX user groups also make it available to their members. The T_EX Live home page is <http://tug.org/texlive>.



Late-Breaking News

Production Notes

Mimi Burbank

This is the final issue of 2002, and represents a double issue. We are slowly making progress toward catching up, but we still need articles.

Some new members have joined the *TUGboat* production team—Karl Berry, our new President, and William Adams, who is the editor of the 2003 conference proceedings.

Karl and Robin Laakso have been soliciting articles and we have had a wonderful response from the *TeX* community. Thanks to all of you who have responded to the call for articles; we hope many more step up and submit papers. For more information about how to do this, please refer to the *TUGboat* web page: <http://www.tug.org/TUGboat>.

Output The final camera copy was prepared at CSIT on a Debian Linux computer, using the *TeX Live* 2003 setup, i386-linux, which is based on Web2C implementation of *TeX*, version 7.5.2.

One curiosity occurred with production of the Beccari article on Typesetting classical Greek philology (p. 276). *TeX*'s output routine insisted on producing the running heads in Greek—until I realized that I could achieve the same output if I forced a page break where I needed it, and the running heads reverted to the usual roman. Whew.

All of the articles contained in this issue were generated as PDF files, many by PDFTeX or ps2pdf. We also used ConTeXt macros to overlay multiple pages, where one article ends and another begins on the same page; these were kindly provided by Hans Hagen. This is the first time our printer has not had to strip pages for us. Thank you Hans!

Coming Next Volume Coming in 2003 will be the proceedings of the 24th annual meeting of the *TeX* Users Group, held at the Outrigger Waikoloa Beach Resort on the Kohala Coast of the Big Island, Hawaii. Also in 2003, we plan on providing the proceedings of EuroTeX 2003, thanks to Yannis Haralambous.

- ◊ Mimi Burbank
CSIT
Florida State University
Tallahassee, FL 32306-4120
mimi@csit.fsu.edu

TUG Business

Report: TUG 2003 Election

The number of candidates nominated for the open offices in the 2003 TUG election fell short of the number which would require a ballot.

The office of President was open, as were up to 13 positions on the Board of Directors. There was one candidate for President, Karl Berry, and 10 candidates for the Board: Barbara Beeton, Kaja Christiansen, Susan DeMeritt, Stephanie Hogue, Ross Moore, Cheryl Ponchin, Samuel Rhoads and Philip Taylor. (Arthur Ogawa and Michael Sofka are continuing board members, whose terms expire in 2005.)

According to the TUG Election Procedures, when the number of candidates is fewer than the number of open positions, all candidates who have met the qualifications are declared elected by acclamation. The term of the President expires as of the annual meeting in 2005; the terms of Board members in this class expire at the meeting in 2007.

Since no ballots were mailed, TUG members have not had the opportunity to read the biographies and personal statements of the candidates. Without this information it is difficult to know each candidate's particular interests, and their vision for the future of TUG. The information which would have accompanied the ballot follows this report, to introduce these individuals to the membership.

Since the election date, the President has exercised his prerogative and appointed two additional candidates who were not already on the Board to assume office immediately: James Hefferon and Gerree Pecht. Information from these two directors is included below, although technically it would not have appeared on the ballot.

The Committee acknowledges the diligent work of our office manager, Robin Laakso, in receiving, organizing, and validating membership of nominees and their respective nominators.

Arthur Ogawa
for the Elections Committee

New Members of the TUG Board

Barbara Beeton



Biography:

TeX Users Group: charter member of TUG; charter member of TUG Board of Directors; *TUGboat* production staff since 1980, Editor since 1983; committees: publications, bylaws, elections; chair, Technical Working Group on Extended Math Font Encoding; liaison from Board to Knuth Scholarship Committee 1991–1992.

Employed by American Mathematical Society: Staff Specialist for Composition Systems; involved with typesetting of mathematical texts since 1973; assisted in initial installation of TeX at AMS in 1979; implemented the first AMS document styles; created the map and ligature structure for AMS cyrillic fonts.

Standards organizations: active 1986–1997 in: ANSI X3V1 (Text processing: Office & publishing systems), ISO/IEC JTC1/SC18/WG8 (Document description and processing languages); developing the standard ISO/IEC 9541:1991 Information technology—Font information interchange.

AFII (Association for Font Information Interchange): Board of Directors, Secretary 1988–1996.

STIX representative to the Unicode Technical Committee for adoption of additional math symbols.

Personal statement:

TUG has changed over the years, with its transition from an appointed to an elected Board. Those charged with shaping its future direction have tried to do so in a way that encourages participation by all members, not just a few. Similarly, the typographic landscape has changed as well, and though the object that is our focus—TeX—is still a tool of undeniable utility, it is just part of a growing pool of text processing software, some of it borrowing from the features that first attracted us to TeX. I maintain my commitment to Don Knuth’s original goals for this tool: high typographic quality and portability. Within this framework, my goal is to continue working for unconstrained communication among TeX users, to encourage exploration of techniques consistent with the typographic excellence we have come to expect,

and to act as a historian of the TeX community when that is appropriate.

I expect to retire from the AMS in about four years, so this will probably be my final candidacy for the TUG Board.

Karl Berry



Biography:

First contact in 1982. Subsequently, many installations at many organizations (not to mention many readings of the *TeXbook* and *METAFONT book*). Co-author of *TeX for the Impatient* (now freely available), one of the first comprehensive non-Knuthian books on TeX.

I was the maintainer of the Unix port of TeX (i.e., Web2c) for several years in the 1990’s. Along with Web2c, I developed Kpathsea, a freely redistributable library for path searching and variants of three DVI drivers that use it; Eplain, a macro package that extends plain TeX; *modes.mf*, a collection of METAFONT modes and adaptations; a list of short font names for portable use within TeX across platforms; and assorted minor projects. I am also the maintainer and primary developer for GNU Texinfo, a TeX-based documentation format.

Besides such programming tasks, I’ve also produced the usual books, articles, collections, and ephemera, studied typeface design, and co-written several articles on reading research and mathematical analysis of type.

For TUG, I serve on the technical council and various committees, co-sponsored the creation of the TeX Development Fund this year (2002), and act as one of the system administrators for the *tug.org* server. I was a TUG board member for two terms before deciding to run for president this year.

Personal statement:

I believe TUG can best serve the TeX community by working as an organization in partnership with the other user groups worldwide, and sponsoring worthwhile technical projects that will increase interest in TeX in the larger computing world.

Kaja P. Christiansen**Biography:**

I was born in Warsaw, Poland. After obtaining an MSc in Mathematics at the University of Warsaw, I eventually moved to Denmark. I came to love my new country, where I have now lived and worked for more than 20 years.

My job at the Department of Computer Science of the University of Aarhus involves system administration, and software support, including the responsibility for all aspects of a well-functioning TeX & friends: local styles, in-house classes and (very) frequent user support, and maintenance. The department has about 550 students, 80 employees, a large number of active research groups, and close ties to the BRICS Research Centre.

I heard about TeX for the first time in fall of 1979. In Palo Alto at the time, I wanted to audit courses at Stanford; my top priority was lectures by Prof. Donald Knuth but that, I was told, was not possible as Prof. Knuth was on leave due to work on a *text processing project* ... This project was TeX! Back home, it didn't take long till we had a runnable system and thus introduced an early version of TeX in Denmark.

Personal statement:

I have served as the chair of TUG's Technical Council since 1997 and co-sponsored the creation of the TeX Development Fund. I share system administrator's responsibilities for the TUG server (whose access to the Internet is currently facilitated by my Department). In my rôle as a member of the board, my special interests have been projects of immediate value to the TeX community: TeX Live, *TUGboat* and TUG's web site.

Since September 2002, I have served as the president of the Danish TeX Users Group (DK-TUG).

Susan DeMeritt**Biography:**

My name is Susan DeMeritt, I live in Lakeside, CA (just outside San Diego).

I am employed by the Center for Communications Research, La Jolla, in San Diego, California for almost 14 years doing technical typing in the Publications Department. I started the position learning TeX and working up to L^AT_EX₂ ε . I enjoy using L^AT_EX₂ ε to typeset mathematical and scientific papers.

I have been a member of the TeX Users Group since 1989. I have been a member of the Board of Directors since March of 1998, and Secretary since 2001. I really enjoy being part of the Board of Directors of the TeX Users Group and I hope my participation has been helpful.

I have successfully taught (along with Cheryl Ponchin) two L^AT_EX classes, one at Duke University and one at the University of Delaware.

Jim Hefferon**Biography:**

I got a PhD in Mathematics in 1986 and soon after I got interested in writing an undergraduate textbook in *Linear Algebra*, (which I have made available on the web with the L^AT_EX source). I spent many hours fussing with the symbols and was unable to make it look right. I figured that I needed a program that knows how to make the stuff as readable as possible, and obviously that led me to TeX.

In the course of using the system I became incredibly impressed by the community. I wanted to help out, so when Robin (Fairbairns) complained one day about the lack of USA CTAN mirrors, I offered to set one up. I eventually took over the main American node of CTAN, one of the core servers. I'm enjoying that work a great deal.

Stephanie Hogue

Biography:

Stephanie Hogue has been a member of TUG since 1991. She has over twenty years experience in typesetting mathematical documents. During her fifteen years with the Wharton School at the University of Pennsylvania, she typeset research and class materials and provided *T_EX* and *L^AT_EX* support for the faculty of the Finance Department. After leaving the Wharton School, Stephanie freelanced for five years as “The TypeWright,” offering *T_EX* and *L^AT_EX* typesetting services.

In 2001, she accepted a position with AlphaSimplex Group, a quantitative asset management company located in Cambridge, Massachusetts. In her current position as Data Archivist, she is responsible for supporting the use of *L^AT_EX* by the research group at AlphaSimplex.

Stephanie served on the Conference Committee for *T_EX NE* (held in New York in Spring 1998) and as the Co-Chair of the Program Committee for TUG 1999 (Vancouver) and TUG 2001 (Delaware). She joined the Board of Directors in 1999 and is presently on the Program Committee for TUG 2003 (Hawaii).

Personal statement:

One of the major accomplishments of the outgoing President and Board was the IRS reclassification of TUG as a non-profit organization. TUG faces a number of challenges, and the non-profit status should give a welcome boost to our fund-raising efforts.

Some of the areas on which I would like to focus are:

- developing support mechanisms to assist users with the numerous *T_EX*-based systems and hundreds of related packages;
- utilizing the web and other electronic technologies for both support and education; and
- integrating our efforts more closely with the international community of local *T_EX* user groups.

Gerree Pecht

Biography:

My name is Gerree P. Pecht. I'm a Technical Research Publications Specialist in the Mathematics department at Princeton University.

I joined Princeton University in 1975 as a Technical Research Secretary I. Eventually graduating to the highest title (at the time) Technical Research Secretary IV. I was introduced to my first computer by Professor Jeff Ullman (now at Stanford); I type(set) his book (with Al Aho of Bell Labs) around 1978. ... the process was called Phototypesetting ...

I taught myself the necessary macro packages (ms macros: designed by a scientist at Bell Labs—nroff, troff) necessary for document preparation ... the file was transferred to a role of film on Kodak printer ... the printer processed it the old-fashioned way ... developing took place in a dark room ... pages were hung to dry and then Xeroxed ...

... the start of camera-ready-copies?

I was introduced to *L^AT_EX* by Professor Sedgewick (he was Chair of the CS department and was typesetting his own book in *L^AT_EX* at the time) while working in the Computer Science department in the mid-1980's.

I started using *L^AT_EX* right from the beginning ... (I wasn't aware that there was such a thing as “plain *T_EX*” ...) I'm self-taught (*L^AT_EX* (L. Lamport), *A Guide to L^AT_EX₂ ε* (K. Daly), *L^AT_EX Graphics Companion* (Goossens, Rahtz, Mittelbach), *Math into L^AT_EX* (Gratzer), *T_EX for the Impatient* (Abrahams) ...) In my constant search for latest developments in the *T_EX* world, I came across a book by George Gratzer called *Math into T_EX: a Simple Introduction to A_MS-L^AT_EX* I'm known for *bundling* many packages including all of the above ... as each package contain some unique conventions, commands, declarations, etc., each useful when producing a customized book, manuscript, slides, graphics, etc.

I very frequently design (special/highly technical) macros to accommodate the project at hand in the preparation of all aspects of lectures, transparencies (using *SLiT_EX* program-highlighting certain formulas and words in color). And combining graphics with many formulas is commonplace in my position as a Technical Research Publications Specialist in the department.

I willingly and unconditionally “share” my many customized templates with faculty/staff/students throughout the University(s) and am eager to help one-on-one. I don't know how I

became the “in-house” technical typesetter for the faculty ... typesetting many complicated Math books, but I’m proud to be referred to as the L^AT_EX “guru” and the department’s resource for graduate students (who are required to typeset their Ph.D. thesis in some form of T_EX).

I just completed typesetting two highly technical Mathematics texts for Professor Elias M. Stein. The books were done in L^AT_EX (graphics, tables, figures, everything!!) and even if I have to say so myself ... they turned out absolutely elegant!!!

Up until the summer of 2001, I worked on a Unix platform. Since then, the system was changed to Linux platform.

Not so long ago ... someone mentioned to me that “L^AT_EX” (T_EX-formatting) ... “is dead.” So why then are universities and the technical/scientific (including government) agencies using it? Seems to me it’s alive and well ... all-around ... I’m a great promoter for the TUG organization.

I have met quite a few wonderful T_EXies through attending TUG conferences and via telephone/email I look forward to this “new” relationship with TUG.

My credo: L^AT_EX it! L^AT_EX it! A_MS-L^AT_EX it! A_MS-T_EX it!!! Use S_LT_EX packages as well as the graphics packages as the Princeton Tiger would say it’s Grrreat!!

Cheryl Ponchin

Biography:

My name is Cheryl Ponchin, I live in Princeton, New Jersey.

I am employed by the Center for Communications Research in Princeton. I have been a technical typist for almost 20 years. I started with T_EX and I am now using L^AT_EX₂ ε as well as many of the different packages available. I enjoy using this software to typeset mathematical and scientific papers.

I have been a member of the T_EX Users Group since 1989. I have been a member of the TUG Board of Directors since March of 1998. I really enjoy being part of the TUG group.

I have taught (with Sue DeMerritt) two L^AT_EX classes for TUG (the University of Delaware and Duke University). We will also be teaching another class at the University of Hawaii at Hilo in July. I am also teaching classes at Princeton University. I have also reviewed *A Guide to L^AT_EX₂ ε* , which was very interesting and rewarding for me.

Samuel E. Rhoads



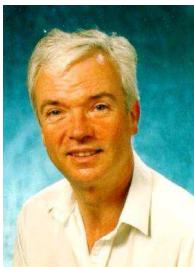
Biography:

I hold a bachelors, a masters and a doctor of arts degree in mathematics. I took a job teaching math at the University of Guam in 1973 and came to Hawai‘i from Guam in 1981. I’ve been teaching math and computer science since the early ’60s. I am now a professor in the Information and Computer Science department at Honolulu Community College in Honolulu, Hawai‘i.

My interest in T_EX began in the ’80s. In the late ’80s I decided that I wanted to write a textbook for the beginning computer science class—I thought I could write a better book than the books that were available—and I decided to try to do it in T_EX so I could retain control of how the book appeared. I later came to understand that I didn’t know enough about publishing to really design my own book, but it was fun to be able to do it anyway. I prepared a short paper describing the experience and presented that paper at TUG 91 (“Authors new to T_EX publish a textbook with a publisher new to T_EX,” *TUGboat* 12(3), pages 387–393).

Wm. C. Brown Publishers published the book in 1990. A textbook for the second course in computer science was also completed and published in 1992. I also wrote and published (in T_EX) a book of Star Charts for the night sky as seen from Hawai‘i. That book is published by the Bishop Museum Press, and is now in its third edition.

I use T_EX (actually L^AT_EX) in much of what I do. In particular, whenever I do something mathematical and want the output to look “right,” I use T_EX. I always use T_EX to prepare exams for the math classes that I teach, but I use a word processor for routine correspondence and other printed material—like this biography—when it’s too much trouble to use T_EX. If I were able to, I’d design a word processor that made T_EX easier to use. If I could use it as easily as I use MS Word, I would use T_EXfor everything I print.

Philip Taylor**Biography:**

Philip Taylor has been a *TEX* devotee ever since a visit (many years ago) to British Petroleum where he saw equipment *identical* to that which he was using (a Digital VAX, with Digital LN03 printer) producing output infinitely better than anything he could manage. Having found that BP were accomplishing this using *TEX*, he immediately took a copy on magnetic tape, and has never looked back.

Although he now spends more time on electronic publishing than on typesetting *per se*, he is still a staunch advocate of *TEX*, and is a member of the TUG Board, Chairman of the TUG Bursary Committee, a past Chairman of the UK *TEX* Users' Group, Programme Committee Chairman for Euro*TEX*'99, and Technical Director of the NTS project. He sees his rôle within the Board as representing the interests of "the ordinary member," and tries to ensure that Board decisions afford equal opportunities and rights to all members.

Financial statements for 2001 and 2002

Robin Laakso

For purposes of comparison, this financial report covers both of the years 2001 and 2002. Also, due to an oversight, our annual financial report was not included in any *TUGboat* 2001 issue, so this serves to remedy that. As always, the accounts have been reviewed by TUG's accountant but have not been audited.

In 2001 and for most of 2002, the *TEX* Users Group was a not-for-profit corporation under section 501(c)(6) the Internal Revenue code. The IRS determined TUG would be exempt from federal income tax as described in section 501(c)(3) as of September 10, 2002. TUG notified members about the change in U.S. income tax status shortly after we were notified on February 27, 2003. The principal benefit of being classified 501(c)(3) is that contributions to TUG are now generally tax-deductible (less any value received), at least in the United States.

TUG is funded primarily by annual dues from members. Unfortunately, we had a drop in membership from 2001 to 2002, from approximately 1900 to 1800 members; however, this resulted in only a slight drop in membership revenue. Because of write offs (non-renewing institutional and subscriber memberships) and other irregularities, such as joint membership dues paid after the membership year, the "Membership Dues" figure (as seen on the Profit & Loss statement) only approximately reflects dues activity each year.

TUG's largest annual expense items are payroll, *TUGboat* production and mailing, and software production and mailing. Because the last three issues of *TUGboat* Volume 21 (2000) are included along with Volume 22 (2001), the "TUGboat Production/Mailing" P&L line item in 2001 is inflated by about \$25,000. Had Volume 21 been accounted for in the year the expense should have occurred (as is the case in 2002 and later), net income in 2001 would be a gain of about \$10,500 instead of the reported loss of \$14,507.

This verbal portion of the report is intended to highlight major features contained in the financial statements, but obviously does not include the detail necessary to explain activities within each account. If you would like to learn more about TUG's finances or have a particular comment or question, please contact the TUG office.

◊ Robin Laakso
TUG Executive Director
office@tug.org

TeX Users Group
Balance Sheet Prev Year Comparison
As of December 31, 2002

	Dec 31, 02	Dec 31, 01
ASSETS		
Current Assets		
Checking/Savings		
OregonTelco CD 80144	128,258.16	0.00
BofA Maximizer 21203-18374	51,779.66	100,160.66
BoA Maximizer 21208-18966	0.00	16,847.69
BOA CD - 21202-09486	0.00	37,856.10
BOA Checking - 21203-10859	5,729.83	-1,393.08
BOA Money Mkt Bursry 2120411698	1,330.11	2,934.43
Petty Cash	10.00	15.41
Total Checking/Savings	<u>187,107.76</u>	<u>156,421.21</u>
Accounts Receivable		
Accounts Receivable	7,270.00	5,362.12
Total Accounts Receivable	<u>7,270.00</u>	<u>5,362.12</u>
Other Current Assets		
Employee Advances	0.00	-1,500.00
Total Other Current Assets	<u>0.00</u>	<u>-1,500.00</u>
Total Current Assets	<u>194,377.76</u>	<u>160,283.33</u>
Fixed Assets		
Fixed Assets	7,059.12	8,030.91
Total Fixed Assets	<u>7,059.12</u>	<u>8,030.91</u>
TOTAL ASSETS	<u>201,436.88</u>	<u>168,314.24</u>
LIABILITIES & EQUITY		
Liabilities		
Current Liabilities		
Accounts Payable	40,124.01	44,540.12
Total Accounts Payable	<u>40,124.01</u>	<u>44,540.12</u>
Other Current Liabilities		
Deferred conference donations	610.00	0.00
Deferred conference income	7,360.00	0.00
Deferred contributions	1,500.00	0.00
Deferred member income	21,175.00	0.00
AMS Prepaid Memberships	1,800.00	1,800.00
Payroll Liabilities	1,211.58	1,498.01
Total Other Current Liabilities	<u>33,656.58</u>	<u>3,298.01</u>
Total Current Liabilities	<u>73,780.59</u>	<u>47,838.13</u>
Total Liabilities	<u>73,780.59</u>	<u>47,838.13</u>
Equity		
Restricted Bursary as of 12/31	1,330.11	3,029.43
Restricted LaTeX as of 12/31	167.50	687.50
Unrestricted as of 1/1	118,978.50	131,491.53
Current Yr Chg in Restrict Acct	0.00	-225.00
Net Income	7,180.18	-14,507.35
Total Equity	<u>127,656.29</u>	<u>120,476.11</u>
TOTAL LIABILITIES & EQUITY	<u>201,436.88</u>	<u>168,314.24</u>

TeX Users Group
Profit & Loss Prev Year Comparison
January through December 2002

	Jan - Dec 02	Jan - Dec 01
Ordinary Income/Expense		
Income		
Membership Dues	125,215	126,614
Product Sales	3,050	1,372
Contributions Income	5,065	5,259
Annual Conference		-1,088
Annual Regional Conference	-363	
Annual Pre-conf Class	-314	-1,580
Interest Income	5,130	6,166
Advertising Income	1,345	1,045
Bursary	301	983
LaTeX 3	-520	-780
2003 Annual Conference	-411	
Total Income	<u>138,498</u>	<u>137,991</u>
Cost of Goods Sold		
TUGboat Prod/Mailing	24,189	54,213
Software Production/Mailing	13,659	13,250
Postage/Delivery - Members	4,184	4,636
Conf Expense, office + overhead		4,588
Member Renewal	420	517
Copy/Printing for members	60	316
Total COGS	<u>42,512</u>	<u>77,520</u>
Gross Profit	<u>95,986</u>	<u>60,471</u>
Expense		
Contributions made by TUG	5,942	4,000
Office Overhead	8,021	7,518
Payroll Exp	60,460	56,272
Contract Labor	375	90
Professional Fees	14,886	1,203
Credit card/Bank charges	3,137	3,231
Depreciation Expense	2,786	2,665
Interest Expense	3	
Total Expense	<u>95,610</u>	<u>74,979</u>
Net Ordinary Income	<u>376</u>	<u>-14,508</u>
Other Income/Expense		
Other Income		
Prior year adjust (01-02)	6,806	
Total Other Income	<u>6,806</u>	
Net Other Income	<u>6,806</u>	
Net Income	<u>7,182</u>	<u>-14,508</u>

Bugs in *Computers & Typesetting*

6 July 2003

This is a list of all substantial corrections made to *Computers & Typesetting* since the publication of the Millennium Edition at the close of the year 2000. (More precisely, it lists errors corrected since the 16th printing of Volume A, the 7th printing of Volume B, the 6th printing of Volume C, the 4th printing of Volume D, and the 5th printing of Volume E.) Corrections made to the softcover version of *The T_EXbook* are the same as corrections to Volume A. Corrections to the softcover version of *The METAFONTbook* are the same as corrections to Volume C. Changes to the mini-indexes and master indexes of Volumes B, D, and E are not shown here unless they are not obviously derivable from what has been shown. Some (or all) of these errors have been corrected in the most recent printings.

Page A16, line 7 from the bottom	(6/30/01)
----------------------------------	-----------

Ten-point type is different from **magnified five-point type**.

Page A17, line 7	(6/30/01)
------------------	-----------

fications that grow in geometric ratios—something like equal-tempered tuning

Page A51, lines 18–20	(6/30/01)
-----------------------	-----------

`ff` yields ff; `fi` yields fi; `f1` yields fl; `ffi` yields ffi; `ffl` yields ffl;
`“` yields “; `”` yields ”; `!‘` yields ¡; `?‘` yields ¡;
`--` yields –; `---` yields —.

Page A52, line 7 from the bottom	(6/30/01)
----------------------------------	-----------

`\ae`, `\AE` æ, Å (Latin ligature and Scandinavian letter AE)

Page A71, line 15	(6/30/01)
-------------------	-----------

One of the interesting things that can happen when glue stretches and

Page A180, line 20	(6/30/01)
--------------------	-----------

Challenge number 5: $k = 1.38065 \times 10^{-16} \text{ erg K}^{-1}$.

Page A254, line 12 from the bottom becomes two lines	(4/09/01)
--	-----------

```
\output=\unvbox255
\ifnum\outputpenalty<10000 \penalty\outputpenalty\fi}
```

Page A292, lines 13–16	(6/30/01)
------------------------	-----------

■ `\mathchoice`{`<filler>`}{`<math mode material>`}{`<filler>`}{`<math mode material>`} `<filler>` {`<math mode material>`}. Four math lists, which are defined as in the second alternative of a `<math field>`, are recorded in a “choice item” that is appended to the current list.

Page A306, line 7	(6/30/01)
-------------------	-----------

instead of a shelfful. In fact, the latter idea—to insert an italic correction—is prefer-

Page A308, lines 25 and 26 (6/17/02)

```
\def\appendroman#1#2#3{\expandafter\def\expandafter#1\expandafter
{\csname\expandafter\gobble\string#2\romannumeral#3\endcsname}}
```

Page A311, line 14 (12/2/02)

```
\def\\{\if\space\next\ % assume that \next is unexpandable
```

Page A323, line 12 from the bottom (6/30/01)

18.31. $k=1.38065 \times 10^{-16} \text{rm}, \text{erg}, K^{-1}.$.

Page A364, lines 12–15 from the bottom (01/21/03)

```
\def\loggingall{\tracingcommands=2 \tracingstats=2
\tracingpages=1 \tracingoutput=1 \tracinglostchars=1
\tracingmacros=2 \tracingparagraphs=1 \tracingrestores=1
\showboxbreadth=\maxdimen \showboxdepth=\maxdimen \errorstopmode}
\def\tracingall{\tracingonline=1 \loggingall}
```

Page A364, line 5 from the bottom (01/21/03)

```
\def\fmtversion{3.14159265} % identifies the current format
```

Page A450, lines 14–16 from the bottom (12/19/02)

```
s1tic1 exp1 x3p1 pi3a1i1a1i2a1i2al1 i2d1 do1 ci1 io1 ou2 us1
```

(where subscripts that aren't shown are zero), and this yields

```
.0s0u1p0e0r1c0a0l1i0f0r0a0g1i0l4i0s1t2i0c1e0x3p2i3a0l2i1d0o1c2i0o2u2s0.
```

Page A451, line 15 (1/30/01)

Connecticut Yankee come out with only nine or ten bad hyphens:

Page A451, line 23 (1/30/01)

```
mo-er-der-mohren-mut-ter-mar-mor-mon-u-menten-macher.
```

Page A454, lines 23–30 (6/30/01)

 If a suitable starting letter is found, let it be in font *f*. Hyphenation is abandoned unless the `\hyphenchar` of *f* is a number between 0 and 255, inclusive. If this test is passed, TeX continues to scan forward until coming to something that's not one of the following three “admissible items”: (1) a character in font *f* whose `\lccode` is nonzero; (2) a ligature formed entirely from characters of type (1); (3) an implicit kern. The first inadmissible item terminates this part of the process; the trial word consists of all the letters found in admissible items. Notice that all of these letters are in font *f*.

Page A461, right column (7/08/01)

`*\char`, 43–45, 76, 86, 155, 283, [286](#),

Page A466, left column (7/09/01)

`*\floatingpenalty`, [123–124](#), 272, 281, [363](#).

Page A470, left column (1/21/03)

`\loggingall`, [364](#).

Page A473, left column (6/30/01)

orphans, see widow words.

Page Bvii, bottom two lines (12/20/02)

all of those changes. I now believe that the final bug was discovered and removed on 20 December 2002. The finder's fee has converged to \$327.68.

Page B2, line 10 from the bottom (12/20/02)

```
define banner ≡ `This is TeX, Version 3.141592' { printed when TeX starts }
```

Page B3, new paragraph to follow line 9 (12/20/02)

Incidentally, Pascal's standard *round* function can be problematical, because it disagrees with the IEEE floating-point standard. Many implementors have therefore chosen to substitute their own home-grown rounding procedure.

Page B8, line 2 (5/04/01)

statements will be meaningful. We insert the label '*exit*' just before the '**end**' of a procedure in

Page B30, line -4 (5/04/01)

```
begin update_terminal; { now the user sees the prompt for sure }
```

Page B84, lines 22 and 27 (5/04/01)

```
ignore = 9 { characters to ignore ( ^@ ) }
active_char = 13 { characters that invoke macros ( ~ ) }
```

Page B139, line 20 (12/19/02)

```
begin while (state = token_list) ∧ (loc = null) ∧ (token_type ≠ v_template) do
  end_token_list; { conserve stack space }
```

Page B206, line 14 (10/30/02)

used input files like `webmac.tex`.

Page B206, new paragraph to follow line 22 (12/20/02)

The following procedures don't allow spaces to be part of file names; but some users seem to like names that are spaced-out. System-dependent changes to allow such things should probably be made with reluctance, and only when an entire file name that includes spaces is "quoted" somehow.

Page B256, line 25 (12/20/02)

```
cur_glue: real; { glue seen so far }
cur_g: scaled; { rounded equivalent of cur_glue times the glue ratio }
begin cur_g ← 0; cur_glue ← float_constant(0);
this_box ← temp_ptr; g_order ← glue_order(this_box); g_sign ← glue_sign(this_box);
```

Page B258, line 5 from the bottom (12/20/02)

```
begin g ← glue_ptr(p); rule_wd ← width(g) − cur_g;
```

Page B258, bottom line (12/20/02)

```
begin cur_glue ← cur_glue + stretch(g); vet_glue(float(glue_set(this_box)) * cur_glue);
cur_g ← round(glue_temp);
```

Page B259, line 4 (12/20/02)

```
begin cur_glue  $\leftarrow$  cur_glue - shrink(g); vet_glue(float(glue_set(this_box)) * cur_glue);
cur_g  $\leftarrow$  round(glue_temp);
```

Page B259, new line to precede old line 7 (12/20/02)

```
rule_wd  $\leftarrow$  rule_wd + cur_g;
```

Page B260, line 21 (12/19/02)

```
else begin lx  $\leftarrow$  lr div (lq + 1);
```

Page B261, line 9 (12/20/02)

```
cur_glue: real; { glue seen so far }
cur_g: scaled; { rounded equivalent of cur_glue times the glue ratio }
begin cur_g  $\leftarrow$  0; cur_glue  $\leftarrow$  float_constant(0);
this_box  $\leftarrow$  temp_ptr; g_order  $\leftarrow$  glue_order(this_box); g_sign  $\leftarrow$  glue_sign(this_box);
```

Page B262, line 10 from the bottom (12/20/02)

```
begin g  $\leftarrow$  glue_ptr(p); rule_ht  $\leftarrow$  width(g) - cur_g;
```

Page B262, line 6 from the bottom (12/20/02)

```
begin cur_glue  $\leftarrow$  cur_glue + stretch(g); vet_glue(float(glue_set(this_box)) * cur_glue);
cur_g  $\leftarrow$  round(glue_temp);
```

Page B262, line 2 from the bottom (12/20/02)

```
begin cur_glue  $\leftarrow$  cur_glue - shrink(g); vet_glue(float(glue_set(this_box)) * cur_glue);
cur_g  $\leftarrow$  round(glue_temp);
```

Page B263, new line to precede old line 2 (12/20/02)

```
rule_ht  $\leftarrow$  rule_ht + cur_g;
```

Page B264, line 10 (12/19/02)

```
else begin lx  $\leftarrow$  lr div (lq + 1);
```

Page B280, lines 23 and 24 (4/08/01)

or unset nodes; in particular, each mlist item appears in the variable-size part of *mem*, so the *type* field is always present.

Page B299, line 9 (12/20/02)

```
if type(r) = kern_node then { unneeded italic correction }
```

Page B332, line 6 (12/19/02)

is being scanned, or when no alignment preamble is active.

Page B332, line 8 (12/19/02)

```
begin if (scanner_status = aligning)  $\vee$  (cur_align = null) then
```

Page B382, line 6 (1/01/01)

between ‘fl’ and ‘y’, then $m = 2$, $t = 2$, and y_1 will be a ligature node for ‘fl’ followed by an

Page B386, line 11 (4/08/01)

qi(2), qi(6): begin cur_r ← rem_byte(q); { |=:, |=:> }

Page B472, new paragraph to follow line 10 (12/20/02)

A devious user might force an *endv* command to occur just about anywhere; we must defeat such hacks.

Page B472, replacement for what used to be line 13 (12/20/02)

```
begin base_ptr ← input_ptr; input_stack[base_ptr] ← cur_input;
while (input_stack[base_ptr].index_field ≠ v_template) ∧
    (input_stack[base_ptr].loc_field = null) ∧
    (input_stack[base_ptr].state_field = token_list) do decrec(base_ptr);
if (input_stack[base_ptr].index_field ≠ v_template) ∨
    (input_stack[base_ptr].loc_field ≠ null) ∨
    (input_stack[base_ptr].state_field ≠ token_list) then
    fatal_error(`(interwoven_alignment_preambles_are_not_allowed)`);
if cur_group = align_group then
```

Page B475, line 12 (7/01/01)

end; { now we are in vertical mode, working on the list that will contain the display }

Page C11, line 11 (10/11/01)

the area below the bar to the area above it equal to $(\sqrt{5} + 1)/2 \approx 1.61803$, the

Page C29, illustration for exercise 4.11 (9/09/01)

[points 2 and 5 should not be labeled twice]

Page C156, line 15 from the bottom (9/09/01)

be the values they had upon entry to the group.)

Page C171, line 16 from the bottom (6/18/02)

<loop> → <loop header> : <loop text> endfor

Page C179, line 7 from the bottom (9/09/01)

next time METAFONT gets to the end of an input line, it will stop reading from the

Page C204, line 3 from the bottom (7/08/01)

slightly. If *autorounding* > 1, you get even more changes: Paths are perturbed slightly

Page C238, lines 9 and 8 from the bottom (7/08/01)

tance is $\text{length}(z_4 - z_1)$. But there's a slicker solution: Just calculate

abs ypart((z₁ - z₂) rotated -angle(z₃ - z₂)).

Page C286, line 25 (9/09/01)

problem; it would simply have put **ENDFOR** into the replacement text of **asts**, because

Page C289, line 7 (9/09/01)

```
if if pair x: x>(0,0) else: false fi: A else: B fi.
```

Page C292, line 10 from the bottom (9/09/01)

be known by saying ‘**if** known $p - q: p = q$ **else: false fi**’; transforms could be handled

Page C313, bottom line (6/30/01)

— LA ROCHEFOUCAULD, *Maximes* (1665)

Page C346, left column (6/18/02)

*:, 169, 171, 317–319.

Page C346, right column (7/09/01)

***angle**, 29, 67, 72, 107, 135, 211, 238.

Page C352, left column (6/30/01)

La Rochefoucauld, François VI, 313.

Page C357, right column (7/08/01)

***true**, 55, 64–65, 170, 210.

Page Dvii, bottom two lines (12/21/02)

corporates all of those changes. I now believe that the final bug was discovered on 22 January 2001, and removed in version 2.71828. The finder’s fee has converged to \$327.68.

Page D2, line –17 (12/21/02)

define *banner* ≡ ‘This is METAFONT, Version 2.71828’ { printed when METAFONT starts }

Page D2, lines 4 and 5 from the bottom (12/23/02)

types; there are no ‘**var**’ parameters, except in the case of files or in the system-dependent *paint_row* procedure; there are no tag fields on variant records; there are no *real* variables; no procedures are declared local to other procedures.)

Page D8, line 2 (5/04/01)

statements will be meaningful. We insert the label ‘*exit*’ just before the ‘**end**’ of a procedure in

Page D28, line –8 (5/04/01)

begin *update_terminal*; { now the user sees the prompt for sure }

Page D42, replacement for lines 8–13 (12/23/02)

Notice that if 64-bit integer arithmetic were available, we could simply compute $(2^{29} * p + q) \text{ div } (2 * q)$. But when we are restricted to Pascal’s 32-bit arithmetic we must either resort to multiple-precision maneuvering or use a simple but slow iteration. The multiple-precision technique would be about three times faster than the code adopted here, but it would be comparatively long and tricky, involving about sixteen additional multiplications and divisions.

Page D43, line 20 (12/23/02)

language or 64-bit substitute is advisable.

Page D44, lines 24–26 (12/23/02)

Once again it is a good idea to use 64-bit arithmetic if possible; otherwise `take_scaled` will use more than 2% of the running time when the Computer Modern fonts are being generated.

Page D101, line 21 (7/08/01)

```
define subscr_head_loc(#) ≡ # + 1 { where value, subscr_head, and attr_head are }
```

Page D180, lines 22 and 23 (1/26/01)

$(y, -x)$ will appear in node p . Similarly, a fourth-octant transformation will have been applied after the transition, so we will have $x_{coord}(q) = -x$ and $y_{coord}(q) = y$.

Page D184, line 18 (12/21/02)

```
chopped: integer; { positive if data truncated, negative if data dangerously large }
```

Page D184, line 25 (12/21/02)

```
if (internal[autorounding] > 0)  $\wedge$  (chopped = 0) then xy-round;
```

Page D184, line 27 (12/21/02)

```
if (internal[autorounding] > unity)  $\wedge$  (chopped = 0) then diag-round;
```

Page D184, line 32 (12/21/02)

```
if (internal[autorounding] ≤ 0)  $\vee$  (chopped ≠ 0) then print-spec(",\u22a5after\u22a5subdivision")
```

Page D185, lines 15–19 (12/21/02)

```
define procrustes(#) ≡ if abs(#) ≥ dmax then
    if abs(#) > max_allowed then
        begin chopped ← 1;
        if # > 0 then # ← max_allowed else # ← −max_allowed;
        end
    else if chopped = 0 then chopped ← −1
```

Page D185, old line 22 (12/21/02)

```
p ← cur-spec; k ← 1; chopped ← 0; dmax ← half(max_allowed);
```

Page D185, old line 28 (12/21/02)

```
if chopped > 0 then
```

Page D196, lines 7 and 8 (1/26/01)

where $x'(t) \geq 0$ we have *right_type* = *first_octant* or *right_type* = *eighth_octant*; in regions where $x'(t) \leq 0$, we have *right_type* = *fifth_octant* or *right_type* = *fourth_octant*.

Page D511, line 17 (7/03/01)

from appearing again.

Page E7, line 11 (12/21/02)

hair, vair, stem, curve, ess, flare, dot_size, bar, slab,

Page E7, line 11 (12/21/02)

crisp, tiny, fine;

and *thin_join* should not be less than *fine*.

Page E9, line 9 (7/03/01)

[92] [123] [124])))

Page E19, line 19 (11/7/01)

cap_notch_cut 46/36 31/36 25/36 24/36 22/36 25/36

Page E41, line 8 (12/21/02)

extra_endchar \leftarrow *extra_endchar* & "charcode:=charcode+code_offset;"

Page E53, line 7 (12/21/02)

numeric *mid_thickness*; *mid_thickness* = Vround $^{1/3}$ [*vair, stem*];

Page E377, lines 3 and 4 from the bottom (12/22/02)

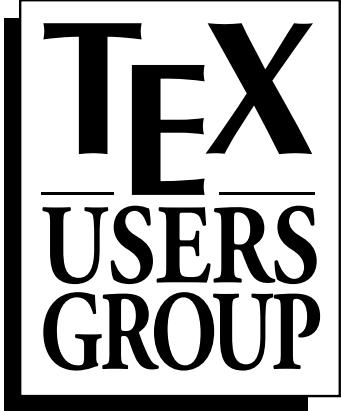
```

path p_-; p_- = z$$l{z@1 - z$$l} ... darkness[z@1, .5[z@2, z$$l]] ... z@2
      --- z$l -- z$r -- z@0 -- z$$r -- cycle;
if (y$ > y$)  $\neq$  (ypart precontrol 1 of p_- > ypart postcontrol 1 of p_-):
  p_- = z$$l{z@1 - z$$l} ... darkness[z@1, .5[z@2, z$$l]]
      --- z$l -- z$r -- z@0 -- z$$r -- cycle; fi
filldraw p_-; % arm and beak

```

Page E577, right column (12/23/02)

padded, 103–111, 117–121, 549.



Promoting the use of
TeX throughout the
world.

mailing address:

P.O. Box 2311
Portland, OR 97208-2311 USA

shipping address:

1466 NW Naito Parkway
Suite 3141
Portland, OR 97209-2820 USA

phone: +1 503-223-9994
fax: +1 503-223-3960
email: office@tug.org
web: http://www.tug.org

President Karl Berry
Vice-President Kaja Christiansen
Treasurer Samuel Rhoads
Secretary Susan DeMeritt
Executive Director Robin Laakso

2004 T_EX Users Group Membership Form

TUG membership rates are listed below. Please check the appropriate boxes and mail the completed form with payment (in US dollars, drawn on a US bank) to the mailing address at left. If paying by credit/debit card, you may alternatively fax the form to the number at left or join online at <http://tug.org/join.html>. The web page also provides more information than we have room for here.

Status (check one) New member Renewing member

	Rate	Amount
<input type="checkbox"/> Early bird membership for 2004 (TUGboat, software) After May 31, dues are \$75.	\$65	_____
<input type="checkbox"/> Special membership for 2004 (TUGboat, software) You may join at this special rate (\$45 after May 31) if you are a senior, student, new graduate, or from a country with a modest economy. See http://tug.org/join.html .	\$35	_____
<input type="checkbox"/> Subscription for 2004 (TUGboat, software) non-voting	\$85	_____
<input type="checkbox"/> Institutional membership for 2004 (TUGboat, software) Includes up to seven individual memberships.	\$500	_____

Last year's materials

If you were not a TUG member in 2003, this is your option to receive software immediately.

<input type="checkbox"/> T _E X Live 2003 software 2 CD's and 1 DVD which also includes CTAN.	\$15	_____
<input type="checkbox"/> CTAN 2003 CD-ROMs	\$15	_____
<input type="checkbox"/> TUGboat Volume 24	\$15	_____

Voluntary donations

<input type="checkbox"/> General TUG contribution	_____
<input type="checkbox"/> Bursary Fund contribution Financial assistance for attending the TUG Annual Meeting.	_____
<input type="checkbox"/> T _E X Development Fund contribution Financial assistance for technical projects.	_____
<input type="checkbox"/> Request sending of CTAN on CD (shipped on DVD to everyone)	n/a

Total \$ _____

Tax deduction: \$30 of the early bird membership fee is deductible, at least in the US.

Multi-year orders: To join for more than one year at this year's rate, just multiply.

Payment (check one) Payment enclosed Visa/MasterCard/AmEx

Account Number: _____

Exp. date: _____ Signature: _____

Privacy: TUG uses your personal information only to send products, publications, notices, and (for voting members) official ballots. TUG neither sells its membership list nor provides it to anyone outside of its own membership.

Electronic notices will generally reach you much earlier than printed ones. However, you may choose not to receive any email from TUG, if you prefer.

Do not send me any TUG notices via email.

TUG may prepare a printed or electronic membership directory, available to TUG members only. If you would like to be listed in such a publication, please check this box.

Do include my information in a published members-only TUG directory.

Name _____

Department _____

Institution _____

Address _____

City _____ State/Province _____

Postal code _____ Country _____

Email address _____

Phone _____ Fax _____

Position _____ Affiliation _____

TeX Consulting & Production Services

North America

Loew, Elizabeth

President, TeXniques, Inc.
675 Massachusetts Avenue, 6th Floor
Cambridge, MA 02139
(617) 876-2333; Fax: (781) 344-8158
Email: 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 TeX files.

Ogawa, Arthur

40453 Cherokee Oaks Drive
Three Rivers, CA 93271-9743
(209) 561-4585
Email: arthur.ogawa@teleport.com

Bookbuilding services, including design, copyedit, art, and composition; color is my specialty. Custom TeX macros and LaTeX2\epsilon document classes and packages. Instruction, support, and consultation for workgroups and authors. Application development in LaTeX, TeX, SGML, PostScript, Java, and BC++. Database and corporate publishing. Extensive references.

Veytsman, Boris

2239 Double Eagle Ct.
Reston, VA 20191
(703) 860-0013
Email: boris@lk.net

I provide training, consulting, software design and implementation for Unix, Perl, SQL, TeX, and LaTeX. I have authored several popular packages for LaTeX and latelex2html. I have contributed to several web-based projects for generating and typesetting reports. For more information please visit my web page: <http://users.lk.net/~borisv>.

The Unicorn Collaborative, Inc, Ted Zajdel

115 Aspen Drive, Suite K
Pacheco, CA 94553
(925) 689-7442
Email: contact@unicorn-collab.com

We are a technical documentation company, initiated in 1990, which time, strives for error free, seamless documentation, delivered on time, and within budget. We provide high quality documentation services such as document design, graphic design and copy editing. We have extensive experience using tools such as FrameMaker, TeX, LaTeX, Word, Acrobat, and many graphics programs. One of our specialties is producing technical manuals and books using LaTeX and TeX. Our experienced staff can be trained to use any tool required to meet your needs. We can help you develop, rewrite, or simply copy-edit your documentation. Our broad experience with different industries allows us to handle many types of documentation including, but not limited to, software and hardware systems, communications, scientific instrumentation, engineering, physics, astronomy, chemistry, pharmaceuticals, biotechnology, semiconductor technology, manufacturing and control systems. For more information see our web page <http://www.unicorn-collab.com>.

Outside North America

DocuTeXing: TeX Typesetting Facility

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

DocuTeXing provides high-quality TeX and LaTeX 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.

Information about these services can be obtained from:

TeX Users Group
1466 NW Naito Parkway, Suite 3141
Portland, OR 97208-2311, U.S.A.

Phone: +1 503 223-9994
Fax: +1 503 223-3960
Email: office@tug.org
URL: <http://www.tug.org/consultants.html>